Extended Markovian Process Algebra: A Case Study

Marco Bernardo

Università di Bologna, Dipartimento di Scienze dell'Informazione
Piazza di Porta S. Donato 5, 40127 Bologna, Italy
E-mail: bernardo@cs.unibo.it

Abstract

Extended Markovian Process Algebra (EMPA) is a process algebra suitable for expressing functional and performance aspects of concurrent systems. It is equipped with an operational interleaving semantics, a functional semantics, a Markovian semantics and an operational net semantics.

In this paper we apply an integrated approach based on EMPA for modeling and analyzing concurrent systems to the Dining Philosophers problem. The problem is firstly described by an EMPA term and then it is studied on the semantic models associated with the term.

1 Introduction

The need of integrating the performance analysis of a concurrent system into the design process of the system itself has been widely recognized and stimulated many researchers. The problem is that, in the case when the performance aspect is neglected, time-critical concurrent systems, such as real-time systems and communication protocols, cannot be modeled in a completely satisfactory manner. And, more important, there is no way of estimating the performance of the concurrent systems under consideration. It often happens that a concurrent system is first fully designed and tested for functionality, and afterwards tested for efficiency. As a consequence, if the performance is detected to be poor, the concurrent system has to be redesigned, thus negatively affecting both the design costs and the delivery at a fixed deadline.

In the last two decades a remarkable effort has been made in order to enhance the expressiveness of two of the most important formalisms developed in the theory of concurrency like process algebras [12, 7] and Petri nets [14], by introducing the concept of time. Among the results of this effort, we shall focus our attention on stochastic process algebras and stochastic Petri nets.

Stochastic Petri nets appeared in the 80's (see [10] and the references therein). Like classical Petri nets, they are mathematical models for graphically representing concurrent systems. The main feature is that the notion of state is distributed among net places, hence allowing causal dependencies, conflicts, synchronizations and parallelism among system activities to be explicitly described. Stochastic Petri nets extend the expressiveness of classical Petri nets by associating with each net transition a random variable determining its duration. Therefore, a stochastic Petri net model of a given system can be analyzed both from the functional and the performance point of view. The functional analysis aims at detecting behavioral and structural properties of nets, by resorting, e.g., to the technique of invariants. The performance analysis aims at determining efficiency measures. Since in most of the cases the random variables used for expressing the transition durations are exponentially distributed, performance indices are computed by solving a Markov chain. Modeling and analysis of stochastic Petri nets is supported by existing tools like GreatSPN [5].

Stochastic process algebra appeared in the 90's (see [1, 2, 3] and the references therein). Like classical process algebras, they are abstract languages conceived for representing concurrent systems in a compositional way: they provide the system designer with a small set of powerful operators whereby it is possible to construct process terms from simpler ones, without incurring in the graphical complexity of nets and making the task of detecting or modifying subsystems quite easier. Like stochastic Petri nets, they extend the expressiveness of the classical version by assigning each action a random variable determining its duration, thus producing algebraic representations of concurrent systems amenable to both functional and performance analysis. The functional analysis can be carried out by resorting to methods such as equivalence checking, preorder checking and model checking [6]. The performance analysis permits to obtain quantitative measures by resorting to the study of a Markov chain, since almost all the existing process algebras rely on exponential timing.

Stochastic process algebras and stochastic Petri nets can be exploited to implement an integrated approach for modeling and analyzing concurrent systems which relates their different points of view (centralized vs. distributed) as well as different aspects of their behavior (qualitative vs. quantitative) [4]. The approach is divided into two phases and can be summarized by the scheme reported in Figure 1.
The first phase, interfaced with the system designer, consists of representing the concurrent system as a term of the stochastic process algebra. Because of compositionality, the system designer is allowed to develop the algebraic representation of the system in a modular, stepwise refinement manner. If the stochastic process algebra is equipped with an interleaving semantics accounting for both the qualitative and quantitative part of the system behavior, the interleaving model of the algebraic representation of the system can be projected on a functional model and a performance model which can be analyzed by means of tools like CW [6] and

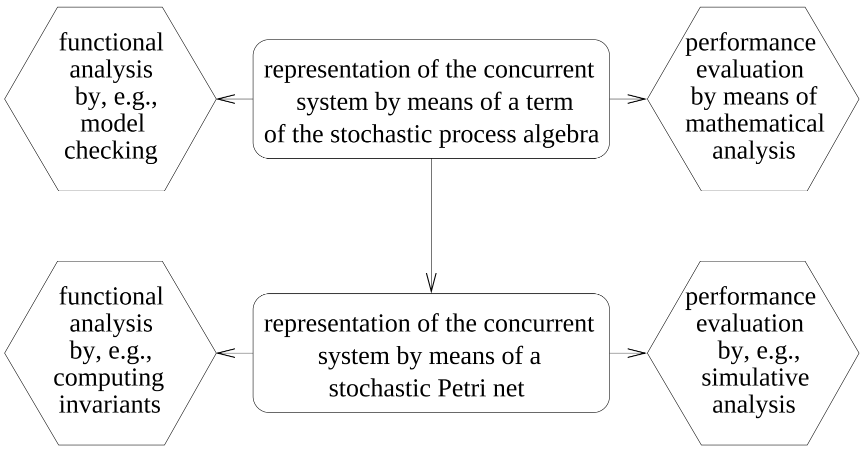

functional analysis by, e.g., model checking
representation of the concurrent system by means of a term of the stochastic process algebra
performance evaluation by means of mathematical analysis
functional analysis by, e.g., computing invariants
representation of the concurrent system by means of a stochastic Petri net
performance evaluation by, e.g., simulative analysis

Figure 1: Integrated approach

SHARPE [15], respectively.
The second phase consists of automatically obtaining from the algebraic representation of the system an equivalent distributed representation. In order for it to be implemented, the stochastic process algebra is required to have a distributed semantics as well. A suitable distributed model might be a stochastic Petri net, because stochastic Petri nets constitute a mature field in which it is possible the description and the analysis of concurrent systems both from the functional and the performance viewpoint, assisted by tools like GreatSPN [5]. The net representation of the concurrent system is derived from the algebraic one without intervention of the system designer, in order to avoid overhead concerning graphical complexity and absence of compositionality. The net representation turns out to be useful in the case when a less abstract representation is required highlighting dependencies and conflicts among system activities, and helpful to detect some properties (e.g., partial deadlock) which can be easily checked only in a distributed setting.

