SPAMR: Extending PAMR with stochastic time[⋆]

Natalia López, Manuel Núñez and Ismael Rodríguez

Dept. Sistemas Informáticos y Computación
Universidad Complutense de Madrid, E-28040 Madrid. Spain.
e-mail: {natalia,mn,isrodrig}@sip.ucm.es

Abstract. In this paper we introduce time information in PAMR (Process Algebra for the Management of Resources). PAMR is a process algebra that simplifies the task of specifying processes whose behavior strongly depend on the resources that they have. One of the drawbacks of PAMR is that there is not an appropriate notion of time. In this paper we will consider that the duration of actions is controlled by a random variable. These random variables will take values, according to some probability distribution functions, that may depend, in particular, on the available resources. We present two examples showing the main features of our stochastic version of PAMR.

1 Introduction

The process algebra PAMR [18], as well as its adaptions/extensions [20,21], represents a formalism to specify systems where resources play a fundamental role. The main advantage of PAMR consists in the separation between the *behavior* of processes (specified in a usual process algebra) and the management of the *resources* that a process can make use of. By doing so, specifications get clearer as many details can be encoded in the resources part. Let us show the main advantages by using a simple example. Let us consider the classical five dining philosophers example. If we try to specify this problem in a CCS-like language, we have to consider that forks are usual processes. Besides, the philosophers must also include some actions indicating *communication* with the forks. In PAMR we may separate between *real* processes (the philosophers) and available resources (the forks). In our case, a philosopher is simply defined as

$$\texttt{philosopher}_i ::= \texttt{think}\,;\texttt{eat}\,;\texttt{philosopher}_i$$

We consider that the philosopher i needs no resources to think. Besides, he needs forks i and $(i \bmod 5)+1$ to perform the action eat.

Another interesting feature of PAMR allows processes to exchange resources among them. By doing so, processes will improve their performances. For example, consider a system where two processes are running. One of them is making intensive use of RAM while the other one is mainly sending packets through a

⋆ Research partially supported by the Spanish MEC project WEST/FAST TIN2006-15578-C02-01 and the Marie Curie RTN *TAROT* (MRTN-CT-2003-505121).

LAN. So, the first process should have a big amount of memory (and a limited access to the local bus) while the situation should be the opposite for the other process. Besides, let us suppose that this situation may change as time passes (for example, the first process may eventually become an I/O-bound process). In order to model such a situation, we could consider a centralizer allocating the available resources to processes. Nevertheless, in PAMR no centralizer is needed. Processes will exchange resources according both to some predefined policy and to some information about their preferences towards resources. So, an exchange is allowed if it is profitable with respect to the chosen policy, for example, if some processes improve and no one deteriorates. This first policy is equivalent to consider that processes are the real owners of resources. Another possibility is that exchanges are allowed only if the whole system improves. The results of this policy are similar to consider a centralizer who looks for an improvement of the global behavior of the system.

Even though PAMR has been already applied to different fields (from concurrent systems to a variant of e-commerce [20] or to the specification of autonomous agents [21,17]) we think that there is (at least) an aspect that could be more satisfactorily covered: The modeling of time. In PAMR, a primitive notion of time could be introduced by using the *necessity function*. In short, each process has a function $n : \mathtt{Act} \times \mathbb{R}_+^n \longrightarrow \mathbb{R}_+ \cup \{\infty\}$. A value $n(a, \bar{x}) = \infty$ indicates that the amount of resources owned by the process (that is, $\bar{x}$) is not enough to perform the action a. If $n(a, \bar{x}) = r \in \mathbb{R}_+$, the value r could be considered as the *time* that the process needs to perform a, given the resources $\bar{x}$. In fact, this approach was somehow implemented in [19]. However, it presents some problems. First, in PAMR there is no notion of concurrent passing of time. In other words, if we have a parallel composition of several processes, the passage of time in one of the processes could not be reflected in the other ones. Second, this notion of time would be completely deterministic, that is, for any n, a, and $\bar{x}$ the *time* that the process needs to perform a is fixed. It seems more interesting to consider a *stochastic* approach where the necessity function returns a random variable.So, we may consider that $n(a, \bar{x}) = \xi$ indicates that, if the process may use the resources $\bar{x}$, then it will perform the action a before time t with probability equal to $F_\xi(t)$, where F_ξ is the probability distribution function associated with ξ. For example, we could have that ξ is uniformly distributed over the interval $[0, \frac{1}{\sum x_i}]$.

In order to include our notion of time, we profit from the study in the field of *stochastic process algebras* (e.g. [8,1,11,3,10]). Even though most of the models that were originally proposed were restricted to only use exponential distributions, there are already several proposals for stochastic models with general distributions (e.g. [9,13,5,15,4,14,6,7]). The restriction to *Markovian* models simplifies several of the problems that appear when considering general distributions. In particular, some quantities of interest, like reachability probabilities or steady-state probabilities, can be efficiently calculated by using well known methods on Markov chains. Besides, the (operational) definition of the language is usually simpler than the one for languages allowing general distributions. Nevertheless, this restriction does not allow to properly specify systems where time distributions are not exponential. Moreover, the main weakness of non-exponential models, that is the analysis of properties, can be (partially) overcome by restricting the class of distributions. A good candidate are phase-type distributions, where the analysis of performance measures can be efficiently done in some general cases. There are other proposals dealing with these limitations in a different way. For example, [12,22] show how model checking can be extended to semi-Markov Chains while [15] presents a framework for translating specifications with general distributions into a functional language where simulations can be performed.

In order to cope with the usual problem of general distributions in the scope of parallel compositions, actions are split into two events: *start* and *termination*. A similar mechanism is used in [5,15]. Nevertheless, in those approaches stochastic and action transitions are separated (already in the syntax on the corresponding languages). In our case, transitions must contain both information about the action and the associated random variable. Besides, processes engaged in the execution of an action (that is, they have performed a start event but they did not perform the corresponding termination event) will have some trading limitations. For example, they will not be able to trade resources that they need to perform the corresponding action. These two facts complicate the operational rules for the definition of systems.

The rest of the paper is structured as follows. In Section 2 we present our language, that we call SPAMR and its operational semantics. In Section 3 we present two examples showing the main features of SPAMR. Finally, in Section 4 we present our conclusions and some lines for future work.

2 The Language SPAMR

In this section we present the syntax and operational semantics of our language. In SPAMR, as well as in PAMR, systems are defined in two steps. First, we introduce the notion of process. A *process* consists of a behavior (expressed in a LOTOS-like language) together with some information about resources (the amounts that it owns, how they are consumed/produced after performing actions, etc). A *system* is defined as the parallel composition of several processes. These processes may perform exchanges of resources so that they improve with respect to a given policy. Once no more exchanges can be made, indicating that a (somehow) good distribution of resources has been reached, processes may perform actions (possibly by synchronizing with other processes). Next we introduce some preliminary notations.

