A Formal Approach to the Integration of Performance Aspects in the Modeling and Analysis of Concurrent Systems

Marco Bernardo Lorenzo Donatiello Roberto Gorrieri

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

Abstract

A formal approach for modeling and analyzing concurrent systems is proposed which integrates performance characteristics in the early stages of the design process. The approach relies on both stochastically timed process algebras and stochastically timed Petri nets in order to exploit their complementary advantages. The approach is instantiated to the case of EMPA (Extended Markovian Process Algebra), introduced together with the collection of its four semantics and the notion of equivalence that are required in order to implement the approach. Finally, the case study of the alternating bit protocol is presented to illustrate the adequacy of the approach.

1 Introduction

The desirability of taking into account the performance aspects of a concurrent system in the early stages of its design has been widely recognized [YK82, Fer86, Har86, BV88]. Nevertheless, it often happens that a concurrent system is tested for efficiency only after it has been fully designed and tested for functionality. This results in two problems. On the one hand, the detection of poor performance causes the system to be designed again, so that the cost of the project increases and the deadline for the delivery of the system might not be fulfilled. On the other hand, functionality related tests and performance related tests are carried out on two different models of the system, so that one has to make sure that these two models are consistent, i.e. they really describe (different aspects of) the same system.

In the past two decades a remarkable effort has been taking place in order to make existing formal description techniques suitable to support performance modeling and analysis. The key feature common to all of the proposals is to enhance the expressiveness of the existing formal description techniques by introducing the concept of time, represented either in a deterministic way or in a stochastic way.

Stochastically timed Petri nets (see [Ajm90] and the references therein) are probably the most successful formal description technique which accounts for functional as well as performance characteristics of concurrent systems, due to the underlying well-established theory and the related tool support. Once we get a stochastically timed Petri net as a model for a given concurrent system, both its functional and performance characteristics are described, and these can then be separately analyzed on two different projected models (a classical Petri net and a stochastic process) obtained from the same integrated model (the stochastically timed Petri net), so we are guaranteed that the projected models are consistent. However, two shortcomings still need to be addressed: lack of compositionality, i.e. the capability of constructing nets by composing smaller ones, and inability to perform an integrated analysis, i.e. an analysis carried out directly on the integrated model, which can be much more efficient as there is no need to build projected models.

Both drawbacks can be overcome by resorting to stochastically timed process algebras (see [PAPM93, PAPM94, PAPM95, PAPM96, PAPM97] and the references therein). The reason is that, first of all, stochastically timed process algebras naturally provide compositionality, since they are algebraic languages composed of 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. Second, functional and performance properties of a system modeled by means of a term of a stochastically timed process algebra can be investigated not only on two consistent projected semantic models (a transition system labeled only on the type of the actions and a stochastic process), but also directly on the integrated semantic model (a transition system labeled with both the type and the duration of the actions) provided that a suitable notion of integrated equivalence is developed.

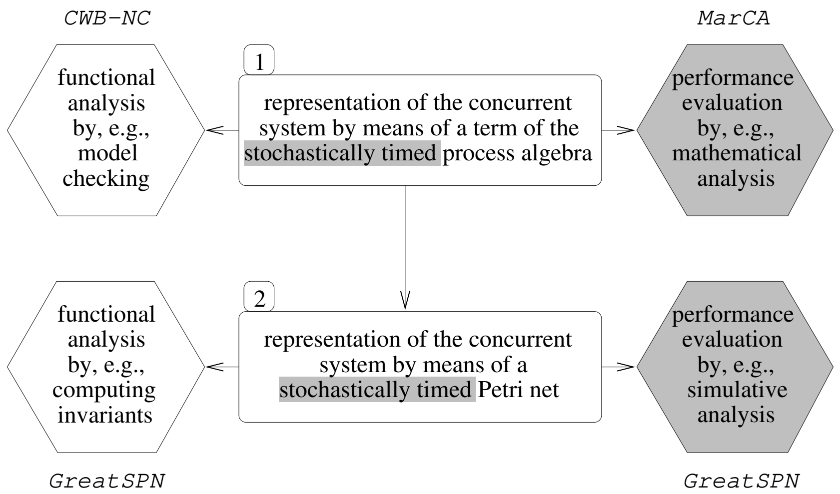


Figure 1: Integrated approach

The purpose of this paper is to combine stochastically timed process algebras and stochastically timed Petri nets so as to devise a formal approach for modeling and analyzing concurrent systems which should allow us to cope with the problems cited at the beginning of this section. Actually, the approach we are going to introduce results in three orthogonal integrations:

(*i*) The first integration relates the two different formalisms, hence two different views of concurrent systems according to [Old91]. The abstract view is provided by process terms: they give an algebraic representation of system components and their interactions, whose semantic model is obtained by interleaving actions of concurrent components. The concrete view is provided instead by Petri nets: they give a machine-like representation of systems with the explicit description of concurrency. This integration results in the two phases depicted in Fig. 1.

(*ii*) The second integration relates functional and performance aspects of concurrent systems. This integration is depicted in Fig. 1 by means of the contrast between the nonshaded part and the shaded part.

(*iii*) The third integration consists of exploiting several existing tools tailored for specific purposes in order to analyze the various models.

Let us explain in more detail the two phases in light of the three orthogonal integrations mentioned above.

1. The first phase requires the designer to specify the concurrent system as a term of the stochastically timed process algebra. Because of compositionality, the designer is allowed to develop the algebraic representation of the system in a modular way: every subsystem can be modeled separately, then these models can be put together through the operators of the algebra. From the algebraic representation, an integrated interleaving semantic model is automatically derived in the form of a transition system labeled on both the type and the duration of the actions. The integrated interleaving semantic model can be analyzed as a whole by a notion of integrated equivalence or is projected on a functional semantic model and a performance semantic model that can be analyzed by means of tools like CWB-NC [CS96] and MarCA [Ste94], respectively.

   The functional analysis can be carried out by resorting to methods such as equivalence checking, preorder checking and model checking [CPS93]. Equivalence checking verifies whether a process term meets the specification of a given system in the case when the specification is a process term as well. Preorder checking requires that the specification is still a process term treated as the minimal requirement to be met, owing to the fact that specification can contain don't care points. Model checking requires specifications to be formalized as modal logic formulas to be satisfied, expressing assertions about safety, liveness, or fairness constraints.

The performance analysis permits obtaining quantitative measures by typically resorting to the study of a Markov chain.

2. The second phase consists of automatically obtaining from the algebraic representation of the system an equivalent representation in the form of a stochastically timed Petri net. The net representation turns out to be useful whenever a less abstract representation is required highlighting dependencies, conflicts, and synchronizations among system activities, and helpful detecting some properties (e.g., partial deadlock) that can be easily checked only in a distributed setting. Additionally, the net representation is usually more compact than the integrated interleaving semantic model resulting from the algebraic representation, since concurrency is kept explicit instead of being simulated by alternative computations obtained by interleaving actions of concurrent components. The functional and performance analysis of the net representation can be assisted by tools like GreatSPN [Chi91].

   The functional analysis aims at detecting behavioral and structural properties of nets (see, e.g., [Mur89]), i.e. both properties depending on the initial marking of the net and properties depending only upon the structure of the net. Concerning structural analysis, the technique of net invariants is frequently used. Such a technique (see, e.g., [Rei85]) consists of computing the solutions of linear equation systems based on the incidence matrix of the net under consideration. These solutions single out places that do not change their token count during transition firings or indicate how often each transition has to fire in order to reproduce a given marking. By means of these solutions, properties such as boundedness, liveness, and deadlock can be studied.

   The performance analysis aims at determining efficiency measures by resorting to either the numerical solution of a Markov chain or the event driven simulation of the net.

Since the two phases above are complementary, the choice between them is made according to the adequacy of the related representation with respect to the analysis of the concurrent system under consideration and the availability of the corresponding tools. In any case, the designer is forced to start with an algebraic representation of the system in order to take advantage of compositionality of algebras and avoid graphical complexity of nets.

In order to implement the integrated approach, we have to choose a class of stochastically timed Petri nets and then a stochastically timed process algebra having possibly the same expressive power. The class of stochastically timed Petri nets we have chosen is that of Generalized Stochastic Petri Nets (GSPNs) [ABC84, ABCC87] because they have been extensively studied and successfully applied. Since in the literature there is no stochastically timed process algebra having the same expressive power as GSPNs, we have developed a new one called Extended Markovian Process Algebra (EMPA) on the basis of MTIPP [GHR93b] and PEPA [Hil96], which is endowed with expressive features typical of GSPNs. The name of the algebra stems from the fact that action durations are mainly expressed by means of exponentially distributed random variables (hence Markovian), but it is also possible to express prioritized probabilistic actions having duration zero as well as actions whose duration is unspecified (hence Extended). In order to support the various phases and analyses of the integrated approach, EMPA has been equipped with a collection of semantics as well as a notion of integrated equivalence based on ideas in [LS91, HR94, Hil96, Buc94, Tof94, Mil89], as depicted in Fig. 2. Each term has an integrated interleaving semantics represented by a labeled transition system (LTS for short) whose labels consist of both the type and the duration of the actions and an integrated net semantics represented by a GSPN. From the integrated interleaving semantic model, two projected semantic models can be obtained: a functional model given by a LTS labeled only on the type of the actions, and a performance model given by a Markov chain (MC for short).

As it can be noted, although the integrated approach has in principle a general validity, in this paper we study its instantiation to the exponential case. The restriction to exponentially distributed durations [1] sim-

[1] Actually, in Sect. 5.1 we shall see that also phase type distributions are somehow expressible with EMPA.