As shown in [4], this integrated approach can be instantiated to Extended Markovian Process Algebra (EMPA), as it is a stochastic process algebra fulfilling the needed requirements. In fact, it is supplied with an interleaving semantics, from which a functional and a performance semantics can be derived, as well as a distributed semantics.

The purpose of this paper is that of showing an application of the integrated approach based on EMPA to a given problem. The paper is organized as follows. In Section 2 EMPA is briefly introduced. In Section 3 the Dining Philosophers problem is tackled using the integrated approach above. In Section 4 some concluding remarks are reported.

2 Extended Markovian Process Algebra

In this section we briefly present the syntax and the semantics for EMPA. Due to lack of space, the reader is invited to consult [4] for a complete presentation.

Each EMPA *action* is described by a pair $<a, \tilde{\lambda}>$ consisting of the *type* of the action and the *rate* of the action, respectively. Based on the type, actions are divided into *external* and *internal* ones: we denote with τ the only internal action type. Based on rate, we distinguish among passive, timed and immediate actions. The duration of a *passive* action is fixed only upon the synchronization with a nonpassive action of the same type. Its rate is denoted with 0. Passive actions are thus used for representing waitings, play a prominent role in synchronizations and allow us to express nondeterministic choices. The duration of a *timed* action is exponentially distributed with parameter $\lambda \in \mathbb{R}^+$ and its rate is λ. The duration of an *immediate* action is zero and its rate is denoted with $\infty_{l,w}$, where l is the *priority level* and w is the *weight*. Immediate actions are then used for representing activities which are irrelevant from the performance evaluation point of view, and allow us to express explicit probabilistic choices. Let $Act = AType \times ARate$ be the set of actions, where $ARate = \{0\} \cup \mathbb{R}_+ \cup Inf$ with $Inf = \{\infty_{l,w} \mid l \in \mathbb{N}_+ \wedge w \in \mathbb{R}_+\}$.

Let Con be a set of *constants*, and let $\Phi = \{\varphi : AType \longrightarrow AType \mid \varphi(\tau) = \tau \wedge \varphi(AType - \{\tau\}) \subseteq AType - \{\tau\}\}$ be a set of *relabeling functions*. The set $\mathcal{L}$ of EMPA processes 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 $a \in AType$, $\tilde{\lambda} \in ARate$, $L \subseteq AType - \{\tau\}$, $H \subseteq AType$, $\varphi \in \Phi$, $S \subseteq AType - \{\tau\}$, $A \in Con$. Apart from the *null term* "$\underline{0}$", the *prefix operator* "$<a, \tilde{\lambda}>.\_$", the *functional abstraction (or hiding) operator* "$\_/L$" and the *functional relabeling operator* "$\_[\varphi]$" whose meaning is exactly the same as that of the analogous operators of classical process algebras, we turn our attention to the *temporal restriction operator* "$\_\backslash H$", the *alternative composition operator* "$\_+\_$" and the *parallel composition operator* "$\_\|_S\_$".

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*. Such a notion concerns terms which cannot execute passive actions, so it singles out terms which can model real systems because the activities performed by these systems are completely specified from the performance viewpoint.

The *alternative composition operator* "$\_+\_$" expresses a choice between two terms. Unlike classical process algebras, such a choice is not necessarily nondeterministic since its nature depends upon the rates of the actions executable by the various alternatives.

The parallel composition operator expresses the parallel composition of two terms according to the TCSP synchronization discipline on action types [7]. The synchronization discipline on action rates we adopted requires that at most one nonpassive action is involved.

Because of the presence of binary operators such as the alternative composition and the parallel composition ones, the situation in which several nonpassive actions are simultaneously executable can arise. Both in the case of the alternative composition (due to mutual exclusion) and in the case of the parallel composition (as we shall adopt an interleaving model, hence allowing only one action at a time to be performed), we need a mechanism choosing the action to be executed. We adopted the *race policy*: the action sampling the least duration succeeds. As a consequence, immediate actions take precedence over timed ones. We assume that passive actions are not involved in the priority mechanism.

Each EMPA term E is supplied with four semantic models.

The *operational interleaving semantics* $\mathcal{I}[\![E]\!]$ is a labeled transition system (LTS) obtained by applying the rules reported in Table 1. Let us call *potential move* of a given term a pair composed of an action executable by the term and a target term obtained by performing the action. The idea is that of inductively computing all the potential moves of a given term regardless of priorities, and then imposing the race policy by discarding those potential moves having a lower priority. Computing inductively all the potential moves of a given term instead of a single potential move at a time is motivated, in a stochastic framework, by the fact that the actual executability as well as the execution probability of an action depend upon all the actions which are executable at the same time when it is executable. As a consequence, only if we know all the potential moves of the subterms of a given term, we can correctly determine its transitions and their rates.
Relation $\vdash$, defined in the lower part of the table, computes the multiset of all the potential moves of each term, regardless of action priorities. The way in which such a multiset is worked out should result clear from the informal explanation of the semantics of operators. The only caution concerns the need of normalizing the rates of potential moves resulting from the synchronization of the same active 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). Consider for instance term

$$<a, \lambda>.\underline{0} \|_{\{a\}} (<a, 0>.\underline{0} \|_{\emptyset} <a, 0>.\underline{0})$$

