Modeling and Analyzing Concurrent Systems with MPA

Marco Bernardo Lorenzo Donatiello Roberto Gorrieri [*]

Dipartimento di Matematica, Università di Bologna,

Piazza di Porta S. Donato 5, 40127 Bologna, Italy.

Abstract

Process algebras are one of the main tools for modeling and analyzing concurrent systems. However, they can be used to describe only the functional aspect of system behavior. Recently, the relevance of integrating performance evaluation within the process of specification, design and implementation of concurrent systems has been widely recognized. Hence, an effort has been made in order to handle also the temporal aspect of system behavior.

In this paper the stochastic process algebra MPA (Markovian Process Algebra) is briefly introduced, together with its operational interleaving semantics, its markovian semantics and its operational net semantics. A concurrent system is described as a term of MPA. The operational interleaving semantics (defined by following Plotkin's structured operational semantics approach, augmented with two transformations) associates a labeled transition system with each MPA term. The markovian semantics is defined through an algorithm which transforms labeled transition systems into state transition rate diagrams of homogeneous continuous time Markov chains. The operational net semantics is defined by following an extension of Plotkin's structured operational semantics approach to nets (as proposed by Degano-De Nicola-Montanari and Olderog), yielding a generalized stochastic Petri net.

A modeling technique for concurrent systems based on MPA is also proposed which integrates different points of view of concurrent systems as well as the qualitative and quantitative analysis of concurrent systems. Finally, some examples are shown which demonstrate the expressiveness and the compositionality of MPA.

Keywords: I.10, II.12, III.1, III.5, III.6, IV.1

Additional Keywords: process algebra, integration of functional and performance modeling, interleaving semantics, markovian semantics, net semantics.

[*]**Contact Author**:voice: +39-51-354494, fax: +39-51-354490, e-mail: `gorrieri@cs.unibo.it`.

1 Introduction

Since the 60's a lot of work has been done in order to devise modeling techniques which are adequate to represent concurrent systems. Process algebras are one of the most relevant results of this work and represent a very useful tool for modeling and analyzing such systems. As a matter of fact,

- Process algebras are *abstract languages* conceived for defining formally concurrent systems in a *compositional* way. I.e., they provide the designer with a restricted set of powerful operators by means of which constructing more and more complex systems from simpler ones; as a consequence, the designer is led to use a hierarchical and modular design style.

- With process algebras concurrent systems can be given formal descriptions which are more easily understandable and more easily modifiable than the ones obtained by many other models.

Classical process algebras, like CCS [19], TCSP [15] and LOTOS [5], cannot be used to evaluate the performance of concurrent systems because they allow to describe only the *functional aspect* of the behavior of concurrent systems. Neglecting the *temporal aspect* of the behavior of concurrent systems is a remarkable drawback for the expressiveness of the classical process algebras because:

- Time-critical concurrent systems (e.g. real time systems and communication protocols) cannot be modeled in a completely satisfactory manner; moreover,

- The performance of the concurrent systems cannot be estimated. It often happens that a concurrent system is firstly fully designed and tested for functionality and afterwards tested for efficiency; so, if the performance is detected to be poor, the concurrent system has to be redesigned with negative consequences for both the design costs and the delivery at a fixed deadline.

The relevance of integrating the analysis of the performance of a concurrent system into the design process of the system itself has recently stimulated many researchers. In the framework of process algebras, this need is satisfied by introducing the concept of time in order to express the durations of the activities performed by the concurrent systems being modeled; when such durations are expressed in a deterministic way, the process algebra is said to be a *temporal process algebra* (see, e.g., [20, 23, 11, 12]), whereas when such durations are expressed through random variables, the process algebra is said to be a *stochastic process algebra* [13, 14].

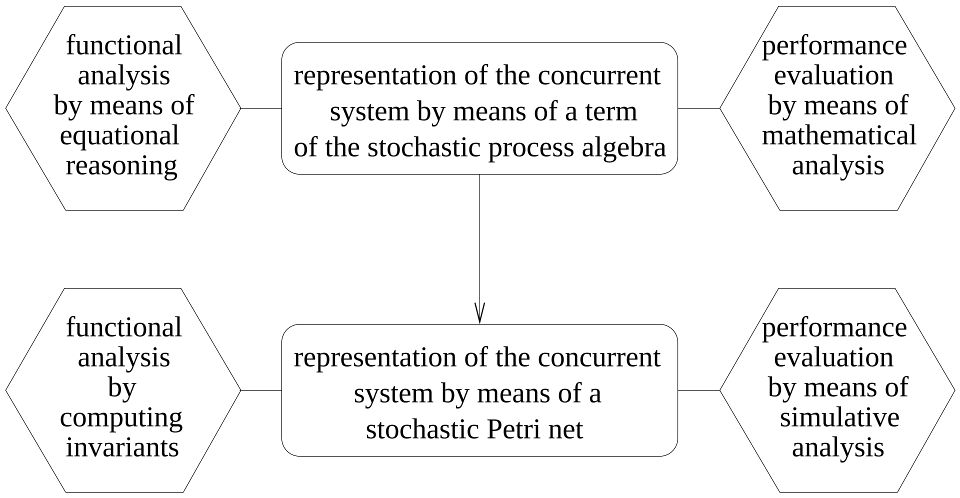
In this paper we present a stochastic process algebra named *MPA (Markovian Process Algebra)*, which is supplied with an operational interleaving semantics, a markovian semantics and an operational net semantics. Therefore, there are three semantic models associated with each MPA term: an *interleaving model* represented by a labeled transition system [22], a *stochastic model* represented by the state transition rate diagram of a homogeneous continuous time Markov chain (HCTMC) [16, 17], and a *distributed model* represented by a generalized stochastic Petri net (GSPN) [18].