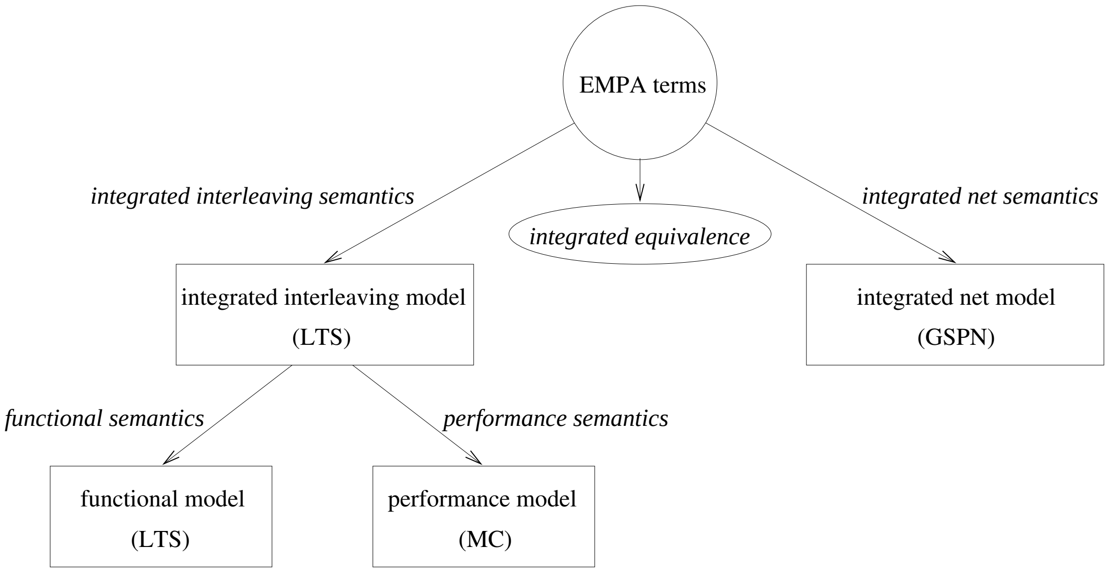
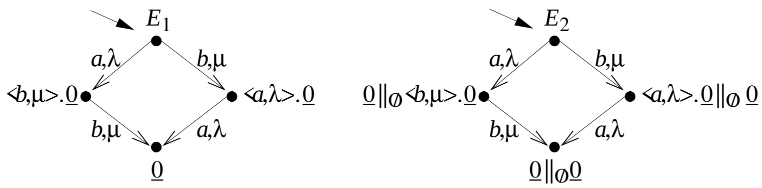


Figure 2: EMPA semantics and equivalence

plifies the performance evaluation, as the performance model turns out to be a MC. Also, such a restriction affects the semantic treatment, because the memoryless property of the exponential distribution allows us to define an integrated semantics for EMPA through the interleaving approach, in the same style as classical process algebras. For instance, suppose we are given an action a whose duration is exponentially distributed with rate λ and an action b whose duration is exponentially distributed with rate μ. Then let us consider a term E_1 that executes either a followed by b or (operator "$\_ + \_$") b followed by a depending on whether the initial a is completed before the initial b or not, and a term E_2 that executes a in parallel with (operator "$\_ \|_{\emptyset} \_$") b:

$$\begin{array}{lcl} E_1 & \equiv & <a, \lambda>.<b, \mu>.\underline{0} + <b, \mu>.<a, \lambda>.\underline{0} \\ E_2 & \equiv & <a, \lambda>.\underline{0} \,\|_{\emptyset} <b, \mu>.\underline{0} \end{array}$$

The LTSs representing their integrated interleaving semantics are isomorphic:



This is correct from the functional point of view by definition of interleaving and also from the performance point of view due to the memoryless property of the exponential distribution [Kle75]: if we assume that E_2 completes a before b, then the residual time to the completion of b is still exponentially distributed with rate μ, so the rate labeling the transition from state $\underline{0} \,\|_{\emptyset} <b, \mu>.\underline{0}$ to state $\underline{0} \,\|_{\emptyset} \underline{0}$ is μ itself instead of μ conditional on λ.

This paper, which is an extended and revised version of [BDG94a, BDG94b, BDG94c, BDG94d, BDG94e, BDG95, BDG96], is organized as follows. In Sect. 2 we introduce EMPA by giving the syntax of its terms and the meaning of its operators. In Sect. 3 the integrated interleaving semantics is defined together with the related functional semantics, while the performance semantics is presented in Sect. 4. In Sect. 5 we stress the expressiveness of EMPA, as well as the advantages of compositionality, by showing some examples about queueing systems. In Sect. 6 we briefly report on a notion of integrated equivalence presented in [BG98]. In Sect. 7 we define the integrated net semantics and we investigate the relationship with the integrated interleaving semantics. In Sect. 8 we apply the integrated approach to a case study: the alternating bit protocol. Finally, in Sect. 9 we report some concluding remarks on related work, tool support, and open problems. We would like to point out that this paper does not contain proofs of results concerned with the integrated equivalence or a comparative study of the expressive power of EMPA: the interested reader is therefore referred to the companion paper [BG98].

2 Syntax and Informal Semantics for EMPA

In this section we introduce EMPA by showing the syntax of its terms and explaining the meaning of its operators. This section is organized as follows. In Sect. 2.1 we introduce the concept of action together with a classification of actions based on their types and rates. In Sect. 2.2 we define the syntax of terms and we informally explain the meaning of each operator. Finally, in Sect. 2.3 we illustrate the execution policy we have adopted to choose among several simultaneously executable actions.

2.1 Actions: Types and Rates

The building blocks of EMPA are *actions*. Each action is a pair $<a, \tilde{\lambda}>$ consisting of the *type* of the action and the *rate* of the action. The type denotes the kind of the action (e.g. transmission of a message), while the rate indicates the speed at which the action occurs from the point of view of an external observer: rates are used as a concise way to denote the random variables specifying the duration of the actions. Depending on the type, like in classical process algebras, actions are divided into *external* and *internal* depending on whether they are observable or not: as usual, we denote by τ the only internal action type we use. Moreover, we have the following classification according to the rates:

- *Active actions* are actions whose rate is specified. An active action can be either exponentially timed or immediate:
  - *Exponentially timed actions* are actions whose rate is a positive real number. Such a number is interpreted as the parameter of the exponentially distributed random variable specifying the duration of the action. We recall that an exponentially distributed random variable X has probability distribution function $F_X(t) = \Pr[X \leq t] = 1 - e^{-\lambda \cdot t}$ for any $t \in \mathbb{R}_+$, expected value $1/\lambda$ and variance $1/\lambda^2$, thus it is uniquely identified by its parameter $\lambda \in \mathbb{R}_+$.
  - *Immediate actions* are actions whose rate, denoted by $\infty_{l,w}$, is infinite. Such actions have duration zero, and each of them is given a *priority level* $l \in \mathbb{N}_+$ and a *weight* $w \in \mathbb{R}_+$.
- *Passive actions* are actions whose rate, denoted by $*$, is undefined. The duration of a passive action is fixed only by synchronizing it with an active action of the same type.

The classification of actions based on their rates implies that exponentially timed actions model activities that are relevant from the performance point of view, immediate actions model logical events as well as activities that are either irrelevant from the performance point of view or unboundedly faster than the others, and passive actions model activities waiting for the synchronization with timed activities and allow for pure nondeterminism. While exponentially timed actions of EMPA are exactly the same as exponentially timed actions of [HR94, Hil96], immediate actions and passive actions are different from those adopted in other stochastically timed process algebras. In particular, immediate actions of EMPA, which have the same structure as immediate transitions of GSPNs, differ from the immediate actions of [HRW95] since these have neither associated priorities nor weights. Moreover, passive actions of EMPA, which resemble actions of classical process algebras, differ from both the passive actions of [HR94] since these have an associated duration and the passive actions of [Hil96] because these have an associated weight. It is worth noting that the coexistence of different kinds of actions provides EMPA with a considerable expressive power. The reader interested in a detailed comparison with process algebras including priorities, probabilities, and/or time is referred to [BG98].

We denote the set of actions by $Act = AType \times ARate$ where $AType$ is the set of types and $ARate = \mathbb{R}_+ \cup Inf \cup \{*\}$, with $Inf = \{\infty_{l,w} \mid l \in \mathbb{N}_+ \wedge w \in \mathbb{R}_+\}$, is the set of rates. We use $a, b, \ldots$ as metavariables for $AType$, $\tilde{\lambda}, \tilde{\mu}, \ldots$ for $ARate$, and $\lambda, \mu, \ldots$ for $\mathbb{R}_+$. Finally, we denote by $APLev = \{-1\} \cup \mathbb{N}$ the set of *action priority levels*, and we assume that $* < \lambda < \infty_{l,w}$ for all $\lambda \in \mathbb{R}_+$ and $\infty_{l,w} \in Inf$.

2.2 Syntax of Terms and Informal Semantics of Operators

Let *Const* be a set of *constants*, ranged over by $A, B, \ldots$, and let $ARFun = \{\varphi : AType \longrightarrow AType \mid \varphi(\tau) = \tau \wedge \varphi(AType - \{\tau\}) \subseteq AType - \{\tau\}\}$ be a set of *action relabeling functions* ranged over by $\varphi, \varphi', \ldots$.

Definition 2.1 The set $\mathcal{L}$ of *process terms* of EMPA is generated by the following syntax

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

where $L, S \subseteq AType - \{\tau\}$. The set $\mathcal{L}$ will be ranged over by $E, F, \ldots$. ■

In the rest of the section we informally explain the semantics of the operators: the formal semantics will be presented in Sect. 3.2.

The *null term* "$\underline{0}$" is the term that cannot execute any action.

The *prefix operator* "$<a, \tilde{\lambda}>.\_$" denotes the sequential composition of an action and a term: term $<a, \tilde{\lambda}>.E$ can execute action $<a, \tilde{\lambda}>$ and then behaves as term E.

The *functional abstraction operator* "$\_/L$" abstracts from the type of the actions: term E/L behaves as term E except that the type of each executed action is turned into τ whenever it is in L. The meaning of this operator is the same as that of the hiding operator of CSP [Hoa85], thereby providing a means to encapsulate or ignore functional information.

The *functional relabeling operator* "$\_[\varphi]$" changes the type of the actions: term $E[\varphi]$ behaves as term E except that the type of each executed action is modified according to φ. The meaning of this operator is the same as that of the relabeling operator of CCS [Mil89], thus providing a means to obtain more compact algebraic descriptions.

The *alternative composition operator* "$\_ + \_$" expresses a choice between two terms: term $E_1 + E_2$ behaves as either term E_1 or term E_2 depending on whether an action of E_1 or an action of E_2 is executed first. As we shall see in Sect. 2.3, the way in which the choice is resolved depends on the kind of the actions involved in the choice itself.

The *parallel composition operator* "$\_ \,\|_S\, \_$" expresses the concurrent execution of two terms according to two synchronization disciplines. The synchronization discipline on action types is the same as that of CSP [Hoa85], hence two actions can synchronize only if they have the same type, and this coincides with the resulting type. The synchronization discipline on action rates states that action $<a, \tilde{\lambda}>$ can be synchronized with action $<a, \tilde{\mu}>$ only if $\min(\tilde{\lambda}, \tilde{\mu}) = *$, and the resulting rate is given by $\max(\tilde{\lambda}, \tilde{\mu})$ up to normalization. In other words, in a synchronization at most one active action can be involved and its rate determines the rate of the synchronization itself, up to normalization as explained in Sect. 3.2. The main reason behind the adoption of such a synchronization discipline on action rates is its simplicity, both from the modeling point of view and from the semantic treatment point of view. The expressive power resulting from this apparently restrictive discipline has been investigated in [BG98].

In order to avoid ambiguities, we assume the binary operators to be left associative and we introduce the following operator precedence relation: functional abstraction = functional relabeling > prefix > alternative composition > parallel composition.

Finally, EMPA is equipped with constants as well as a set $Def : Const \twoheadrightarrow \mathcal{L}$ of related *defining equations*. In order to guarantee the correctness of recursive definitions given by means of constants, we restrict ourselves to the set $\mathcal{G}$ of closed and guarded terms [BG98].

2.3 Execution Policy

Because of the presence of binary operators such as the alternative composition and the parallel composition, the situation in which several active actions are simultaneously executable can arise. Both in the case of the alternative composition operator (due to the choice it expresses) and in the case of the parallel composition operator (as we have adopted an interleaving model, hence representing the execution of only one action at a time, which is consistent with the fact that two exponentially timed actions cannot terminate at the same time), we need a mechanism for choosing the action to be executed. In stochastically timed frameworks, such a mechanism is usually referred to as the *execution policy* [ABBCCC89].