which is constructed from the parallel composition of a term that can execute a timed action of type a with a term deriving from the parallel composition of two independent terms each of which can execute a passive action of type a. This term has two potential moves each of which must have rate equal to $\lambda/2$. This operation is carried out by function $Norm$.
Relation $\longrightarrow$, defined in the upper part of the table, enforces the race policy by selecting among the potential moves of a term those having the highest priority level (function *Race*), and then merges together all the potential moves in which actions with the same type and priority as well as the same target term occur (function *Melt*). Merging is required by the fact that idempotency does not hold for the alternative composition operator. For instance, term

$$\frac{E \vdash PM}{E \xrightarrow{a,\tilde{\lambda}} E'} \quad [(<a,\tilde{\lambda}>, E') \in Melt(Race(PM))]$$

$$<a,\tilde{\lambda}>.E \vdash \{\!|\, (<a,\tilde{\lambda}>, E) \,|\!\}$$

$$\frac{E \vdash PM}{\begin{array}{ll} E/L \vdash & \{\!|\, (<a,\tilde{\lambda}>, E'/L) \mid (<a,\tilde{\lambda}>, E') \in PM \wedge a \notin L \,|\!\} \oplus \\ & \{\!|\, (<\tau,\tilde{\lambda}>, E'/L) \mid (<a,\tilde{\lambda}>, E') \in PM \wedge a \in L \,|\!\} \end{array}}$$

$$\frac{E \vdash PM}{E \backslash H \vdash \{\!|\, (<a,\tilde{\lambda}>, E' \backslash H) \mid (<a,\tilde{\lambda}>, E') \in PM \wedge \neg(a \in H \wedge \tilde{\lambda} = 0) \,|\!\}}$$

$$\frac{E \vdash PM}{E[\varphi] \vdash \{\!|\, (<\varphi(a),\tilde{\lambda}>, E'[\varphi]) \mid (<a,\tilde{\lambda}>, E') \in PM \,|\!\}}$$

$$\frac{E_1 \vdash PM_1 \quad E_2 \vdash PM_2}{E_1 + E_2 \vdash PM_1 \oplus PM_2}$$

$$\frac{E_1 \vdash PM_1 \quad E_2 \vdash PM_2}{\begin{array}{ll} E_1 \|_S E_2 \vdash & \{\!|\, (<a,\tilde{\lambda}>, E_1' \|_S E_2) \mid a \notin S \wedge (<a,\tilde{\lambda}>, E_1') \in PM_1 \,|\!\} \oplus \\ & \{\!|\, (<a,\tilde{\lambda}>, E_1 \|_S E_2') \mid a \notin S \wedge (<a,\tilde{\lambda}>, E_2') \in PM_2 \,|\!\} \oplus \\ & \{\!|\, (<a,\tilde{\gamma}>, E_1' \|_S E_2') \mid a \in S \wedge \\ & \qquad (<a,\tilde{\lambda}>, E_1') \in PM_1 \wedge \\ & \qquad (<a,\tilde{\mu}>, E_2') \in PM_2 \wedge \\ & \qquad \tilde{\gamma} = Norm(a, \tilde{\lambda}, \tilde{\mu}, PM_1, PM_2) \,|\!\} \end{array}}$$

$$\frac{E \vdash PM}{A \vdash PM} \quad [A \stackrel{\Delta}{=} E]$$

Table 1: Inductive rules for EMPA interleaving semantics

$$<a, \lambda>.E + <a, \lambda>.E$$

is equivalent to term

$$<a, 2 \cdot \lambda>.E$$

due to a well known property of exponential distributions.

The *functional semantics* $\mathcal{F}[\![E]\!]$ is a LTS obtained from $\mathcal{I}[\![E]\!]$ by dropping action rates.

The *performance semantics*, hereafter called *Markovian semantics*, $\mathcal{M}[\![E]\!]$ is a homogeneous continuous-time Markov chain (HCTMC) [8] obtained from $\mathcal{I}[\![E]\!]$ by applying an algorithm divided into two phases. The first phase removes all the states having immediate transitions and the immediate transitions themselves, because the sojourn time in such states is zero. The second phase merges states which are equivalent according to the notion of lumping [8], in order to minimize the space state of the HCTMC obtained at the end of the first phase.

The *operational net semantics* $\mathcal{L}oc\mathcal{ON}[\![E]\!]$ is a generalized stochastic Petri net (GSPN) [11] obtained by applying an extension of the structured operational semantics approach [13] such that: net places correspond to the sequential subterms of E and its derivatives and are constructed by means of a decomposition function, net transitions are defined by induction on the syntactical structure of the sets of sequential terms by using rules similar to those reported in Table 1, and net markings correspond roughly to E and its derivatives. The approach is called location-oriented because, by the definition of the decomposition function, all the information about the syntactical structure of terms is encoded inside places: this causes nets associated with terms to be safe. The operational net semantics turns out to be consistent with the operational interleaving semantics since it meets both the functional and the performance retrievability principles. Moreover, it represents all the intended concurrency of terms because it satisfies the concurrency principle as well.

3 The Dining Philosophers problem

In this section we want to tackle a classical problem in the world of the concurrency by using the integrated approach proposed in Section 1. The aim is not that of developing a solution of the problem, but to describe the scenario generating the problem itself by means of an EMPA term and then analyze this scenario by considering the semantic models associated with such a term.

The problem we wish to examine is the *Dining Philosophers problem.* The scenario of such a problem is the following. There are n philosophers each of which is independent from the others. Each philosopher performs only two activities: thinking and eating. When a philosopher is hungry, he stops thinking and moves to a table. There is a plate of rice in the center of the table, and there are n chopsticks on the table each of which is placed at the same distance from its neighboring chopsticks. Before eating, each philosopher must pick up both the chopstick on his left and the chopstick on his right, i.e. he cannot eat using only one chopstick. The amount of time taken by a philosopher to eat is finite. After eating, each philosopher must put down both the chopsticks. Thereafter, he starts thinking again.