With a stochastic process algebra we can exploit a modeling technique for concurrent systems which integrates:

- Different points of view of concurrent systems, and

- The qualitative and the quantitative analysis of concurrent systems.

Such a modeling technique can be summarized by the following scheme


functional analysis by means of equational reasoning
representation of the concurrent system by means of a term of the stochastic process algebra
performance evaluation by means of mathematical analysis
functional analysis by computing invariants
representation of the concurrent system by means of a stochastic Petri net
performance evaluation by means of simulative analysis

which indicates that:

- The first phase consists of specifying the concurrent system as a term of the stochastic process algebra (which is MPA in our case), so as to obtain a first representation, easy to understand and compositional. With this algebraic representation, it is possible to:
  - Perform a functional analysis of the concurrent system by means of *equivalence checking*, *preorder checking* and *model checking* [9]. Moreover, if the equivalence relation is a congruence, analysis can be performed by equational reasoning defined on its axiomatization. Such analyses, which can be also computer-aided (see, e.g., [9]), can detect qualitative properties of the concurrent system (e.g. deadlock) and can also help in minimizing the state space of the system representation.
  - Evaluate the performance of the concurrent system by resorting to the study of its associated HCTMC, which can be assisted by computer [25]
- The second phase consists of translating the term of the stochastic process algebra into a stochastic Petri net (which is a GSPN in our case), giving a distributed representation of the same concurrent system. Such a translation makes explicit the parallelism and the causal dependencies among the activities of the concurrent system itself, but the price to pay is that such a model usually has a remarkable graphic complexity. Moreover, it is not easily compositional, hence making hard, in general, the detection and the analysis of its subsystems. With this stochastic Petri net representation, it is possible to:
  - Perform a qualitative analysis of the concurrent system by, e.g., computing net invariants [7].
  - Evaluate the performance of the concurrent systems by resorting to existing tools for simulation on stochastic Petri nets, e.g. [8].

The purpose of this work is not that of formally defining MPA together with its three semantics, rather that of giving an idea about what can be done by using MPA (e.g., by means of the modeling technique above). The reader interested in the mathematical details is invited to consult [1, 2, 3, 4]. [1]

This paper is organized as follows. In Section 2 the syntax and the informal semantics of MPA are introduced. In Sections 3 to 5 three examples are reported to explain how to generate the operational interleaving semantics, the markovian semantics and the operational GSPN semantics of a given MPA term. In Section 6 some more interesting examples are shown in which queueing systems are modeled with MPA. Finally, in Section 7 some concluding remarks are reported.

2 Syntax and informal semantics of MPA

In this section we shall introduce the syntax of MPA terms and we shall informally describe the semantics of MPA operators.

2.1 Actions: type and rate

The first step in the definition of MPA consists of introducing a set of *actions* which model the atomic activities performed by concurrent systems. Each MPA action is described by a pair $<a, \tilde{\lambda}>$ consisting of the *type* of the action and the *rate* of the action. Depending on the type, MPA actions are subdivided into:

- *External or observable actions*, i.e. actions whose execution can be seen by an external observer; we shall denote with Com the set of external action types.
- *Internal or invisible actions*, i.e. actions whose execution cannot be seen by an external observer; we shall denote with τ the only internal action type we shall use.

Depending on the rate, MPA actions are subdivided into:

- *Passive actions*, i.e. actions whose execution rate is zero. The duration of a passive action is undefined and is fixed only upon a synchronization with a nonpassive action of the same type. Passive actions are used for modeling waitings.
- *Active actions*, i.e. actions whose execution rate is nonzero, are divided into:
  - *Timed actions*, i.e. actions whose execution rate is finite; the duration of a timed action is expressed by an exponentially distributed random variable with parameter given by the action rate.
  - *Immediate actions*, i.e. actions whose execution rate is infinite. Such actions have duration zero and are used to model activities whose duration is irrelevant from the performance evaluation point of view. Each immediate action has a priority level l and a weight p associated with it.

We shall denote with

$$Act = Com \cup \{\tau\}$$

the set of action types, ranged over by $a, b, c, \ldots$. Furthermore,

$$Rate = \{0\} \cup \mathbb{R}^+ \cup Inf,\ Inf = \{\infty_{l,p} \mid l \in \mathbb{N}^+ \wedge p \in \mathbb{R}^+\},$$

shall denote the set of action rates. We shall use $\tilde{\lambda}, \tilde{\mu}, \tilde{\gamma}, \ldots$ as metavariables for $Rate$ and $\lambda, \mu, \gamma, \ldots$ as metavariables for $\mathbb{R}^+$.

[1]The technical reports are available via anonymous FTP from the area `ftp.cs.unibo.it:/pub/TR/UBLCS` in compressed PostScript format.

2.2 Syntax of terms and informal semantics of operators

Let Con be a set of *constants*, ranged over by $A, B, C, \ldots$. Each constant A is used as a shorthand for a MPA term E through its *defining equation* $A \stackrel{\Delta}{=} E$.

Let Φ be a set of *relabeling functions* which transform action types into other action types preserving their observability; more accurately, let

$$\Phi = \{\varphi : Act \longrightarrow Act \mid \varphi(\tau) = \tau \wedge \forall a \in Com. \, \varphi(a) \in Com\}.$$

We shall use $\varphi, \varphi', \varphi'', \ldots$ as metavariables for Φ.

The set $\mathcal{L}$ of MPA *terms* is defined as the set of terms E generated by the following syntax:

$$E ::= \underline{0} \mid {<}a, \tilde{\lambda}{>}.E \mid E/L \mid E\backslash H \mid E[\varphi] \mid E + E \mid E\|_S E \mid A$$

where $a \in Act$, $\tilde{\lambda} \in Rate$, $L \subseteq Com$, $H \subseteq Act$, $\varphi \in \Phi$, $S \subseteq Com$, $A \in Con$.

Now we shall informally explain the semantics of MPA operators.