Consider a term enabling two exponentially timed actions $<a, \lambda>$ and $<b, \mu>$. In this case we adopt the *race policy*: the action sampling the least duration succeeds. This implies that (i) the random variable describing the *sojourn time* in the state corresponding to the term above is the minimum of the exponentially distributed random variables describing the durations of the two actions, and (ii) the *execution probability* of the two actions is determined as well by the exponentially distributed random variables describing their durations. In order to compute the two quantities above, we exploit the property that the minimum of n independent exponentially distributed random variables is an exponentially distributed random variable whose rate is the sum of the n original rates [Kle75]. As a consequence, for the term above we have that the sojourn time of the corresponding state is exponentially distributed with rate $\lambda + \mu$ (hence the mean sojourn time is $1/(\lambda + \mu)$) and the execution probabilities of the two actions are $\lambda/(\lambda + \mu)$ and $\mu/(\lambda + \mu)$, respectively.

Another important consequence of the adoption of the race policy is that immediate actions take precedence over exponentially timed actions. If we consider a term enabling actions $<a, \lambda>$ and $<b, \infty_{l,w}>$, then only the latter action can be actually executed since its duration is zero whereas the former action cannot sample duration zero from its associated exponential distribution.

Consider now a term enabling two immediate actions $<a, \infty_{l,w}>$ and $<b, \infty_{l',w'}>$. Since both actions have the same duration and hence the race policy does not apply, we choose the action to execute according to the *preselection policy*: only the actions having the highest priority level are executable, and each of them is given a probability execution proportional to its own weight. The sojourn time of the state corresponding to the term above is zero. If $l > l'$ ($l' > l$), then only action $a, \infty_{l,w}$ ($b, \infty_{l',w'}$) is actually executable. If $l = l'$, then the execution probabilities of the two actions are $w/(w + w')$ and $w'/(w + w')$, respectively.

Finally, consider a term enabling two passive actions $<a, *>$ and $<b, *>$. Since the duration of passive actions is undefined, and they are assigned neither priority levels nor weights, they can be chosen according to neither the race policy nor the preselection policy. This means that passive actions can be viewed as actions of classical process algebras, hence the term above expresses a purely nondeterministic choice, where nondeterminism refers to the absence of a mechanism that specifies how the choice is resolved.

3 Integrated Interleaving Semantics of EMPA Terms

In order to implement the first phase of the integrated approach of Fig. 1, we provide each EMPA term with a formally defined integrated semantics based on LTSs whose labels consist of both the type and the rate of actions, from which two projected semantic models describing either functionality or performance can be derived.

This section is organized as follows. In Sect. 3.1 we recall some notions about LTSs since they are the semantic model in this framework. In Sect. 3.2 we define the integrated interleaving semantics of EMPA terms, we introduce the related concepts of functional and performance semantics, and we formalize the property of performance closure.

3.1 Rooted Labeled Transition Systems

In this section we recall the definition of LTS and some related notions [Par81].

Definition 3.1 A *rooted 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*;
- $\longrightarrow \subseteq S \times U \times S$ is called *transition relation*;
- $s_0 \in S$ is called the *initial state*. ■

In the graphical representation of a LTS, states are drawn as black dots and transitions are drawn as arrows between pairs of states with the appropriate labels; the initial state is pointed to by an unlabeled arrow.

Below we recall two notions of equivalence for LTSs. The first one, isomorphism, considers two LTSs to be equivalent if they have the same number of states, and any pair of corresponding states have identically labeled transitions toward any pair of corresponding states. The second one, bisimilarity, is coarser than the previous one since it considers two LTSs to be equivalent if any pair of corresponding states have identically labeled transitions toward any pair of corresponding states, regardless of the number of states.

Definition 3.2 Let $Z_1 = (S_1, U, \longrightarrow_1, s_{01})$ and $Z_2 = (S_2, U, \longrightarrow_2, s_{02})$ be two LTSs.

- Z_1 is *isomorphic* to Z_2 if and only if there exists a bijection $\beta : S_1 \longrightarrow S_2$ such that:
  - $\beta(s_{01}) = s_{02}$;
  - for each $s, s' \in S_1$ and for each $u \in U$
  $$s \xrightarrow{u}_1 s' \iff \beta(s) \xrightarrow{u}_2 \beta(s')$$
- Z_1 is *bisimilar* to Z_2 if and only if there exists a relation $\mathcal{B} \subseteq S_1 \times S_2$ such that:
  - $(s_{01}, s_{02}) \in \mathcal{B}$;
  - for each $(s_1, s_2) \in \mathcal{B}$ and for each $u \in U$
    - whenever $s_1 \xrightarrow{u}_1 s'_1$, then $s_2 \xrightarrow{u}_2 s'_2$ and $(s'_1, s'_2) \in \mathcal{B}$;
    - whenever $s_2 \xrightarrow{u}_2 s'_2$, then $s_1 \xrightarrow{u}_1 s'_1$ and $(s'_1, s'_2) \in \mathcal{B}$. ∎

3.2 Integrated Interleaving Semantics

The main problem to tackle when defining the semantics for EMPA is that the actions executable by a given term may have different priority levels, and only those having the highest priority level are actually executable. Let us call the *potential move* of a given term a pair composed of an action executable by that term when ignoring priority levels and the derivative term obtained by executing that action; let us denote by $PMove = Act \times \mathcal{G}$ the set of all the potential moves. To solve the problem above, we compute inductively the multiset [2] of the potential moves of a given term regardless of priority levels, and then we select those having the highest priority level. This is motivated in our framework by the fact that the actual executability as well as the execution probability of an action depend upon *all* the actions that are executable at the same time when it is executable: only if we know all the potential moves of a given term, we can correctly determine the transitions of the corresponding state and their rates. This is clarified by the following example.

Example 3.3 Consider term
$$E \equiv <a, \infty_{3,1}>.E_1 + <e, \infty_{2,1}>.A + <g, *>.\underline{0}$$
where
$$\begin{array}{rcl} E_1 & \equiv & <b, \lambda>.(\underline{0} \,\|_{\emptyset}\, \underline{0}) + <c, \infty_{1,1}>.E_2 \\ E_2 & \equiv & <h, \xi>.E_3 + <h, \xi>.E_3 \\ E_3 & \equiv & <d, \mu>.\underline{0} \,\|_{\{d\}}\, (<d, *>.\underline{0} \,\|_{\emptyset}\, <d, *>.\underline{0}) \\ A & \stackrel{\Delta}{=} & <f, \gamma>.A \end{array}$$
Suppose we apply to E standard semantic rules for classical process algebras, thereby disregarding priority levels, probabilities, and durations. Then we obtain the LTS in Fig. 3(a) where
$$\begin{array}{rcl} E_4 & \equiv & \underline{0} \,\|_{\{d\}}\, (\underline{0} \,\|_{\emptyset}\, <d, *>.\underline{0}) \\ E_5 & \equiv & \underline{0} \,\|_{\{d\}}\, (<d, *>.\underline{0} \,\|_{\emptyset}\, \underline{0}) \end{array}$$
where transitions are in exact correspondence with the potential moves.

Now assume that priority levels are taken into account. Then lower priority transitions must be pruned, thus resulting in the LTS in Fig. 3(b): note that the passive transition labeled with $<g, *>$ has not been discarded. The new LTS is obtained by means of an auxiliary function we shall call *Select*.

[2] We use "{|" and "|}" as brackets for multisets, "$\_ \oplus \_$" and "$\_ \ominus \_$" to denote multiset union and difference, $\mathcal{M}u_{fin}(S)$ ($\mathcal{P}_{fin}(S)$) to denote the collection of finite multisets (sets) over set S, $M(s)$ to denote the multiplicity of element s in multiset M, and $\pi_i(M)$ to denote the multiset obtained by projecting the tuples in multiset M on their i-th component. Thus, e.g., $(\pi_1(PM_2))(<a, *>)$ in the fifth part of Table 1 denotes the multiplicity of tuples of PM_2 whose first component is $<a, *>$.

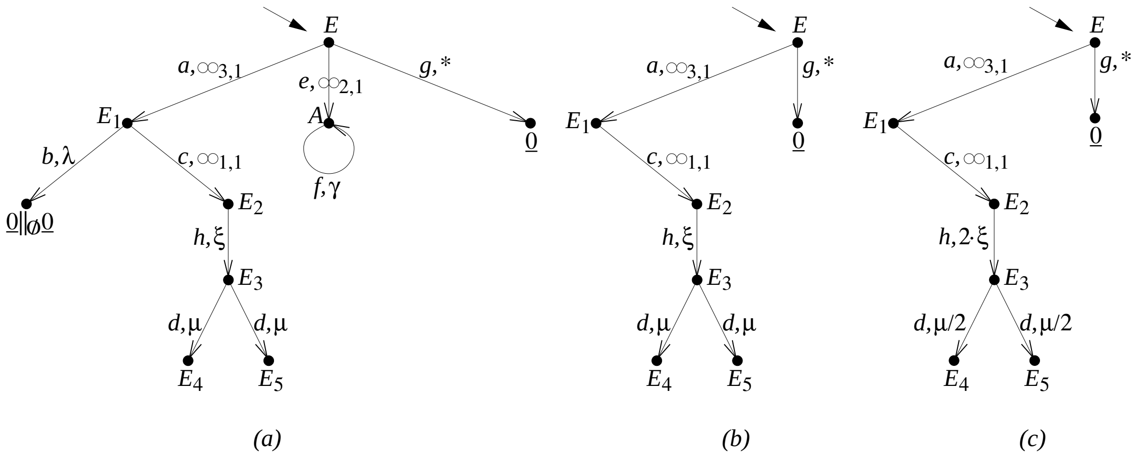


Figure 3: Integrated interleaving models for Ex. 3.3

Finally, consider the rate of the transition from E_2 to E_3 and the rates of the two transitions from E_3 to E_4 and E_5. In the correct semantic model for E, such rates have to be like in Fig. 3(c). Concerning the transition from E_2 to E_3, its rate is $2 \cdot \xi$ instead of ξ because in E_2 two exponentially timed actions with rate ξ occur and the race policy has been adopted. The problem is that both exponentially timed actions have the same type and results in the same derivative term, so with classical semantic rules only one transition is produced. The same problem arises in the case of immediate actions. To overcome this, instead of producing e.g. two transitions with two different auxiliary labels [HR94], one transition having multiplicity two (which incidentally requires the adoption of a variant of LTS as a semantic model) [Hil96], or directly one transition with the correct rate by means of auxiliary semantic rules [GHR93a], we keep track of the multiplicity of potential moves and then we construct transitions by using an auxiliary function we shall call *Melt* that merges together those potential moves having the same action type, the same priority level, and the same derivative term. The rate of transitions derived by merging potential moves is computed by means of another auxiliary function we shall call *Min* to remind the adoption of the race policy.

Concerning the transitions from E_3 to E_4 and E_5, their rate is $\mu/2$ instead of μ because in E_3 only one exponentially timed action with rate μ occurs: the value $\mu/2$ stems from the assumption that independent passive actions have the same probability to participate in a synchronization. The same considerations would hold if in E_3 we had an immediate action instead of an exponentially timed action or alternative passive actions instead of independent passive actions. In all of these cases a normalization of rates is required, and this is carried out by means of an auxiliary function we shall call *Norm*.