The problem is interesting in the theory of concurrency because the philosophers represent competing users and the chopsticks represent shared resources. Each solution of the problem must then guarantee the maximum parallelism between the philosophers and the mutually exclusive usage of the chopsticks, besides avoiding deadlock and possibly starvation.

Let us consider a possible description of the previously outlined scenario with EMPA. Suppose for simplicity that $n = 3$, so that we deal with only three philosophers $\{P_0, P_1, P_2\}$ and three chopsticks $\{C_0, C_1, C_2\}$; following this numbering, we assume that P_i has C_i on his right and C_{i+1} on his left where the sum is modulo n. Suppose also that P_i must pick up (and put down) C_i before C_{i+1}; this is simply a choice because here we only want to model the scenario, but it can also be viewed as an attempt to solve the problem (although it is well known that it causes deadlock). The action types we shall use in the description are the following: t standing for "think", pu_i standing for "pick up C_i", e standing for "eat", pd_i standing for "put down C_i". Now we have to decide the action rates. Assuming that the amounts of time taken by P_i for thinking and for eating are exponentially distributed with rate λ_i and μ_i, respectively, the action type t will have $\{\lambda_0, \lambda_1, \lambda_2\}$ as possible rates while the action type e will have $\{\mu_0, \mu_1, \mu_2\}$. Since actions whose type is either pu_i or pd_i are irrelevant from the performance viewpoint, they will be taken to be immediate actions from the point of view of P_i and passive actions from the point of view of C_i.
The scenario can be described with the following term DP_3:

$$DP_3 \triangleq (P_0 \|_\emptyset P_1 \|_\emptyset P_2) \|_S (C_0 \|_\emptyset C_1 \|_\emptyset C_2) \text{ where } S = \{pu_i, pd_i \mid 0 \leq i \leq 2\}$$
$$P_i \triangleq <t, \lambda_i>.<pu_i, \infty_{1,1}>.<pu_{i+1}, \infty_{1,1}>.<e, \mu_i>.<pd_i, \infty_{1,1}>.<pd_{i+1}, \infty_{1,1}>.P_i$$
$$C_i \triangleq <pu_i, 0>.<pd_i, 0>.C_i$$

Note that it has been very easy to model the scenario because of compositionality. We have first described the whole system as the parallel composition of terms representing philosophers and chopsticks, augmented with the synchronization interface S. Then we have modeled each philosopher and each chopstick separately.

The operational interleaving semantics $\mathcal{I}[\![DP_3]\!]$ is presented in Figure 2. The shorthands used for state names and transition labels are reported in Table 2, where $P_i \triangleq <t, \lambda_i>.P_i^1$, $P_i^1 \triangleq <pu_i, \infty_{1,1}>.P_i^2$, $P_i^2 \triangleq <pu_{i+1}, \infty_{1,1}>.P_i^3$, $P_i^3 \triangleq <e, \mu_i>.P_i^4$, $P_i^4 \triangleq <pd_i, \infty_{1,1}>.P_i^5$, $P_i^5 \triangleq <pd_{i+1}, \infty_{1,1}>.P_i$, $C_i \triangleq <pu_i, 0>.C_i^1$, and $C_i^1 \triangleq <pd_i, 0>.C_i$.
The LTS $\mathcal{I}[\![DP_3]\!]$ has 61 states and a regular structure due to the symmetry in the roles of the three philosophers. Only 13 states over 61 are tangible, i.e. have no