2.2.1 Null term

The *null term* "$\underline{0}$" is a zero-ary operator which models a deadlocked system, i.e. a system which cannot execute actions any more.

2.2.2 Prefix operator

The *prefix operator* "." expresses the sequential composition of an action and a term. Hence, the system ${<}a, \tilde{\lambda}{>}.E$ executes the action ${<}a, \tilde{\lambda}{>}$ and then behaves as E.

2.2.3 Functional abstraction operator

The *functional abstraction operator* "/" expresses the abstraction from the type of actions whose type is in a given set. Hence, the system E/L behaves as E, but the actions whose type appears in L are transformed into the invisible action τ. I.e., the functional abstraction operator is essentially the hiding operator of TCSP [15].

2.2.4 Temporal restriction operator

The *temporal restriction operator* "\" prevents the execution of passive actions whose type is in a given set. Hence, the system $E\backslash H$ behaves as E, but the passive actions whose type appears in H cannot be executed.

2.2.5 Relabeling operator

The *relabeling operator* "[]" changes the type of the actions. Indeed, the system $E[\varphi]$ behaves as E, up to transformations of action types as indicated by the relabeling function φ. Note that the relabeling operator and the functional abstraction operator have clearly distinct roles; as a matter of fact, the relabeling operator preserves the observability of the action types, whilst the functional abstraction operator can only transform observable actions into invisible ones.

The relabeling operator avoids the duplication of terms representing components of a given system which differ only for action types (see the end of Section 6.3 for an example).

2.2.6 Alternative composition operator

The *alternative composition operator* "+" expresses a choice between two terms. Hence, the system $E_1 + E_2$ behaves as either E_1 or E_2.

2.2.7 Parallel composition operator

The *parallel composition operator* "$\|$" expresses the parallel composition of two terms according to the TCSP synchronization discipline on types. Hence, the system $E_1 \|_S E_2$, where S is called the *synchronization set*, can execute asynchronously actions whose type does not appear in S from E_1 *or* E_2, and synchronously actions whose type appears in S from E_1 *and* E_2. The actual executability of synchronizations may depend on the rate of the involved actions:

- Action $<a, \lambda>$ or $<a, \infty_{l,p}>$ can be synchronized with action $<a, 0>$, hence modeling the temporal closure of passive actions (see Section 3 for more details).

- Action $<a, 0>$ can be synchronized with action $<a, 0>$, hence allowing to model n-way synchronizations, $n > 2$, where $n-1$ passive actions and one active action are involved.

- Action $<a, \tilde{\lambda}>$ cannot be synchronized with action $<a, \tilde{\mu}>$ when $\tilde{\lambda} \neq 0 \neq \tilde{\mu}$, because we require that in a synchronization at most one active action is involved. So, the rate of the resulting action is uniquely determined by the rate of its active subaction. We think that this choice leads to the adoption of a clearer modular design style.

2.2.8 Recursive definition

The meaning of a defining equation $A \stackrel{\Delta}{=} E$ is that the constant A behaves as the term E. Constants can be used for defining recursive terms. For instance, if $A \stackrel{\Delta}{=} E$ and E contains an occurrence of A, then the definition of A is recursive (obviously, the recursion may be indirect in more complex cases).

A very relevant property is the so-called *guarded closure*, which concerns the correct definition of (recursive) terms from the functional point of view.

- A guardedly closed term is *closed* so it does not contain constants no more instantiable due to the lack of the corresponding defining equations.

- A guardedly closed term is *guarded* so it cannot result in an infinite sequence of substitutions of a constant with another constant without executing actions.

In the following we shall consider only guardedly closed terms and we shall denote with $\mathcal{G}$ the set of guardedly closed terms in $\mathcal{L}$.

2.3 Race policy

Since several active actions can be simultaneously executable, it is necessary to choose a policy determining which of them is to be executed first. In this framework we adopt the *race policy*, which chooses the active action having the least duration; since by definition immediate actions have duration zero whereas timed actions cannot sample durations zero from their exponential distributions, immediate actions take precedence over timed actions.

It is worth noting that each immediate action is equipped with a priority level and a weight; therefore, when several immediate actions are simultaneously executable, only those having the highest priority level are actually executable and the choice among them is probabilistically made by giving each of them an execution probability proportional to its weight.

Before continuing, we recall the following property of the exponentially distributed random variables.

Proposition 2.1 If $X_1, X_2, \ldots, X_n$ are $n \geq 2$ exponentially distributed random variables with parameters $\lambda_1, \lambda_2, \ldots, \lambda_n$, respectively, and a random variable Y is defined by

$$Y = \min_{1 \leq i \leq n} X_i,$$

then Y is an exponentially distributed random variable with parameter

$$\lambda = \sum_{i=1}^{n} \lambda_i.$$

■

As a consequence of the previous remarks and the above proposition, we have that:

- If in a state n timed actions of the form $<a_1, \lambda_1>$,$<a_2, \lambda_2>$,$\ldots$,$<a_n, \lambda_n>$ are executable and no immediate actions are executable, then the execution probability of the timed action $<a_k, \lambda_k>$ is given by
$$\frac{\lambda_k}{\sum_{i=1}^{n} \lambda_i}.$$
- If in a state n immediate actions of the form $<a_1, \infty_{l,p_1}>$,$<a_2, \infty_{l,p_2}>$,$\ldots$,$<a_n, \infty_{l,p_n}>$ are executable and no immediate actions with higher priority are executable, then the execution probability of the immediate action $<a_k, \infty_{l,p_k}>$ is given by
$$\frac{p_k}{\sum_{i=1}^{n} p_i}.$$

We shall also assume that the passive actions executable in a given state have the same nonzero execution probability.