The reader is invited to look again at this example after examining the formal definition of the semantics, in order to verify that the LTS of Fig. 3(c) is exactly the result of the application to E of the rules in Table 1 equipped with the auxiliary functions mentioned above. ■

The formal definition of the integrated interleaving semantics for EMPA is based on the transition relation $\longrightarrow$, which is the least subset of $\mathcal{G} \times Act \times \mathcal{G}$ satisfying the inference rule in the first part of Table 1. This rule selects the potential moves that have the highest priority level (or are passive), and then merges together those having the same action type, the same priority level and the same derivative term. The first operation is carried out through functions $Select : \mathcal{M}u_{fin}(PMove) \longrightarrow \mathcal{M}u_{fin}(PMove)$ and $PL : Act \longrightarrow APLev$, which are defined in the third part of Table 1. The second operation is carried out through function $Melt : \mathcal{M}u_{fin}(PMove) \longrightarrow \mathcal{P}_{fin}(PMove)$ and partial function $Min : (ARate \times ARate) \longrightarrow\!\!\!\!\!\!\circ\;\; ARate$, which are defined in the fourth part of Table 1. We recall that function *Melt*, whose introduction is motivated by the drawback cited in the example above, avoids burdening transitions with auxiliary labels as well as keeping track of the fact that some transitions may have multiplicity greater than one. We also point out that the name *Min* should recall the adoption of the race policy: the minimum of a set of random variables has to be computed. We regard *Min* as an associative and commutative operation, thus we take the liberty to apply it to multisets of rates.

$$\frac{(<a,\tilde{\lambda}>,E') \in Melt(Select(PM(E)))}{E \xrightarrow{a,\tilde{\lambda}} E'}$$

$$PM(\underline{0}) = \emptyset$$

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

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

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

$$PM(E_1 + E_2) = PM(E_1) \oplus PM(E_2)$$

$$\begin{aligned} PM(E_1 \parallel_S E_2) = {} & \{\!|\, (<a,\tilde{\lambda}>,E'_1 \parallel_S E_2) \mid a \notin S \wedge (<a,\tilde{\lambda}>,E'_1) \in PM(E_1) \,|\!\} \oplus \\ & \{\!|\, (<a,\tilde{\lambda}>,E_1 \parallel_S E'_2) \mid a \notin S \wedge (<a,\tilde{\lambda}>,E'_2) \in PM(E_2) \,|\!\} \oplus \\ & \{\!|\, (<a,\tilde{\gamma}>,E'_1 \parallel_S E'_2) \mid a \in S \wedge \\ & \qquad (<a,\tilde{\lambda}_1>,E'_1) \in PM(E_1) \wedge \\ & \qquad (<a,\tilde{\lambda}_2>,E'_2) \in PM(E_2) \wedge \\ & \qquad \tilde{\gamma} = Norm(a,\tilde{\lambda}_1,\tilde{\lambda}_2,PM(E_1),PM(E_2)) \,|\!\} \end{aligned}$$

$$PM(A) = PM(E) \quad \text{if } A \triangleq E$$

$$\begin{aligned} Select(PM) = \{\!|\, (<a,\tilde{\lambda}>,E) \in PM \mid & \forall (<b,\tilde{\mu}>,E') \in PM \,.\, PL(<a,\tilde{\lambda}>) \geq PL(<b,\tilde{\mu}>) \vee \\ & PL(<a,\tilde{\lambda}>) = -1 \,|\!\} \end{aligned}$$

$$PL(<a,*>) = -1 \qquad PL(<a,\lambda>) = 0 \qquad PL(<a,\infty_{l,w}>) = l$$

$$\begin{aligned} Melt(PM) = \{(<a,\tilde{\lambda}>,E) \mid & \exists \tilde{\mu} \in ARate \,.\, (<a,\tilde{\mu}>,E) \in PM \wedge \\ & \tilde{\lambda} = Min\{\!|\, \tilde{\gamma} \mid (<a,\tilde{\gamma}>,E) \in PM \wedge PL(<a,\tilde{\gamma}>) = PL(<a,\tilde{\mu}>) \,|\!\}\} \end{aligned}$$

$$* \; Min \; * = * \qquad \lambda_1 \; Min \; \lambda_2 = \lambda_1 + \lambda_2 \qquad \infty_{l,w_1} \; Min \; \infty_{l,w_2} = \infty_{l,w_1+w_2}$$

$$Norm(a,\tilde{\lambda}_1,\tilde{\lambda}_2,PM_1,PM_2) = \begin{cases} Split(\tilde{\lambda}_1, 1/(\pi_1(PM_2))(<a,*>)) & \text{if } \tilde{\lambda}_2 = * \\ Split(\tilde{\lambda}_2, 1/(\pi_1(PM_1))(<a,*>)) & \text{if } \tilde{\lambda}_1 = * \end{cases}$$

$$Split(*,p) = * \qquad Split(\lambda,p) = \lambda \cdot p \qquad Split(\infty_{l,w},p) = \infty_{l,w \cdot p}$$

Table 1: Inductive rules for EMPA integrated interleaving semantics

The multiset $PM(E) \in \mathcal{M}u_{fin}(PMove)$ of potential moves of $E \in \mathcal{G}$ is defined by structural induction in the second part of Table 1 according to the intuitive meaning of operators explained in Sect. 2.2. It is worth noting that, unlike the definition of the semantics for classical process algebras, we compute all the potential moves of a term at once instead of computing one potential move at a time, since this is the most convenient way to correctly determine the transitions (*Select*) and their rates (*Melt* and *Norm*). In order to enforce the *bounded capacity assumption* [Hil94], which establishes that the rate at which an activity is carried out cannot be increased by synchronizing it with other activities, in the rule for the parallel composition operator a *normalization* is required which suitably computes the rates of potential moves resulting from the synchronization of the same active action with several independent or alternative passive actions. The normalization operates in such a way that applying *Min* to the rates of the synchronizations involving the active action gives as a result the rate of the active action itself, and that each synchronization is assigned the same execution probability. This normalization is carried out through partial function $Norm : (AType \times ARate \times ARate \times \mathcal{M}u_{fin}(PMove) \times \mathcal{M}u_{fin}(PMove)) \multimap ARate$ and function $Split : (ARate \times \mathbb{R}_{]0,1]}) \longrightarrow ARate$, which are defined in the fifth part of Table 1. Note that $Norm(a, \tilde{\lambda}_1, \tilde{\lambda}_2, PM_1, PM_2)$ is defined if and only if $\min(\tilde{\lambda}, \tilde{\mu}) = *$, which is the condition on action rates we have required in Sect. 2.2 in order for a synchronization to be permitted. The name *Split* comes from the way this function is used to calculate the performance semantics in Sect. 4.3.

Example 3.4 Consider term

$$E \equiv E_1 \,\|_{\{a\}} (E_2 \,\|_{\emptyset}\, E_3)$$

where

$$\begin{array}{lcl} E_1 & \equiv & <a, \lambda>.\underline{0} \\ E_2 & \equiv & <a, *>.\underline{0} + <a, *>.\underline{0} \\ E_3 & \equiv & <a, *>.\underline{0} \end{array}$$

Then E_1 has one potential move $(<a, \lambda>, \underline{0})$, E_2 has one potential move $(<a, *>, \underline{0})$ with multiplicity two, and E_3 has one potential move $(<a, *>, \underline{0})$. As a consequence, $E_2 \,\|_{\emptyset}\, E_3$ has both potential move $(<a, *>, \underline{0} \,\|_{\emptyset}\, E_3)$ with multiplicity two and potential move $(<a, *>, E_2 \,\|_{\emptyset}\, \underline{0})$. Therefore, when computing the potential moves for E, function *Norm* produces both $(<a, \lambda/3>, \underline{0} \,\|_{\{a\}} (\underline{0} \,\|_{\emptyset}\, E_3))$ with multiplicity two and $(<a, \lambda/3>, \underline{0} \,\|_{\{a\}} (E_2 \,\|_{\emptyset}\, \underline{0}))$, and subsequently function *Melt* produces both $(<a, 2 \cdot \lambda/3>, \underline{0} \,\|_{\{a\}} (\underline{0} \,\|_{\emptyset}\, E_3))$ and $(<a, \lambda/3>, \underline{0} \,\|_{\{a\}} (E_2 \,\|_{\emptyset}\, \underline{0}))$, as expected. ■

Definition 3.5 The *integrated interleaving semantics* of $E \in \mathcal{G}$ 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 $\uparrow E \times Act \times \uparrow E$. ■

Definition 3.6 $E \in \mathcal{G}$ is *performance closed* if and only if $\mathcal{I}[\![E]\!]$ does not contain passive transitions. We denote by $\mathcal{E}$ the set of performance closed terms of $\mathcal{G}$. ■

Borrowing the terminology of GSPNs, a state of $\mathcal{I}[\![E]\!]$ is called *tangible* if it has at least one outgoing exponentially timed transition and *vanishing* if it has at least one outgoing immediate transition. Because of function *Select*, a tangible state has only outgoing exponentially timed transitions and, possibly, passive transitions; likewise, a vanishing state has only outgoing immediate transitions of the same priority level and, possibly, passive transitions. If the term at hand is performance closed, which means that it is completely specified from the performance standpoint, then neither tangible states nor vanishing states have outgoing passive transitions.

Given a term $E \in \mathcal{G}$, its integrated interleaving semantics $\mathcal{I}[\![E]\!]$ fully represents the behavior of E because transitions are decorated by both the action type and the action rate, hence both the functional aspects and the performance aspects are described. In order to fully implement the first phase of the integrated approach of Fig. 1, we need to derive two projected semantic models concerning functionality and performance, respectively. One can think of obtaining the *functional semantics* $\mathcal{F}[\![E]\!]$ and the *performance semantics* $\mathcal{P}[\![E]\!]$ of term E from its integrated interleaving semantics $\mathcal{I}[\![E]\!]$ by simply dropping action rates and action types, respectively. As a matter of fact, this is the case for the functional semantics, and also for the performance semantics whenever only exponentially timed transitions or only immediate transitions are involved. Below we introduce the definition of the functional semantics, while the definition of the performance semantics is deferred to Sect. 4 since it requires a more careful treatment due to the possible coexistence of exponentially timed and immediate transitions.

Definition 3.7 The *functional semantics* of $E \in \mathcal{G}$ is the LTS

$$\mathcal{F}[\![E]\!] = (\uparrow E, AType, \longrightarrow_{E,\mathcal{F}}, E)$$

where $\longrightarrow_{E,\mathcal{F}}$ is the restriction of $\longrightarrow_E$ to $\uparrow E \times AType \times \uparrow E$. ■

4 Performance Semantics of EMPA Terms

In this section we complete the description of the implementation of the first phase of the integrated approach of Fig. 1 by showing the performance projection of the integrated interleaving semantics, i.e. the performance semantics, for performance closed terms only.

Since in EMPA the durations of timed actions are expressed through exponentially distributed random variables, it is natural to associate with each term a MC acting as a performance model. Given a term $E \in \mathcal{E}$, its performance semantics $\mathcal{P}[\![E]\!]$, hereafter called *Markovian semantics* and denoted by $\mathcal{M}[\![E]\!]$, is derived by adequately manipulating $\mathcal{I}[\![E]\!]$. Formally, $\mathcal{M}[\![E]\!]$ represents the state transition diagram of the MC associated with E, so it is defined as a variant of a LTS, called probabilistically rooted labeled transition system (p-LTS), in which there is no initial state but a probability mass function that specifies, for every state, the probability that it is the initial state.