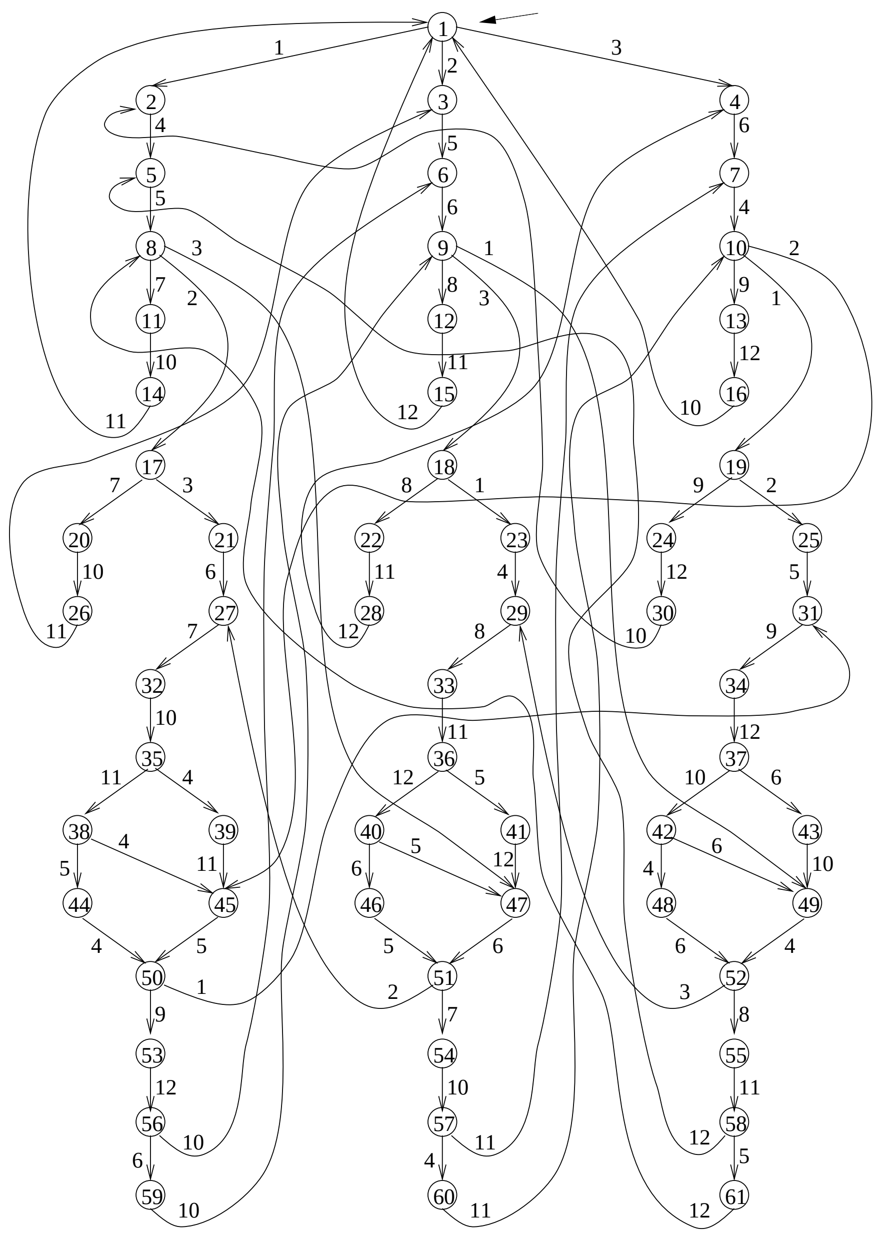


Figure 2: Operational interleaving semantics of DP_3

s_1	$P_0P_1P_2C_0C_1C_2$	s_2	$P_0^1P_1P_2C_0C_1C_2$	s_3	$P_0P_1^1P_2C_0C_1C_2$
s_4	$P_0P_1P_2^1C_0C_1C_2$	s_5	$P_0^2P_1P_2C_0^1C_1C_2$	s_6	$P_0P_1^2P_2C_0C_1^1C_2$
s_7	$P_0P_1P_2^2C_0C_1C_2^1$	s_8	$P_0^3P_1P_2C_0^1C_1^1C_2$	s_9	$P_0P_1^3P_2C_0C_1^1C_2^1$
s_{10}	$P_0P_1P_2^3C_0^1C_1C_2^1$	s_{11}	$P_0^4P_1P_2C_0^1C_1^1C_2$	s_{12}	$P_0P_1^4P_2C_0C_1^1C_2^1$
s_{13}	$P_0P_1P_2^4C_0^1C_1C_2^1$	s_{14}	$P_0^5P_1P_2C_0C_1^1C_2$	s_{15}	$P_0P_1^5P_2C_0C_1C_2^1$
s_{16}	$P_0P_1P_2^5C_0^1C_1C_2$	s_{17}	$P_0^3P_1^1P_2C_0^1C_1^1C_2$	s_{18}	$P_0P_1^3P_2^1C_0C_1^1C_2^1$
s_{19}	$P_0^1P_1P_2^3C_0^1C_1C_2^1$	s_{20}	$P_0^4P_1^1P_2C_0^1C_1^1C_2$	s_{21}	$P_0^3P_1^1P_2^1C_0^1C_1^1C_2$
s_{22}	$P_0P_1^4P_2^1C_0C_1^1C_2^1$	s_{23}	$P_0^1P_1^3P_2^1C_0C_1^1C_2^1$	s_{24}	$P_0^1P_1P_2^4C_0^1C_1C_2^1$
s_{25}	$P_0^1P_1^1P_2^3C_0^1C_1C_2^1$	s_{26}	$P_0^5P_1^1P_2C_0C_1^1C_2$	s_{27}	$P_0^3P_1^1P_2^2C_0^1C_1^1C_2^1$
s_{28}	$P_0P_1^5P_2^1C_0C_1C_2^1$	s_{29}	$P_0^2P_1^3P_2^1C_0^1C_1^1C_2^1$	s_{30}	$P_0^1P_1P_2^5C_0^1C_1C_2$
s_{31}	$P_0^1P_1^2P_2^3C_0^1C_1^1C_2^1$	s_{32}	$P_0^4P_1^1P_2^2C_0^1C_1^1C_2^1$	s_{33}	$P_0^2P_1^4P_2^1C_0^1C_1^1C_2^1$
s_{34}	$P_0^1P_1^2P_2^4C_0^1C_1^1C_2^1$	s_{35}	$P_0^5P_1^1P_2^2C_0C_1^1C_2^1$	s_{36}	$P_0^2P_1^5P_2^1C_0^1C_1C_2^1$
s_{37}	$P_0^1P_1^2P_2^5C_0^1C_1^1C_2$	s_{38}	$P_0P_1^1P_2^2C_0C_1C_2^1$	s_{39}	$P_0^5P_1^1P_2^3C_0^1C_1^1C_2^1$
s_{40}	$P_0^2P_1P_2^1C_0^1C_1C_2$	s_{41}	$P_0^3P_1^5P_2^1C_0^1C_1^1C_2^1$	s_{42}	$P_0^1P_1^2P_2C_0C_1^1C_2$
s_{43}	$P_0^1P_1^3P_2^5C_0^1C_1^1C_2^1$	s_{44}	$P_0P_1^2P_2^2C_0C_1^1C_2^1$	s_{45}	$P_0P_1^1P_2^3C_0^1C_1C_2^1$
s_{46}	$P_0^2P_1P_2^2C_0^1C_1C_2^1$	s_{47}	$P_0^3P_1P_2^1C_0^1C_1^1C_2$	s_{48}	$P_0^2P_1^2P_2C_0^1C_1^1C_2$
s_{49}	$P_0^1P_1^3P_2C_0C_1^1C_2^1$	s_{50}	$P_0P_1^2P_2^3C_0^1C_1^1C_2^1$	s_{51}	$P_0^3P_1P_2^2C_0^1C_1^1C_2^1$
s_{52}	$P_0^2P_1^3P_2C_0^1C_1^1C_2^1$	s_{53}	$P_0P_1^2P_2^4C_0^1C_1^1C_2^1$	s_{54}	$P_0^4P_1P_2^2C_0^1C_1^1C_2^1$
s_{55}	$P_0^2P_1^4P_2C_0^1C_1^1C_2^1$	s_{56}	$P_0P_1^2P_2^5C_0^1C_1^1C_2$	s_{57}	$P_0^5P_1P_2^2C_0C_1^1C_2^1$
s_{58}	$P_0^2P_1^5P_2C_0^1C_1C_2^1$	s_{59}	$P_0P_1^3P_2^5C_0^1C_1^1C_2^1$	s_{60}	$P_0^5P_1P_2^3C_0^1C_1^1C_2^1$
		s_{61}	$P_0^3P_1^5P_2C_0^1C_1^1C_2^1$		