The possibility of having several simultaneously executable actions in a given state is due to the two binary operators of MPA, i.e. the alternative composition operator and the parallel composition operator. Concerning the alternative composition operator, in Section 2.2.6 it is stated that such an operator expresses a choice between two terms and now we can understand that such a choice is of a probabilistic nature, as based on the race policy. To be more precise, such a choice takes:

- An *implicit* form when it concerns the choice between two timed actions, because in this case the execution probability of each of the two timed actions is implicitly determined by the race policy; or
- An *explicit* form when it concerns the choice between two immediate actions at the same priority level, because in this case the execution probability of each of the two immediate actions is explicitly determined by its weight.

3 Operational interleaving semantics of MPA terms

The *operational interleaving semantics* of a term $E \in \mathcal{G}$ is represented by the labeled transition system $\mathcal{I}[\![E]\!]$. This is generated inductively by the structured operational semantics rules, then refined by means of two transformations [2].

The rules are designed by following Plotkin's structured operational semantics approach [22], so their definition proceed by induction on the syntactical structure of MPA terms. Furthermore, they permit to deduce transitions on a by need basis.

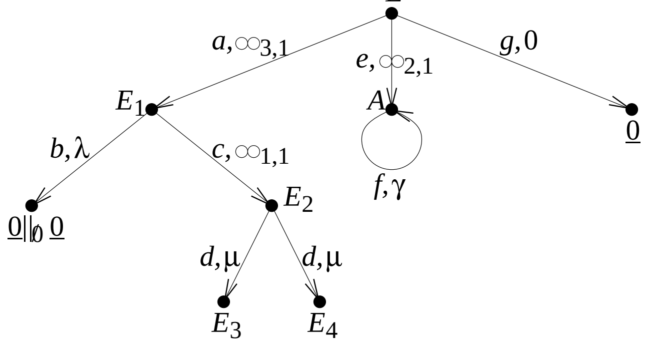
The first transformation is necessary because the structured operational semantics rules partly ignore the race policy. In fact, they take into account neither the priority of immediate actions over timed actions nor the different priority levels existing among immediate actions. The first transformation guarantees that all the active transitions exiting from a given state have the same priority level.

The second transformation is necessary because the structured operational semantics rules do not assign the correct temporal parameter to identically labeled transitions deriving from the synchronization of a timed action with several passive actions (of the same type) which are either independent of each other or mutually exclusive. Thus, this transformation modifies the temporal parameter of the appropriate transitions.

The two transformations above can be performed interactively with the structured operational semantics rules, i.e. it is not necessary to generate the whole state space before applying them. This is due to the fact that we consider only terms in $\mathcal{G}$, hence infinite branching is avoided and only a finite amount of time is needed to generate the transitions from a given state.

Based on the operational interleaving semantics, we introduce the properties of functional closure and temporal closure, which concern the complete definition of terms from the functional point of view and the temporal point of view, respectively.
A term $E \in \mathcal{G}$ is said to be *functionally closed* iff $\mathcal{I}[\![E]\!]$ is isomorphic to $\mathcal{I}[\![E/Com]\!]$, i.e. it cannot execute observable actions. We shall denote with $\mathcal{F}$ the set of terms in $\mathcal{L}$ that are functionally closed.
A term $E \in \mathcal{G}$ is said to be *temporally closed* iff $\mathcal{I}[\![E]\!]$ is isomorphic to $\mathcal{I}[\![E \backslash Act]\!]$, i.e. it cannot execute passive actions. We shall denote with $\mathcal{T}$ the set of terms in $\mathcal{L}$ that are temporally closed; we shall also denote with $\mathcal{E}$ the set of terms in $\mathcal{L}$ that are guardedly and temporally closed, i.e. $\mathcal{E} = \mathcal{G} \cap \mathcal{T}$, and we shall denote with $\mathcal{E}^+$ the set of terms in $\mathcal{E}$ whose operational interleaving semantics does not contain cycles of immediate transitions.

Example 3.1 Consider the term $E \equiv (< a, \infty_{3,1} > .E_1 + < e, \infty_{2,1} > .A) + < g,\, 0 > .\underline{0}$ where $E_1 \equiv <b, \lambda>.(\underline{0} \|_{\emptyset} \underline{0}) + <c, \infty_{1,1}>.E_2$, $A \overset{\Delta}{=} <f, \gamma>.A$, $E_2 \equiv <d, \mu>.\underline{0} \|_{\{d\}} (<d, 0>.\underline{0} \|_{\emptyset} <d, 0>.\underline{0})$.
The labeled transition system obtained by applying the structured operational semantics rules is



where $E_3 \equiv \underline{0}\|_{\{d\}}(\underline{0}\|_{\emptyset}<d,0>.\underline{0})$, $E_4 \equiv \underline{0}\|_{\{d\}}(<d,0>.\underline{0}\|_{\emptyset}\underline{0})$.
The labeled transition system obtained by applying the first transformation is

$$\begin{tikzcd}
 & & & E \arrow[dll, "{a,\infty_{3,1}}"'] \arrow[dr, "{g,0}"] & \\
 & E_1 \arrow[d, "{c,\infty_{1,1}}"'] & & & \underline{0} \\
 & E_2 \arrow[dl, "{d,\mu}"'] \arrow[dr, "{d,\mu}"] & & & \\
E_3 & & E_4 & &
\end{tikzcd}$$

Note that passive actions are not involved in the priority mechanism. Finally, the labeled transition system $\mathcal{I}[\![E]\!]$ obtained by applying the second transformation is

$$\begin{tikzcd}
 & & & E \arrow[dll, "{a,\infty_{3,1}}"'] \arrow[dr, "{g,0}"] & \\
 & E_1 \arrow[d, "{c,\infty_{1,1}}"'] & & & \underline{0} \\
 & E_2 \arrow[dl, "{d,\mu/2}"'] \arrow[dr, "{d,\mu/2}"] & & & \\
E_3 & & E_4 & &
\end{tikzcd}$$

Note that the correct temporal parameter of both the transition from E_2 to E_3 and the transition from E_2 to E_4 is $\mu/2$ instead of μ. This is due to the fact that in E_2 only a timed action occurs with rate μ hence the global exit rate from E_2 must be μ. ■

By forgetting the rate in transition labels, we get a classic transition system labeled on action types, on which we can define any notion of *functional equivalence*, e.g. bisimulation [19].