Definition 1. We consider $\mathbb{R}_+ = \{x \in \mathbb{R} \mid x \geq 0\}$. For any $r \in \mathbb{R}_+$, we have that $\texttt{trunc}(r)$ denotes the natural number resulting from discarding the decimal part of the real r.

We will usually denote *vectors* in $\mathbb{R}^n$ (for $n \geq 2$) by $\overline{x}, \overline{y}, \ldots$ Given $\overline{x} \in \mathbb{R}^n$, x_i denotes its *i-th* component. Let $\overline{x}, \overline{y} \in \mathbb{R}^n$. We define the addition of $\overline{x}$ and $\overline{y}$ as the vector $\overline{x} + \overline{y} = (x_1 + y_1, \ldots, x_n + y_n)$. We write $\overline{x} \leq \overline{y}$ if we have $x_i \leq y_i$ for all $1 \leq i \leq n$.

Let A be a set and $n, m \geq 1$. We denote by $A^{n\times m}$ the *matrices* of dimension $n \times m$ of elements of A (we use calligraphic letters $\mathcal{E}, \mathcal{E}_1 \dots$ to range over $A^{n\times m}$). □

Stochastic information is introduced by means of *random variables*. We will consider that the sample space (that is, the domain of random variables) is the set of real numbers $\mathbb{R}$ and that random variables take positive values only in $\mathbb{R}^+$, that is, given a random variable ξ we have $F_\xi(t) = 0$ for all $t < 0$. The reason for this restriction is that random variables will always be associated with time distributions.

Definition 2. We denote by $\mathcal{V}$ the set of random variables. We will extend this set with a special symbol $\xi_\infty \notin \mathcal{V}$. It will be used to represent the case when an action cannot be performed (because the process does not own enough resources). We consider $\mathcal{V}_* = \mathcal{V} \cup \{\xi_\infty\}$ ($\xi, \psi, \dots$ to range over $\mathcal{V}_*$).

Let ξ be a random variable. Its *probability distribution function*, denoted by F_ξ, is the function $F_\xi : \mathbb{R} \to [0, 1]$ such that $F_\xi(x) = \mathrm{P}(\xi \leq x)$, where $\mathrm{P}(\xi \leq x)$ is the probability that ξ assumes values less than or equal to x.

We consider a *distinguished* random variable. By *unit* we denote a random variable such that $F_{unit}(x) = 1$ for all $x \geq 0$, that is, *unit* is distributed as the Dirac distribution in 0.

Let ξ_1 and ξ_2 be independent random variables with probability distribution functions F_{ξ_1} and F_{ξ_2}, respectively. We define the *combined addition* of ξ_1 and ξ_2, denoted by $\xi_1 \oplus \xi_2$, as the random variable with probability distribution function defined as $F_{\xi_1 \oplus \xi_2}(x) = F_{\xi_1}(x) + F_{\xi_2}(x) - F_{\xi_1}(x) \cdot F_{\xi_2}(x)$. This operator can be generalized to an arbitrary (finite) number of random variables. Let $\Psi = \{\xi_i\}_{i \in I}$ be a non-empty finite set of independent random variables. We define the *combined addition* of the variables in Ψ, denoted by $\oplus\Psi$, as the random variable with probability distribution function defined, for any $x \in \mathbb{R}$, as $F_{\oplus\Psi}(x) = \sum_{\emptyset \subset \Phi \subseteq \Psi} (-1)^{(|\Phi|+1)} F_{\otimes\Phi}(x)$ where $F_{\otimes\Psi}(x) = \prod_{i \in I} F_{\xi_i}(x)$. Note that for singleton sets of random variables, $\Psi = \{\xi\}$, we have $\oplus\Psi = \xi$. We consider, for convenience, $F_{\oplus\emptyset}(x) = 0$ for all $x \in \mathbb{R}$.

Let ξ and ϕ be two random variables. We say that ξ and ϕ are *identically distributed*, denoted by $\xi \asymp \phi$, if for any $t \geq 0$, we have that $F_\xi(t) = F_\phi(t)$. □

The operator $\oplus$ can be used when random variables are combined. Let us note that this operator does not correspond with the usual definition of addition of random variables (we will denote that addition of random variables by $+$). Actually, it is easy to show that $F_{\oplus\Psi}$ computes the probability distribution function associated with the random variable $\min(\Psi)$. Finally, let us remark that if we consider the addition of random variables, for all random variable ξ we have that $\xi + unit = \xi$.

Example 1. Let us show some examples of random variables and the probability distribution functions governing their behavior. Let us consider the following probability distribution functions:

$$F_1(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ \frac{x}{3} & \text{if } 0 < x < 3 \\ 1 & \text{if } x \geq 3 \end{cases}$$

$$F_2(x) = \begin{cases} 0 & \text{if } x < 4 \\ 1 & \text{if } x \geq 4 \end{cases}$$

$$F_3(x) = \begin{cases} 1 - e^{-3\cdot x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

If a random variable ξ_1 has as associated probability distribution function F_1 then we say that ξ_1 is uniformly distributed in the interval $[0, 3]$. Uniform distributions allow us to keep compatibility with time intervals in (non-stochastic) timed models in the sense that the same *weight* is assigned to all the times in the interval. If ξ_2 has as associated probability distribution function F_2 then we say that ξ_2 follows a Dirac distribution in 4. The idea is that the corresponding delay will be equal to 4 time units. Dirac distributions allow us to simulate deterministic delays appearing in timed models. If ξ_3 has as associated probability distribution function F_3 then we say that ξ_3 is exponentially distributed with parameter 3. $\square$

One of the components of a process will be given by its *behavior*. These behaviors will be specified in a LOTOS-like processes algebra, but any other similar formalism could be used. We consider a fixed set of visible actions $\texttt{Act}$ ($a, b, \ldots$ to range over $\texttt{Act}$). We assume the existence of a special action $\tau \notin \texttt{Act}$ which represents the internal behavior of a process. We also consider the set of internal actions $\texttt{Act}_\tau$ such that $\tau \in \texttt{Act}_\tau$, $\texttt{Act}_\tau \cap \texttt{Act} = \emptyset$, and there exists a bijection $f : \texttt{Act} \longrightarrow (\texttt{Act}_\tau - \{\tau\})$ such that for all $a \in \texttt{Act}$, we will denote $f(a)$ by τ_a. We denote by $\texttt{ACT}$ the set of actions, that is, $\texttt{ACT} = \texttt{Act} \cup \texttt{Act}_\tau$ ($\alpha, \alpha', \ldots$ to range over $\texttt{ACT}$). Let us note that we do not consider a unique internal action, even though an *external* observer will not be able to distinguish them. The difference among them comes from the fact that they will need different resources to be performed. So, if a process needs a set of resources $\overline{x}$ to perform a visible action a, and this action is *hidden*, the resulting action, that is τ_a, needs the same amount of resources $\overline{x}$ to be performed. Sets of internal actions appear in other models for concurrent processes (for example, for I/O automata [16]). Finally, we consider a set of process variables Id.