l_1	t, λ_0	l_2	t, λ_1	l_3	t, λ_2	l_4	$pu_0, \infty_{1,1}$
l_5	$pu_1, \infty_{1,1}$	l_6	$pu_2, \infty_{1,1}$	l_7	t, μ_0	l_8	t, μ_1
l_9	t, μ_2	l_{10}	$pd_0, \infty_{1,1}$	l_{11}	$pd_1, \infty_{1,1}$	l_{12}	$pd_2, \infty_{1,1}$

p_1	$((dec(P_0)\|_\emptyset id)\|_\emptyset id)\|_S id$	p_2	$((id\|_\emptyset dec(P_1))\|_\emptyset id)\|_S id$
p_3	$(id\|_\emptyset dec(P_2))\|_S id$	p_4	$id\|_S((dec(C_0)\|_\emptyset id)\|_\emptyset id)$
p_5	$id\|_S((id\|_\emptyset dec(C_1))\|_\emptyset id)$	p_6	$id\|_S(id\|_\emptyset dec(C_2))$
p_7	$((dec(P_0^1)\|_\emptyset id)\|_\emptyset id)\|_S id$	p_8	$((id\|_\emptyset dec(P_1^1))\|_\emptyset id)\|_S id$
p_9	$(id\|_\emptyset dec(P_2^1))\|_S id$	p_{10}	$((dec(P_0^2)\|_\emptyset id)\|_\emptyset id)\|_S id$
p_{11}	$((id\|_\emptyset dec(P_1^2))\|_\emptyset id)\|_S id$	p_{12}	$(id\|_\emptyset dec(P_2^2))\|_S id$
p_{13}	$((dec(P_0^3)\|_\emptyset id)\|_\emptyset id)\|_S id$	p_{14}	$((id\|_\emptyset dec(P_1^3))\|_\emptyset id)\|_S id$
p_{15}	$(id\|_\emptyset dec(P_2^3))\|_S id$	p_{16}	$id\|_S((dec(C_0^1)\|_\emptyset id)\|_\emptyset id)$
p_{17}	$id\|_S((id\|_\emptyset dec(C_1^1))\|_\emptyset id)$	p_{18}	$id\|_S(id\|_\emptyset dec(C_2^1))$
p_{19}	$((dec(P_0^4)\|_\emptyset id)\|_\emptyset id)\|_S id$	p_{20}	$((id\|_\emptyset dec(P_1^4))\|_\emptyset id)\|_S id$
p_{21}	$(id\|_\emptyset dec(P_2^4))\|_S id$	p_{22}	$((dec(P_0^5)\|_\emptyset id)\|_\emptyset id)\|_S id$
p_{23}	$((id\|_\emptyset dec(P_1^5))\|_\emptyset id)\|_S id$	p_{24}	$(id\|_\emptyset dec(P_2^5))\|_S id$

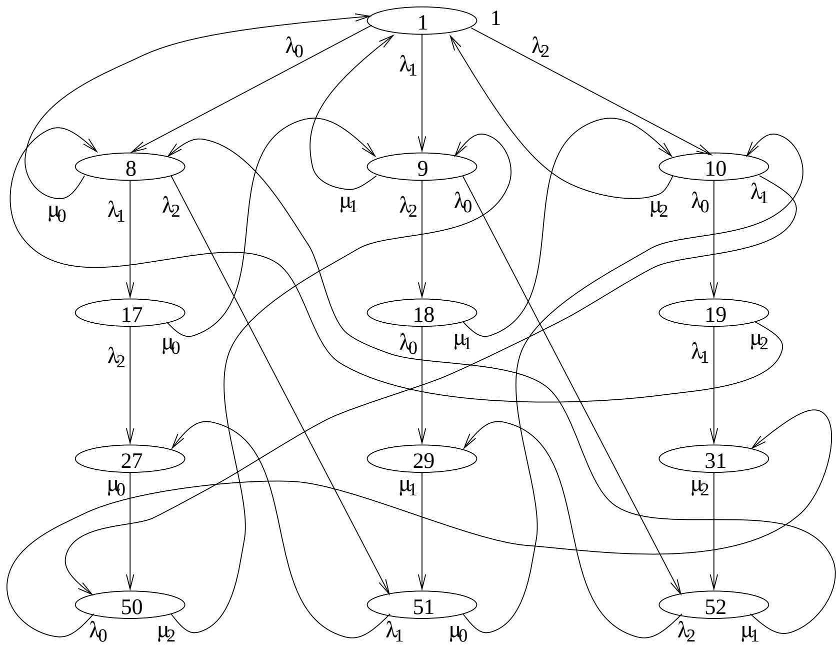
Table 2: State names, transition labels and place names

immediate transitions. $\mathcal{I}[\![DP_3]\!]$ has no passive transitions, so DP_3 is temporally closed.
Following the proposed approach, we can use the LTS $\mathcal{F}[\![DP_3]\!]$ (obtained from $\mathcal{I}[\![DP_3]\!]$ by dropping action rates) to detect some functional properties. For example, we see that each state has at least one incoming transition and one outgoing transition. As a consequence, the system has neither terminating states nor deadlock states. At first glance, the absence of deadlock could be surprising because if we considered a purely functional description of the same scenario (which can be obtained by simply considering each action as if it were passive), then we would find the deadlock state $P_0^2 P_1^2 P_2^2 C_0^1 C_1^1 C_2^1$: each of the three philosophers stopped thinking, moved to the table and picked up the chopstick on his right, thus waiting forever for the chopstick on his left to become free. The absence of deadlock in $\mathcal{F}[\![DP_3]\!]$ stems from the timing associated with the two activities performed by the three philosophers, and in particular from the property of unbounded support of the exponential distributions. Such a property guarantees that the probability that all the three philosophers stop thinking simultaneously is zero.
The LTS $\mathcal{F}[\![DP_3]\!]$ could be fully analyzed by using tools like CW [6].