4 Markovian semantics of MPA terms

The *markovian semantics* of a term $E \in \mathcal{E}^+$ is represented by the state transition rate diagram of the HCTMC $\mathcal{M}[\![E]\!]$. [2] This is obtained by applying to $\mathcal{I}[\![E]\!]$ an algorithm organized in three phases [2].

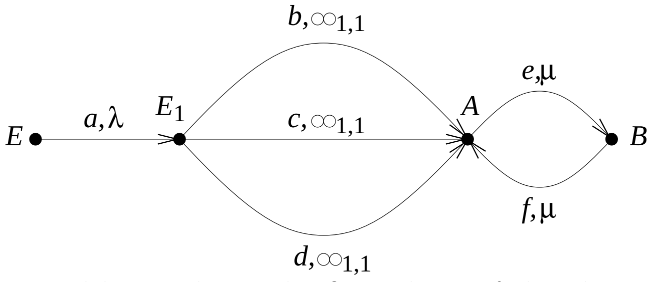
The first phase, besides discarding action types, eliminates all the immediate transitions occurring in $\mathcal{I}[\![E]\!]$, because immediate transitions cannot be present in the state transition rate diagram of a HCTMC.

The second phase produces a state transition rate diagram of a HCTMC by imposing that between each ordered pair of states there is at most one transition (using Proposition 2.1).

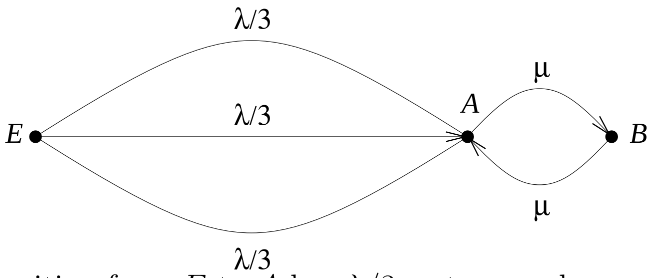
[2]For the basic notions about Markov chains, the reader is referred to the textbooks [16, 17].

The third phase detects and merges states which are equivalent according to the definition of *lumping* [16], so as to minimize the state transition rate diagram of the HCTMC obtained at the end of the previous phase.

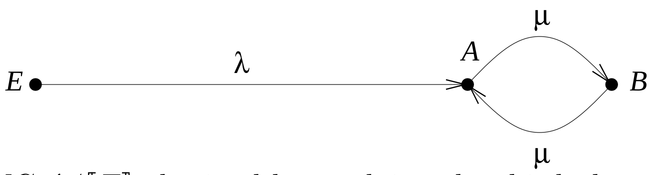
Example 4.1 Consider the term $E \equiv <a, \lambda>.E_1$, $E_1 \equiv <b, \infty_{1,1}>.A + <c, \infty_{1,1}>.A + <$ d,
$\infty_{1,1}>.A$, $A \stackrel{\Delta}{=} <e, \mu>.B$, $B \stackrel{\Delta}{=} <f, \mu>.A$.
The labeled transition system $\mathcal{I}[\![E]\!]$ is



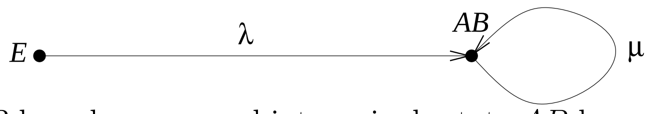
The automaton obtained by applying the first phase of the algorithm is



Note that each transition from E to A has $\lambda/3$ as temporal parameter because $1/3$ was the probability of each action which labeled one of the immediate transitions from E_1 to A. The HCTMC obtained by applying the second phase of the algorithm is



Finally, the HCTMC $\mathcal{M}[\![E]\!]$ obtained by applying the third phase of the algorithm is



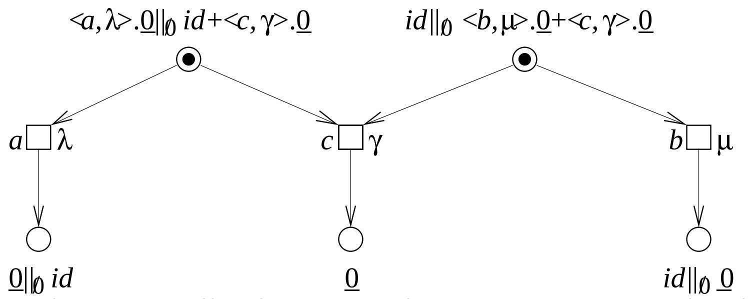
Note that A and B have been merged into a single state AB because they are equivalent according to the definition of lumping, i.e. they have the same global transition rate to each of the sets of states in the final HCTMC. ■

One of the possible notions of *temporal equivalence* can be defined in a natural way: Two systems will be considered equivalent from the temporal point of view if they give rise to isomorphic HCTMCs.

5 Operational GSPN semantics of MPA terms

The *operational GSPN semantics* of a term $E \in \mathcal{E}$ is represented by the generalized stochastic Petri net $\mathcal{GSPN}[\![E]\!]$. [3] This is obtained by applying an extension [4] of the structured operational semantics approach proposed by Degano-De Nicola-Montanari and then refined by the same authors [10] and by Olderog [21]. This approach aims to establish a correspondence between net places and sequential subterms and between net markings and terms, and it is based on both a decomposition function defined over the set of terms and a set of inference rules which generate net transitions.