Definition 3. The set of *basic processes*, denoted by $\mathcal{B}$, is given by the following BNF-expression $B ::= \text{stop} \mid X \mid \alpha \,;B \mid B + B \mid B \parallel_A B \mid \texttt{hide } A \texttt{ in } B \mid X := B$, where $\alpha \in \texttt{ACT}, A \subseteq \texttt{Act}$, and $X \in Id$. $\square$

The term stop defines the process that performs no actions. The process $\alpha \,; B$ performs the action α and, after that, it behaves like B. Choice is represented by $B + B'$, and the process will behave either like B or like B'. The expression $B \parallel_A B'$ represents the parallel composition of B and B' (with synchronization

$$(\text{Pre})\frac{}{\alpha;B \xrightarrow{\alpha} B} \qquad (\text{Rec})\frac{B\{X{:=}B/X\} \xrightarrow{\alpha} B'}{X{:=}B \xrightarrow{\alpha} B'}$$

$$(\text{Cho1})\frac{B_1 \xrightarrow{\alpha} B_1'}{B_1{+}B_2 \xrightarrow{\alpha} B_1'} \qquad (\text{Cho2})\frac{B_2 \xrightarrow{\alpha} B_2'}{B_1{+}B_2 \xrightarrow{\alpha} B_2'}$$

$$(\text{Par1})\frac{B_1 \xrightarrow{\alpha} B_1' \wedge \alpha{\notin}A}{B_1\|_A B_2 \xrightarrow{\alpha} B_1'\|_A B_2} \quad (\text{Par2})\frac{B_2 \xrightarrow{\alpha} B_2' \wedge \alpha{\notin}A}{B_1\|_A B_2 \xrightarrow{\alpha} B_1\|_A B_2'}$$

$$(\text{Par3})\frac{B_1 \xrightarrow{a} B_1' \wedge B_2 \xrightarrow{a} B_2' \wedge a{\in}A}{B_1\|_A B_2 \xrightarrow{a} B_1'\|_A B_2'}$$

$$(\text{Hid1})\frac{B_1 \xrightarrow{\alpha} B_1' \wedge \alpha{\notin}A}{\texttt{hide } A \texttt{ in } B_1 \xrightarrow{\alpha} \texttt{hide } A \texttt{ in } B_1'}$$

$$(\text{Hid2})\frac{B_1 \xrightarrow{a} B_1' \wedge a{\in}A}{\texttt{hide } A \texttt{ in } B_1 \xrightarrow{\tau_a} \texttt{hide } A \texttt{ in } B_1'}$$

Fig. 1. Operational Semantics for the Base Language.

set A). The term `hide` A `in` B will perform any action that B performs if the action is not in A; an action $a \in A$ will be hidden, becoming τ_a. The term $X := B$ represents a (possibly) recursive definition of a process. In Figure 1 the operational semantics for behaviors is presented. The rules are quite standard. Let us remind that $B\{B'/X\}$ represents the replacement of the free occurrences of the variable X in B by the term B'. Let us also remark that, in rule (Hid2), the result of hiding a visible action a is not τ but τ_a. Let us also note that the parallel operator works in an interleaving way. Concurrent execution of actions (by several processes) is considered only in the scope of a system. The following definition is given to compute the actions that a basic process may immediately perform.

Definition 4. Let $B \in \mathcal{B}$. We define its *set of immediate actions*, denoted by $\texttt{Imm}(B)$, as $\texttt{Imm}(B) = \{\alpha \in \texttt{ACT} \mid \exists\, B' \in \mathcal{B} :\ B \xrightarrow{\alpha} B'\}$. □

As we have already commented, a process consists of its basic behavior (previously defined), a set of assigned resources, and some information relating both resources and behavior.

Definition 5. Let us consider that there exists a number $m > 0$ of different resources. A *process* is a tuple $P = (B, \overline{x}, u, n, c)$, where $B \in \mathcal{B}$ (the *basic process* defining its behavior), $\overline{x} = (x_1, \ldots x_m) \in \mathbb{R}_+^m$ (the *amounts* of resources), $u : \mathcal{P}(\texttt{ACT}) \times \mathbb{R}_+^m \longrightarrow \mathbb{R}$ (the *utility* function), $n : \texttt{ACT} \times \mathbb{R}_+^m \longrightarrow \mathcal{V}_*$ (the *necessity* function), and $c : \texttt{ACT} \times \mathbb{R}_+^m \longrightarrow \mathbb{R}^m$ (the *consumption* of resources function). We denote by $\mathcal{P}$ the set of processes.

Given a process P_i, we will usually consider $P_i = (B_i, \overline{x_i}, u_i, n_i, c_i)$, that is, indices will denote the process to which $B, \overline{x}, \ldots$ are related. □

Next we briefly explain the components of a process. If $P = (B, \overline{x}, u, n, c)$ is a process then $\overline{x}$ indicates that P owns x_i units of the i-th resource. The *utility* function u contains *preferences* between *baskets* of resources. This function is applied to the set of actions that B can immediately perform and to a set of resources. It returns a real value. If we have that $u(A, \overline{x}) < u(A, \overline{y})$ then P would prefer the basket $\overline{y}$ to the basket $\overline{x}$, considering that $A = \texttt{Imm}(B)$. Given the fact that the utility function is only applied to sets of immediate actions, we suppose $u(\emptyset, \overline{x}) = 0$. That is, a *deadlocked* process does not need any resource, so they will be shared with the rest of the processes. The utility function plays a fundamental role in the exchange of resources as it allows to decide whether a new basket would improve the situation of the process.

The *necessity* function n relates the resources and the speed of execution of actions. Thus, if $n(\alpha, \overline{x}) = \xi \neq \xi_\infty$ then the action α will be performed by P, that owns the resources $\overline{x}$, following the probability distribution function F_ξ. In other words, α will be performed before time t with probability $F_\xi(t)$. If $n(\alpha, \overline{x}) = \xi_\infty$ then we have that the owned resources are not enough to perform α. It is important to note that, usually, random variables will depend on the resources the process owns. Finally, we assume $n(a, \overline{x}) = n(\tau_a, \overline{x})$.

The *consumption* of resources function indicates the consumed/produced resources after performing an action. That is, $c(\alpha, \bar{x}) = \bar{y}$ means that, after performing α, the set of resources of the process varies from $\bar{x}$ to $\bar{y}$. A necessary condition for a process to perform an action α is $c(\alpha, \bar{x}) \geq \bar{0}$. We do not allow *debts*, even transient ones, because they could generate inconsistencies. For example, such debts could produce that two processes simultaneously use a printer.

