Towards Performance Evaluation with General Distributions in Process Algebras*

Mario Bravetti, Marco Bernardo and Roberto Gorrieri

Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
E-mail: {bravetti, bernardo, gorrieri}@cs.unibo.it

Abstract. We present a process algebra for the performance modeling and evaluation of concurrent systems whose activity durations are expressed through general probability distributions. We first determine the class of generalized semi-Markov processes (GSMPs) as being the class of stochastic processes on which we must rely for performance evaluation to be possible. Then we argue that in this context the right semantics for algebraic terms is a variant of the ST semantics which accounts for both functional and performance aspects. The GSMP based process algebra we propose is introduced together with its formal semantics, an example of performance evaluation, and a notion of probabilistic bisimulation based equivalence accounting for action durations which is shown to be a congruence.

1 Introduction

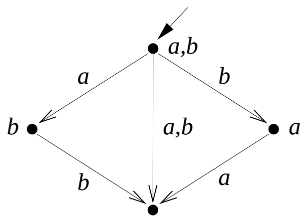
The aim of this paper is to develop a core calculus suitable for the performance evaluation of concurrent systems whose activity durations are expressed through general probability distributions. In order for performance evaluation to be possible, it is necessary to resort to a class of stochastic processes known and studied in the literature so that the solution techniques already developed for such a class can be exploited. In the past five years, a lot of work has been done in order to enrich the expressiveness of existing process algebras with the representation of exponentially distributed durations (see e.g. [13, 17, 4, 5, 3]). These algebras, called Markovian process algebras, have the advantage to rely on the class of Markov chains, which can be easily analyzed in order to obtain performance measures. The price to pay is a strong limitation in the expressiveness of these algebras, since not even deterministic durations can be represented. Some previous efforts have been made in order to develop calculi for general distributions [14, 1, 9, 23, 11], but in these approaches it is not shown on what underlying well known performance model the calculus is based.

The stochastic processes mainly studied in the literature for performance evaluation purposes are in increasing order of expressivity: (continuous time)

* The full version of the paper is available at ftp://ftp.cs.unibo.it/pub/techreports/98-06.ps.gz . This research has been partially funded by Progetto Strategico CNR "Modelli e Metodi per la Matematica e l'Ingegneria".

Markov chains (MCs), semi-Markov processes (SMPs), and generalized semi-Markov processes (GSMPs). The difference among them lies in the set of instants of process life which satisfy the Markov property, i.e. those instants such that the future behavior of the stochastic process depends only on the current state of the process and not on its past behavior. For MCs the Markov property holds in every instant of process life, for SMPs it holds only in the instants of state change, and for GSMPs it never holds, but can be retrieved through a different representation of process states (each state is turned into a continuous infinity of states) by the standard technique of [10].

Since MCs can represent only activities with an exponentially distributed duration (only these distribution have the required memoryless property), the only candidates for representing systems with generally distributed durations are SMPs and GSMPs, and we now show that GSMPs are actually needed for our purposes. Consider the example of two activities a and b executed in parallel, the former with a deterministic duration 5 and the latter with a geometrically distributed duration with parameter 0.5. This situation can be represented in this way:



$$PDF(a) = Det(5), PDF(b) = Geom(.5)$$

Each state is labeled with the set of activities which are in execution during the period of time the system sojourns in the state. In the beginning both activities are in contemporaneous execution and the system sojourns in the first state until one activity (or both contemporaneously) terminates. When this happens the system performs the transition labeled with the terminated action(s). Suppose that a terminates before b and the system reaches the state labeled with b. In this state the activity b continues its execution until it terminates. As a consequence the sojourn time of the system in the state labeled with b (which is given by the residual distribution of activity b) is not determined simply by the fact that the system is in this state, but depends on the time b has already spent in execution in the first state. Hence the process is Markovian not even in the instant when this state is entered.

This example shows that even the simple case of two parallel activities with generally distributed durations cannot be represented by an SMP. The process of the example is, instead, a GSMP. This can be seen as follows. If we imagine to give a different representation of the process where we replace the state labeled with b with infinitely many states each denoting a different spent lifetime for the activity b, we can retrieve the Markov property. The sojourn time in each of the newly derived states would then be determined by the state itself (it would be given by the distribution of activity b conditioned on a particular value for the spent lifetime) and not by the previous behavior of the process.

GSMPs have been introduced by Matthes [21]. In our approach we consider a general definition for GSMPs which leaves out the constraint of [21] that disallows actions to start and terminate at the same instant, imposed in order to analyze such processes. This because we do not want to restrict a priori the set of systems that can be modeled by the algebra: we instead treat performance analysis related problems at the level of the performance model.

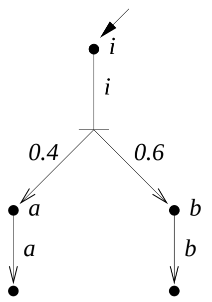
A GSMP is a stochastic process defined on a set of states $\{s \mid s \in \mathcal{S}\}$. In each state s there is a set of active elements $ElSt(s)$ taken from a set El. Each element of El has an associated generally distributed lifetime. Whenever in a state s a set of advancing elements ter contemporaneously terminate, the process moves to the state $s' \in \mathcal{S}$ with a given probability $P(s, ter, s')$. The transitions of the GSMP of the previous example have all default probability 1. In the case there are many transitions that start from a common state s and that refer to a common set of terminating activities ter (see next example about choice) we depict a single line starting from s and labeled with ter that branches (in the point marked with a little bar) in many arrows, each labeled with the probability associated with the corresponding transition.

By relying on GSMPs, we can sometimes make exact analysis by employing the notion of *insensitivity* introduced by Matthes in [21]. The distributions occurring in an insensitive GSMP can be replaced with exponential distributions with the same mean, so the problem reduces to solve a MC. For GSMPs which do not respect the constraints considered by Matthes, we can in some cases obtain insensitivity by employing the structural simplification techniques presented in [15]. These techniques can be combined with approximation techniques such as i.e. replacing some general distributions with phase-type distributions in order to obtain the result of insensitivity for the others. In the worst case, with this last approach all the distributions are replaced by phase-type ones and mathematical analysis is still possible.