Example 5.1 Consider the term $E \equiv <a, \lambda>.\underline{0} \|_{\emptyset} <b, \mu>.\underline{0} + <c, \gamma>.\underline{0}$; then $\mathcal{GSPN}[\![E]\!]$ is



Note that E has been initially decomposed into two sequential subterms due to the presence of an occurrence of the parallel composition operator; a place has been associated with each of these sequential subterms. ■

6 Describing queueing systems with MPA

In this section we shall demonstrate the expressiveness and the compositionality of MPA by describing three derived stochastic distributions and some queueing systems.

A *queueing system* is a model largely used for performance evaluation to represent a *service center* composed of a *waiting queue* and a given number of *servers*; they provide a certain *service* (following a given *discipline*) to the *customers* arriving at the service center. The customer arrival process and the customer service process are described as stochastic processes.

6.1 Poisson, hyperexponential and Erlang distributions

Suppose we want to model a Poisson arrival process with parameter $\lambda \in \mathbb{R}^+$, i.e. an arrival process such that the customer interarrival time is exponentially distributed with parameter $\lambda \in \mathbb{R}^+$. Such a process can be modeled in MPA through the term

$$Poisson \triangleq <a, \lambda>.Poisson$$

Suppose we want to model an arrival process following an hyperexponential distribution with order n and parameters $p_1, p_2, \ldots, p_n \in \mathbb{R}_{]0,1]}, \lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{R}^+$. I.e. an arrival process such that the customer interarrival time has a probability distribution function which is a convex linear combination with coefficients $p_1, p_2, \ldots, p_n \in \mathbb{R}_{]0,1]}$ of n exponential probability distribution functions with parameters $\lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{R}^+$. Such a process can be

[3]For the definition of GSPN, the reader is referred to the introductive paper [18].

modeled in MPA through the terms

$$Hexp_n \triangleq {<}a_1, \infty_{1,p_1}{>}.Exp_1 + {<}a_2, \infty_{1,p_2}{>}.Exp_2 + \ldots + {<}a_n, \infty_{1,p_n}{>}.Exp_n$$

$$Exp_i \triangleq {<}a, \lambda_i{>}.Hexp_n,\ 1 \leq i \leq n$$

Finally, suppose we want to model a service process following an Erlang distribution with order n and parameter μ, i.e. a service process such that the customer service time is the sum of n independent identically distributed random variables with parameter μ. Such a process can be modeled in MPA through the term

$$Erlang_n \triangleq {<}s_1, \mu{>}.{<}s_2, \mu{>}.\ldots.{<}s_n, \mu{>}.Erlang_n$$

6.2 Queueing system M/M/1

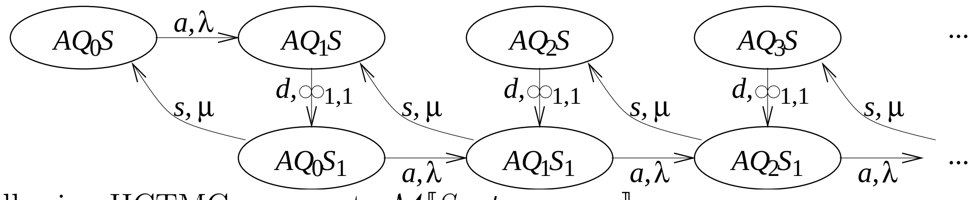
A queueing system $M/M/1$ with *arrival rate* λ and *service rate* μ is a queueing system composed of an unbounded FIFO queue and a single server such that the customer arrival process is a Poisson process with parameter λ and the customer service process is an exponential process with parameter μ (the number of customers is assumed to be unbounded).

Let a be the type of the action "a customer arrives at the queue of the service center", d be the type of the action "a customer is delivered by the queue to a server" and s be the type of the action "a customer is served by a server and then leaves the service center". Then, a queueing system $M/M/1$ can be modeled with MPA as follows:

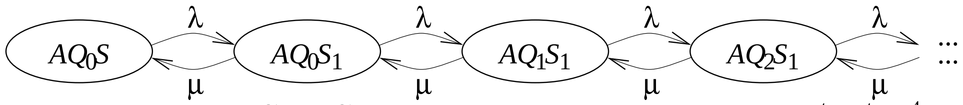
- $Arrivals \triangleq {<}a, \lambda{>}.Arrivals$;
- $Queue_0 \triangleq {<}a, 0{>}.Queue_1$,
  $Queue_i \triangleq {<}a, 0{>}.Queue_{i+1} + {<}d, 0{>}.Queue_{i-1},\ i > 0$;
- $Server \triangleq {<}d, \infty_{1,1}{>}.S_1$,
  $S_1 \triangleq {<}s, \mu{>}.Server$;
- $System_{M/M/1} \triangleq Arrivals \|_{\{a\}} (Queue_0 \|_{\{d\}} Server)$.

It is worth noting that we have described the arrival process, the queue and the server independently of each other, and that then we have composed their descriptions in order to obtain the model of the whole system. Notice also that all the actions of the queue are passive.