Systems will perform transitions according to the possible transitions of processes. These transitions are defined by the following operational rule:

$$\frac{B \xrightarrow{\alpha} B' \ \wedge \ n(\alpha, \overline{x}) = \xi \in \mathcal{V} \ \wedge \ c(\alpha, \overline{x}) \geq \overline{0}}{P \xrightarrow{(\alpha, \xi)} P'}$$

where $P = (B, \overline{x}, u, n, c)$ and $P' = (B', c(\alpha, \overline{x}), u, n, c)$. Intuitively, a process may perform an action if its corresponding behavior can, it has enough resources to perform it, and this performance does not produce any debts.

A system will be the parallel composition of several processes. We allow m among n synchronization. Let us consider the compositions of the processes $P_1, \ldots, P_n$. Each process has a synchronization set A_i. The process P_i is allowed to asynchronously perform any action in $\texttt{ACT} - A_i$; if P_i is able to perform an action $a \in A_i$ then P_i has to synchronize with the rest of processes P_j such that $a \in A_j$. In order to deal with the usual semantic problems of general distributions in the scope of a parallel operator, we have chosen an approach inspired in [5,15]. That is, actions are split into *start* (belonging to the set $\texttt{ACT}^+ = \{\alpha^+ \mid \alpha \in \texttt{ACT}\}$) and *termination* (belonging to the set $\texttt{ACT}^- = \{\alpha^- \mid \alpha \in \texttt{ACT}\}$). So, labels appearing in the forthcoming operational semantics for systems may contain elements from $\texttt{ACT}^+ \cup \texttt{ACT}^-$.

Definition 6. Let $A_1, \ldots A_n \subseteq \texttt{Act}$. A *system* S is a parallel composition of n processes $P_1, \ldots P_n$ synchronizing, respectively, in the set A_i. We denote the system S by ${}_M\|_n^{A_i} P_i$, where $M \subseteq \{1, \ldots, n\} \times \texttt{ACT} \times \mathcal{V} \times \mathbb{R}^m$. We denote the set of systems by $\mathcal{S}$. □

$$\frac{\texttt{valid}(S,\mathcal{E}) \ \wedge \ \texttt{allowed}(S,\mathcal{E})}{S \overset{\mathcal{E}}{\rightsquigarrow} {}_{M}\|_n^{A_i}\ P_i' \ \left[\forall\, 1 \leq i \leq n:\ P_i' = (B_i, \overline{x_i} - \sum_j \mathcal{E}_{ij} + \sum_j \mathcal{E}_{ji}, u_i, n_i, c_i)\right]}$$

$$\frac{S \not\rightsquigarrow \ \wedge P_j \xrightarrow{(\alpha,\xi)} P_j' \ \wedge \ \alpha \notin A_j \ \wedge \ j \notin \mathrm{Ind}(M)}{S \xrightarrow{(\alpha^+,\xi)_{\{j\}}} {}_{M_1}\|_n^{A_i}\ P_i'' \quad \left[\forall\, 1 \leq i \leq n,\ P_i'' = \begin{cases} P_j' \text{ if } i = j \\ P_i \text{ if } i \neq j \end{cases}\right]}$$

$$\frac{S \not\rightsquigarrow \ \wedge S \not\xrightarrow{\texttt{Act}^+} \ \wedge (j,\alpha,\xi,\bar{x}) \in M \ \wedge \ \alpha \notin A_j}{S \xrightarrow{(\alpha^-,\xi)_{\{j\}}} {}_{M_2}\|_n^{A_i}\ P_i}$$

$$\frac{S \not\rightsquigarrow \ \wedge P_j \xrightarrow{(a,\xi)} P_j' \ \wedge \ a \in A_j \ \wedge \ \forall\, k \in B(a) : (k \notin \mathrm{Ind}(M) \ \wedge \ \exists\, P_k' : P_k \xrightarrow{(a,\xi_k)} P_k')}{S \xrightarrow{(a^+,\psi)_{B(a)}} {}_{M_3}\|_n^{A_i}\ P_i'' \quad \left[\forall\, 1 \leq i \leq n,\ P_i'' = \begin{cases} P_i' \text{ if } a \in A_i \\ P_i \text{ if } a \notin A_i \end{cases}\right]}$$

$$\frac{S \not\rightsquigarrow \ \wedge S \not\xrightarrow{\texttt{Act}^+} \ \wedge (j,a,\psi,\bar{x}) \in M \ \wedge \ a \in A_j \ \wedge \ \forall k \in B(a) : (\exists\, \overline{x_k} : (k,a,\psi,\overline{x_k}) \in M)}{S \xrightarrow{(a^-,\psi)_{B(a)}} {}_{M_4}\|_n^{A_i}\ P_i}$$

where

$S = {}_{M}\|_n^{A_i}\ P_i,\ P_i = (B_i, \overline{x_i}, u_i, n_i, c_i)$	$\mathrm{Ind}(M) = \{j \mid \exists\, \alpha, \xi, \bar{x} : (j,\alpha,\xi,\bar{x}) \in M\}$
$M_1 = M \cup \{(j,\alpha,\xi,\texttt{Blocked}(P_j,\alpha))\}$	$M_2 = M - \{(j,\alpha,\xi,\bar{x})\}$
$B(a) = \{k \mid 1 \leq k \leq n \ \wedge \ a \in A_k\}$	$\psi = \max\{n_k(a,\overline{x_k}) \mid k \in B(a)\}$
$M_3 = M \cup \{(k,a,\psi,\texttt{Blocked}(P_k,a)) \mid k \in B(a)\}$	$M_4 = M - \{(k,a,\psi,\overline{x_k}) \mid k \in B(a)\}$

Fig. 2. Operational Semantics for Systems.