This section is organized as follows. In Sect. 4.1 we introduce some notions about p-LTSs since they are the means whereby the semantic model is expressed in this framework. In Sect. 4.2 we recall some notions about MCs. In Sect. 4.3 we define the Markovian semantics of EMPA terms.

4.1 Probabilistically Rooted Labeled Transition Systems

In this section we present the definition of p-LTS as well as the related notions of p-isomorphism and p-bisimilarity we have introduced.

Definition 4.1 A *probabilistically rooted labeled transition system (p-LTS)* is a quadruple

$$(S, U, \longrightarrow, P)$$

such that:

- $S, U, \longrightarrow$ are defined as for a LTS;
- $P : S \longrightarrow \mathbb{R}_{[0,1]}$ is called *initial state probability function* and is such that $\sum_{s \in S} P(s) = 1$. ■

In the graphical representation of a p-LTS, states and transitions are drawn as in a LTS, and each state is labeled with its initial state probability unless it is zero. In this paper we consider only p-LTSs whose set of labels is contained in $\mathbb{R}_+ \cup Inf$, such that the transitions leaving a state are either all labeled with elements of $\mathbb{R}_+$ or all labeled with elements of Inf having the same priority level. The notions of equivalence for such p-LTSs (p-isomorphism and p-bisimilarity) carry over from the corresponding notions for LTSs. In particular, p-bisimilarity is developed according to [LS91], so it considers two p-LTSs to be equivalent if any pair of corresponding states have the same aggregated rate to reach the same equivalence class.

Definition 4.2 Let $Z_1 = (S_1, \mathbb{R}_+ \cup Inf, \longrightarrow_1, P_1)$ and $Z_2 = (S_2, \mathbb{R}_+ \cup Inf, \longrightarrow_2, P_2)$ be two p-LTSs.

- Z_1 is *p-isomorphic* to Z_2 if and only if there exists a bijection $\beta : S_1 \longrightarrow S_2$ such that:

– for each $s \in S_1$

$$P_1(s) = P_2(\beta(s))$$

– for each $s, s' \in S_1$ and for each $\tilde{\lambda} \in \mathbb{R}_+ \cup \mathit{Inf}$

$$s \xrightarrow{\tilde{\lambda}}_1 s' \iff \beta(s) \xrightarrow{\tilde{\lambda}}_2 \beta(s')$$

- Z_1 is *p-bisimilar* to Z_2 if and only if there exists an equivalence relation $\mathcal{B} \subseteq (S_1 \cup S_2) \times (S_1 \cup S_2)$ such that:

  – for each $C \in (S_1 \cup S_2)/\mathcal{B}$

$$\sum_{s \in C \cap S_1} P_1(s) = \sum_{s \in C \cap S_2} P_2(s)$$

  – whenever $(s_1, s_2) \in \mathcal{B} \cap (S_1 \times S_2)$, then for each $C \in (S_1 \cup S_2)/\mathcal{B}$

$$\mathit{Min}\{\!|\, \tilde{\lambda} \mid s_1 \xrightarrow{\tilde{\lambda}}_1 s_1' \wedge s_1' \in C \cap S_1 \,|\!\} = \mathit{Min}\{\!|\, \tilde{\lambda} \mid s_2 \xrightarrow{\tilde{\lambda}}_2 s_2' \wedge s_2' \in C \cap S_2 \,|\!\}$$ ■

4.2 Markov Chains

In this section we recall some notions and properties about MCs [Kle75]. We shall start with the continuous time variant.

Definition 4.3 A *continuous time Markov chain (CTMC)* is a continuous time stochastic process $X = \{X(t) \mid t \in T\}$ with discrete state space S_X such that, for each $n \in \mathbb{N}_+$, $i_0, \dots, i_{n-1}, i_n \in S_X$, $t_0, \dots, t_{n-1}, t_n \in T$ where $t_0 < \dots < t_{n-1} < t_n$, it turns out

$$\Pr\{X(t_n) = i_n \mid X(t_{n-1}) = i_{n-1} \wedge \dots \wedge X(t_0) = i_0\} = \Pr\{X(t_n) = i_n \mid X(t_{n-1}) = i_{n-1}\}$$ ■

Definition 4.4 Let X be a CTMC.

- The *transition matrix* of X from time $t \in T$ to time $t' \in T$ is matrix $\mathbf{P}_X(t, t')$ defined by
$$\mathbf{P}_X(t, t') = [\Pr\{X(t') = j \mid X(t) = i\}]_{i,j \in S_X}$$

- The *infinitesimal generator* of X at time $t \in T$ is matrix $\mathbf{Q}_X(t)$ defined by
$$\mathbf{Q}_X(t) = [q_{i,j}(t)]_{i,j \in S_X} = \lim_{\Delta t \to 0} \frac{\mathbf{P}_X(t, t + \Delta t) - \mathbf{I}}{\Delta t}$$
where $\mathbf{I}$ is the identity matrix.

- X is a *homogeneous CTMC (HCTMC)* if and only if its infinitesimal generator is independent of the time.

- The *state probability distribution function* of X at time $t \in T$ is vector $\boldsymbol{\pi}_X(t)$ defined by
$$\boldsymbol{\pi}_X(t) = [\Pr\{X(t) = i\}]_{i \in S_X}$$

- The *steady state probability distribution function* of X is vector $\boldsymbol{\pi}_X$ defined by
$$\boldsymbol{\pi}_X = \lim_{t \to \infty} \boldsymbol{\pi}_X(t)$$ ■

A HCTMC X is represented by means of its infinitesimal generator when we wish to determine its state probability distribution functions, from which performance indices of interest can be derived. Whenever the steady state probability distribution function exists, it can be determined by solving

$$\begin{array}{rcl} \boldsymbol{\pi}_X \cdot \mathbf{Q}_X & = & \mathbf{0} \\ \sum_{i \in S_X} \boldsymbol{\pi}_X[i] & = & 1 \end{array}$$

The HCTMC X can equivalently be represented by means of the p-LTS

$$(S_X, \mathbb{R}_+, \{(i, q_{i,j}, j) \in S_X \times \mathbb{R}_+ \times S_X \mid q_{i,j} > 0\}, \boldsymbol{\pi}_X(0))$$

Similar definitions and properties hold for the discrete time variant.

Definition 4.5 A *discrete time Markov chain (DTMC)* is a discrete time stochastic process $X = \{X_n \mid n \in \mathbb{N}\}$ with discrete state space S_X such that, for each $n \in \mathbb{N}_+$, $i_0, \ldots, i_{n-1}, i_n \in S_X$, it turns out

$$\Pr\{X_n = i_n \mid X_{n-1} = i_{n-1} \wedge \ldots \wedge X_0 = i_0\} = \Pr\{X_n = i_n \mid X_{n-1} = i_{n-1}\}$$ ■

Definition 4.6 Let X be a DTMC.

- The *transition matrix* of X at step $n \in \mathbb{N}$ is matrix $\mathbf{P}_X(n)$ defined by
$$\mathbf{P}_X(n) = [\Pr\{X_{n+1} = j \mid X_n = i\}]_{i,j \in S_X}$$
- X is a *homogeneous DTMC (HDTMC)* if and only if its transition matrix is independent of the time.
- The *state probability distribution function* of X at step $n \in \mathbb{N}$ is vector $\boldsymbol{\pi}_X(n)$ defined by
$$\boldsymbol{\pi}_X(n) = [\Pr\{X_n = i\}]_{i \in S_X}$$
- The *steady state probability distribution function* of X is vector $\boldsymbol{\pi}_X$ defined by
$$\boldsymbol{\pi}_X = \lim_{n \to \infty} \boldsymbol{\pi}_X(n)$$ ■

A HDTMC X is represented by means of its transition matrix when we wish to determine its state probability distribution functions, from which performance indices of interest can be derived. Whenever the steady state probability distribution function exists, it can be determined by solving

$$\begin{aligned} \boldsymbol{\pi}_X \cdot \mathbf{P}_X &= \mathbf{P}_X \\ \sum_{i \in S_X} \boldsymbol{\pi}_X[i] &= 1 \end{aligned}$$

The HDTMC X can equivalently be represented by means of the p-LTS

$$(S_X, \mathbb{R}_{]0,1]}, \{(i, p_{i,j}, j) \in S_X \times \mathbb{R}_{]0,1]} \times S_X \mid p_{i,j} > 0\}, \boldsymbol{\pi}_X(0))$$

We conclude with the notion of *ordinary lumping* [Sch84], which results in an aggregation method that allows an exact analysis of a MC to be carried out on a smaller stochastic process which still is a MC. Exact analysis refers to the fact that, whenever the steady state probability distribution function of the original MC exists, the steady state probability of each macrostate of the lumped MC is the sum of the steady state probabilities of the original states it contains. Though quite helpful, this aggregation should be avoided when it may cause information loss, e.g. as a consequence of merging together states having different weights with respect to a given performance measure. We now give the definition for the continuous time case (in the discrete time case, transition probabilities substitute for transition rates).

Definition 4.7 Let X be a HCTMC. A partition Λ of S_X is an *ordinary lumping* of X if and only if for every $C_i, C_j \in \Lambda$ and $h, l \in C_i$

$$\sum_{k \in C_j} q_{h,k} = \sum_{k \in C_j} q_{l,k}$$

If this is the case, the *ordinarily lumped* HCTMC X' obtained from X has state space Λ and infinitesimal generator $\mathbf{Q}'_{X'}$ where $q'_{i,j} = \sum_{k \in C_j} q_{h,k}$ for some $h \in C_i$. ■

It is easily seen that, if X is a MC and X' is the MC obtained from X via the ordinary lumping Λ, then the p-LTSs underlying X and X' are p-bisimilar via the reflexive, symmetric and transitive closure of the relation that associates each state of X with the state of X' that contains it.

4.3 Markovian Semantics

The Markovian semantics of a performance closed term is a HDTMC or a HCTMC depending on whether the underlying integrated interleaving semantic model has only immediate transitions or not.