Once recognized that we have to produce a GSMP in order to do mathematical analysis when dealing with general distributions, the problem is how to develop a calculus suitable for generating GSMPs. The system in the previous example could be represented by term $a.\underline{0} \,||\, b.\underline{0}$. As far as Markovian process algebras are concerned, the memoryless property of exponential distributions allows algebraic terms to be given a simple interleaving semantics. For example, thanks to this property, it is equivalent to consider the element b as starting in the first state or in the state labeled with b only. In this way the actions can be considered as being executed atomically in the state where they terminate and we have that the expansion law $a.\underline{0} \,||\, b.\underline{0} = a.b.\underline{0} + b.a.\underline{0}$ still holds (since the exponential distribution is continuous, a and b cannot terminate at the same instant). The price to pay for using an interleaving semantics is an unnatural semantics for the alternative composition operator "$\_ + \_$" which must be resolved with the *race policy*. This means e.g. that in term $a.\underline{0} + b.\underline{0}$ the two actions a and b are considered as being in parallel execution and the one that terminates first resolves the choice. However in the context of exponential distributions this policy can be considered as acceptable since it causes the semantic models of terms to be adherent to MCs where choices are expressed in the same way.

When general distributions are considered we can no more rely on the memoryless property. In order to describe correctly the parallel execution of actions, at the semantic level we have to represent actions that start in a certain state, evolve through several states, and terminate in another state (in the previous example both a and b start in the first state and may terminate in another state). Therefore, an action can no more be considered as being executed atomically in a single transition, but is characterized in the semantics by the two events of *action start* and *action termination*.

Some previous efforts [14, 1, 23] have been made in order to try to adapt the interleaving semantics to deal with general distributions by considering actions as starting in the first state they become enabled. In order for the starting point of actions to be observable the interleaving semantics had to be enriched with additional information: in [14] transitions are enriched with *start references*, in [23] transitions are enriched with information about causality relations among actions, and in [1] actions must be differently identified with indexes before the semantic rules are applied. As a matter of fact these semantics are not actually interleaving since the expansion law $a.\underline{0} \,||\, b.\underline{0} = a.b.\underline{0} + b.a.\underline{0}$ is no longer valid. Moreover the unnatural semantics for operator "_ + _" can no longer be justified by the structure of the underlying stochastic process, since contrary to Markov chains in GSMPs choices and durations are expressed in a separated way. For example an intuitive semantics of $a.\underline{0} + b.\underline{0}$ should generate a GSMP like the following one:



$$PDF(a) = Det(5), PDF(b) = Geom(.5), PDF(i) = Det(0)$$

where the element i is an auxiliary dummy element with zero duration which causes the process to leave immediately the first state. [2] This GSMP first performs a probabilistic choice between a and b according to probabilities 0.4 and 0.6 and then executes the selected action. This corresponds to using the *preselection policy* instead of the race policy to solve choices.

[2] An equivalent representation of this GSMP which allows to leave out the element i can be obtained by associating directly with the states labeled with a and b the probability of being the initial state (see [7]).

Therefore, when general distributions are considered and actions are no longer atomic, there is no advantage in trying to keep the semantics in an interleaving atomic form. In fact the need of representing general distributions does not introduce any great new problem. It is simply sufficient that a standard semantics is employed where actions are represented at the semantic level as a combination of *action start* and *action termination* and the termination of an action is uniquely related to its start. A semantics of this kind is the ST semantics (see e.g. [12, 2, 8]). As we will see in Sect. 2, with an ST semantics the preselection policy is naturally obtained by associating the choice of an action with the event of action start.

In the light of these observations we note that the two events of *start* and *termination* for timed activities are somehow expressed also in [11], where an algebra is presented which can represent timed activities through "clocks" and instantaneous atomic actions. However in this approach the events of start and termination of a clock are expressed directly within process terms, instead of being naturally represented at the semantic level as in the ST semantics. In fact even if the authors recognize the need of a precise description of clocks for representing general distributions, they still employ an interleaving semantics for actions, thus creating an unnecessary distiction between actions and clocks. It is worth noting that in the paradigm of [11] the description of clocks determines not only the performance of the system but also its functional behavior. In our approach, instead, specifying performance merely consists of quantifying the durations of actions.

As a consequence of the previous observations and discussions, in this paper we propose *Generalized Semi-Markovian Process Algebra (GSMPA)*, a process algebra defined through a variant of the ST semantics which accounts for both functional and performance aspects and allows to generate performance models in the form of GSMPs. The remainder of the paper is organized as follows. In Sect. 2 we present the semantic model of GSMPA. In Sect. 3 we show how to specify a system. In Sect. 4 we formally define the semantics for GSMPA. In Sect. 5 we present an example of performance evaluation. In Sect. 6 we introduce a probabilistic bisimulation based notion of equivalence for GSMPA which we show to be a congruence. Finally, in Sect. 7 we report some concluding remarks.

2 A Semantic Model for Durational Actions

The GSMPA semantics is defined by following the variant of the ST semantics of [2]. The aim of obtaining semantic models which can be easily transformed into GSMPs determines the choice of a paradigm which relates the termination of actions with their start through a technique of *action identification* like that of [2] rather than e.g. through a mechanism involving pointers like that of [8]. By identifying each action appearing in a process term with a different representation we can easily map at the semantic level each action to a different element of a GSMP. The resulting semantic models would then describe the evolution of actions just as for the elements in a GSMP. In [2] the actions of a term are distinguished by labeling each of them with a different index before the operational semantic rules are applied. We employ a different approach for identifying actions which, among other things, allow for the operational semantics to be applied directly on process terms.

We first point out that, for the purpose of defining an ST semantics, it is not necessary to identify *every* action of a term as a different action. For example two consecutive actions of a term executed by the same sequential process cannot be concurrently executed, but are causally related. Thus the two actions cannot overlap during their execution and once an action has started it must terminate before the other one can start. Therefore even if the two actions are not distinguished by the identification mechanism the event of an action termination is still uniquely related to the event of its start (there is no ambiguity about which action is terminating).

As a consequence in our approach we choose to identify actions through their *location*, i.e. the position with respect to the parallel composition operators of the term, thus following the technique introduced in [6]. Since as in [4, 5, 3] we adopt the CSP [18] synchronization policy for the parallel composition operator, we have that the actions of the whole system (hereafter called *top level actions*) can be formed by the synchronization of several *local actions* with the same type, each executed by a single sequential process. We express the location of a top level action by composing the locations of the related local actions. The set Loc of *action locations* is generated by the syntax: $loc ::= \bullet \mid \swarrow loc \mid \searrow loc \mid \langle loc | loc \rangle$. [3] We define the set AId of *(top level) action identifiers* by $AId = AType \times Loc$ where $AType$ is the set of action types ranged over by $a, b, c, \ldots$.