The set M will store information about which processes are performing actions, that is, they started an action but they did not finish it. Any element $(i,\alpha,\xi,\bar{y}) \in M$ represents that the i-th process is currently performing α, with respect to the random variable ξ, and it is using the resources given by $\bar{y}$. We will always assume that M is initially empty. In Figure 2 we present the operational semantics for systems. The first rule indicates a exchange of resources. The predicate $\texttt{valid}(S,\mathcal{E})$ controls that processes are not giving resources that they do not own or that they are currently using/producing/consuming. The predicate $\texttt{allowed}(S,\mathcal{E})$ detects whether the exchange is correct with respect to the chosen policy. We will formally define these two predicates later. As a consequence of the exchange, P_i gives to P_j the quantities of resources indicated by $\mathcal{E}_{ij}$, while P_i receives from P_j the quantities given by $\mathcal{E}_{ji}$. We will denote by $\rightsquigarrow^*$ the reflexive and transitive closure of $\rightsquigarrow$. Once we have a situation where no exchange can be performed (that is, $S \not\rightsquigarrow$) the processes will perform *usual* transitions. Let us remark that $S \not\rightsquigarrow$ indicates that the resources have been (somehow) well distributed. The second rule indicates that a process may start the performance of a non-synchronizing action if it is not currently performing another action (i.e. $j \notin \mathrm{Ind}(M)$). In this case, we need to include information about this performance in the set M (see the definition of M_1 at the bottom of Figure 2). In particular, we need to indicate which resources will be *blocked* while performing that action (we will formally introduce this concept after this explanation). The fourth rule is similar but considering synchronizing actions. Let us remark that processes may perform the same action at different speeds (in particular, depending on the resources that they own). We take the slowest, that is, we consider a random variable distributed as the maximum of the corresponding random variables (see the definition of ψ at the bottom of Figure 2). The remaining rules deal with termination of actions. The third rule says that if a process P_j may terminate an action (that is, a tuple with index j belongs to M) and no other process may start an action immediately (i.e. $S \not\xrightarrow{\mathtt{Act}^+}$) then the system can terminate this action. Let us note that $S \not\xrightarrow{\mathtt{Act}^+}$ holds iff $\mathrm{Ind}(M) \cup \{j \mid P_j \not\longrightarrow\} = \{1, \ldots, n\}$. In this case, the corresponding tuple is removed from M. A similar situation appears in the last rule.

Next we define the remaining predicates. If a process is performing an action then there will be resources that it cannot exchange. These resources are those created/consumed during the performance of the action and the ones that the process is using (those adding utility for that particular action). So, these resources will be *blocked* for possible exchanges.

Definition 7. Let $P = (B, \overline{x}, u, n, c)$ be a process. We say that P *depends* on the resource j to perform an action $\alpha \in \mathtt{ACT}$, denoted by $\mathtt{Depends}(P, j, \alpha)$, if there exist $\overline{x'}, \overline{x''} \leq \bar{x}$ such that $\overline{x''} = (x'_1, x'_2, \ldots, x'_j + \epsilon, \ldots, x'_m)$, for some $\epsilon > 0$, and $u(\{\alpha\}, \overline{x'}) < u(\{\alpha\}, \overline{x''})$.

The tuple of *blocked* resources for P while performing and action $\alpha \in \mathtt{ACT}$, denoted by $\mathtt{Blocked}(P, \alpha)$, is defined as $c(\alpha, \overline{x}) - \bar{x} + \bar{y}$, where for all $1 \leq j \leq m$ we have $y_j = x_j$ if $\mathtt{Depends}(P, j, \alpha)$ and $y_j = 0$ otherwise. □

Let us note that P will perform exchanges by taking into account that it will own (after performing α) the resources given by $c(\alpha, \overline{x})$. This is so because the operational rule for processes *updates* the set of owned resources. However, the net creation/consumption of resources, that is $c(\alpha, \overline{x}) - \bar{x}$, cannot be used in these exchanges. In addition, the resources that the process is using for performing α are also excluded. All these resources are *liberated* when the action is finished (by removing the corresponding information from M). Let us also remark that a process may receive (after future exchanges, before completing the performance of α) new quantities of the blocked resources. Nevertheless, these resources will neither improve the performance of the action being performed nor will be added to the set of blocked resources.

The $\mathtt{valid}(S, \mathcal{E})$ predicate controls that a exchange is possible. It holds if the processes do not give resources that they do not own and the diagonal of the matrix is filled with $\bar{0}$. In addition, blocked resources (that is, those included in the set M) cannot be exchanged.

Definition 8. Let $S = {}_M\|_n^{A_i} P_i$ be a system, where for all $1 \leq i \leq n$ we have $P_i = (B_i, \overline{x_i}, u_i, n_i, c_i)$. We say that $\mathcal{E} \in (\mathbb{R}_+^m)^{n*n}$ is a *valid exchange matrix for* S, denoted by $\texttt{valid}(S, \mathcal{E})$, if for all $1 \leq i \leq n$ we have $\mathcal{E}_{ii} = \bar{0}$, and $\sum_j \mathcal{E}_{ij} \leq \overline{x_i} - \overline{x}$ if there exists $\alpha, \xi : (i, \alpha, \xi, \overline{x}) \in M$, and $\sum_j \mathcal{E}_{ij} \leq \overline{x_i}$ otherwise. □

The $\texttt{allowed}(S, \mathcal{E})$ predicate detects whether an exchange conforms the chosen policy. In [18] two different policies were introduced, but others could be defined.[1] In this paper we will only consider the *preserving utility policy*. Under this assumption, exchanges are allowed only if, after the exchange, at least one process improves and no process gets worse. Intuitively, processes are the owners of the resources and they will not give them up if they do not receive a *compensation*.

Definition 9. For all $1 \leq i \leq n$ let $P_i = (B_i, \overline{x_i}, u_i, n_i, c_i)$, let $S = {}_M\|_n^{A_i} P_i$ be a system, and $\mathcal{E} \in (\mathbb{R}_+^m)^{n \times n}$ be such that $\texttt{valid}(S, \mathcal{E})$. The $\texttt{allowed}(S, \mathcal{E})$ predicate holds if for all $1 \leq i \leq n$ we have $u_i(\texttt{Imm}(B_i), \overline{x_i}) \leq u_i(\texttt{Imm}(B_i), \overline{x_i} - \sum_j \mathcal{E}_{ij} + \sum_j \mathcal{E}_{ji})$ and there exists $1 \leq k \leq n$ such that we have $u_k(\texttt{Imm}(B_k), \overline{x_k}) < u_k(\texttt{Imm}(B_k), \overline{x_k} - \sum_j \mathcal{E}_{kj} + \sum_j \mathcal{E}_{jk})$. □

3 Examples

In this section we will show how SPAMR can be used to specify and analyze concurrent systems where both resources and actions taking time to be performed play an important role. First, we present a classic and simple example: The readers/writers problem. Next, we give a more elaborated example where three researchers share some hardware in a laboratory.

During the rest of the section we assume that an undefined value of the utility, necessity, or consumption functions is set to an arbitrary value. Actually, these cases will not be possible because they will correspond to an action (or a set of actions for the utility function) not reachable by the corresponding processes. For the sake of simplicity in the presentation, we will use probability distribution functions instead of random variables. We will consider that $U(t_1, t_2)$ denotes a random variable uniformly distributed on the interval $[t_1, t_2]$; $E(\lambda)$ denotes a random variable exponentially distributed with parameter λ; $\delta(t)$ denotes a Dirac distribution in t. Let us remember that a Dirac distribution models deterministic time, that is, $\delta(t)$ indicates a delay of t time units.