The following labeled transition system represents $\mathcal{I}[\![System_{M/M/1}]\!]$:

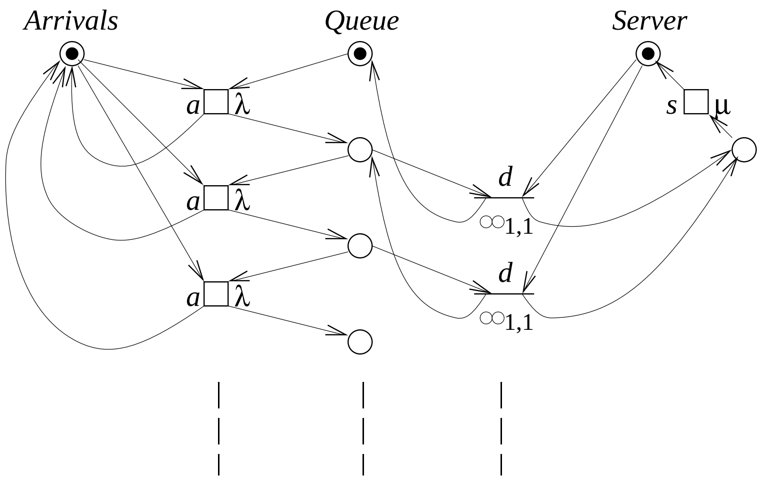


The following HCTMC represents $\mathcal{M}[\![System_{M/M/1}]\!]$:



and it is isomorphic to the HCTMC underlying a queueing system $M/M/1$. [4]

The following net represents $\mathcal{GSPN}[\![System_{M/M/1}]\!]$:



6.3 Queueing system with forks and joins

In this section we present an advanced example of modeling with MPA; it refers to a queueing system with a fork and a join. Such a system can be informally specified as follows:

- The service requests arrive at the system forming a Poisson process with parameter λ. The number of requests is assumed to be unbounded.
- Each service request arrived at the system goes through a fork which splits the request into n subrequests and sends each of them to one of the n service centers of the system.
- The service center $k, 1 \leq k \leq n$, - completely independent of the other $n-1$ service centers - is formed by an unbounded FIFO queue and a server with service rate equal to μ_k (in other words, the service center k is a queueing system $M/M/1$ with service rate μ_k).
- The n servers are linked to a join which collects the subrequests they have satisfied. Then the join puts together such satisfied subrequests taking into account which requests originated them, so as to issue satisfied requests. We assume that issuing satisfied requests takes precedence over receiving satisfied subrequests.

Let r be the type of the action "a request arrives at the fork", sr_k be the type of the action "a subrequest is sent by the fork to service center k", dsr_k be the type of the action "a subrequest is delivered by queue k to server k", ssr_k be the type of the action "a subrequest is satisfied by server k", sr'_k be the type of the action "a satisfied subrequest is sent by server k to the join", r' be the type of the action "a satisfied subrequest is issued". Then, the queueing system with fork and join previously specified can be modeled with MPA in the following way:

[4]In [3] we proved that this result holds for each queueing systems of the class M/M, thus supporting our claim that we have captured the correct markovian semantics.

- $QSFJ \triangleq Input\|_{\{r\}} System\|_{\{r'\}} Output$;

- $Input \triangleq <r, \lambda>.Input$;

- $System \triangleq Fork\|_{\{sr_k \mid 1 \le k \le n\}}(Center_1\|_{\emptyset}Center_2\|_{\emptyset} \ldots \|_{\emptyset}Center_n)\|_{\{sr'_k \mid 1 \le k \le n\}} Join$;

- $Fork \triangleq <r, 0>.<sr_1, \infty_{2,1}>.<sr_2, \infty_{2,1}>. \ldots .<sr_n, \infty_{2,1}>.Fork$;

- $Center_k \triangleq Queue_{k,0}\|_{\{dsr_k\}} Server_k$;

- $Queue_{k,0} \triangleq <sr_k, 0>.Queue_{k,1}$,
  $Queue_{k,i} \triangleq <sr_k, 0>.Queue_{k,i+1} + <dsr_k, 0>.Queue_{k,i-1}$, $i > 0$;

- $Server_k \triangleq <dsr_k, \infty_{1,1}>.<ssr_k, \mu_k>.<sr'_k, \infty_{1,1}>.Server_k$;

- $Join \triangleq J_{1,0}\|_{\{r'\}} J_{2,0}\|_{\{r'\}} \ldots \|_{\{r'\}} J_{n,0}$,
  $J_{k,0} \triangleq <sr'_k, 0>.J_{k,1}$,
  $J_{k,i} \triangleq <sr'_k, 0>.J_{k,i+1} + <r', 0>.J_{k,i-1}$, $i > 0$;

- $Output \triangleq <r', \infty_{2,1}>.Output$.

Such a system can be obviously made complex at will; for instance, the service centers could be other than queueing systems $M/M/1$ or there could be more forks or more joins. As an interesting case, consider a queueing system with fork and join where the first service center is in turn a queueing system with fork and join. This case is interesting because it shows both the usefulness of the functional abstraction operator to perform information hiding and the usefulness of the relabeling operator to obtain more concise models. This new queueing system is modeled with MPA as follows:

- $QSFJ' \triangleq Input\|_{\{r\}} System'\|_{\{r'\}} Output$;

- $System' \triangleq Fork\|_{\{sr_k \mid 1 \le k \le n\}}(System''\|_{\emptyset}Center_2\|_{\emptyset} \ldots \|_{\emptyset}Center_n)\|_{\{sr'_k \mid 1 \le k \le n\}} Join$;

- $System'' \triangleq (System/L)[\varphi]$,

where

- $L = \{sr_k \mid 1 \le k \le n\} \cup \{dsr_k \mid 1 \le k \le n\} \cup \{ssr_k \mid 1 \le k \le n\} \cup \{sr'_k \mid 1 \le k \le n\}$;

- $\varphi = \{(r, sr_1), (r', sr'_1)\} \cup I_{Act - \{r, r'\}}$.

Notice that the functional abstraction on L applied to $System$ isolates $System$ from $System'$, hence preventing actions of $System$ and $System'$ having the same type from interfering, whereas the relabeling function φ applied to $System/L$ allows the synchronization between $System$ and the fork and the join of $System'$.

7 Conclusions and related works

In this paper we have briefly introduced the stochastic process algebra MPA, illustrating its expressive power with a few examples. Furthermore, we have proposed a modeling technique – based on MPA – which integrates different views of concurrent systems (centralized vs. distributed) as well as the different aspects of their behaviour (functional vs. temporal).

