Giving a Net Semantics to Markovian Process Algebra

Marco Bernardo, Lorenzo Donatiello, Roberto Gorrieri

Università di Bologna, Dipartimento di Scienze dell'Informazione
Piazza di Porta S. Donato 5, I-40127 Bologna, Italy

Abstract

In this paper we define two compositional net semantics for the stochastic process algebra Markovian Process Algebra (MPA), based on a class of stochastic Petri nets.

The first semantics is operational in style and is defined by structural induction. We prove that both the functional and performance interleaving semantics are retrievable: given a process term E, the reachability graph and the Markov chain obtained from the net for E are the same as we obtain directly for E, according to MPA specification. Then, we present the other net semantics, which is defined in a denotational style. This additional semantics is proposed only to clarify that all the intended concurrency is expressed in the operational net model.

Finally, we comment on an integrated specification strategy for concurrent systems, which is available also because of the operational net semantics described here.

1 Introduction

Process algebras [15, 11, 4] constitute one of the most important tools for modeling and analyzing concurrent systems. In recent years, a remarkable research effort has been made to tackle and solve an important problem of process algebras: their inability to express performance aspects of concurrent systems. One of the most promising solutions to this problem consists of considering actions as durational: if durations are expressed by means of random variables, then we have *stochastic process algebras* [9, 10, 2, 5].

In this paper we consider *Markovian Process Algebra (MPA)* [3]. Each of its terms is equipped with two semantic interpretations: the labeled transition system $\mathcal{I}[\![E]\!]$ is the operational interleaving model of term E and represents the behavior of E, whilst the Markov chain $\mathcal{M}[\![E]\!]$ is the stochastic model of E and describes the performance aspect of E. If we restrict ourselves to the operational interleaving semantics, we see that it is not a truly concurrent semantics. In fact, it is not possible to distinguish the parallel execution of, e.g., two actions a and b from the alternative execution of their sequentializations ab and ba: in process algebraic terms, $a|b$ is equivalent to $ab+ba$. From this point of view, a more satisfactory semantics can be obtained if we map terms to Petri nets [18], because their states are distributed and it is then possible to express that some part of the process is completely independent of the rest of the process.

The main purpose of this paper is to introduce a net semantics for MPA in operational style and to prove that it is consistent with the original interleaving semantics, from both a functional and a performance point of view. By resorting to a suitable extension of the structured operational semantics approach [17], an operational net semantics for a process algebra can be defined. The idea, due to Degano-De Nicola-Montanari [8] and Olderog [16], consists of associating with each term E a place/transition net such that: places correspond to the sequential subterms of E and its derivatives, transitions are defined by induction on the syntactical structure of the sets of sequential subterms, and markings correspond roughly to E and its derivatives. Here we want to extend such an idea to MPA; since it is a stochastic process algebra, its terms will be translated into stochastic Petri nets.

The reason why we are interested in this net description is for integration purposes. In fact, with a stochastic process algebra like MPA we can exploit a *compositional* modeling technique for concurrent systems which integrates their different views (*centralized* vs. *distributed* – shown in this paper by the double representation as a labeled transition system and as a Petri net), as well as different aspects of their behavior (*qualitative-functional* vs. *quantitative-performance* – shown in stochastic process algebras and in stochastic Petri nets by the double representation as a functional model and as a performance model). Such a modeling technique, described in Section 9, is at the base of a software tool, we are designing, for a fully integrated computer aided – and possibly automatic – qualitative and quantitative analysis of concurrent systems.

This paper is organized as follows. In Section 2 some notions concerning MPA are recalled. In Section 3 a specific class of stochastic Petri nets is introduced. In Section 4 an operational net semantics is defined which translates terms into elements of the class above. The adequacy of this semantics is assessed in Sections 5 and 6 by verifying that it satisfies the *functional retrievability principle* and the *stochastic retrievability principle*. Then, in Section 7 we propose another net semantics, defined in a denotational style, in order to show that the operational net semantics meets the *concurrency principle*. In Section 8 a simple example is presented describing a queueing system. Finally, in Section 9 a few concluding remarks are reported.

2 MPA: a brief overview

In this section we introduce the syntax of terms and we informally describe the semantics of operators. For more details, the reader is invited to consult [3]. [1]

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 action is described by a pair $<a, \tilde{\lambda}>$ consisting of the *type* of the action and the *rate* of the action, respectively. Depending on the type, actions are subdivided into:

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

Depending on the rate, 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, in turn 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 w associated with it.

We denote with AType the set of action types, with $\mathit{ARate} = \{0\} \cup \mathbb{R}^+ \cup \mathit{Inf}$, $\mathit{Inf} = \{\infty_{l,w} \mid l \in \mathbb{N}^+ \wedge w \in \mathbb{R}^+\}$, the set of action rates, and with $\mathit{Act} = \mathit{AType} \times \mathit{ARate}$ the set of actions.

2.2 Syntax and informal semantics

Let Con be a set of *constants*, and $\Phi = \{\varphi : \mathit{AType} \longrightarrow \mathit{AType} \mid \varphi(\tau) = \tau \wedge \varphi(\mathit{AType} - \{\tau\}) \subseteq \mathit{AType} - \{\tau\}\}$ be a set of *relabeling functions*.

Definition 2.1 The set $\mathcal{L}$ of MPA processes is defined as the set of *process 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 $L, S \subseteq \mathit{AType} - \{\tau\}$, $H \subseteq \mathit{AType}$, $A \in \mathit{Con}$. ■

[1] This technical report is available via `ftp` from the area `ftp.cs.unibo.it:/pub/TR/UBLCS` in compressed PostScript format, or via `www` with the URL `http://www.cs.unibo.it`.

The *null term* "$\underline{0}$" is a nullary operator representing a term which can execute no action.
The *prefix operator* "$<a, \tilde{\lambda}>.\_$" expresses the sequential composition of an action and a term.
The *functional abstraction operator* "$\_/L$" expresses the abstraction from the type of actions whose type is in L, i.e. the action type is turned to τ.
The *temporal restriction operator* "$\_\backslash H$" prevents the execution of passive actions whose type is in H. Based on this operator, we define the notion of *temporal closure*: a term is temporally closed if it cannot execute passive actions. Thus, the temporal closure property singles out terms which can model real systems because the durations of activities performed by these systems are completely specified.
The *relabeling operator* "$\_[\varphi]$" changes the type of the actions (executed by the term to which it is applied) according to φ.
The *alternative composition operator* "$\_+\_$" expresses a choice between two terms which has a probabilistic nature, as based on the race policy (see Section 2.3).
The *parallel composition operator* "$\_\|_S\_$" expresses the parallel execution of two terms: $E_1\|_S E_2$, 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 $<a, \infty_{l,w}>$ can be synchronized with action $<a, 0>$, hence modeling the temporal closure of passive actions.
- 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 $\min(\tilde{\lambda}, \tilde{\mu}) > 0$, 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.