3.1 The Readers and Writers Problem

We suppose n readers and m writers which may access a file. Any number of readers may simultaneously read from the file, but when a writer holds access to

[1] The choice of a *good* policy is not a trivial task. Actually, it is impossible to choose a *perfect* policy because this problem is related with the social welfare aggregator problem. Arrow's *impossibility theorem* [2] shows that there does not exist such an aggregator fulfilling a certain set of *desirable* properties.

the file, neither readers nor other writers are allowed to access it. The behaviors for readers and writers are given by:

$$\begin{aligned} \mathit{Reader} &:= \mathit{read} \,;\, \mathit{other\_tasks} \,;\, \mathit{Reader} \\ \mathit{Writer} &:= \mathit{write} \,;\, \mathit{other\_tasks} \,;\, \mathit{Writer} \end{aligned}$$

We consider that there is only one resource, the *access* to the file, and that there are n units of it. We initially assign a unit of the resource to each reader. For any process, its utility function is defined as:

$$\begin{aligned} u(\{\mathit{read}\}, \bar{x}) &= \begin{cases} K_1 & \text{if } \mathtt{trunc}(x) \geq 1 \\ 0 & \text{otherwise} \end{cases} \\ u(\{\mathit{write}\}, \bar{x}) &= \begin{cases} K_2 & \text{if } x = n \\ 0 & \text{otherwise} \end{cases} \\ u(\{\mathit{other\_tasks}\}, \bar{x}) &= K_3 \end{aligned}$$

where $K_1, K_2 > 0$, and $K_3 \geq 0$. Let us comment on the way the access of a writer forbids the access of any reader or any other writer. For a writer to access the file, it must own all the *access* resources of the system. Also note that if a reader desires to read, it needs at least one unit of the resource; additional units do not increase utility. Regarding necessity functions, they can be defined as follows:

$$\begin{aligned} n(\mathit{read}, \bar{x}) &= \begin{cases} U(K_4, K_5) & \text{if } \mathtt{trunc}(x) \geq 1 \\ \xi_\infty & \text{otherwise} \end{cases} \\ n(\mathit{write}, \bar{x}) &= \begin{cases} U(K_6, K_7) & \text{if } x = n \\ \xi_\infty & \text{otherwise} \end{cases} \\ n(\mathit{other\_tasks}, \bar{x}) &= E(K_8) \end{aligned}$$

where $0 \leq K_4 < K_5$, $0 \leq K_6 < K_7$, and $K_8 > 0$. Note that resources are neither consumed nor created. So, $c(\alpha, \bar{x}) = \bar{x}$. Finally, the system is:

$$\mathit{Readers\_Writers} = {}_{\emptyset}\|_{n+m}^{A_i}\, P_i$$

where, for all $1 \leq i \leq n$ we have $P_i = (\mathit{Reader}, 1, u, n, c)$, and for all $1 \leq i \leq m$ we have $P_{n+i} = (\mathit{Writer}, 0, u, n, c)$. Besides, for all $1 \leq i \leq n + m$ we have $A_i = \emptyset$. The following result shows that this system cannot get deadlocked. Let us remark that the definition of the utility functions plays an important role in this absence of deadlocks. The proof follows from the fact that if a local equilibrium is reached then there exists (at least) a process such that either it is willing to perform *other tasks* (no resources are needed to perform this action) or it has utility greater than zero. Let us remark that no process gets utility by having less number of accesses that the ones it needs. In both cases, this process will be able to perform its corresponding action.

Lemma 1. (*Absence of Deadlocks for Readers_Writers*). Let us consider a system S such that

$$\mathit{Readers\_Writers} \rightsquigarrow^* S_1 \xrightarrow{(\alpha_1,\xi_1)_{B_1}} S_1' \rightsquigarrow^* S_2 \cdots \xrightarrow{(\alpha_k,\xi_k)_{B_k}} S_n' \rightsquigarrow^* S$$

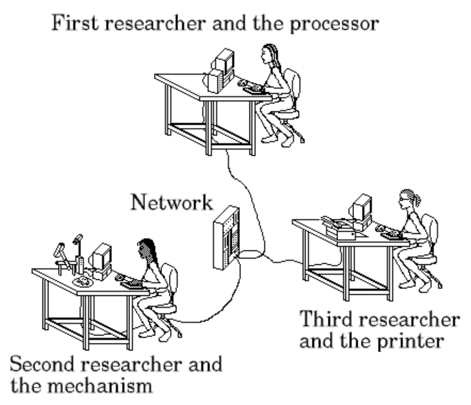
First researcher and the processor
Network
Third researcher and the printer
Second researcher and the mechanism

Fig. 3. Three Researchers in a Laboratory.

where for all $1 \leq i \leq k$ we have $\alpha_i \in \texttt{ACT}^- \cup \texttt{ACT}^+$ and $B_i \subseteq \{1, \ldots, n+m\}$. If S is not a local equilibrium then there exist $S', \mathcal{E}$ such that $S \overset{\mathcal{E}}{\rightsquigarrow} S'$; otherwise, there exist S', α, ξ, B such that $S \xrightarrow{(\alpha,\xi)_B} S'$. $\square$

3.2 The Three Researchers

We present a more complex example that shows most of the characteristics of the language. We consider the situation depicted in Figure 3. We have a laboratory where three (female) researchers, that we call R_1, R_2, and R_3, are making some experiments. Each researcher is placed in front of a computer. One of the computers has a very powerful *processor* that allows to make complex computations. The second computer has a *mechanism* that allows to make some experiments to confirm the analysis given by the previous computations. Finally, the third computer has a *printer*. Each researcher can access *directly* the resource (processor, mechanism or printer) placed in her computer; the other resources are accessed remotely. So, in this example we have four resources: processor, mechanism, printer and network.

The behavior of a researcher is defined as follows. First, she thinks and discusses with the other researchers via electronic chat until an idea comes out.

$$\mathit{Researcher}_i := \mathit{brain\_storming} \; ; \mathit{Computing}_i$$

Afterwards, she tries to access the first computer. If she is R_1 and she has enough *quantity* of the processor, she will be able to perform the computations on her computer. Afterwards, she will analyze the results. If the results confirm her hypothesis, she will try to perform an experiment; otherwise, she will start to think on another idea. The process is similar for the second and third researchers. The only difference is that they must send her data through the channel, then the computations will be done, and then they will receive the results through the channel.

$$Computing_1 := comp_1 \; ; analize\_comp_1 \; ; \begin{pmatrix} good \; ; Experimenting_1 \\ + \\ bad \; ; Researcher_1 \end{pmatrix}$$

$$\underset{[i \in \{2,3\}]}{Computing_i} := \begin{array}{l} send\_data\_comp_i \; ; comp_i ; \\ rec\_data\_comp_i \; ; analize\_comp_i \; ; \end{array} \begin{pmatrix} good \; ; Experimenting_i \\ + \\ bad \; ; Researcher_i \end{pmatrix}$$