As the various (transformations producing the) semantics we have proposed for MPA can be fully mechanized, future work will be devoted to implement a software tool which associates to each MPA term the collection of its three semantics. On this basis, the next step should consist of integrating this tool with the various ones, already available, tailored for specific purposes. For instance, the interleaving semantics is a labelled transition system which can be input for AUTO [24]; the Markov chain associated to a term can be given as input to SHARPE [25]; its Petri net to PAPETRI [7] and its GSPN to GreatSPN [8]. Therefore, our work opens a perspective to a fully integrated tool which can help in making computer aided – and possibly automatic –the qualitative and quantitative analysis of concurrent systems.

The development of MPA has been influenced by the stochastic process algebras MTIPP [13] and PEPA [14]. The definition of MPA tries to overcome some problems concerning expressiveness and semantics which are present in both MTIPP and PEPA. In particular, MPA is equipped with immediate actions with associated priority levels and weights, hence probabilistic choices can be naturally expressed (see term $Hexp$ in Section 6.1). Active actions and passive actions have clearly distinct roles due to the constraint on the synchronization rule. Furthermore, MPA is provided with the temporal restriction operator - by means of which the property of temporal closure can be defined - and the relabeling operator - which permits to obtain more concise models. Finally, the operational interleaving semantics and the markovian semantics of MPA have been carefully defined; finally, MPA is the first stochastic process algebra supplied with a net semantics.

The work that has led to the birth of MPA is obviously open to future developments. For instance, we would like to identify an equivalence relation over MPA terms which takes into account at the same time both the functional and the temporal aspects of the behavior of concurrent systems. Secondly, it is interesting to investigate whether the markovian semantics is compositional, after defining some suitable operators over HCTMCs in correspondence to those of MPA. Thirdly, we would like to define a net semantics which generates more compact models (e.g. by exploiting inhibitor arcs in the line of [6]). Finally, we would like to be able to model timed activities whose durations are not expressible by means of random variables following an exponential, a hyperexponential or an Erlang distribution; this would lead to the development of a semi-markovian process algebra, left for future research.

References

[1] M. Bernardo, “Verso l'Integrazione di Modelli di Concorrenza Stocastici”, Master Thesis, University of Bologna (Italy), March 1994.

[2] M. Bernardo, L. Donatiello, R. Gorrieri, “MPA: a Stochastic Process Algebra”, Technical Report 94-10, University of Bologna (Italy), 1994.

[3] M. Bernardo, L. Donatiello, R. Gorrieri, “Describing Queueing Systems with MPA”, Technical Report 94-11, University of Bologna (Italy), 1994.

[4] M. Bernardo, L. Donatiello, R. Gorrieri, “Operational GSPN semantics of MPA”, Technical Report 94-12, University of Bologna (Italy), 1994.

[5] T. Bolognesi, E. Brinksma, “Introduction to the ISO Specification Language LOTOS”, *Computer Networks and ISDN Systems* 14, 1987.

[6] N. Busi, R. Gorrieri, G. Siliprandi, “Distributed Conflicts in Communicating Systems”, Technical Report 94-8, University of Bologna (Italy), 1994.

[7] G. Berthelot, C. Johnen, L. Petrucci, “PAPETRI: Environment for the Analysis of Petri Nets”, in Proc. *Computer Aided Verification* (CAV’90), LNCS 531, Springer, 1991.

[8] G. Chiola, “GreatSPN 1.5 Software Architecture”, in Proc. of the *Fifth International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, Turin (Italy), 1991.

[9] R. Cleaveland, J. Parrow, B. Steffen, “The Concurrency Workbench: A Semantics-Based Tool for the Verification of Concurrent Systems”, in ACM *Transactions on Programming Languages and Systems* 15, 1993.

[10] P. Degano, R. De Nicola, U. Montanari, “On the Consistency of Truly Concurrent Operational and Denotational Semantics”, in Proc. of *Logics in Computer Science*, IEEE Press, Edinburgh (UK), 1988.

[11] G. Ferrari, U. Montanari, “Observing Time-Complexity of Concurrent Programs” Technical Report, University of Pisa (Italy), 1993.

[12] R. Gorrieri, M. Roccetti, “Towards Performance Evaluation in Process Algebras”, in Proc. *Algebraic Methodology and Software Technology*, Springer, Twente (NL), 1993.

[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 Tutorials of Performance Conference, LNCS 729, Springer, 1993.

[14] J. Hillston, “PEPA: Performance Enhanced Process Algebra”, Technical Report, University of Edinburgh (UK), 1993.

[15] C. A. R. Hoare, “Communicating Sequential Processes”, Prentice Hall, 1985.

[16] J. G. Kemeny, J. L. Snell, “Finite Markov Chains”, Springer, 1977.

[17] L. Kleinrock, “Queueing Systems”, Wiley, 1975.

[18] M. A. Marsan, “Stochastic Petri Nets: an Elementary Introduction”, LNCS 424, Springer, 1990.

[19] R. Milner, “Communication and Concurrency”, Prentice Hall, 1989.

[20] F. Moller, C. Tofts, “A Temporal Calculus of Communicating Systems”, LNCS 458, Springer, 1990.

[21] E. -R. Olderog, “Nets, Terms and Formulas. Three Views of Concurrent Processes and their Relationship”, Cambridge University Press, 1991.

[22] G. Plotkin, “A Structural Approach to Operational Semantics”, Tech. report of Aarhus University, 1981.

[23] G. Reed, W. Roscoe, “A Timed Model for CSP”, LNCS 226, Springer, 1986.

[24] V. Roy, R. de Simone, “Auto / Autograph”, in Proc. CAV’90, LNCS 531, Springer, 1991.

[25] R.A. Sahner, K.S. Trivedi, “Reliability Modelling Using SHARPE”, *IEEE Trans. Reliabil.* 13 (2), 186-193, 1987.