Example 1. Consider $(a.E_1 \,||_\emptyset\, b.E_2) \,||_{\{b\}}\, b.E_3$. Then $a_{\swarrow\swarrow}$ is the identifier of the action formed by the single local action identified with a executed by the leftmost sequential process. Moreover $b_{\langle \searrow | \rangle}$ is the identifier of the action formed by the synchronization of the two local actions with type b. ■

Identifying actions through locations is further justified in the framework of performance modeling since it allows the *compositional specification of action durations*, i.e. it allows to specify the system performance by associating duration distributions with local actions and by expressing the synchronization paradigm used for the different action types (see Sect. 3.2 for further details).

Our approach differs from that of [2] also in the way we record in the states of semantic models the event of an action start. Since in CSP the actions obtained by synchronization are observable, we cannot simply label in some way the local action (or the local actions in the syncronization case) within the process term as done in [2]. Consider for example the term $(a.\underline{0} \,||_\emptyset\, a.\underline{0}) \,||_{\{a\}} (a.\underline{0} \,||_\emptyset\, a.\underline{0})$. In this term two actions may start execution before any action terminate. By labeling local actions inside the term we could not distinguish e.g. if the two actions that have started are $a_{\langle \swarrow | \swarrow \rangle}$ and $a_{\langle \searrow | \searrow \rangle}$, or $a_{\langle \swarrow | \searrow \rangle}$ and $a_{\langle \searrow | \swarrow \rangle}$.

In our approach we keep the information about started actions separated from terms and we represent the system states by pairs $\langle E, Exec \rangle$, where E is a

[3] In the following we omit the $\bullet$ when writing locations.

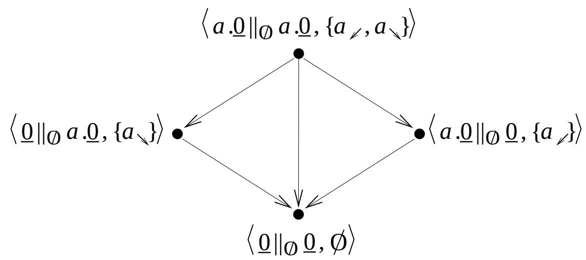
process term and *Exec* is a label expressing the set of the identifiers of *actions in execution*. On the other hand this approach allows to generate semantic models very similar to GSMPs where states are labeled with active elements, and does not require the definition of a particular syntax for state terms.

Moreover, the need of obtaining semantic models close to GSMPs has another effect on the definition of GSMPA semantics. In order to define a GSMP starting from a semantic model we have to describe the system behavior also in the case of contemporaneous termination of actions. This can be done very naturally by defining termination transitions as in the step semantics [22]. For example consider the following GSMPA specification:

$$a.\underline{0} \,\|_{\emptyset}\, a.\underline{0}$$

$$PDF_{\swarrow}(a) = Det(5), PDF_{\searrow}(a) = Geom(.5)$$

where, due to the compositional specification of durations (see Sect. 3.2), we express the local action durations by referring to the locations of the sequential processes they belong to. Once the two actions have started execution the system reaches the state $\langle a.\underline{0} \,\|_{\emptyset}\, a.\underline{0}, \{a_{\swarrow}, a_{\searrow}\}\rangle$. The evolution of this state at the semantic level is depicted below.



$$PDF(a_{\swarrow}) = Det(5), PDF(a_{\searrow}) = Geom(.5)$$

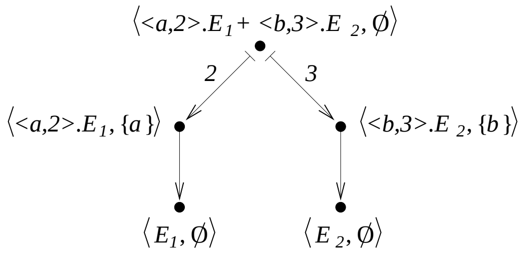
where the transitions presented are termination transitions. In semantic models the termination transitions are not labeled with terminating actions since this would be a redundant information. The actions a transition refers to can easily be derived from the state labels. This semantic model maps directly into the GSMP presented in Sect. 1 where the element a correspond to action $a_{\swarrow}$ and the element b correspond to action $a_{\searrow}$. It is worth noting that the termination of both actions in execution is assumed to be possible irregardless of duration distributions (e.g. if in the example above $PDF_{\searrow}(a) = Det(4)$, only the rightmost computation would be actually possible). This is necessary in order to achieve the congruence property when defining the notion of equivalence as we will see in Sect. 6.

The last enhancement of the ST semantics we have to make in order to allow for performance evaluation concerns the *quantification of choices*. While in classical process algebras choices are solved in a purely nondeterministic way, in GSMPA we need to give a probabilistic quantification of choices in order to obtain semantic models fully specified from the performance point of view. As we have already sketched in Sect. 1, in GSMPA choices are solved through the preselection policy. Each action of a term has an associated *weight* and choices among actions are carried out by giving each of them a probability proportional to its weight. For example, consider the following GSMPA specification:

$$<a, 2>.E_1 + <b, 3>.E_2$$

$$PDF(a) = Det(5), PDF(b) = Geom(.5)$$

At the semantic level the system is represented as follows:



$$PDF(a) = Det(5), PDF(b) = Geom(.5)$$

where each transition $\longmapsto$ represent the event of an action start. Since the choice of an action for execution corresponds to the event of action start, the transitions $\longmapsto$ are labeled with weights and are called *choice transitions.* Like termination transitions, choice transitions are not labeled with the starting action since this information can easily be derived from the state labels. This semantic model has a clear correspondence with the GSMP presented in Sect. 1.

3 Specification of a Concurrent System

The specification of a system is composed of the specification of its behavior (a term of GSMPA) and the specification of the duration of its actions. The separation of the two specifications reflects the same separation at the semantic model level stemming from the structure of GSMPs.

3.1 Specification of System Behavior

Representation of Actions Each *action* is represented as $<a^v_{loc}, w>$ and consists of a *type* a, a *location* loc, a *visibility* v, and a *weight* w.

The type and the location of an action together constitute the *identifier* of the action as described in Sect. 2. The *visibility* v of an action specifies if the action must be executed internally or may synchronize with other actions. In GSMPA we cannot use a distinguished unique action type for invisible actions such as the traditional τ, because in case some actions of a process are made hidden to the environment (by means of the hiding operator "$\_/L$") we must preserve the identifiers of actions (which include the action types) in order to distinguish actions belonging to the internal behavior of the process so that its performance is modeled correctly. As a consequence, visibility reduces to an action attribute. We define the set $AVis$ of *action visibilities*, ranged over by v, by $AVis = \{o, h\}$ where o stands for *observable* (default value) and h stands for *hidden*. Moreover we define the set $AVId$ of *(top level) action identifiers with visibility* by $AVId = AId \times AVis$. The *weight* $w \in \mathbb{R}_+$ of an action is used, as stated in Sect. 2, for choosing actions according to the preselection policy (the default value for the weight is 1).