Definition 4.8 Let $E \in \mathcal{E}$ be such that $\mathcal{I}[\![E]\!]$ contains only immediate transitions. The *Markovian semantics* of E is the p-LTS

$$\mathcal{M}[\![E]\!] = (\uparrow E, \mathbb{R}_{]0,1]}, \longrightarrow_{E,\mathcal{M}}, P_{E,\mathcal{M}})$$

where:

- $\longrightarrow_{E,\mathcal{M}}$ is the least subset of $\uparrow E \times \mathbb{R}_{]0,1]} \times \uparrow E$ such that $F \xrightarrow{p}_{E,\mathcal{M}} F'$ whenever
$$p = \sum \{\!| \, w \mid F \xrightarrow{a,\infty_{l,w}}_E F' \, |\!\} / \sum \{\!| \, w \mid F \xrightarrow{a,\infty_{l,w}}_E F'' \, |\!\}$$
- $P_{E,\mathcal{M}} : \uparrow E \longrightarrow \mathbb{R}_{[0,1]}$, $P_{E,\mathcal{M}}(F) = \begin{cases} 1 & \text{if } F \equiv E \\ 0 & \text{if } F \not\equiv E \end{cases}$. ■

Definition 4.9 Let $E \in \mathcal{E}$ be such that $\mathcal{I}[\![E]\!]$ contains only exponentially timed transitions. The *Markovian semantics* of E is the p-LTS

$$\mathcal{M}[\![E]\!] = (\uparrow E, \mathbb{R}_{+}, \longrightarrow_{E,\mathcal{M}}, P_{E,\mathcal{M}})$$

where:

- $\longrightarrow_{E,\mathcal{M}}$ is the least subset of $\uparrow E \times \mathbb{R}_{+} \times \uparrow E$ such that $F \xrightarrow{\lambda}_{E,\mathcal{M}} F'$ whenever
$$\lambda = \sum \{\!| \, \mu \mid F \xrightarrow{a,\mu}_E F' \, |\!\}$$
- $P_{E,\mathcal{M}} : \uparrow E \longrightarrow \mathbb{R}_{[0,1]}$, $P_{E,\mathcal{M}}(F) = \begin{cases} 1 & \text{if } F \equiv E \\ 0 & \text{if } F \not\equiv E \end{cases}$. ■

When $E \in \mathcal{E}$ is such that $\mathcal{I}[\![E]\!]$ contains both exponentially timed and immediate transitions, a HCTMC can still be derived by removing the immediate transitions and the related vanishing states, which is justified from a stochastic point of view by the fact that the sojourn time in a vanishing state is zero. We now present the algorithm transforming $\mathcal{I}[\![E]\!]$ into $\mathcal{M}[\![E]\!]$ whenever both kinds of transitions coexist. Due to its generality, such an algorithm can be regarded as an alternative to the technique of the embedded MC, which has been used e.g. to define the MC underlying a GSPN [ABC84].

The first step of the algorithm consists of

1. dropping action types,
2. removing selfloops composed of an immediate transition (hereafter called immediate selfloops for short),
3. changing the weight of each immediate transition into the corresponding execution probability, and
4. determining the initial state probability function.

Formally, from $\mathcal{I}[\![E]\!] = (\uparrow E, Act, \longrightarrow_E, E)$ we obtain the p-LTS $\mathcal{P}_1[\![E]\!] = (S_{E,1}, \mathbb{R}_+ \cup \mathit{Inf}, \longrightarrow_{E,1}, P_{E,1})$ where: [3]

- $S_{E,1} = \uparrow E$.
- Let $PM_1(s) = \mathit{Melt}(\{\!| \, (\tilde{\lambda}, s') \mid s \xrightarrow{a,\tilde{\lambda}}_E s' \, |\!\})$ for any $s \in S_{E,1}$. Then $\longrightarrow_{E,1}$ is the least subset of $S_{E,1} \times (\mathbb{R}_+ \cup \mathit{Inf}) \times S_{E,1}$ such that:
  - If s is tangible and $(\lambda, s') \in PM_1(s)$, then $s \xrightarrow{\lambda}_{E,1} s'$.

[3] With abuse of notation, we apply function Melt to multisets of pairs whose first components are rates instead of actions.

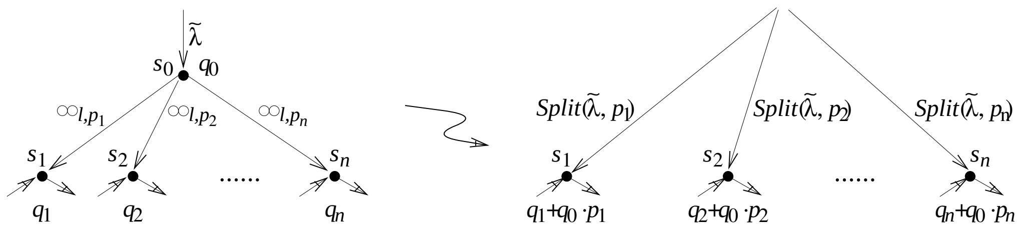


Figure 4: Graph reduction rule

  - If s is vanishing and there are exactly $m \geq 1$ potential moves (∞_{l,w_j}, s_j), $1 \leq j \leq m$, in $PM_1(s)$ such that $s_j \not\equiv s$, then there are m transitions $s \xrightarrow{\infty_{l,w_j/w}}_{E,1} s_j$, $1 \leq j \leq m$, where $w = \sum_{j=1}^{m} w_j$.

- $P_{E,1} : S_{E,1} \longrightarrow \mathbb{R}_{[0,1]}$, $P_{E,1}(s) = \begin{cases} 1 & \text{if } s \equiv E \\ 0 & \text{if } s \not\equiv E \end{cases}$.

The k-th step, $k \geq 2$, consists of applying the graph reduction rule in Fig. 4 to a given vanishing state $s_0 \in S_{E,k-1}$. With this step we thus consider a fork of immediate transitions that is treated by

1. eliminating the related vanishing state as well as the immediate transitions themselves,
2. splitting the transitions entering the state upstream the fork,
3. removing immediate selfloops created by splitting immediate transitions leaving one of the states downstream the fork and entering the state upstream the fork, and
4. distributing the initial state probability associated with the state upstream the fork among the states downstream the fork.

Formally, if we assume that the vanishing state considered at the k-th step is the one in Fig. 4, we build the p-LTS $\mathcal{P}_k[\![E]\!] = (S_{E,k}, \mathbb{R}_+ \cup Inf, \longrightarrow_{E,k}, P_{E,k})$ where:

- $S_{E,k} = S_{E,k-1} - \{s_0\}$.
- Let $PM_k(s) = Melt(\{\!| (\tilde{\lambda}, s') \mid s \xrightarrow{\tilde{\lambda}}_{E,k-1} s' \wedge s' \not\equiv s_0 |\!\} \oplus \{\!| (Split(\tilde{\lambda}, p_i), s_i) \mid s \xrightarrow{\tilde{\lambda}}_{E,k-1} s_0 \wedge 1 \leq i \leq n |\!\})$ for any $s \in S_{E,k}$. Then $\longrightarrow_{E,k}$ is the least subset of $S_{E,k} \times (\mathbb{R}_+ \cup Inf) \times S_{E,k}$ such that:
  - If s is tangible, or vanishing but $s \notin \{s_i \mid 1 \leq i \leq n\}$, and $(\tilde{\lambda}, s') \in PM_k(s)$, then $s \xrightarrow{\tilde{\lambda}}_{E,k} s'$.
  - If s is vanishing, $s \equiv s_i$ and there are exactly $m \geq 1$ potential moves (∞_{l,p_j}, s_j), $1 \leq j \leq m$, in $PM_k(s)$ such that $s_j \not\equiv s$, then there are m transitions $s \xrightarrow{\infty_{l,p_j/p}}_{E,k} s_j$, $1 \leq j \leq m$, where $p = \sum_{j=1}^{m} p_j$.
- $P_{E,k} : S_{E,k} \longrightarrow \mathbb{R}_{[0,1]}$, $P_{E,k}(s) = \begin{cases} P_{E,k-1}(s) & \text{if } s \notin \{s_i \mid 1 \leq i \leq n\} \\ P_{E,k-1}(s) + P_{E,k-1}(s_0) \cdot p_i & \text{if } s \equiv s_i \end{cases}$.

Definition 4.10 Let $E \in \mathcal{E}$ be such that $\mathcal{I}[\![E]\!]$ contains both exponentially timed and immediate transitions. The *Markovian semantics* of E is the p-LTS

$$\mathcal{M}[\![E]\!] = (S_{E,\mathcal{M}}, \mathbb{R}_+, \longrightarrow_{E,\mathcal{M}}, P_{E,\mathcal{M}})$$

obtained by applying the algorithm above. ■

We conclude by proving the correctness of the algorithm.

Theorem 4.11 Let $E \in \mathcal{E}$ be such that $\mathcal{I}[\![E]\!]$ contains both exponentially timed and immediate transitions.

(i) For every $k \in \mathbb{N}_+$ and $s \in S_{E,k}$ vanishing, $\sum \{\!| p \mid s \xrightarrow{\infty_{l,p}}_{E,k} s' |\!\} = 1$.

(ii) For every $k \in \mathbb{N}_+$, $\sum_{s \in S_{E,k}} P_{E,k}(s) = 1$.

(iii) The elimination of immediate selfloops is correct from the performance viewpoint.

(iv) $\mathcal{M}[\![E]\!]$ is unique.

(v) If $\mathcal{I}[\![E]\!]$ has finitely many states, then the algorithm terminates after $O(|\uparrow E|)$ steps.

Proof Let $E \in \mathcal{E}$ be such that $\mathcal{I}[\![E]\!]$ contains both exponentially timed and immediate transitions.

(i) We proceed by induction on $k \in \mathbb{N}_+$:

- If $k = 1$ then the result immediately follows from the definition of $\longrightarrow_{E,1}$.
- Let $k > 1$ and let the result hold for $k-1$. Suppose that the fork considered at step k is the one depicted in Fig. 4, and let $s \in S_{E,k}$:
  - If $s \notin \{s' \in S_{E,k-1} \mid s' \xrightarrow{\infty_{l',p_0}}_{E,k-1} s_0\}$ then either s is tangible (hence the result is not concerned with it), or s is vanishing but none of its immediate transitions enters s_0, so the result holds by the induction hypothesis or, if it is downstream the fork, by the renormalization performed at step k.
  - Let $s \in \{s' \in S_{E,k-1} \mid s' \xrightarrow{\infty_{l',p_0}}_{E,k-1} s_0\}$. If s is downstream the fork, then the result trivially follows by the renormalization carried out at step k. Assume that s is not downstream the fork. From the induction hypothesis it follows that

$$\begin{aligned}
&\textstyle\sum \{\!| p \mid s \xrightarrow{\infty_{l',p}}_{E,k} s' |\!\} = \\
&\quad \textstyle\sum \{\!| p \mid s \xrightarrow{\infty_{l',p}}_{E,k} s' \wedge s' \not\equiv s_0 |\!\} + \sum \{\!| p_0 \cdot p_i \mid s \xrightarrow{\infty_{l',p_0 \cdot p_i}}_{E,k} s_i |\!\} = \\
&\quad \textstyle\sum \{\!| p \mid s \xrightarrow{\infty_{l',p}}_{E,k-1} s' \wedge s' \not\equiv s_0 |\!\} + p_0 \cdot \sum \{\!| p_i \mid s_0 \xrightarrow{\infty_{l,p_i}}_{E,k-1} s_i |\!\} = \\
&\quad \textstyle\sum \{\!| p \mid s \xrightarrow{\infty_{l',p}}_{E,k-1} s' \wedge s' \not\equiv s_0 |\!\} + p_0 = \\
&\quad \textstyle\sum \{\!| p \mid s \xrightarrow{\infty_{l',p}}_{E,k-1} s' |\!\} = 1
\end{aligned}$$