Finally, MPA is equipped with constants. Each constant A is used as a shorthand for a term E through its *defining equation* $A \stackrel{\Delta}{=} E$: its meaning is that constant A behaves as E. Constants can be used for defining recursive terms; the correctness of such recursive definitions can be assessed by the notion of *guarded closure* [3]. In the following we consider only the set $\mathcal{G}$ of guardedly closed terms in $\mathcal{L}$.

2.3 Race policy

Since several active actions may be simultaneously executable, and since in an interleaving model only one action at a time can be done, it is necessary to choose a policy determining which of them is to be executed. In this framework we adopt the *race policy*, which chooses the active action having the least duration. As a consequence, immediate actions take precedence over timed actions. Furthermore, since each immediate action is equipped with a priority level and a weight, when several immediate actions are simultaneously executable only those having the highest priority level are actually executable: the choice among them is probabilistically made by giving each of them an execution probability proportional to its weight.

We conclude by mentioning a problem common to all the stochastic process algebras. In the case of a classical process algebra, $a.E + a.E$ can only perform a thus becoming E, so it is equivalent to $a.E$. In the case of a stochastic process algebra such as MPA, given $<a, \lambda>.E + <a, \lambda>.E$ we must remember that there are two executable actions because the race policy has been adopted and therefore the exit rate from this term is not λ but 2λ [3]. To obtain this, we take into account the multiplicity of action executions by means of the formalism of multisets. Some relevant definitions about them are recalled below.

Definition 2.2 Given a set S, a *multiset* over S is a function $M : S \longrightarrow \mathbb{N}$, and a *finite multiset* over S is a function $M : S \longrightarrow \mathbb{N}$ such that the set $dom(M) = \{s \in S \mid M(s) \neq 0\}$ is finite. The value $M(s)$ is called the *multiplicity* of element s. We denote with $\mathcal{M}u(S)$ the set of all the multisets over S, with $\mathcal{M}u_{fin}(S)$ the set of all the finite multisets over S, and with $\mathcal{P}_{fin}(S)$ the set of all the finite sets over S. [2] ■

Definition 2.3 Given a set S, let $M_1, M_2, M \in \mathcal{M}u(S)$ and $M' \in \mathcal{M}u(S \times S)$.

- $s \in M \Longleftrightarrow M(s) > 0$;
- $M_1 \subseteq M_2 \Longleftrightarrow \forall s \in S.\, M_1(s) \leq M_2(s)$.
- $M = M_1 \oplus M_2 \Longleftrightarrow \forall s \in S.\, M(s) = M_1(s) + M_2(s)$.
- $M = M_1 - M_2 \Longleftrightarrow \forall s \in S.\, M(s) = \max(M_1(s) - M_2(s), 0)$;
- $M' = M_1 \otimes M_2 \Longleftrightarrow \forall (s_1, s_2) \in S \times S.\ M'(s_1, s_2) = M_1(s_1) \cdot M_2(s_2)$. ■

2.4 Operational interleaving semantics

In this section we present the semantics in a formal way by resorting to the *structured operational semantics (SOS) approach* [17]. Since this approach generates labeled transition systems, we firstly recall some notions about them.

Definition 2.4 A *labeled transition system (LTS)* is a quadruple
$$(S, U, \longrightarrow, s_0)$$
such that:

- S is a set whose elements are called *states*;
- U is a set whose elements are called *labels*;
- $\longrightarrow\ \in \mathcal{M}u(S \times U \times S)$ is called *transition (multi)relation*;
- $s_0 \in S$ is called the *initial state*. ■

[2] We use "$\{|$" and "$|\}$" as brackets for multisets, and "$\emptyset$" to denote the empty multiset.

Definition 2.5 Let $A_1 = (S_1, U, \longrightarrow_1, s_{01})$ and $A_2 = (S_2, U, \longrightarrow_2, s_{02})$ be two LTSs.

- A_1 is *strongly bisimilar* to A_2 if and only if there exists a relation $\mathcal{B} \subseteq S_1 \times S_2$ such that:
  - $(s_{01}, s_{02}) \in \mathcal{B}$;
  - $\forall (s_1, s_2) \in \mathcal{B}.\, \forall u \in U.$
    - $\forall s_1' \in S_1.\, s_1 \xrightarrow{u}_1 s_1' \Longrightarrow \exists s_2' \in S_2.\, s_2 \xrightarrow{u}_2 s_2' \wedge (s_1', s_2') \in \mathcal{B}$;
    - $\forall s_2' \in S_2.\, s_2 \xrightarrow{u}_2 s_2' \Longrightarrow \exists s_1' \in S_1.\, s_1 \xrightarrow{u}_1 s_1' \wedge (s_1', s_2') \in \mathcal{B}$.
- A_1 is *isomorphic* to A_2 if and only if there exists a bijection $\beta : S_1 \longrightarrow S_2$ such that:
  - $\beta(s_{01}) = s_{02}$;
  - $\forall s, s' \in S_1.\, \forall u \in U.$
    $\longrightarrow_1(s, u, s') = \longrightarrow_2(\beta(s), u, \beta(s'))$. ■

The application of the SOS approach to MPA results in the LTS whose set of states is $\mathcal{G}$, whose set of labels is Act and whose transition relation $\longrightarrow$ is the least multiset over $\mathcal{G} \times Act \times \mathcal{G}$ generated by the axioms and the inference rules reported in Table 1.

Definition 2.6 The *operational interleaving semantics* of a term $E \in \mathcal{G}$ is the LTS
$$\mathcal{I}[\![E]\!] = (\uparrow E, Act, \longrightarrow_E, E)$$
where:

- $\uparrow E$ is the least subset of $\mathcal{G}$ such that:
  - $E \in \uparrow E$;
  - if $E_1 \in \uparrow E$ and $E_1 \xrightarrow{a, \tilde{\lambda}} E_2$, then $E_2 \in \uparrow E$;
- $\longrightarrow_E$ is the restriction of $\longrightarrow$ to $\uparrow E$. ■

The SOS rules reported in Table 1 have been carefully designed for two reasons. The first one is that they must take into account the priority of immediate actions over timed actions as well as the different priority levels existing among immediate actions. As a consequence, all the active transitions exiting from a given state must have the same priority level.
This has been enforced in the SOS rules for the alternative and the parallel composition operators by predicate *highest priority level*: $HPL(<a, \tilde{\lambda}>, E)$ holds true, at the top level of the induction, whenever $<a, \tilde{\lambda}>$ either is a passive action or has the highest priority level among the actions in the multiset $EAct(E)$ of the actions executable by E (see [3] for more details).