We define the set $LAct$ of *local actions* by $LAct = AType \times AVis \times \mathbb{R}_+$ and the set Act of *(top-level) actions* by $Act = AVId \times \mathbb{R}_+$.

Syntax of Terms and Informal Semantics of Operators Let $Const$ be a set of *constants* ranged over by $A, B, C, \ldots$, and let $ARFun = \{\varphi : AType \longrightarrow AType\}$ be a set of *action relabeling functions*.

Definition 1. The set $\mathcal{L}$ of *process terms* of GSMPA is generated by the following syntax

$$E ::= \underline{0} \mid <a^v, w>.E \mid E/L \mid E[\varphi] \mid E + E \mid E \|_S E \mid A$$

where $<a^v, w> \in LAct$ and $L, S \subseteq AType$. Set $\mathcal{L}$ will be ranged over by $E, E', E'', \ldots$. We denote by $\mathcal{G}$ the set of closed and guarded terms of $\mathcal{L}$. ■

The meaning of the operators is the standard one (as in CSP/CCS). In particular it is worth noting that: "$\_/L$" turns the visibility of the actions whose type belong to L into h, "$\_[\varphi]$" modifies also the type of hidden actions, "$\_ + \_$" expresses a choice which is resolved according to the the weights of the actions executable by the two terms, and "$\_ \|_S \_$" assignes to an action obtained by synchronization the *product* of the weights of the actions that synchronize [24].

3.2 Specification of Action Durations

The durations of the actions are specified in a compositional way as follows.

We identify sequential processes through locations just as we do for actions. The set PId of *process identifiers* is generated by the syntax: $pid ::= \bullet \mid pid\swarrow \mid pid\searrow$.[4] If we denote by $Dist$ the set of probability distribution functions, the specification of durations includes:

- For every sequential process pid a function $PDF_{pid} : AType \longrightarrow Dist$ that associates with every local action of process pid a duration distribution.
- For any action type $a \in AType$ a function $PDFSynch_a : (Dist \times Dist) \longrightarrow Dist$ that determines the duration of an action derived by the synchronization of two actions of type a. This operator determines the synchronization paradigm used for action type a and can be *arbitrary* (it should be at least commutative and associative).

When specifying the durations of a concurrent system described by a term E, what is actually necessary is to specify the durations of the local actions executable by the sequential processes of E and the synchronization paradigm for the action types belonging to the synchronization sets occurring in E.

[4] Contrary to action locations, process locations are defined through a left recursive syntax since this is convenient for the definition of the function $PDFCalc$ of Sect. 4.2. In the following we omit the $\bullet$ when writing process identifiers so that we express a location in the same way for both actions and processes.

4 Integrated Semantics for GSMPA

In this section we define the integrated semantics for GSMPA. The structure of this section parallels the structure of the previous one in order to take the separate specification of behaviors and durations into account.

4.1 Representation of System Behavior

We now define the formal integrated semantics for GSMPA according to Sect. 3.1 in the form of a labeled transition system (LTS) with labeled states $\langle E, Exec\rangle$ belonging to the set $LabS$ defined as $\mathcal{G} \times AVis$ and with two types of transitions related to the two events of action beginning and action ending.

Choice transitions represent the choice of actions for execution according to the preselection policy and are labeled with the weight w associated with the chosen action. They are represented by the transition relation $\longmapsto$ defined as the least subset of $LabS \times \mathbb{R}_+ \times LabS$ that satisfies the inference rule in the first part of Table 1. This rule determines the transitions leaving a state $\langle E, Exec\rangle$ beginning from the multiset of choice moves of the state $CM(\langle E, Exec\rangle)$. As in [5] we represent the choice moves of a state by a multiset since several moves with the same weight and the same derivative state may be inferred. This multiset is defined by structural induction as the least element of $\mathcal{M}u_{fin}(L_C \times LabS)$ satisfying the rules in the second part of Table 1. [5] A choice move represents the choice of a single action and is a pair composed of the weight of the action and the state reached after the choice. Note that in the definition of $CM(\langle E, Exec\rangle)$ we consider only the sets $Exec$ such that $\langle E, Exec\rangle$ is actually reachable from an initial state $\langle F, \emptyset\rangle$. The transition relation is determined from $CM(\langle E, Exec\rangle)$ through function $melt : \mathcal{M}u_{fin}(L_C \times LabS) \longrightarrow \mathcal{P}_{fin}(L_C \times LabS)$, defined in the third part of Table 1, which merges together the choice moves with the same derivative state by summing their weights. The auxiliary functions $left : \mathcal{P}(AVId) \longrightarrow \mathcal{P}(AVId)$, $right : \mathcal{P}(AVId) \longrightarrow \mathcal{P}(AVId)$, $hide : \mathcal{P}(AVId) \longrightarrow \mathcal{P}(AVId)$, and $relab : \mathcal{P}(AVId) \longrightarrow \mathcal{P}(AVId)$ defined in the fourth part of Table 1.

Termination transitions represent the contemporaneous termination of a set of actions in execution according to the race policy. They are represented by the transition relation $\longrightarrow$ defined as the least subset of $LabS \times LabS$ that satisfies the inference rule in the first part of Table 2. This rule determines a transition leaving a state $\langle E, Exec\rangle$ (that is reachable from an initial state) for each nonempty subset Ter of $Exec$ using the function $TM(\langle E, Exec\rangle) : \mathcal{P}(Exec) - \{\emptyset\} \longrightarrow \mathcal{G}$ that describes the termination moves of the state. Since there is one and only one derivative term for each nonempty subset of $Exec$ we represent the termination moves of a state by a function. This function is defined in the second part of Table 2 by structural induction on states with a nonempty set $Exec$.