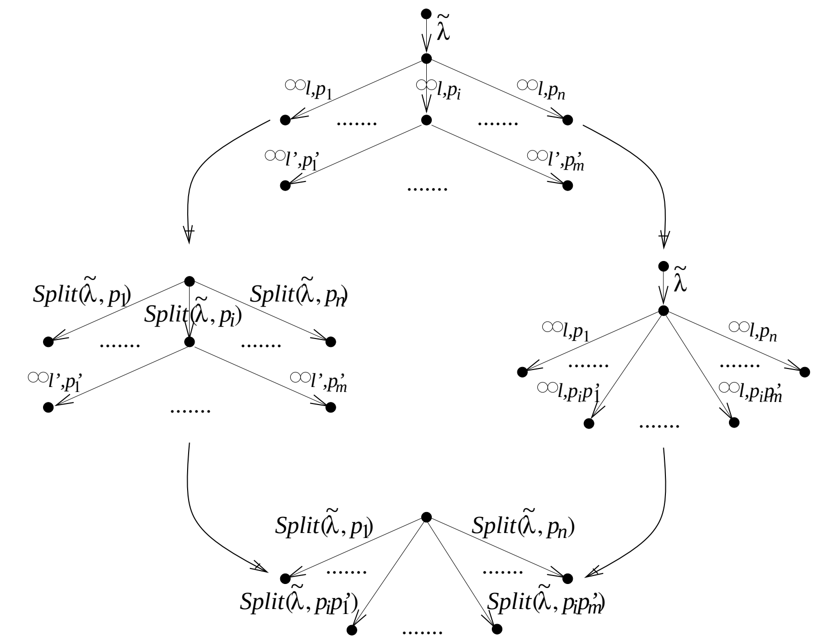
(ii) We proceed by induction on $k \in \mathbb{N}_+$:

- If $k = 1$ then the result immediately follows from the definition of $P_{E,1}$.
- Let $k > 1$ and let the result hold for $k-1$. Suppose that the fork considered at step k is the one depicted in Fig. 4. From the induction hypothesis and (i) it follows that

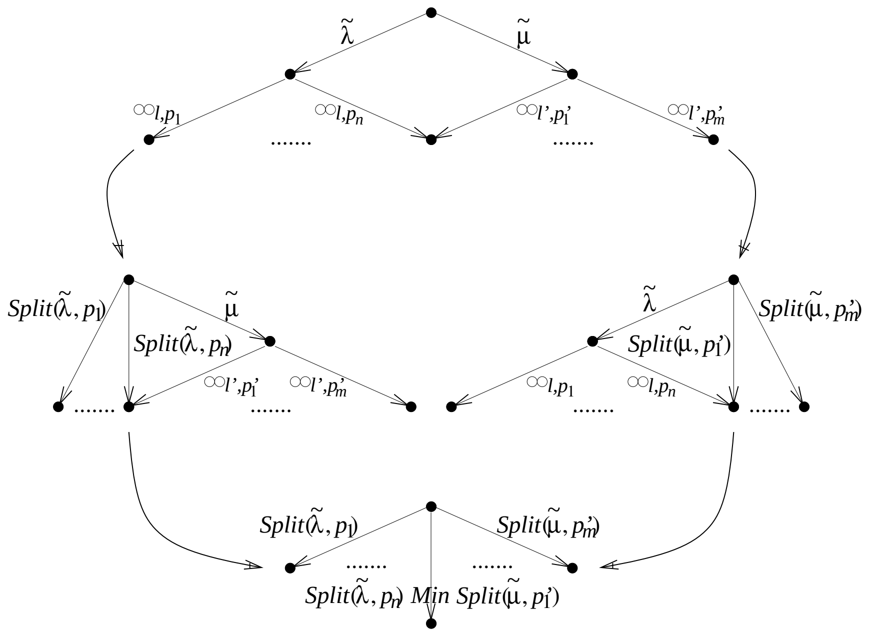
$$\begin{aligned}
&\textstyle\sum_{s \in S_{E,k}} P_{E,k}(s) = \\
&\quad \textstyle\sum_{s \in S_{E,k} - \{s_i \mid 1 \le i \le n\}} P_{E,k-1}(s) + \sum_{1 \le i \le n} (P_{E,k-1}(s_i) + P_{E,k-1}(s_0) \cdot p_i) = \\
&\quad \textstyle\sum_{s \in S_{E,k} - \{s_i \mid 1 \le i \le n\}} P_{E,k-1}(s) + \sum_{1 \le i \le n} P_{E,k-1}(s_i) + P_{E,k-1}(s_0) \cdot \sum_{1 \le i \le n} p_i = \\
&\quad \textstyle\sum_{s \in S_{E,k}} P_{E,k-1}(s) + P_{E,k-1}(s_0) = \\
&\quad \textstyle\sum_{s \in S_{E,k-1}} P_{E,k-1}(s) = 1
\end{aligned}$$

(iii) Let us modify the fork of immediate transitions depicted in Fig. 4 by assuming that s_0 has also an immediate selfloop labeled with $\infty_{l,q}$, where $\sum_{i=1}^{n} p_i + q = 1$ due to (i). Let us unfold the immediate selfloop by introducing the set of states $\{s_{0,j} \mid j \in \mathbb{N}_+\}$ such that:

- the immediate selfloop is replaced by a transition labeled with $\infty_{l,q}$ from s_0 to $s_{0,1}$;
- for all $j \in \mathbb{N}_+$, $s_{0,j}$ has a transition labeled with ∞_{l,p_i} reaching s_i, and a transition labeled with $\infty_{l,q}$ reaching $s_{0,j+1}$.



(a)



(b)

Figure 5: Confluence of the graph reduction rule in absence of immediate selfloops

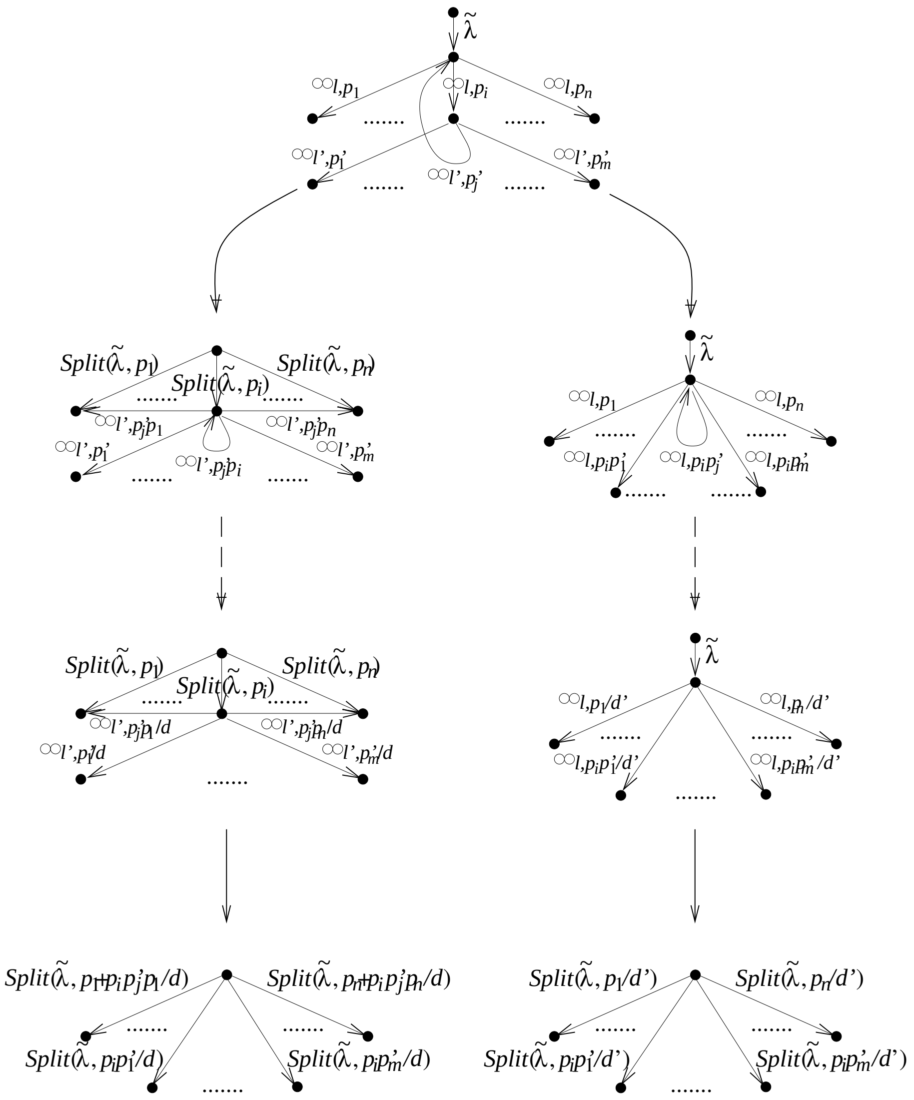


Figure 6: Confluence of the graph reduction rule in presence of immediate selfloops

Starting from s_0, the probability of reaching $s_{0,j}$ after j transition executions is q^j, while the probability of reaching s_i within j transition executions is $\sum_{h=0}^{j-1} p_i \cdot q^h$. As j grows, these probabilities approach 0 and $p_i/(1-q) = p_i / \sum_{r=1}^{n} p_r$, respectively.

(iv) The uniqueness of $\mathcal{M}[\![E]\!]$ stems from the confluence of the graph reduction rule in Fig. 4. To prove confluence, we proceed by induction on the length of the longest cycle of immediate transitions in $\mathcal{I}[\![E]\!]$.

- If the length of the longest cycle of immediate transitions is $c \leq 1$, then the first step eliminates all the cycles of immediate transitions (if any). In this case, at each step no immediate selfloop arises, thus making unnecessary the possible renormalization of execution probabilities at states downstream the fork. The confluence of the graph reduction rule then follows. Given two forks of immediate transitions, there are the three cases below:
  * There exists a state downstream a fork and upstream the other fork. Fig. 5(a) shows that confluence holds in this case. This is achieved by property $Split(Split(\tilde{\lambda}, p), p') = Split(\tilde{\lambda}, p \cdot p')$.
  * There exists at least one state downstream both forks. Fig. 5(b) shows that confluence holds in this case as well.
  * There is no state shared by the two forks. In such a case, it is obvious that the order in which the two forks are considered is irrelevant.
- Suppose that the length of the longest cycle of immediate transitions is $c \geq 2$, and assume that the result holds whenever the length of the longest cycle of immediate transitions is $< c$. Consider the application of the graph reduction rule to one of the states in the cycle:
  * If no immediate selfloop arises, the confluence is preserved by this step as shown above.
  * If an immediate selfloop arises, the confluence is still preserved by this step as shown in Fig. 6 due to property $Split(\tilde{\lambda}, p)\ Min\ Split(\tilde{\lambda}, p') = Split(\tilde{\lambda}, p + p')$. In fact, by exploiting (i), it turns out that

$$\begin{aligned} d = {} & \textstyle\sum_{1 \leq h \leq n \wedge h \neq i} p'_j \cdot p_h + \sum_{1 \leq r \leq m \wedge r \neq j} p'_r = \\ & \textstyle\sum_{1 \leq h \leq n} p'_j \cdot p_h - p'_j \cdot p_i + \sum_{1 \leq r \leq m \wedge r \neq j} p'_r = \\ & \textstyle p'_j - p'_j \cdot p_i + \sum_{1 \leq r \leq m \wedge r \neq j} p'_r = \\ & \textstyle\sum_{1 \leq r \leq m} p'_r - p'_j \cdot p_i = 1 - p'_j \cdot p_i \end{aligned}$$

and

$$\begin{aligned} d' = {} & \textstyle\sum_{1 \leq h \leq n \wedge h \neq i} p_h + \sum_{1 \leq r \leq m \wedge r \neq j} p_i \cdot p'_r = \\ & \textstyle\sum_{1 \leq h \leq n} p_h - p_i + \sum_{1 \leq r \leq m} p_i \cdot p'_r - p_i \cdot p'_j = \\ & 1 - p_i + p_i - p_i \cdot p'_j = 1 - p_i \cdot p'_j \end{aligned}$$

and for each $h = 1, \dots, n$ such that $h \neq i$

$$\begin{aligned} p_h + p_i \cdot p'_j \cdot p_h / d = {} & p_h (1 + p_i \cdot p'_j / (1 - p'_j \cdot p_i)) = \\ & p_h (1 - p'_j \cdot p_i + p_i \cdot p'_j)/(1 - p'_j \cdot p_i) = p_h / d' \end{aligned}$$

  The effect of such an application of the graph reduction rule is to shorten the longest cycle of immediate transitions, so the induction hypothesis can be exploited.