$$<a, \tilde{\lambda}>.E \xrightarrow{a,\tilde{\lambda}} E$$

$$\frac{E \xrightarrow{a,\tilde{\lambda}} E'}{E/L \xrightarrow{a,\tilde{\lambda}} E'/L} \ [a \notin L] \qquad \frac{E \xrightarrow{a,\tilde{\lambda}} E'}{E/L \xrightarrow{\tau,\tilde{\lambda}} E'/L} \ [a \in L]$$

$$\frac{E \xrightarrow{a,\tilde{\lambda}} E'}{E \backslash H \xrightarrow{a,\tilde{\lambda}} E' \backslash H} \ [\neg(a \in H \wedge \tilde{\lambda} = 0)]$$

$$\frac{E \xrightarrow{a,\tilde{\lambda}} E'}{E[\varphi] \xrightarrow{\varphi(a),\tilde{\lambda}} E'[\varphi]}$$

$$\frac{E_1 \xrightarrow{a,\tilde{\lambda}} E}{E_1 + E_2 \xrightarrow{a,\tilde{\lambda}} E} \ [HPL(<a, \tilde{\lambda}>, E_1 + E_2)] \qquad \frac{E_2 \xrightarrow{a,\tilde{\lambda}} E}{E_1 + E_2 \xrightarrow{a,\tilde{\lambda}} E} \ [HPL(<a, \tilde{\lambda}>, E_1 + E_2)]$$

$$\frac{E_1 \xrightarrow{a,\tilde{\lambda}} E_1'}{E_1 \|_S E_2 \xrightarrow{a,\tilde{\lambda}} E_1' \|_S E_2} \ [a \notin S \wedge HPL(<a, \tilde{\lambda}>, E_1 \|_S E_2)] \qquad \frac{E_2 \xrightarrow{a,\tilde{\lambda}} E_2'}{E_1 \|_S E_2 \xrightarrow{a,\tilde{\lambda}} E_1 \|_S E_2'} \ [a \notin S \wedge HPL(<a, \tilde{\lambda}>, E_1 \|_S E_2)]$$

$$\frac{E_1 \xrightarrow{a,\tilde{\lambda}} E_1' \quad E_2 \xrightarrow{a,\tilde{\mu}} E_2'}{E_1 \|_S E_2 \xrightarrow{a,\tilde{\gamma}} E_1' \|_S E_2'} \ [a \in S \wedge \min(\tilde{\lambda}, \tilde{\mu}) = 0 \wedge \tilde{\gamma} = \max(\tilde{\lambda}, \tilde{\mu}) / \mathcal{N}(a, \tilde{\lambda}, \tilde{\mu}, E_1, E_2) \wedge HPL(<a, \tilde{\gamma}>, E_1 \|_S E_2)]$$

$$\frac{E \xrightarrow{a,\tilde{\lambda}} E'}{A \xrightarrow{a,\tilde{\lambda}} E'} \ [A \stackrel{\Delta}{=} E]$$

Table 1: SOS rules for MPA interleaving semantics

The second reason is that the SOS rules must normalize the rates of identically labeled transitions exiting from the same state and deriving from the synchronization of the same timed action with several passive actions which are either independent of each other (i.e. composed in parallel with a synchronization set not containing the action type at hand), or mutually exclusive (i.e. composed in alternative) [3].
This has been enforced in the SOS rules for the parallel composition operator by means of function *normalize*: if $\min(\tilde{\lambda}, \tilde{\mu}) = 0$, then $\mathcal{N}(a, \tilde{\lambda}, \tilde{\mu}, E_1, E_2)$ is equal to either 1 if $\max(\tilde{\lambda}, \tilde{\mu}) \notin \mathbb{R}^+ \cup Inf$, or $EAct(E_2)(<a, 0>)$ if $\tilde{\lambda} \in \mathbb{R}^+ \cup Inf$, or $EAct(E_1)(<a, 0>)$ if $\tilde{\mu} \in \mathbb{R}^+ \cup Inf$.

Based on the operational interleaving semantics, we formally define the notion of temporal closure.

Definition 2.7 A term $E \in \mathcal{G}$ is said to be *temporally closed* if and only if $\mathcal{I}[\![E]\!]$ is isomorphic to $\mathcal{I}[\![E \backslash AType]\!]$. ■

We denote with $\mathcal{T}$ the set of terms in $\mathcal{L}$ that are temporally closed. We 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 denote with $\mathcal{E}'$ the set of terms in $\mathcal{E}$ whose operational interleaving semantics does not contain cycles of immediate transitions.

2.5 Markovian semantics

The *Markovian semantics* of a term $E \in \mathcal{E}'$ is the homogeneous continous time Markov chain (HCTMC) [12] denoted with $\mathcal{M}[\![E]\!]$, obtained by applying to $\mathcal{I}[\![E]\!]$ an algorithm organized in three phases [3]. The first phase eliminates all the immediate transitions occurring in $\mathcal{I}[\![E]\!]$. The second phase produces a HCTMC by imposing that between each ordered pair of states there is at most one transition. The third phase detects and merges states which are equivalent according to the notion of lumping [12].

3 Stochastic Petri nets

In this paper we shall be concerned with the class of the *generalized stochastic Petri nets (GSPNs)* [14], which are essentially place/transition nets [18] equipped with inhibitor arcs whose transitions are either timed (i.e. their durations are expressed by means of exponentially distributed random variables) or immediate (i.e. their durations are zero, so they take precedence over timed ones). Furthermore, GSPN transitions are subdivided into priority levels and have weights which can depend on the current marking. The race policy is adopted whenever several transitions are simultaneously executable. [3]

Since GSPNs do not admit passive transitions, we propose a new class of nets by introducing three restrictions (1-safeness, absence of inhibitor arcs, weight functions independent of the current marking) and one

[3]We assume the reader familiar with the formalism of GSPNs.

extension (presence of passive transitions not involved in the priority mechanism).

Definition 3.1 A *passive generalized stochastic Petri net (PGSPN)* is a tuple
$$(P, U, T, M_0, L, W)$$
such that:

- P is a set whose elements are called *places*;
- U is a set whose elements are called *labels*;
- $T \in \mathcal{M}u(\mathcal{M}u_{fin}(P) \times U \times \mathcal{M}u_{fin}(P))$ whose elements are called *(multi)transitions*;
- $M_0 \in \mathcal{M}u_{fin}(P)$ is called the *initial marking*;
- (P, U, T, M_0) is 1-safe;
- $L : T \twoheadrightarrow \mathbb{N}$ is called *priority function* and is such that:
  - $L(t) = 0$ if t is timed;
  - $L(t) \in \mathbb{N}^+$ if t is immediate;
  - $L(t)$ is undefined if t is passive;
- $W : T \twoheadrightarrow \mathbb{R}^+$ is called *weight function* and is such that:
  - $W(t)$ is the rate of the exponential distribution associated with t if $L(t) = 0$;
  - $W(t)$ is the weight of t if $L(t) \in \mathbb{N}^+$;
  - $W(t)$ is undefined if $L(t)$ is undefined. ■

As usual, we denote with "$[\rangle$" the transition firing relation and with $R(M)$ the set of markings reachable from marking M by transition firing. As a consequence, the *reachability graph* of N is defined as the LTS $\mathcal{RG}[\![N]\!] = (R(M_0), U, [\rangle, M_0)$.