[5] We denote by $\mathcal{M}u_{fin}(S)$ the set of finite multiset over S, we use $\{\!|$ and $|\!\}$ as multiset parentheses, and we use $\oplus$ to denote multiset union.

$$\frac{(w, \langle E', Exec' \rangle) \in melt(CM(\langle E, Exec \rangle))}{\langle E, Exec \rangle \overset{w}{\longmapsto} \langle E', Exec' \rangle}$$

$$CM(\langle \underline{0}, \emptyset \rangle) = \emptyset$$

$$CM(\langle {<}a^v, w{>}.E, \emptyset \rangle) = \{\!|\, (w, \langle {<}a^v, w{>}.E, \{a^v\} \rangle) \,|\!\}$$

$$CM(\langle {<}a^v, w{>}.E, \{a^v\} \rangle) = \emptyset$$

$$\begin{aligned} CM(\langle E/L, hide(Exec, L) \rangle) = \{\!|\, & (w, \langle E'/L, hide(Exec \cup \{a^v_{loc}\}, L) \rangle) \mid \\ & (w, \langle E', Exec \cup \{a^v_{loc}\} \rangle) \in CM(\langle E, Exec \rangle) \,|\!\} \end{aligned}$$

$$\begin{aligned} CM(\langle E[\varphi], relab(Exec, \varphi) \rangle) = \{\!|\, & (w, \langle E'[\varphi], relab(Exec \cup \{a^v_{loc}\}, \varphi) \rangle) \mid \\ & (w, \langle E', Exec \cup \{a^v_{loc}\} \rangle) \in CM(\langle E, Exec \rangle) \,|\!\} \end{aligned}$$

$$CM(\langle E_1 + E_2, \emptyset \rangle) = CM(\langle E_1, \emptyset \rangle) \oplus CM(\langle E_2, \emptyset \rangle)$$

$$\begin{aligned} CM(\langle E_1 \parallel_S E_2, Exec \rangle) = \; & \{\!|\, (w, \langle E'_1 \parallel_S E_2, Exec \cup \{a^v_{\swarrow\, loc}\} \rangle) \mid (a \notin S \vee v = h) \wedge \\ & \quad (w, \langle E'_1, left(Exec) \cup \{a^v_{loc}\} \rangle) \in CM(\langle E_1, left(Exec) \rangle) \,|\!\} \oplus \\ & \{\!|\, (w, \langle E_1 \parallel_S E'_2, Exec \cup \{a^v_{\searrow\, loc}\} \rangle) \mid (a \notin S \vee v = h) \wedge \\ & \quad (w, \langle E'_2, right(Exec) \cup \{a^v_{loc}\} \rangle) \in CM(\langle E_2, right(Exec) \rangle) \,|\!\} \oplus \\ & \{\!|\, (w, \langle E'_1 \parallel_S E'_2, Exec \cup \{a^o_{<loc' \mid loc''>}\} \rangle) \mid a \in S \wedge \\ & \quad (w', \langle E'_1, left(Exec) \cup \{a^o_{loc'}\} \rangle) \in CM(\langle E_1, left(Exec) \rangle) \wedge \\ & \quad (w'', \langle E'_2, right(Exec) \cup \{a^o_{loc''}\} \rangle) \in CM(\langle E_2, right(Exec) \rangle) \wedge \\ & \quad w = w' \cdot w'' \,|\!\} \end{aligned}$$

$$CM(\langle A, \emptyset \rangle) = CM(\langle E, \emptyset \rangle) \qquad A \stackrel{\Delta}{=} E$$

$$\begin{aligned} melt(CM) = \{ (w, \langle E, Exec \rangle) \mid \; & \exists w' : (w', \langle E, Exec \rangle) \in CM \; \wedge \\ & w = \textstyle\sum \{\!|\, w'' \mid (w'', \langle E, Exec \rangle) \in CM \,|\!\} \} \end{aligned}$$

$$left(vids) = \{a^v_{loc} \mid a^v_{\swarrow loc} \in vids \vee \exists loc' : a^v_{\langle loc | loc' \rangle} \in vids\}$$

$$right(vids) = \{a^v_{loc} \mid a^v_{\searrow loc} \in vids \vee \exists loc' : a^v_{\langle loc' | loc \rangle} \in vids\}$$

$$hide(vids, L) = \{a^h_{loc} \mid a^v_{loc} \in vids \wedge a \in L\} \cup \{a^v_{loc} \in vids \mid a \notin L\}$$

$$relab(vids, \varphi) = \{\varphi(a)^v_{loc} \mid a^v_{loc} \in vids\}$$

Table 1. Rules for choice transitions

$$\frac{TM(\langle E, Exec\rangle)(Ter) = E'}{\langle E, Exec\rangle \longrightarrow \langle E', Exec - Ter\rangle} \quad Ter \subseteq Exec,\ Ter \neq \emptyset$$
$TM(\langle <a^v, w>.E, \{a^v\}\rangle)(\{a^v\}) = E$ $TM(\langle E/L, hide(Exec, L)\rangle)(hide(Ter, L)) = TM(\langle E, Exec\rangle)(Ter)/L$ $TM(\langle E[\varphi], relab(Exec, \varphi)\rangle)(relab(Ter, \varphi)) = TM(\langle E, Exec\rangle)(Ter)[\varphi]$ $TM(\langle E_1 \parallel_S E_2, Exec\rangle)(Ter) = E_1' \parallel_S E_2'$ where : $E_1' \equiv \begin{cases} TM(\langle E_1, left(Exec)\rangle)(left(Ter)) & \text{if } left(Ter) \neq \emptyset \\ E_1 & \text{if } left(Ter) = \emptyset \end{cases}$ $E_2' \equiv \begin{cases} TM(\langle E_2, right(Exec)\rangle)(right(Ter)) & \text{if } right(Ter) \neq \emptyset \\ E_2 & \text{if } right(Ter) = \emptyset \end{cases}$

Table 2. Rules for termination transitions

4.2 Computation of Action Durations

The function $PDF : AId \longrightarrow Dist$ that assigns to each identifier its distribution of duration is computed as follows

$$PDF(a_{loc}) = PDFCalc(a_{loc}, \bullet)$$

where function $PDFCalc : (AId \times PId) \longrightarrow Dist$ deals with functions $PDFSynch_a$ and PDF_{pid}, defined at the specification level, in the following way: [6]

$$PDFCalc(a_{\langle loc_1 | loc_2\rangle}, pid) = PDFSynch_a(PDFCalc(a_{loc_1}, pid\swarrow), PDFCalc(a_{loc_2}, pid\searrow))$$
$$PDFCalc(a_{\swarrow loc}, pid) = PDFCalc(a_{loc}, pid\swarrow)$$
$$PDFCalc(a_{\searrow loc}, pid) = PDFCalc(a_{loc}, pid\searrow)$$
$$PDFCalc(a, pid) = PDF_{pid}(a)$$

4.3 Definition of the Integrated Semantics

Definition 2. The *integrated semantics* of $E \in \mathcal{G}$ is given by the following tuple composed of a LTS and a set of duration distributions

$$\mathcal{I}[\![E]\!] = (S_E, \longrightarrow_E, \longmapsto_E, r_E; PDF_E)$$

where S_E is the set of states reachable from the initial state $r_E = \langle E, \emptyset\rangle$ via $\longmapsto_E$ (the restriction of $\longmapsto$ to $S_E \times \mathbb{R}_+ \times S_E$) and $\longrightarrow_E$ (the restriction of $\longrightarrow$ to $S_E \times S_E$), and PDF_E is the restriction of function PDF to the set $Ids_E = \{a_{loc} \mid \exists \langle E, Exec\rangle \in S_E, v \in AVis : a^v_{loc} \in Exec\}$ of the identifiers of the actions that may be executed by E. ■

[6] We assume the two functions $PDFSynch_a$ and PDF_{pid} to be always defined.