In the case of the experimenting process, R_2 does not need the channel to perform the experiment, while the other two researchers need it. Nevertheless, this fact will not be reflected in the specification of the behaviors, but in the utility and necessity functions: They will not depend on the amount of the channel that the second researcher has, while they will for the other two researchers. After performing the experiment, if the results confirm their hypothesis then they will print; otherwise, they will discard the idea.

$$\underset{[i \in \{1,2,3\}]}{Experimenting_i} := experiment_i \; ; analyze\_exp_i \; ; \begin{pmatrix} good \; ; Printing_i \\ + \\ bad \; ; Researcher_i \end{pmatrix}$$

Finally, R_3 will print the results, and will start to think on another idea. The other two researchers need to send the data to the printer through the channel. Note that in this case the printer will not send back any results.

$$Printing_3 := print_3 \; ; Researcher_3$$

$$\underset{[i \in \{2,3\}]}{Printing_i} := send\_data\_print_i \; ; print_i \; ; Researcher_i$$

Next we define the rest of the components of the processes. We suppose that each researcher initially owns her local resource, and that they have the same amount of broad-band. Without losing generality, we consider that both the total amount of processor and of broadband are equal to 1. That is, $\overline{x_1} = (1, 0, 0, \frac{1}{3})$,

$\overline{x_2} = (0, 1, 0, \frac{1}{3})$, $\overline{x_3} = (0, 0, 1, \frac{1}{3})$. Regarding utility functions we have:

$$
\begin{array}{ll}
u_i(\{brain\_storming\}, \bar{z}) & = \begin{cases} z_4 \text{ if } z_4 < \frac{1}{3} \\ \frac{1}{3} \text{ otherwise} \end{cases} \\
{\scriptstyle [i\in\{1,2,3\}]} & \\
u_i(\{comp_i\}, \bar{z}) & = C_i \cdot z_1 \\
{\scriptstyle [i\in\{1,2,3\}]} & \\
u_2(\{experiment_2\}, \bar{z}) & = E_2 \cdot \texttt{trunc}(z_2) \\
u_i(\{experiment_i\}, \bar{z}) & = E_i \cdot \texttt{trunc}(z_2) \cdot z_4^2 \\
{\scriptstyle [i\in\{1,3\}]} & \\
u_i(\{print_i\}, \bar{z}) & = P_i \cdot \texttt{trunc}(z_3) \\
{\scriptstyle [i\in\{1,2,3\}]} & \\
u_1(\{send\_data\_print_1\}, \bar{z}) & = z_4 \\
u_2(\{c\}, \bar{z}) & = z_4 \\
{\scriptstyle [c\in\{send\_data\_comp_2, rec\_data\_comp_2, send\_data\_print_2\}]} & \\
u_3(\{c\}, \bar{z}) & = z_4 \\
{\scriptstyle [c\in\{send\_data\_comp_3, rec\_data\_comp_3,\}]} & \\
u_i(A_i, \bar{z}) & = 0 \\
{\scriptstyle [i\in\{1,2,3\} \,\wedge\, A_i\in\{\{good, bad\}, \{analize\_comp_i\}\}]} &
\end{array}
$$

Let us comment on the previous definitions. If the researchers are willing to perform a brainstorm, then they will need the broadband to use the chat application. Due to the three researchers are committed to chat together, a limit in the utility of the broadband must be imposed to avoid any researcher to keep the whole resource. After a new idea comes out, each researcher will be willing to perform a computation. In this case, their utility depends only on the amount of (time) processor that they own. The additional constant (i.e. $C_i > 0$) measures the *propensity* of the researcher i to make computations. A similar situation appears for the case of printing. On the contrary, the utility when trying to perform an experiment does not depend only on the *amount* of the mechanism; in the case of R_1 and R_3, it will also depend on the amount of broadband that they own because they are supposed to interact with the mechanism. Let us note that in these last two cases, researchers get no utility if they do not exclusively own the mechanism or the printer, respectively (this is indicated by using the *truncate* function). In the case of the processor and the network, a fraction of the whole amount reports utility greater than zero. If the researchers try to send data, their utilities depend only on the amount of broadband that they have. Finally, no resources are needed to perform the rest of the actions. In this case, the utility is given by a constant.

The execution time will sometimes depend only on the amount of resources (e.g. $comp_i$) and sometimes will also depend on a constant indicating the skills of the researcher (e.g. $experiment_i$).

$$\begin{aligned}
&\underset{[i\in\{1,2,3\}]}{n_i(comp_i,\bar{z})} &&= \begin{cases} E(C\cdot z_1) \text{ if } z_1>0 \\ \xi_\infty \quad\quad\quad \text{otherwise}\end{cases} \\
&n_2(experiment_2,\bar{z}) &&= \begin{cases} E(E_2) \text{ if } \mathtt{trunc}(z_2)\ge 1 \\ \xi_\infty \quad\quad \text{otherwise}\end{cases} \\
&\underset{[i\in\{1,3\}]}{n_i(experiment_i,\bar{z})} &&= \begin{cases} E(E_i\cdot z_4^2) \text{ if } \mathtt{trunc}(z_2)\ge 1 \ \wedge\ z_4>0 \\ \xi_\infty \quad\quad\quad \text{otherwise}\end{cases} \\
&\underset{[i\in\{1,2,3\}]}{n_i(print_i,\bar{z})} &&= \begin{cases} U(P_1,P_2) \text{ if } \mathtt{trunc}(z_3)\ge 1 \\ \xi_\infty \quad\quad\quad \text{otherwise}\end{cases} \\
&\underset{[c\in\{send\_data\_print_1\}]}{n_1(c,\bar{z})} &&= \xi \\
&\underset{[c\in\{send\_data\_comp_2,rec\_data\_comp_2,send\_data\_print_2\}]}{n_2(c,\bar{z})} &&= \xi \\
&\underset{[c\in\{send\_data\_comp_3,rec\_data\_comp_3\}]}{n_3(c,\bar{z})} &&= \xi \\
&\underset{[i\in\{1,2,3\}]}{n_i(good,\bar{z})} &&= n_i(bad,\bar{z}) = \delta(BG) \\
&\underset{[i\in\{1,2,3\}]}{n_i(brain\_storming,\bar{z})} &&= \begin{cases} E(T_i\cdot z_4) \text{ if } z_4>0 \\ \xi_\infty \quad\quad\quad \text{otherwise}\end{cases} \\
&\underset{[i\in\{1,2,3\}]}{n_i(analize\_comp_i,\bar{z})} &&= E(AC_i) \\
&\underset{[i\in\{1,2,3\}]}{n_i(analyze\_exp_i,\bar{z})} &&= E(AE_i)
\end{aligned}$$

where

$$\xi = \begin{cases} U(\frac{K_1}{z_4},\frac{K_2}{z_4}) \text{ if } z_4>0 \\ \xi_\infty \quad\quad\quad \text{otherwise}\end{cases}$$

Let us note that the delay associated with choosing whether a result is *good* or *bad* is deterministic as the (random) time for the analysis is consumed by the previous actions ($analize\_comp_i$ and $analyze\_exp_i$). In this system, resources are neither created nor consumed. So, $c_i(\alpha,\bar{z})=\bar{z}$. The laboratory is defined as:

$$Laboratory = {}_{\emptyset}\|_{3}^{A_i}\ P_i$$

where for all $1\le i\le 3$ we have $P_i=(B_i,\overline{x_i},u_i,n_i,c_i)$ and $A_i=\{brain\_storming\}$.