4 Operational net semantics for MPA

The net semantics for MPA associates a PGSPN with every term and is given in operational style by defining its set of transitions by means of an inference system. In the next sections we assess its merits based on the following three principles:

- *Functional retrievability principle* [16]: the interleaving semantics of each term should be retrievable from its net semantics. Such a principle is usually formalized by requiring that, for each term, the LTS representing the operational interleaving semantics of the term is strongly bisimilar (or isomorphic) to the LTS representing the reachability graph of the net semantics of the term.
- *Stochastic retrievability principle*: the stochastic semantics of each term should be retrievable from its net semantics. Such a principle can be formalized by requiring that, for each term, the HCTMC representing the Markovian semantics of the term is lumping equivalent (i.e. isomorphic after lumping) to the HCTMC underlying the net semantics of the term.
- *Concurrency principle* [16]: the intended concurrency of each term should be represented by its net semantics. Such a principle is usually formalized by requiring that the operational net semantics coincides with a denotational net semantics, assumed as a reference point.

The first step in the definition of the operational net semantics consists of establishing a correspondence between net places and sequential subterms, thus inducing a correspondence between net markings and terms. We define the set $\mathcal{V}$ of places as the set of terms V generated as follows
$$V ::= \underline{0} \mid <a, \tilde{\lambda}>.E \mid V/L \mid V\backslash H \mid V[\varphi] \mid V+V \mid V\|_S id \mid id\|_S V \mid A$$
and we define the *decomposition function*
$$dec : \mathcal{G} \longrightarrow \mathcal{M}u_{fin}(\mathcal{V})$$
by induction on the syntactical structure of the terms in $\mathcal{G}$ in the following way:

- $dec(\underline{0}) = \{\!|\, \underline{0} \,|\!\}$;
- $dec(<a, \tilde{\lambda}>.E) = \{\!|\, <a, \tilde{\lambda}>.E \,|\!\}$;
- $dec(E/L) = \{\!|\, V/L \mid V \in dec(E) \,|\!\}$;
- $dec(E\backslash H) = \{\!|\, V\backslash H \mid V \in dec(E) \,|\!\}$;
- $dec(E[\varphi]) = \{\!|\, V[\varphi] \mid V \in dec(E) \,|\!\}$;
- $dec(E_1+E_2) = \{\!|\, V_1+V_2 \mid V_1 \in dec(E_1) \wedge V_2 \in dec(E_2) \,|\!\}$;
- $dec(E_1\|_S E_2) = \{\!|\, V\|_S id \mid V \in dec(E_1) \,|\!\} \oplus \{\!|\, id\|_S V \mid V \in dec(E_2) \,|\!\}$;
- $dec(A) = dec(E)$ if $A \triangleq E$,

where $Q \in \mathcal{M}u_{fin}(\mathcal{V})$ is said to be *complete* if and only if there exists $E \in \mathcal{G}$ such that $dec(E) = Q$. We use $V, V', V'', \ldots$ as metavariables for $\mathcal{V}$, $Q, Q', Q'', \ldots$ for $\mathcal{M}u_{fin}(\mathcal{V})$, $R, R', R'', \ldots$ for complete elements of $\mathcal{M}u_{fin}(\mathcal{V})$.

The decomposition function is well defined because we consider only guardedly closed terms. We also would like to point out that, in order to syntactically express the decomposition into sequential subterms, the binary operator "$\_\|_S\_$" has been replaced by the two unary operators "$\_\|_S id$" and "$id\|_S\_$".

Example 4.1 Given term $E \equiv (<a, \lambda>.\underline{0}\|_\emptyset<b, \mu>.\underline{0}) +<c, \gamma>.\underline{0}$, we have that $dec(E) = \{\!|\, (<a, \lambda>.\underline{0}\|_\emptyset id)+ <c, \gamma>.\underline{0}, (id\|_\emptyset<b, \mu>.\underline{0})+<c, \gamma>.\underline{0} \,|\!\}$. ■

The second step consists of introducing an appropriate inference system by means of which net transitions will be constructed. We thus define the relation $\longrightarrow$ as the least multiset over $\mathcal{M}u_{fin}(\mathcal{V}) \times Act \times \mathcal{M}u_{fin}(\mathcal{V})$ generated by the axioms and the inference rules reported in Table 2.

These rules are strictly related to the rules reported in Table 1 for the operational interlaving semantics of MPA terms. The main difference is that here predicate

$$\{<a,\tilde{\lambda}>.E\} \xrightarrow{a,\tilde{\lambda}} dec(E)$$

$$\frac{Q \xrightarrow{a,\tilde{\lambda}} Q'}{Q/L \xrightarrow{a,\tilde{\lambda}} Q'/L}\ [a \notin L] \qquad \frac{Q \xrightarrow{a,\tilde{\lambda}} Q'}{Q/L \xrightarrow{\tau,\tilde{\lambda}} Q'/L}\ [a \in L]$$

$$\frac{Q \xrightarrow{a,\tilde{\lambda}} Q'}{Q\backslash H \xrightarrow{a,\tilde{\lambda}} Q'\backslash H}\ [\neg(a \in H \wedge \tilde{\lambda} = 0)]$$

$$\frac{Q \xrightarrow{a,\tilde{\lambda}} Q'}{Q[\varphi] \xrightarrow{\varphi(a),\tilde{\lambda}} Q'[\varphi]}$$

$$\frac{Q_1 \cup Q_2 \xrightarrow{a,\tilde{\lambda}} Q}{Q_1 \cup (Q_2 + R) \xrightarrow{a,\tilde{\lambda}} Q}\ [Q_1 \cap Q_2 = \emptyset] \qquad \frac{Q_1 \cup Q_2 \xrightarrow{a,\tilde{\lambda}} Q}{Q_1 \cup (R + Q_2) \xrightarrow{a,\tilde{\lambda}} Q}\ [Q_1 \cap Q_2 = \emptyset]$$

$$\frac{Q \xrightarrow{a,\tilde{\lambda}} Q'}{Q \|_S id \xrightarrow{a,\tilde{\lambda}} Q' \|_S id}\ [a \notin S] \qquad \frac{Q \xrightarrow{a,\tilde{\lambda}} Q'}{id \|_S Q \xrightarrow{a,\tilde{\lambda}} id \|_S Q'}\ [a \notin S]$$

$$\frac{Q_1 \xrightarrow{a,\tilde{\lambda}} Q_1' \quad Q_2 \xrightarrow{a,\tilde{\mu}} Q_2'}{Q_1 \|_S id \cup id \|_S Q_2 \xrightarrow{a,\tilde{\gamma}} Q_1' \|_S id \cup id \|_S Q_2'}\ [a \in S \wedge \min(\tilde{\lambda},\tilde{\mu}) = 0 \wedge \tilde{\gamma} = \max(\tilde{\lambda},\tilde{\mu})/\mathcal{N}(a,\tilde{\lambda},\tilde{\mu},Q_1,Q_2)]$$

Table 2: SOS rules for MPA net semantics

HPL is useless since the priority among transitions is inherently enforced by the firing rule.
Only the rules for the alternative composition operator need to be explained. In particular, it is necessary to realize that only one part of the sequential subterms (i.e. Q_2) needs to have an alternative and that such an alternative (i.e. R) must be a complete set of sequential subterms. The completeness of R guarantees that none of its sequential subterms has been previously active, hence Q_2 (which is the alternative of R) has not been discarded yet due to an action previously executed by a sequential subterm of R.