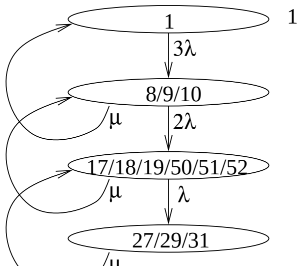
The Markovian semantics $\mathcal{M}[\![DP_3]\!]$ is presented in Figure 3*(a)*.
The HCTMC $\mathcal{M}[\![DP_3]\!]$ has 13 states corresponding to the 13 tangible states of $\mathcal{I}[\![DP_3]\!]$. It also has a regular structure due to the symmetry in the roles of the three philosophers. Since each state has at least one outgoing transition and one incoming transition, the HCTMC is irreducible (i.e., for each pair of states s_j and s_k, the probability to move from s_i to s_j is nonzero). The HCTMC is obviously finite, too.
Following the proposed approach, we can exploit such a HCTMC for assessing some performance indices. In particular, it is interesting to consider the case when $\lambda_0 = \lambda_1 = \lambda_2 \equiv \lambda$ and $\mu_0 = \mu_1 = \mu_2 \equiv \mu$. In such a case the phase of lumping comes into play and the Markovian semantics $\mathcal{M}[\![DP_3]\!]$ is reduced to the HCTMC presented in Figure 3*(b)*. If we look closely at this HCTMC, we immediately realize that it is isomorphic to the HCTMC underlying a QS $M/M/1/\infty/3$ [9], i.e. a QS $M/M/1$ with only three customers. Therefore, after numbering the states from 0 to 3 for simplicity, we have that its steady-state probability vector $\boldsymbol{\pi}$ exists and it is given by [9]

$$\pi_0 = 1/\sum_{k=0}^{3} \left(\frac{\lambda}{\mu}\right)^k \cdot \frac{3!}{(3-k)!}, \qquad \forall k = 1,2,3 \;\; \pi_k = \pi_0 \cdot \left(\frac{\lambda}{\mu}\right)^k \cdot \frac{3!}{(3-k)!}$$

This correspondence is natural if we regard the three philosophers as customers of the service center represented by the table. State s_0' represents the situation in which all the three philosophers are thinking; the rate at which it is possible to move from s_0' to s_1' is $3 \cdot \lambda$. In state s_1' one philosopher is eating while the other ones are thinking; if the eating philosopher terminates before one of the others stops thinking then we move to state s_0' with rate μ, otherwise we move to state s_2' with rate $2 \cdot \lambda$. In state s_2' one philosopher is eating, one philosopher is waiting for



(a)



(b)

Figure 3: Markovian semantics of DP_3

the chopsticks and one philosopher is thinking; if the eating philosopher terminates before the thinking philosopher then the waiting philosopher can start eating and we move to state s'_1 with rate μ, otherwise we move to state s'_3 with rate λ. In state s'_3 one philosopher is eating and the others are waiting for the chopsticks; when the eating philosopher terminates, one of the two waiting philosophers can start eating and we move to state s'_2 with rate μ.
From the steady-state distribution $\boldsymbol{\pi}$ we can derive several steady-state performance indices. For example, $\boldsymbol{\pi}_i$ is the fraction of time during which i philosophers are eating or ready to eat; $(3\lambda\boldsymbol{\pi}_0)^{-1} = (\mu\boldsymbol{\pi}_1)^{-1}$ is the average time that elapses before a philosopher starts eating if none of the others is eating, i.e. it is the average time during which the resources remain completely unused.
The HCTMC $\mathcal{M}[\![DP_3]\!]$ could be fully analyzed by using tools like SHARPE [15].

In order to pass from the first phase of the integrated approach to the second one, we have to leave the interleaving model underlying DP_3 and find a distributed model (i.e., a GSPN in our case) which represents the same scenario in a less abstract way. Doing so, we lose the compositionality (hence the easy readability) of the algebraic term DP_3 but we highlight the parallelism and the causal dependencies among the activities performed by the philosophers.
The operational net semantics $\mathcal{L}oc\mathcal{ON}[\![DP_3]\!]$ is presented in Figure 4, where the names of the places and the labels of the transitions have been shortened as reported in Table 2.
The GSPN $\mathcal{L}oc\mathcal{ON}[\![DP_3]\!]$ comprises 24 places and 18 transitions. Again, it has a regular structure due to the symmetry in the roles of the three philosophers. Since the operational net semantics for EMPA meets the retrievability principles, DP_3 and $\mathcal{L}oc\mathcal{ON}[\![DP_3]\!]$ model exactly the same scenario.
Following the proposed approach, we can exploit such a GSPN for verifying functional and performance properties. For instance, we find that there is no deadlock but if we considered a functional description of the same scenario (i.e. the place/transition net obtained from $\mathcal{L}oc\mathcal{ON}[\![DP_3]\!]$ by assuming that each transition has the same priority), we would find the deadlock marking $\{p_{10}, p_{11}, p_{12}, p_{16}, p_{17}, p_{18}\}$: each of the three philosophers stopped thinking, moved to the table and picked up the chopstick on his right, thus waiting forever for the chopstick on his left to become free. Again, the absence of deadlock is due to the timing chosen for the various activities. In general, functional properties can be detected by computing net invariants.
The GSPN $\mathcal{L}oc\mathcal{ON}[\![DP_3]\!]$ could be fully analyzed both from the functional point of view and from the performance evaluation point of view by using tools like GreatSPN [5]. In particular, this tool permits also a simulative analysis of the modeled system.