4.4 Functional and Performance Semantics

From the integrated model we can derive by projection a functional model and a performance model. The functional model is obtained by removing the quantitative information related to duration of actions and probability of choices. The performance model is a GSMP obtained by abstracting from functional information, i.e. from action types and locations. See [7] for more details.

5 A Simple Example: Queueing Systems $M/D/1/2/2$

A queueing system (QS for short) [19] is a model largely used for performance evaluation purposes to represent a service center composed of a waiting queue and a given number of servers, which provides a certain service (following a given discipline) to the customers arriving at the service center.

Here we consider a QS $M/D/1/2/2$ with two classes of customers: it is composed of a single server and a FIFO queue with capacity 1, providing service to two customers each belonging to a different class. The first (second) customer stays outside the system for a time exponentially distributed with some parameter λ (μ), then requires service. Service times have a deterministic duration k. If we represent by w the waiting of a customer outside the system, by e the event of a customer entering the system, by d the delivery of a customer to the server, by s the service of a customer, and by l the event of a customer leaving the system, such a QS can be modeled as follows with GSMPA by compositionally specifying behaviors and durations:

- $QS_{M/D/1/2/2} \stackrel{\Delta}{=} (Customer \parallel_{\emptyset} Customer) \parallel_{\{e,l\}} (Queue \parallel_{\{d\}} Server)$

  $PDFSynch_e = PDFSynch_l = PDFSynch_d = \max$ [7]

  - $Customer \stackrel{\Delta}{=} w.e.l.Customer$

    $PDF_{\swarrow\swarrow}(w) = Exp(\lambda), PDF_{\swarrow\swarrow}(e) = PDF_{\swarrow\swarrow}(l) = Det(0)$
    $PDF_{\swarrow\searrow}(w) = Exp(\mu), PDF_{\swarrow\searrow}(e) = PDF_{\swarrow\searrow}(l) = Det(0)$

  - $Queue \stackrel{\Delta}{=} e.d.Queue$

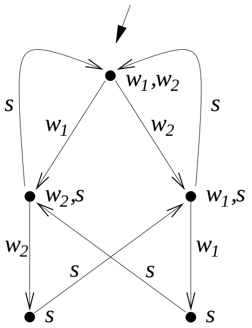
    $PDF_{\searrow\swarrow}(e) = PDF_{\searrow\swarrow}(d) = Det(0)$

  - $Server \stackrel{\Delta}{=} d.s.l.Server$

    $PDF_{\searrow\searrow}(s) = Det(k), PDF_{\searrow\searrow}(d) = PDF_{\searrow\searrow}(l) = Det(0)$

Suppose we want to compute the utilization of the server. The first step consists of deriving by projection a GSMP from $\mathcal{I}[\![QS_{M/D/1/2/2}]\!]$. The resulting GSMP is depicted below.

[7] Operator max computes the pointwise maximum of two functions.



$$PDF(w_1) = Exp(\lambda), PDF(w_2) = Exp(\mu), PDF(s) = Det(k)$$

where element w_1 corresponds to action $w_{\swarrow\swarrow}$, element w_2 to action $w_{\swarrow\searrow}$, and element s to action $s_{\searrow\searrow}$. Note that actions e, d, and l do not occur since the GSMP states with zero sojourn time have been eliminated with the technique presented in [7].

This GSMP is not insensitive due to the behavior of element s in the states labeled with w_1, s and w_2, s. By applying the technique of state amalgamation presented in [15] we merge the state labeled with w_1, s with the state labeled with s it reaches when w_1 terminates, and the state labeled with w_2, s with the state labeled with s it reaches when w_2 terminates. In this way we obtain an insensitive GSMP which in this case turns out to be even an SMP. As we can susbstitute an exponential distribution with rate k^{-1} for the deterministic duration k, we obtain a MC. Finally, to compute the utilization of the server we resort to the technique of rewards as in [3]. With this technique the performance measure of interest is the weighted sum of the steady state probabilities of the states of the Markov chain. In our case we have to single out those states in which the server is providing service. This is accomplished by simply assigning reward 1 to the states whose label includes s. If e.g. $\lambda = 0.2$, $\mu = 0.1$, and $k = 3$, the utilization turns out to be 57%.

6 A Notion of Equivalence for GSMPA

In the following we propose a notion of integrated equivalence for GSMPA which combines a bisimulation in the form of that presented in [2] with the notion of *probabilistic bisimulation* proposed in [20]. According to [20], a probabilistic bisimulation must be an equivalence relation such that two bisimilar terms have the same aggregated probability to reach the same equivalence class. In the case of GSMPA, such a notion must be refined in order to take durations into account. This was a relatively easy task for Markovian process algebras (see e.g. [17, 16, 5]) thanks to the memoryless property of exponential distributions. Now, since general distributions are involved, we have to deal with the fact that the sojourn time in a state depends not only on the duration distributions of the actions in execution in that state, but also on the spent lifetimes of such actions. Consequently, the best we can hope for is to equate some pairs of states whenever we can rely on a correspondence between their sets of actions in execution (following the approach of [2]) such that matching actions have the same duration distribution. In order to describe such a correspondence, we define the notion of consistent action association which relates actions by abstracting from their locations.

Definition 3. A relation $\psi \subseteq AVId \times AVId$ is a *consistent location association* if and only if there exist $vids_1, vids_2 \subseteq AVId$ such that ψ is a bijection from $vids_1$ to $vids_2$ and

$$(a^{v_1}_{loc_1}, b^{v_2}_{loc_2}) \in \psi \Rightarrow (a = b \wedge v_1 = v_2 \wedge PDF(a_{loc_1}) = PDF(b_{loc_2}))$$

We let Ψ be the set of all consistent location associations. ■

As in [2] we deal with a family of bisimulations each indexed by a location association ψ. A bisimulation $\mathcal{B}_\psi$ equates two states, whose actions in execution have been matched according to ψ, whenever they behave the same according to probabilistic bisimulation as far as choice transitions are concerned, and classical bisimulation as far as termination transitions are concerned. More precisely two states are equated if they have: (i) the same aggregated weight to reach corresponding equivalence classes by choosing actions having the same type, the same visibility, and the same duration distribution, where classes are related by a bisimulation indexed by ψ augmented with a pair consisting of two chosen actions, (ii) the same possibility to reach corresponding equivalence classes by terminating a set of actions which are pointwise matched in ψ, where classes are related by a bisimulation indexed by ψ diminished by the terminated actions.