Example 4.2 If we consider term E defined in Example 4.1 together with its decomposition, and we assume that action $<a,\lambda>$ is executed first, we obtain marking $\{|\,\underline{0}\|_\emptyset id, (id\|_\emptyset <b,\mu>.\underline{0}) + <c,\gamma>.\underline{0}\,|\}$ where action $<c,\gamma>$ has already been discarded. In fact, its alternative $id\|_\emptyset<b,\mu>.\underline{0}$ is not complete. ■

The third step consists of associating with each term an appropriate PGSPN by exploiting the previous two steps.

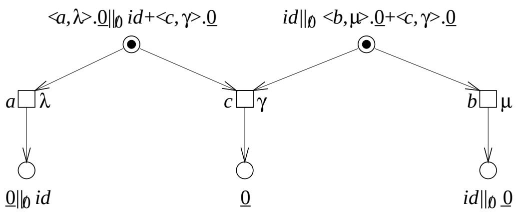
Definition 4.3 The *operational net semantics* of a term $E \in \mathcal{G}$ is the PGSPN

$$\mathcal{PGSPN}[\![E]\!] = (P, U, T, M_0, L, W)$$

where:

- P is the least subset of $\mathcal{V}$ such that:
  - $dom(dec(E)) \subseteq P$;
  - if $dom(Q_1) \subseteq P$ and $Q_1 \xrightarrow{a,\tilde{\lambda}} Q_2$, then $dom(Q_2) \subseteq P$;
- $U = Act$;
- T is the restriction of $\longrightarrow$ to $\mathcal{M}u_{fin}(P)$;
- $M_0 = dec(E)$;
- $L : T \twoheadrightarrow \mathbb{N}$ such that:
  - $L(Q_1, <a,\lambda>, Q_2) = 0$;
  - $L(Q_1, <a,\infty_{l,w}>, Q_2) = l$;
  - $L(Q_1, <a,0>, Q_2)$ is undefined;
- $W : T \twoheadrightarrow \mathbb{R}^+$ such that:
  - $W(Q_1, <a,\lambda>, Q_2) = \lambda$;
  - $W(Q_1, <a,\infty_{l,w}>, Q_2) = w$;
  - $W(Q_1, <a,0>, Q_2)$ is undefined. ■

Example 4.4 If we take term E introduced in Example 4.1, we have that $\mathcal{PGSPN}[\![E]\!]$ is as follows:



■

Now we show two properties of the operational net semantics; the first one follows from the previous construction, the second one can be easily demonstrated with a proof similar to that provided in [16].

Theorem 4.5 Let $E \in \mathcal{G}$. It turns out that:

(*i*) $\mathcal{PGSPN}[\![E]\!]$ is a PGSPN.

(*ii*) $\mathcal{PGSPN}[\![E]\!]$ is finite if each subterm of E of the form $E'/L, E'\backslash H, E'[\varphi], E_1\|_S E_2$ is without constants. ∎

It is interesting to identify a class of terms in $\mathcal{G}$ such that for each term E in this class it turns out that $\mathcal{PGSPN}[\![E]\!]$ is a GSPN; as we can expect, the above class is given by $\mathcal{E}$ and this will be proved later.

5 Functional retrievability principle

In this section we prove that the operational net semantics satisfies the functional retrievability principle. This guarantees that, for each term, the system represented by the term has the same functional behavior as the system represented by the net semantics of the term.

Theorem 5.1 For each $E \in \mathcal{G}$ it turns out that $\mathcal{RG}[\![\mathcal{PGSPN}[\![E]\!]]\!]$ is isomorphic to $\mathcal{I}[\![E]\!]$.

Proof The proof proceeds in three steps.

(1st step) Suppose that the active transitions of PGSPNs are not subdivided into different priority levels, and let $\mathcal{RG}'$ denote the reachability graph resulting from this assumption. Furthermore, suppose that predicate HPL is ignored when applying the rules defining the operational interleaving semantics for MPA, and let $\mathcal{I}'$ denote the LTS resulting from this assumption. Then we can demonstrate, by following the proof developed in [16] Theorem 3.7.18, that $\mathcal{RG}'[\![\mathcal{PGSPN}[\![E]\!]]\!]$ is strongly bisimilar to $\mathcal{I}'[\![E]\!]$ through the binary relation $\mathcal{B} = \{(F, Q) \in \uparrow' E \times R'(\mathit{dec}(E)) \mid Q\; \mathit{swf} \wedge \mathit{dec}(F) = \mathit{upd}(Q)\}$ where:

- The definition of *strongly well formed (swf)* marking is the following:
  - $\{|\,\underline{0}\,|\}$ and $\{|\,<a,\tilde{\lambda}>.E\,|\}$ are swf:
  - if Q is swf then so are Q/L, $Q\backslash H$ and $Q[\varphi]$;
  - if $Q_1 \cup Q_2$ is swf such that $Q_1 \cap Q_2 = \emptyset$ and either $Q_1 = \emptyset$ or not all components in Q_1 contain "+" as their topmost operator, then $Q_1 \cup (Q_2 + R)$ and $Q_1 \cup (R + Q_2)$ are swf;
  - if Q_1 and Q_2 are swf then so is $Q_1\|_S \mathit{id} \cup \mathit{id}\|_S Q_2$.

  This property is satisfied by complete elements of $\mathcal{M}u_{fin}(\mathcal{V})$ and is invariant for transition firing.

- The definition of the *update operation (upd)* on swf markings is the following:
  - if Q is complete then $\mathit{upd}(Q) = Q$;
  - if $Q \equiv Q'/L$ is incomplete then $\mathit{upd}(Q) = \mathit{upd}(Q')/L$;
  - if $Q \equiv Q'\backslash H$ is incomplete then $\mathit{upd}(Q) = \mathit{upd}(Q')\backslash H$;
  - if $Q \equiv Q'[\varphi]$ is incomplete then $\mathit{upd}(Q) = \mathit{upd}(Q')[\varphi]$;
  - if $Q \equiv Q_1 \cup (Q_2 + R)$ or $Q \equiv Q_1 \cup (R + Q_2)$ is incomplete then $\mathit{upd}(Q) = \mathit{upd}(Q_1 \cup Q_2)$;
  - if $Q \equiv Q_1\|_S \mathit{id} \cup \mathit{id}\|_S Q_2$ is incomplete then $\mathit{upd}(Q) = \mathit{upd}(Q_1)\|_S \mathit{id} \cup \mathit{id}\|_S \mathit{upd}(Q_2)$.

  For each swf marking Q, it turns out that $\mathit{upd}(Q)$ is complete.

(2nd step) Now we want to prove, under the same conditions assumed at the beginning of the previous step, that $\mathcal{B}$ is an isomorphism between $\mathcal{RG}'[\![\mathcal{PGSPN}[\![E]\!]]\!]$ and $\mathcal{I}'[\![E]\!]$.