The proof of deadlock-freedom is a little bit more involved in this case because there are more possibilities. Nevertheless, the technique is exactly the same as in Lemma 1. First, let us remark that all the resources where the `truncate` function is applied have one full unit of them. After reaching a local equilibrium S, if all the processes have zero utility then we get a contradiction; otherwise, because of the relation between utility and necessity functions, a process will be able to perform one of its immediate actions.

Lemma 2. (*Absence of Deadlocks for Laboratory*). Let us consider a system S such that

$$\mathit{Laboratory} \rightsquigarrow^* S_1 \xrightarrow{(\alpha_1,\xi_1)_{B_1}} S'_1 \rightsquigarrow^* S_2 \cdots \xrightarrow{(\alpha_n,\xi_n)_{B_n}} S'_n \rightsquigarrow^* S$$

where for all $1 \leq i \leq n$ we have $\alpha_i \in \mathtt{ACT}^- \cup \mathtt{ACT}^+$ and $B_i \subseteq \{1,2,3\}$. If S is not a local equilibrium then there exist $S', \mathcal{E}$ such that $S \overset{\mathcal{E}}{\rightsquigarrow} S'$; otherwise, there exist S', α, ξ, B such that $S \xrightarrow{(\alpha,\xi)_B} S'$. □

Besides, it is guaranteed that the system is starvation-free. This is so because the three researchers are forced to synchronize when brainstorming, so that their progress can help their partners when looking for new ideas. This fact disallows a researcher to advance forever in her research while the other ones are stopped.

4 Conclusions and Future Work

In this paper we have introduced an stochastic version of `PAMR` where random variables have been associated with the performance of actions. We have defined an operational semantics for the new language. Since we do not restrict the probability distribution functions associated with random variables, we need to use complex technicalities to define this semantics. In order to show the usefulness of our language, we have given two examples showing most of its features.

As future work we plan to define semantic frameworks for our language. We have already developed a notion of (strong) bisimulation and two trace-based semantics. The difference between the alternative trace semantics comes from the point of observation: Whether we check only *visible* events or also *internal* ones. In addition, we plan to consider a testing semantics for our language in the line of [13] and a stochastic weak bisimulation following [14].

References

1. M. Ajmone Marsan, A. Bianco, L. Ciminiera, R. Sisto, and A. Valenzano. A LOTOS extension for the performance analysis of distributed systems. *IEEE/ACM Transactions on Networking*, 2(2):151–165, 1994.
2. K.J. Arrow. *Social Choice and Individual Values*. Wiley, 2nd edition, 1963.
3. M. Bernardo and R. Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202(1-2):1–54, 1998.
4. M. Bravetti and P.R. D'Argenio. Tutte le algebre insieme: Concepts, discussions and relations of stochastic process algebras with general distributions. In *Validation of Stochastic Systems, LNCS 2925*, pages 44–88. Springer, 2004.
5. M. Bravetti and R. Gorrieri. The theory of interactive generalized semi-Markov processes. *Theoretical Computer Science*, 282(1):5–32, 2002.
6. P.R. D'Argenio and J.-P. Katoen. A theory of stochastic systems part I: Stochastic automata. *Information and Computation*, 203(1):1–38, 2005.

7. P.R. D'Argenio and J.-P. Katoen. A theory of stochastic systems part II: Process algebra. *Information and Computation*, 203(1):39–74, 2005.
8. N. Götz, U. Herzog, and M. Rettelbach. Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras. In *16th Int. Symp. on Computer Performance Modelling, Measurement and Evaluation, PERFORMANCE'93, LNCS 729*, pages 121–146. Springer, 1993.
9. P.G. Harrison and B. Strulo. SPADES – a process algebra for discrete event simulation. *Journal of Logic Computation*, 10(1):3–42, 2000.
10. H. Hermanns, U. Herzog, and J.-P. Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 274(1-2):43–87, 2002.
11. J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.
12. G.G. Infante López, H. Hermanns, and J.-P. Katoen. Beyond memoryless distributions: Model checking semi-Markov chains. In *Joint Int. Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification, PAPM-PROBMIV'01, LNCS 2165*, pages 57–70. Springer, 2001.
13. N. López and M. Núñez. A testing theory for generally distributed stochastic processes. In *12th Int. Conf. on Concurrency Theory, CONCUR'01, LNCS 2154*, pages 321–335. Springer, 2001.
14. N. López and M. Núñez. Weak stochastic bisimulation for non-markovian processes. In *2nd Int. Conf. on Theoretical Aspects of Computing, ICTAC'05, LNCS 3722*, pages 454–468. Springer, 2005.
15. N. López, M. Núñez, and F. Rubio. An integrated framework for the analysis of asynchronous communicating stochastic processes. *Formal Aspects of Computing*, 16(3):238–262, 2004.
16. N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *6th ACM Symp. on Principles of Distributed Computing, PODC'87*, pages 137–151. ACM Press, 1987.
17. M.G. Merayo, M. Núñez, and I. Rodríguez. Formal specification of multi-agent systems by using EUSMs. In *2nd IPM Int. Symposium on Fundamentals of Software Engineering, FSEN'07, LNCS (to appear)*, 2007.
18. M. Núñez and I. Rodríguez. `PAMR`: A process algebra for the management of resources in concurrent systems. In *21st IFIP WG 6.1 Int. Conf. on Formal Techniques for Networked and Distributed Systems, FORTE'01*, pages 169–185. Kluwer Academic Publishers, 2001.
19. M. Núñez and I. Rodríguez. Encoding `PAMR` into (timed) `EFSMs`. In *22nd IFIP WG 6.1 Int. Conf. on Formal Techniques for Networked and Distributed Systems, FORTE'02, LNCS 2529*, pages 1–16. Springer, 2002.
20. M. Núñez, I. Rodríguez, and F. Rubio. Formal specification of multi-agent e-barter systems. *Science of Computer Programming*, 57(2):187–216, 2005.
21. M. Núñez, I. Rodríguez, and F. Rubio. Specification and testing of autonomous agents in e-commerce systems. *Software Testing, Verification and Reliability*, 15(4):211–233, 2005.
22. K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of stochastic games. In *17th Int. Conf. on Computer Aided Verification, CAV'05, LNCS 3576*, pages 266–280. Springer, 2005.