Before defining the equivalence we give some auxiliary definitions. A relation $\mathcal{B}$ over a set A can be seen as representing a correspondence among the elements of two copies of the same set A in this way: whenever $a\ \mathcal{B}\ b$, a is an element of the first copy of A and b is an element of the second copy of A. If transitivity is guaranteed such a relation can be seen as representing a bijective mapping between classes of elements of the first copy of A and classes of elements of the second copy of A.

Definition 4. Given a set A we say that a relation $\mathcal{B} \subseteq A \times A$ is a *class mapping over* A, denoted $\mathcal{B} \in cmap(A)$, if and only if $\mathcal{B} = \mathcal{B} \circ \mathcal{B}^{-1} \circ \mathcal{B}$ ■

Proposition 5. *Given a set A and a relation $\mathcal{B} \subseteq A \times A$ we have that $\mathcal{B} \in cmap(A)$ if and only if either $\mathcal{B} = \emptyset$ or there is a unique way to find $A_1, A_2 \subseteq A$, π_1 partition of A_1, π_2 partition of A_2, and a bijection $f : \pi_1 \longrightarrow \pi_2$ such that:*

$$\mathcal{B} = \bigcup_{(C_1, C_2) \in f} C_1 \times C_2$$

■

Given a class mapping $\mathcal{B}$ of a set A, we define $crel(\mathcal{B})$ as the extension to the whole A of the class bijection f induced by $\mathcal{B}$.

Definition 6. Given a set A and $\mathcal{B} \in cmap(A)$ we define the *class relation associated with* $\mathcal{B}$, denoted $crel(\mathcal{B}) \subseteq \mathcal{P}(A) \times \mathcal{P}(A)$ as follows:

- if $\mathcal{B} = \emptyset$ then $crel(\mathcal{B}) = \emptyset$,
- otherwise $crel(\mathcal{B}) = f \cup \{(A - A_1, \emptyset), (\emptyset, A - A_2)\} - \{(\emptyset, \emptyset)\}$ where f, A_1 and A_2 are as in the previous proposition. ∎

We are now in a position to define the equivalence for GSMPA.

Definition 7. A *strong generalized semi-Markovian bisimulation family (strong GSMBF)* is a Ψ-indexed family $\mathbf{B} = \{\mathcal{B}_\psi \subseteq LabS \times LabS \mid \psi \in \Psi\}$ of relations over $LabS$ such that:

- $I_{\{\langle E, Exec\rangle \mid E \in \mathcal{G}\}} \subseteq \mathcal{B}_{I_{Exec}} \qquad \forall\ Exec \subseteq AVId$ [8]
- $s_1\ \mathcal{B}_\psi\ s_2 \Rightarrow s_2\ \mathcal{B}_{\psi^{-1}}\ s_1 \qquad \forall\ \psi \in \Psi$
- $s_1\ \mathcal{B}_{\psi'}\ s_2 \wedge s_2\ \mathcal{B}_{\psi''}\ s_3 \Rightarrow s_1\ \mathcal{B}_{\psi' \circ \psi''}\ s_3 \qquad \forall\ \psi', \psi'' \in \Psi$
- $(s_1, s_2) \in \mathcal{B}_\psi \Rightarrow$
  - $s_1 \equiv \langle E_1, dom(\psi)\rangle \wedge s_2 \equiv \langle E_2, range(\psi)\rangle$
  - $\forall\ (C_1, C_2) \in crel(\bigcup_{\substack{a^v_{loc_1}, a^v_{loc_2} \in AVId \\ PDF(a_{loc_1}) = PDF(a_{loc_2})}} \mathcal{B}_{\psi \cup \{a^v_{loc_1}, a^v_{loc_2}\}})$

    $$\sum \{\!|\ w \mid \exists s' \in C_1 : s_1 \stackrel{w}{\longmapsto} s'\ |\!\} = \sum \{\!|\ w \mid \exists s' \in C_2 : s_2 \stackrel{w}{\longmapsto} s'\ |\!\}$$

  - $\forall\ (C_1, C_2) \in crel(\bigcup_{nter \subset dom(\psi)} \mathcal{B}_{\psi|_{nter}})$ [9]

    $$\exists s' \in C_1 : s_1 \longrightarrow s' \Leftrightarrow \exists s' \in C_2 : s_2 \longrightarrow s'$$ ∎

Some consequences of the first three conditions are that: $\forall Exec \subseteq AVId$, $\mathcal{B}_{I_{Exec}}$ is an equivalence relation; $\forall \psi \in \Psi$, $\mathcal{B}_\psi \in cmap(LabS)$; and that the two unions appearing in the last two items always result in a class mapping over $LabS$.

Proposition 8. *Let $\{\mathbf{B_i} \mid i \in I\}$ with $\mathbf{B_i} = \{\mathcal{B}_{\psi,i} \mid \psi \in \Psi\}$ be the set of all strong GSMBFs. Let $\sim_{\mathbf{GSMBF}} = \{\sim_{GSMBF,\psi} \mid \sim_{GSMBF,\psi} = \bigcup_{i \in I} \mathcal{B}_{\psi,i}\}$. Then $\sim_{\mathbf{GSMBF}}$ is the largest strong GSMBF, i.e. for any strong GSMBF $\mathbf{B} = \{\mathcal{B}_\psi \mid \psi \in \Psi\}$ we have $\mathcal{B}_\psi \subseteq \sim_{GSMBF,\psi} \quad \forall\ \psi \in \Psi$.* ∎

Definition 9. We define $\sim_{GSMBE} \subset \mathcal{G} \times \mathcal{G}$, the *strong generalized semi-Markovian bisimulation equivalence (strong GSMBE)*, as follows:

$$\sim_{GSMBE} = \{(E_1, E_2) \in \mathcal{G} \times \mathcal{G} \mid (<E_1, \emptyset>, <E_2, \emptyset>) \in \sim_{GSMBF,\emptyset}\}$$ ∎

Theorem 10. *$\sim_{GSMBE}$ is a congruence with respect to GSMPA operators and recursive definitions.* ∎

---

[8] I_A denotes the identity relation over set A and $I_\emptyset = \emptyset$.

[9] $nter$ denotes a set of nonterminating actions and $\psi\ |_{nter}$ the restriction of ψ to the set $nter$.