Firstly, we have to prove that $\mathcal{B}$ is a function. Given $F \in \uparrow' E$, since F is reachable from E and $\mathcal{B}$ is a strong bisimulation, there must exist $Q \in R'(\mathit{dec}(E))$ such that $(F, Q) \in \mathcal{B}$, i.e. $\mathit{dec}(F) = \mathit{upd}(Q)$. It remains to prove the uniqueness of such a swf reachable marking Q. Suppose that there exist $Q_1, Q_2 \in R'(\mathit{dec}(E))$ swf and different from each other such that $\mathit{upd}(Q_1) = \mathit{upd}(Q_2) = \mathit{dec}(F)$. This can stem only from the fact that there exists at least a pair composed of a subterm G of a place V_1 in Q_1 and a subterm $G + G'$ of a place V_2 in Q_2 which reside in the same position of the syntactical structure of V_1 and V_2 (if such a pair did not exist, Q_1 and Q_2 could not be different from each other). The existence of this pair contradicts the reachability of Q_1. In fact, we recall that the decomposition function dec distributes all the "+" operators between all the appropriate places and when one of these places is part of a marking involved in a transition firing, either it remains unchanged or it gives rise to a new place where the "+" operator disappears and only the alternative involved remains after it has been transformed (see the rules of $\longrightarrow$ for the alternative composition operator).

Secondly, we have to prove that $\mathcal{B}$ is injective. This is trivial, because if there exist $F_1, F_2 \in \uparrow' E$ and $Q \in R'(\mathit{dec}(E))$ such that $\mathit{dec}(F_1) = \mathit{upd}(Q) = \mathit{dec}(F_2)$, then necessarily $F_1 = F_2$.

Thirdly, we have to prove that $\mathcal{B}$ is surjective. This is true because given $Q \in R'(\mathit{dec}(E))$, since Q is reachable from $\mathit{dec}(E)$ and $\mathcal{B}$ is a strong bisimulation, there must exist $F \in \uparrow' E$ such that $(F, Q) \in \mathcal{B}$.

Finally, we have to prove that $\mathcal{B}$ satisfies the isomorphism clauses. This follows immediately from the fact that $\mathcal{B}$ is a bijection fulfilling the bisimilarity clauses.

(3rd step) Now let us take into account both the different priority levels into which the active transitions of PGSPNs are subdivided, and predicate HPL. Since the priority mechanism for MPA actions is exactly the same as the priority mechanism for PGSPN transitions, from the previous step it follows that $\mathcal{RG}[\![\mathcal{PGSPN}[\![E]\!]]\!]$ is isomorphic to $\mathcal{I}[\![E]\!]$. ∎

This theorem simply states that the operational net semantics is sound with respect to the operational interleaving semantics. The usefulness of this net semantics can be appreciated by observing that some properties of systems (e.g. partial deadlock) can be easily checked only in a distributed model like Petri nets.

Corollary 5.2 Let $E \in \mathcal{G}$. Then $\mathcal{PGSPN}[\![E]\!]$ is a GSPN if and only if $E \in \mathcal{E}$.

This corollary is a straightforward consequence of Theorem 5.1 [3]. Whenever $E \in \mathcal{E}$, we denote $\mathcal{PGSPN}[\![E]\!]$ with $\mathcal{GSPN}[\![E]\!]$.

6 Stochastic retrievability principle

In this section we show that the operational net semantics satisfies the stochastic retrievability principle. This guarantees that, for each term, the system represented by the term has the same performance behavior as the system represented by the net semantics of the term. As a consequence, the principles of functional and stochastic retrievability together assure that a term and its net semantics model exactly the same system, hence the functional and performance properties of such a system can be studied by means of one of the two models at will.

Theorem 6.1 For each $E \in \mathcal{E}'$ it turns out that the HCTMC underlying $\mathcal{GSPN}[\![E]\!]$ is lumping equivalent to $\mathcal{M}[\![E]\!]$. ■

This theorem (which is a straightforward consequence of Theorem 5.1), simply states the soundness of the operational net semantics with respect to the Markovian semantics. The usefulness of this net semantics can be appreciated by observing that there exist some software tools (e.g. GreatSPN [6]) which allow to perform a simulative analysis of stochastic Petri nets, whereas similar tools for stochastic process algebras do not exist yet.

7 Concurrency principle

In this section we prove that the operational net semantics satisfies the concurrency principle. The introduction of this principle is due to the fact that retrievability deals only with individual transitions so it does not reject net semantics exhibiting too little concurrency.

To formalize the concurrency principle, we adapt to our stochastic framework some standard operators on nets generally accepted as representing the intended concurrency of terms. In other words, following a standard practice (see, e.g., [16]), we develop a *denotational net semantics* for MPA and then we investigate whether the operational net semantics admits the same concurrent computations as the denotational net semantics. The operators on PGSPNs are defined as follows: [4]

- $\underline{0} = (\{p\}, U, \emptyset, \{\!|\, p \,|\!\}, \emptyset, \emptyset)$;
- $<\!a,\tilde{\lambda}\!>.(P,U,T,M_0,L,W) = (P',U,T',M'_0,L',W')$ where:
  - $P' = P \cup \{p'\}$, $p' \notin P$;
  - $T' = T \oplus \{\!|\, (\{\!|\, p' \,|\!\}, <\!a,\tilde{\lambda}\!>, M_0) \,|\!\}$;
  - $M'_0 = \{\!|\, p' \,|\!\}$;
- $(P,U,T,M_0,L',W)/L = (P,U,T',M_0,L'',W')$ where:
  - $T' = \{\!|\, (M_1,<\!a,\tilde{\lambda}\!>,M_2) \in T \mid a \notin L \,|\!\} \oplus \{\!|\, (M_1,<\!\tau,\tilde{\lambda}\!>,M_2) \mid \exists a \in L.\ (M_1,<\!a,\tilde{\lambda}\!>,M_2) \in T \,|\!\}$;
- $(P,U,T,M_0,L,W)\backslash H = (P,U,T',M_0,L',W')$ where:
  - $T' = \{\!|\, (M_1,<\!a,\tilde{\lambda}\!>,M_2) \in T \mid \neg(a \in H \wedge \tilde{\lambda} = 0) \,|\!\}$;
- $(P,U,T,M_0,L,W)[\varphi] = (P,U,T',M_0,L',W')$ where:
  - $T' = \{\!|\, (M_1,<\!\varphi(a),\tilde{\lambda}\!>,M_2) \mid (M_1,<\!a,\tilde{\lambda}\!>,M_2) \in T \,|\!\}$;
- $(P_1,U,T_1,M_{01},L_1,W_1) + (P_2,U,T_2,M_{02},L_2,W_2) = (P,U,T,M_0,L,W)$ where:
  - $P = (dom(M_{01}) \times dom(M_{02})) \cup P_1 \cup P_2$, $P_1 \cap P_2 = \emptyset$;
  - $T = \{\!|\, (M_1 \otimes M_{02},<\!a,\tilde{\lambda}\!>,M_2) \mid M_1 \subseteq M_{01} \wedge (M_1,<\!a,\tilde{\lambda}\!>,M_2) \in T_1 \,|\!\} \oplus \{\!|\, (M_{01} \otimes M_1,<\!a,\tilde{\lambda}\!>,M_2) \mid M_1 \subseteq M_{02} \wedge (M_1,<\!a,\tilde{\lambda}\!>,M_2) \in T_2 \,|\!\} \oplus T_1 \oplus T_2$;
  - $M_0 = M_{01} \otimes M_{02}$;
- $(P_1,U,T_1,M_{01},L_1,W_1) \|_S (P_2,U,T_2,M_{02},L_2,W_2) = (P,U,T,M_0,L,W)$ where:
  - $P = P_1 \cup P_2$, $P_1 \cap P_2 = \emptyset$;
  - $T = \{\!|\, (M_1,<\!a,\tilde{\lambda}\!>,M_2) \in T_1 \oplus T_2 \mid a \notin S \,|\!\} \oplus \{\!|\, (M_1 \oplus M'_1,<\!a,\tilde{\gamma}\!>,M_2 \oplus M'_2) \mid (M_1,<\!a,\tilde{\lambda}\!>,M_2) \in T_1 \wedge (M'_1,<\!a,\tilde{\mu}\!>,M'_2) \in T_2 \wedge a \in S \wedge \min(\tilde{\lambda},\tilde{\mu}) = 0 \wedge \tilde{\gamma} = \max(\tilde{\lambda},\tilde{\mu})/\mathcal{N}(a,\tilde{\lambda},\tilde{\mu},M_1,M'_1) \,|\!\}$;
  - $M_0 = M_{01} \oplus M_{02}$.