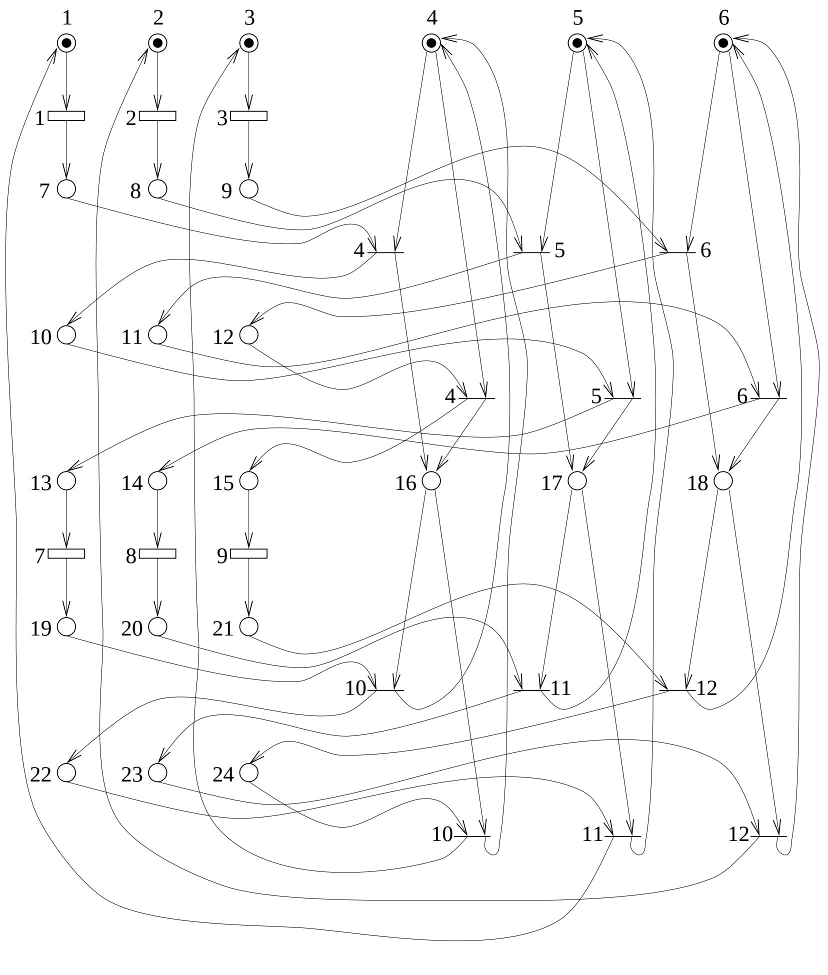


Figure 4: Operational net semantics of DP_3

4 Concluding remarks

In this paper we have introduced the stochastic process algebra EMPA, we have instantiated to it an approach for modeling and analyzing concurrent systems which integrates their different views (centralized vs. distributed) as well as the different aspects of their behavior (qualitative vs. quantitative), and finally we have applied such an approach to a given problem.

As the various semantics for EMPA can be fully mechanized, we are presently designing a software tool which associates with each EMPA term the collection of its 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, given an algebraic description of a concurrent system, the LTS representing its functional semantics can be input for tools like CW [6]; the HCTMC representing its Markovian semantics can be given as input to tools such as SHARPE [15]; and the GSPN representing its operational net semantics can be input for tools like GreatSPN [5].

Therefore, this 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.

Acknowledgements
I would like to thank Lorenzo Donatiello and Roberto Gorrieri for their helpful suggestions. This research has been partially funded by MURST, CNR and ESPRIT BRA n. 8130 LOMAPS.

References

[1] *Proceedings of the 1st Workshop on Process Algebras and Performance Modelling (PAPM '93)*, Edinburgh (UK), J. Hillston and F. Moller editors, May 1993

[2] *Proceedings of the 2nd Workshop on Process Algebras and Performance Modelling (PAPM '94)*, Erlangen (Germany), U. Herzog and M. Rettelbach editors, July 1994

[3] *Proceedings of the 3rd Workshop on Process Algebras and Performance Modelling (PAPM '95)*, Edinburgh (UK), S. Gilmore and J. Hillston editors, to appear as a special issue of the Computer Journal, June 1995

[4] M. Bernardo, L. Donatiello, R. Gorrieri, *"Integrating Performance and Functional Analysis of Concurrent Systems with EMPA"*, Technical Report TR-95-14, University of Bologna (Italy), September 1995

[5] G. Chiola, *"GreatSPN 1.5 Software Architecture"*, in Proc. of the 5th Int. Conf. on Modeling Techniques and Tools for Computer Performance Evaluation, Torino (Italy), Elsevier, pages 121-136, 1991

[6] R. Cleaveland, J. Parrow, B. Steffen, *"The Concurrency Workbench: a Semantics-Based Tool for the Verification of Concurrent Systems"*, in ACM Trans. on Programming Languages and Systems, vol. 15(1), pages 36-72, 1993

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

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

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

[10] M. A. Marsan, *"Stochastic Petri Nets: An Elementary Introduction"*, in LNCS, vol. 424, pages 1-29, Springer-Verlag, 1990

[11] M. A. Marsan, G. Balbo, G. Conte, *"A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems"*, in ACM Trans. on Computer Systems, vol. 2, pages 143-172, 1984

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

[13] E. -R. Olderog, *"Nets, Terms and Formulas"*, Cambridge University Press, Cambridge (UK), 1991

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

[15] R.A. Sahner, K.S. Trivedi, *"Reliability Modelling Using SHARPE"*, in IEEE Trans. on Reliability, vol. 13, pages 186-193, 1987