We recall that, as stated in Sect. 2, in order to get a congruence we need to represent in the semantic models also those computations which are not actually possible because of timing considerations. As an example consider $E_1 \equiv (a.c.\underline{0} \,\|_\emptyset\, b.c.\underline{0}) \,\|_{\{c\}}\, c.\underline{0}$ and $E_2 \equiv (a.c.\underline{0} \,\|_\emptyset\, b.\underline{0}) \,\|_{\{c\}}\, c.\underline{0}$ where any a has duration 4 and any b has duration 5. If the possibility of b terminating before a were not taken into account in the semantics, then it would turn out $E_1 \sim_{GSMBE} E_2$ but $E_1 \,\|_{\{a\}}\, \underline{0} \not\sim_{GSMBE} E_2 \,\|_{\{a\}}\, \underline{0}$.

7 Conclusion

As far as future work is concerned, we are following two research directions. Dealing with both continuous and discrete duration distributions in GSMPA has the potentiality to create a uniform framework for performance evaluation process algebras and real time process algebras. Preliminary work in this direction can be found in [7], where we have added to the core calculus presented in this paper the capability of modeling interrupt mechanisms and we have started investigating how GSMPA scales down to existing deterministically or stochastically timed process algebras when only some duration distributions are considered. Another very promising research direction concerns the connection between the approximation of general distributions through phase type distributions and the potentiality of the ST semantics to support action refinement. An action with a phase type duration distribution may be equivalently represented from the stochastic point of view by a process term employing actions with exponential distributions only. Since ST semantics supports action refinement, substituting equivalent terms for actions in a system specification would be correct also from the functional point of view.

References

1. M. Ajmone Marsan, A. Bianco, L. Ciminiera, R. Sisto, A. Valenzano, *"A LOTOS Extension for the Performance Analysis of Distributed Systems"*, in IEEE/ACM Trans. on Networking 2:151-164, 1994
2. L. Aceto, M. Hennessy, *"Adding Action Refinement to a Finite Process Algebra"*, in Information and Computation 115:179-247, 1994
3. M. Bernardo, *"An Algebra-Based Method to Associate Rewards with EMPA Terms"*, in Proc. of the *24th Int. Coll. on Automata, Languages and Programming (ICALP '97)*, LNCS 1256:358-368, Bologna (Italy), 1997
4. M. Bernardo, L. Donatiello, R. Gorrieri, *"A Formal Approach to the Integration of Performance Aspects in the Modeling and Analysis of Concurrent Systems"*, to appear in Information and Computation, 1998
5. M. Bernardo, R. Gorrieri, *"A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time"*, to appear in Theoretical Computer Science, 1998
6. G. Boudol, I. Castellani, *"Permutation of Transitions: An Event Structure Semantics for CCS and SCCS"*, in Proc. of the *Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, LNCS 354:411-427, Noordwijkerhout (The Netherlands), 1988

7. M. Bravetti, M. Bernardo, R. Gorrieri, *"Generalized Semi Markovian Process Algebra"*, Technical Report UBLCS-97-9, University of Bologna (Italy), October 1997
8. N. Busi, R.J. van Glabbeek, R. Gorrieri, *"Axiomatising ST-Bisimulation Equivalence"*, in Proc. of the *IFIP Working Conf. on Programming Concepts, Methods and Calculi (PROCOMET '94)*, S. Miniato (Italy), 1994
9. E. Brinksma, J.-P. Katoen, R. Langerak, D. Latella, *"A Stochastic Causality-Based Process Algebra"*, in Computer Journal 38:553-565, 1995
10. D. R. Cox, *"The Analysis of non-Markovian Stochastic Processes by the Inclusion of Supplementary Variables"*, in Proc. of the Cambridge Philosophical Society 51:433-440, 1955
11. P.R. D'Argenio, J.-P. Katoen, E. Brinksma, *"A Stochastic Automata Model and its Algebraic Approach"* in Proc. of the *5th Workshop on Process Algebras and Performance Modelling (PAPM '97)*, pp. 1-16, Enschede (The Netherlands), 1997
12. R.J. van Glabbeek, F.W. Vaandrager, *"Petri Net Models for Algebraic Theories of Concurrency"*, in Proc. of the *Conf. on Parallel Architectures and Languages Europe (PARLE '87)*, LNCS 259:224-242, Eindhoven (The Netherlands), 1987
13. N. Götz, U. Herzog, M. Rettelbach, *"Multiprocessor and Distributed System Design: The Integration of Functional Specification and Performance Analysis Using Stochastic Process Algebras"*, in Proc. of the *16th Int. Symp. on Computer Performance Modelling, Measurement and Evaluation (PERFORMANCE '93)*, LNCS 729:121-146, Rome (Italy), 1993
14. N. Götz, U. Herzog, M. Rettelbach, *"TIPP - A Stochastic Process Algebra"*, in Proc. of the *1st Workshop on Process Algebras and Performance Modelling (PAPM '93)*, pp. 31-36, Edinburgh (UK), 1993
15. W. Henderson, D. Lucic, *"Aggregation and Disaggregation through Insensitivity in Stochastic Petri Nets"*, in Performance Evaluation 17:91-114, 1993
16. H. Hermanns, M. Rettelbach, *"Syntax, Semantics, Equivalences, and Axioms for MTIPP"*, in Proc. of the *2nd Workshop on Process Algebras and Performance Modelling (PAPM '94)*, pp. 71-87, Erlangen (Germany), 1994
17. J. Hillston, *"A Compositional Approach to Performance Modelling"*, Cambridge University Press, 1996
18. C.A.R. Hoare, *"Communicating Sequential Processes"*, Prentice Hall, 1985
19. L. Kleinrock, *"Queueing Systems"*, John Wiley & Sons, 1975
20. K.G. Larsen, A. Skou, *"Bisimulation through Probabilistic Testing"*, in Information and Computation 94:1-28, 1991
21. K. Matthes, *"Zur Theorie der Bedienungsprozesse"*, in Trans. of the *3rd Prague Conf. on Information Theory, Stat. Dec. Fns. and Random Processes*, pp. 513-528, 1962
22. M. Nielsen, P.S. Thiagarajan, *"Degrees of Nondeterminism and Concurrency"*, in Proc. of the *4th Conf. on Foundations of Software Technologies and Theoretical Computer Science*, LNCS 181:89-117, 1984
23. C. Priami, *"Stochastic π-Calculus with General Distributions"*, in *Proc. of the 4th Workshop on Process Algebras and Performance Modelling (PAPM '96)*, CLUT, pp. 41-57, Torino (Italy), 1996
24. C. Tofts, *"Processes with Probabilities, Priority and Time"*, in Formal Aspects of Computing 6:536-564, 1994