The effect of these net operators should be easy to understand, except for the definition of the alternative composition operator. Such a definition combines the standard alternative composition operator with the idea of root unwinding which ensures that there are no cycles left at initially marked places; it

[4] The definitions of the set of labels, the priority function and the weight function for the resulting PGSPN of each operator are omitted because they are similar to those reported in Definition 4.3.

then uses the cartesian product to introduce choices between all the pairs of initial transitions of the two nets.

Using the place based strong bisimilarity $\approx$ on nets as formulated in [16] Definition 2.3.8, and following a demonstration similar to that of [16] Theorem 3.8.3, we can now prove that for each operator $\boldsymbol{op}$ we have $\mathcal{PGSPN}[\![\boldsymbol{op}_{MPA}(E_1, \ldots, E_n)]\!] \approx \boldsymbol{op}_{PGSPN}(\mathcal{PGSPN}[\![E_1]\!], \ldots, \mathcal{PGSPN}[\![E_n]\!])$. By exploiting [16] Theorem 2.3.10, this means that the two nets have the same causal semantics, i.e. they have the same concurrent computations.

8 An example

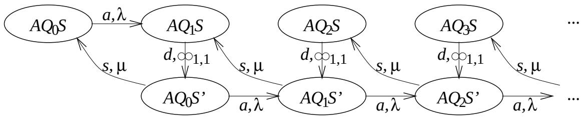
In this section we show the different semantics associated with a MPA description of a simple but interesting system, and we see in particular that the interleaving semantics and the Markovian semantics are both retrievable from the net semantics (other examples can be found in [3]).

The system we wish to consider here is a queueing system $\boldsymbol{M/M/}1$ with arrival rate $\boldsymbol{\lambda}$ and service rate $\boldsymbol{\mu}$ [13]. Let $\boldsymbol{a}$ be the type of the action "a customer arrives at the queue of the service center", $\boldsymbol{d}$ be the type of the action "a customer is delivered by the queue to the server" and $\boldsymbol{s}$ be the type of the action "a customer is served by the server". Then, a queueing system $\boldsymbol{M/M/}1$ can be modeled with MPA as follows:

- $\boldsymbol{System} \triangleq \boldsymbol{Arrivals}\|_{\{a\}}(\boldsymbol{Queue}_0\|_{\{d\}}\boldsymbol{Server})$;
- $\boldsymbol{Arrivals} \triangleq \lt\boldsymbol{a}, \lambda\gt.\boldsymbol{Arrivals}$;
- $\boldsymbol{Queue}_0 \triangleq \lt\boldsymbol{a}, 0\gt.\boldsymbol{Queue}_1$,
  $\boldsymbol{Queue}_h \triangleq \lt\boldsymbol{a}, 0\gt.\boldsymbol{Queue}_{h+1} + \lt\boldsymbol{d}, 0\gt.\boldsymbol{Queue}_{h-1}, \quad h > 0$;
- $\boldsymbol{Server} \triangleq \lt\boldsymbol{d}, \infty_{1,1}\gt.\lt\boldsymbol{s}, \mu\gt.\boldsymbol{Server}$.

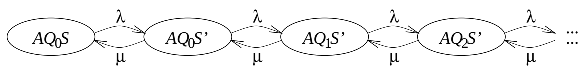
It is worth noting that, thanks to the property of compositionality of MPA, we have described the whole system as the composition of the arrival process with the composition of the queue and the server, and that then we have separately modeled the arrival process, the queue and the server. Notice also that all the actions of the queue are passive.

The interleaving semantics $\mathcal{I}[\![\boldsymbol{System}]\!]$ is given by the following infinite LTS:



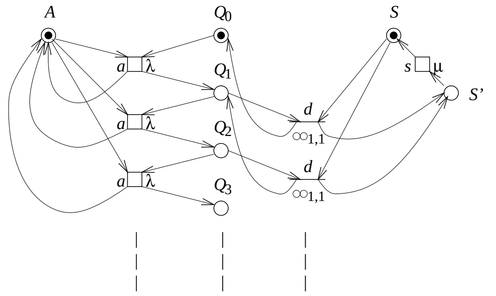
where for $h \geq 0$ the shorthand $\boldsymbol{AQ_hS}$ stands for $\boldsymbol{Arrivals}\|_{\{a\}}(\boldsymbol{Queue}_h\|_{\{d\}}\boldsymbol{Server})$ and $\boldsymbol{AQ_hS'}$ for $\boldsymbol{Arrivals}\|_{\{a\}}(\boldsymbol{Queue}_h\|_{\{d\}}\lt\boldsymbol{s}, \mu\gt.\boldsymbol{Server})$.

The Markovian semantics $\mathcal{M}[\![\boldsymbol{System}]\!]$ is given by the following infinite HCTMC:



and it is isomorphic to the HCTMC underlying a queueing system $\boldsymbol{M/M/}1$ [13].

The net semantics $\mathcal{GSPN}[\![\boldsymbol{System}]\!]$ is given by the following infinite GSPN:

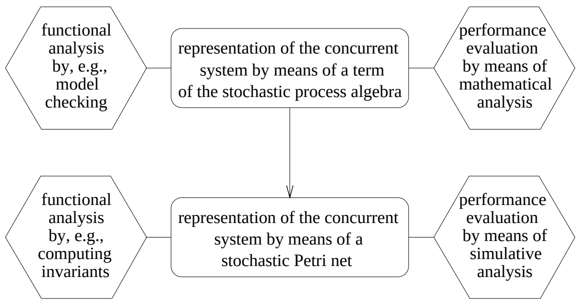


where $\boldsymbol{A}$ stands for $\lt\boldsymbol{a}, \lambda\gt.\boldsymbol{Arrival}\|_{\{a\}}\boldsymbol{id}$, $\boldsymbol{Q}_0$ stands for $\boldsymbol{id}\|_{\{a\}}(\lt\boldsymbol{a}, 0\gt.\boldsymbol{Queue}_1\|_{\{d\}}\boldsymbol{id})$, $\boldsymbol{Q}_h$ stands for $\boldsymbol{id}\|_{\{a\}}((\lt\boldsymbol{a}, 0\gt.\boldsymbol{Queue}_{h+1} + \lt\boldsymbol{d}, 0\gt.\boldsymbol{Queue}_{h-1})\|_{\{d\}}\boldsymbol{id})$, $\boldsymbol{S}$ for $\boldsymbol{id}\|_{\{a\}}(\boldsymbol{id}\|_{\{d\}}\boldsymbol{Server})$, $\boldsymbol{S'}$ for $\boldsymbol{id}\|_{\{a\}}(\boldsymbol{id}\|_{\{d\}}\lt\boldsymbol{s}, \mu\gt.\boldsymbol{Server})$. It is immediate to see that the reachability graph of $\mathcal{GSPN}[\![\boldsymbol{System}]\!]$ and its underlying HCTMC are isomorphic to $\mathcal{I}[\![\boldsymbol{System}]\!]$ and $\mathcal{M}[\![\boldsymbol{System}]\!]$, respectively.

9 Conclusions

In this paper we have defined an operational (and a denotational) net semantics for MPA based on GSPNs. This paper represents a slight improvement over [1], in which the first attempt was made to give a formally defined net semantics, fulfilling retrievability and concurrency requirements, to a stochastic process algebra. [5]

Having an operational net semantics available allows us to implement a compositional modeling technique for concurrent systems which integrates their different views (*centralized* vs. *distributed*) as well as the different aspects of their behavior (*qualitative-functional* vs. *quantitative-performance*). Such a modeling technique is subdivided into two phases and can be summarized by the following scheme:



The first phase consists of specifying the concurrent system as a term of MPA, 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, e.g., equivalence checking or, if the equiva-

[5] Afterwards an informal approach for giving a net semantics to stochastic process algebras has been proposed in [19].

lence relation is a congruence, by equational reasoning defined on its axiomatization. Such an analysis can detect qualitative properties of the concurrent system (e.g. deadlock), can help in minimizing the state space of the system representation, and can be also computer-aided [7]. Moreover, the algebraic representation allows us to evaluate the performance of the concurrent system by resorting to the study of its associated HCTMC, which can be assisted by computer [20]. The second phase consists of translating the MPA term into a GSPN, giving a distributed representation of the same concurrent system. Such a translation highlights the parallelism and the causal dependencies among the activities of the concurrent system (useful, e.g., for detecting partial deadlock), but the price to pay is that such a model usually has a remarkable graphic complexity, and it is not easily compositional, hence making hard, in general, the detection and the analysis of its subsystems. With this GSPN representation it is however possible to exploit a tool like GreatSPN [6] for performing a qualitative analysis of the concurrent system, e.g. by computing net invariants, as well as evaluating the performance of the concurrent system, e.g. by resorting to a discrete event simulation.

As the various semantics for MPA can be fully mechanized, we are presently designing a software tool which associates with 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. Therefore, our work opens a perspective to a fully integrated tool which exploits the compositionality of MPA at the design level and helps in making computer aided – and possibly automatic – the qualitative and quantitative analysis of concurrent systems.

Acknowledgements

This reaserch has been partially funded by MURST, CNR and ESPRIT BRA n. 8130 LOMAPS.

References

[1] M. Bernardo, L. Donatiello, R. Gorrieri, *"Operational GSPN Semantics of MPA"*, Technical Report TR-94-12, University of Bologna (Italy), May 1994

[2] M. Bernardo, L. Donatiello, R. Gorrieri, *"Modeling and Analyzing Concurrent Systems with MPA"*, in Proceedings of the 2nd Workshop on Process Algebras and Performance Modelling, Erlangen (Germany), July 1994

[3] M. Bernardo, L. Donatiello, R. Gorrieri, *"Integrating Performance and Functional Analysis of Concurrent Systems with MPA"*, Technical Report TR-95-12, University of Bologna (Italy), September 1995

[4] T. Bolognesi, E. Brinksma, *"Introduction to the ISO Specification Language LOTOS"*, in Computer Networks and ISDN Systems, vol. 14, 1987

[5] P. Buchholz, *"Markovian Process Algebra: Composition and Equivalence"*, in Proceedings of the 2nd Workshop on Process Algebras and Performance Modelling, Erlangen (Germany), July 1994

[6] G. Chiola, *"GreatSPN 1.5 Software Architecture"*, in Proceedings of the 5th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Turin (Italy), 1991

[7] 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, vol. 15, 1993

[8] P. Degano, R. De Nicola, U. Montanari, *"A Distributed Operational Semantics for CCS Based on Condition/Event Systems"*, in Acta Informatica, vol. 26, 1988

[9] N. Götz, U. Herzog, M. Rettelbach, *"Multiprocessor and Distributed System Design: the Integration of Functional Specification and Performance Analysis Using Stochastic Process Algebras"*, in LNCS, vol. 729, Springer-Verlag, 1993

[10] J. Hillston, *"A Compositional Approach to Performance Modelling"*, Ph.D. Thesis, University of Edinburgh (UK), 1994

[11] C. A. R. Hoare, *"Communicating Sequential Processes"*, Prentice Hall, 1985

[12] J. G. Kemeny, J. L. Snell, *"Finite Markov Chains"*, Springer-Verlag, 1977

[13] L. Kleinrock, *"Queueing Systems"*, Wiley, 1975

[14] M. A. Marsan, G. Balbo, G. Conte, *"A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems"*, in ACM Transactions on Computer Systems, vol. 2, 1984

[15] R. Milner, *"Communication and Concurrency"*, Prentice Hall, 1989

[16] E. -R. Olderog, *"Nets, Terms and Formulas"*, Cambridge University Press, 1991

[17] G. Plotkin, *"A Structural Approach to Operational Semantics"*, Technical Report, Aarhus University (Denmark), 1981

[18] W. Reisig, *"Petri Nets: an Introduction"*, Springer-Verlag, 1985

[19] M. Ribaudo, *"Understanding Stochastic Process Algebras via their Stochastic Petri Nets Semantics"*, in Proceedings of the 2nd Workshop on Process Algebras and Performance Modelling, Erlangen (Germany), July 1994

[20] R.A. Sahner, K.S. Trivedi, *"Reliability Modelling Using SHARPE"*, in IEEE Transactions on Reliability, vol. 13, 1987