(v) If $\mathcal{I}[\![E]\!]$ has finitely many states, then $\mathcal{I}[\![E]\!]$ has finitely many transitions because E is guardedly closed. Therefore, the first phase of the algorithm terminates and the number of steps is bounded by the number of vanishing states in $\uparrow E$. ■

5 Describing Queueing Systems with EMPA

Before continuing with the presentation of the integrated approach of Fig. 1, we wish to dwell upon EMPA. The purpose of this section is to stress that an algebraic formalism like EMPA provides the designer with a *compositional linguistic support* which is usually lacking in the performance evaluation field, thereby easing the modeling process. As an example, we shall consider a full overview of well known system models such as queueing systems with memoryless arrival and service processes (some of which have already been described e.g. in [GHR93b, Hil96]), in order to exercise all the expressive capabilities of EMPA.

A *queueing system (QS)* [Kle75] is a model largely used for performance evaluation purposes to represent a service center composed of a waiting queue and a given number of servers, which provide a certain service (following a given discipline) to the customers arriving at the service center. For example, a QS $M/M/n/q/m$ with arrival rate λ and service rate μ is defined as follows:

1. The customer interarrival time is exponentially distributed with rate λ.

2. The customer service time is exponentially distributed with rate μ.

3. There are n independent servers.

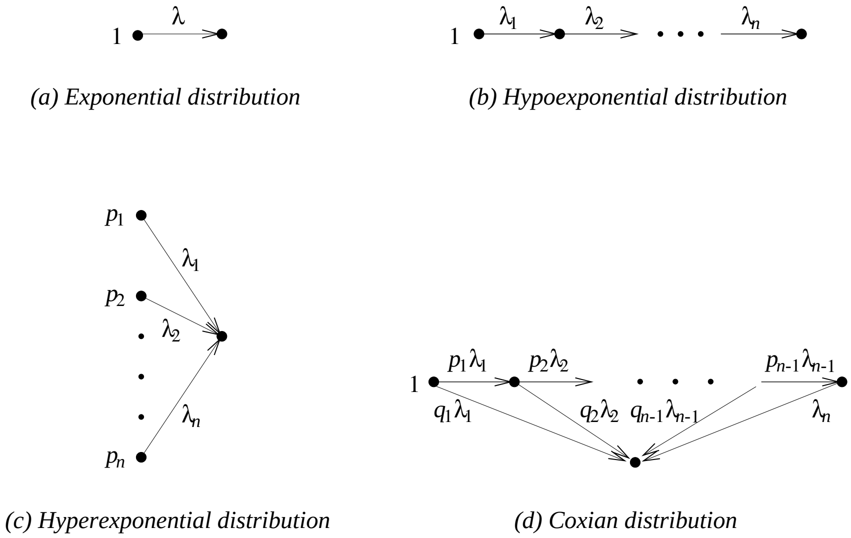
4. There is a FIFO queue with $q - n$ seats.

5. There are m independent customers.

Since the customer arrival process and the customer service process are described as stochastic processes, in Sect. 5.1 we show how to express with EMPA some frequently occurring probability distributions. Then, in Sect. 5.2 we model a QS $M/M/1/q$, and we show that its underlying HCTMC coincides with the Markovian semantics of the algebraic description in order to stress the correctness of the semantics itself. Afterwards, we complicate the model by allowing for a service rate which depends on the workload of the system (Sect. 5.3), by introducing customers requiring different service times (Sect. 5.4) or having different priorities (Sect. 5.5), by considering the service request of each customer as being composed of several subrequests to be processed in parallel after being split and before being rejoined (Sect. 5.6), and by considering a network of QSs instead of a single one where the routing of customers is probabilistic. In each of the cases above we shall succeed to get the desired EMPA model from the algebraic model of the QS $M/M/1/q$ thanks to compositionality and the powerful interplay of the three different kinds of actions.

5.1 Phase Type Distributions

In EMPA it is possible to directly express only actions having exponentially distributed durations as well as zero durations. However, it is worth noting that through the interplay of exponentially timed actions and immediate actions, all the phase type distributions are expressible by means of EMPA.



(a) Exponential distribution (b) Hypoexponential distribution

(c) Hyperexponential distribution (d) Coxian distribution

Figure 7: Phase type distributions

A *phase type distribution* [Neu81] is a continuous distribution function describing the time to absorption in a finite state HCTMC having exactly one absorbing state. Well known examples of phase type distributions are the exponential distribution, the hypoexponential distribution, the hyperexponential distribution, and finally the Coxian distribution, which are characterized in terms of time to absorption in a finite state HCTMC with an absorbing state as outlined in Fig. 7. Since an absorbing state can be modeled by term $\underline{0}$, the distributions above can be easily represented by means of series parallel combinations of exponentially timed actions as follows:

- An exponential distribution with rate $\lambda \in \mathbb{R}_+$ can be modeled by means of term
$$Exp_\lambda \triangleq <a, \lambda>.\underline{0}$$
whose Markovian semantics is p-isomorphic to the HCTMC in Fig. 7(a).

- An n-stage hypoexponential distribution with rates $\lambda_i \in \mathbb{R}_+$, $1 \leq i \leq n$, can be modeled by means of the set of inductively defined terms
$$\begin{aligned} Hypoexp_{m,\lambda_1,\dots,\lambda_m} &\triangleq <a, \lambda_1>.Hypoexp_{m-1,\lambda_2,\dots,\lambda_m}, \quad 2 \leq m \leq n, \\ Hypoexp_{1,\lambda} &\triangleq Exp_\lambda \end{aligned}$$
whose Markovian semantics is p-isomorphic to the HCTMC in Fig. 7(b).

- An n-stage hyperexponential distribution with rates $\lambda_i \in \mathbb{R}_+$, $1 \leq i \leq n$, and branching probabilities $p_i \in \mathbb{R}_{]0,1]}$, $1 \leq i \leq n$, where $\sum_{i=1}^{n} p_i = 1$, can be modeled by means of the set of inductively defined terms
$$\begin{aligned} Hyperexp_{n,\lambda_1,\dots,\lambda_n,p_1,\dots,p_n} &\triangleq H_{n,\lambda_1,\dots,\lambda_n,p_1,\dots,p_n}, \\ H_{m,\lambda_1,\dots,\lambda_m,p_1,\dots,p_m} &\triangleq H_{m-1,\lambda_1,\dots,\lambda_{m-1},p_1,\dots,p_{m-1}} + <a, \infty_{1,p_m}>.Exp_{\lambda_m}, \quad 2 \leq m \leq n, \\ H_{1,\lambda,p} &\triangleq <a, \infty_{1,p}>.Exp_\lambda \end{aligned}$$
whose Markovian semantics is p-isomorphic to the HCTMC in Fig. 7(c).

- An n-stage Coxian distribution with rates $\lambda_i \in \mathbb{R}_+$, $1 \leq i \leq n$, and branching probabilities $p_i, q_i \in \mathbb{R}_{]0,1]}$ where $p_i + q_i = 1$, $1 \leq i \leq n-1$, can be modeled by means of the set of inductively defined terms
$$\begin{aligned} Cox_{m,\lambda_1,\dots,\lambda_m,p_1,\dots,p_{m-1},q_1,\dots,q_{m-1}} &\triangleq <a, \lambda_1>.(<a, \infty_{1,q_1}>.\underline{0} + \\ &\qquad <a, \infty_{1,p_1}>.Cox_{m-1,\lambda_2,\dots,\lambda_m,p_2,\dots,p_{m-1},q_2,\dots,q_{m-1}}), \quad 2 \leq m \leq n, \\ Cox_{1,\lambda} &\triangleq Exp_\lambda \end{aligned}$$
whose Markovian semantics is p-isomorphic to the HCTMC in Fig. 7(d).

The capability of expressing phase type distributions is quite important since many frequently occurring distribution functions are such or can be approximated by means of them. However, it must be noticed that in EMPA phase type distributions cannot be described in a direct manner, so they have to be used carefully. For example, if we consider term $Exp_\lambda + Hyperexp_{2,\lambda_1,\lambda_2,p_1,p_2}$ then we realize that the right hand side term takes precedence over the left hand side term, so the whole term cannot be used to express the choice between an activity whose duration is exponentially distributed and another activity whose duration is hyperexponentially distributed. To overcome this drawback, the system designer should be enabled to describe directly any distribution, as we shall discuss in Sect. 9(8).

5.2 Queueing Systems M/M/1/q

In this section we concentrate on QSs $M/M/1/q$: the absence of the value of the fifth parameter means that the number of customers is unbounded. How can we model a QS $M/M/1/q$ with arrival rate λ and service rate μ? Let a be the action type "a customer arrives at the queue of the service center", d be the action type "a customer is delivered by the queue to the server", and s be the action type "a customer is served by the server". Then the QS under consideration can be modeled with EMPA as follows:

- $QS_{M/M/1/q} \triangleq Arrivals \,\|_{\{a\}} (Queue_0 \,\|_{\{d\}}\, Server)$:
  - $Arrivals \triangleq <a, \lambda>.Arrivals$;

– $Queue_0 \triangleq \lt a, *\gt.Queue_1$,
$Queue_h \triangleq \lt a, *\gt.Queue_{h+1} + \lt d, *\gt.Queue_{h-1}, \quad 0 < h < q-1$,
$Queue_{q-1} \triangleq \lt d, *\gt.Queue_{q-2}$;

– $Server \triangleq \lt d, \infty_{1,1}\gt.\lt s, \mu\gt.Server$.

It is worth noting that we have described the whole system as the composition of the arrival process with the composition of the queue and the server (using action types a and d as interfaces among components), and that then we have separately modeled the arrival process, the queue, and the server. Since the queue is independent of both the arrival rate and the service rate, passive actions have been exploited to represent it. As a consequence, if we want to modify the description by changing the arrival rate or the service rate, only component *Arrivals* or *Server* needs to be modified while component *Queue* is not affected. Additionally, the delivery of a customer to the server can be neglected from the performance point of view: this is achieved by means of the immediate action in component *Server*.

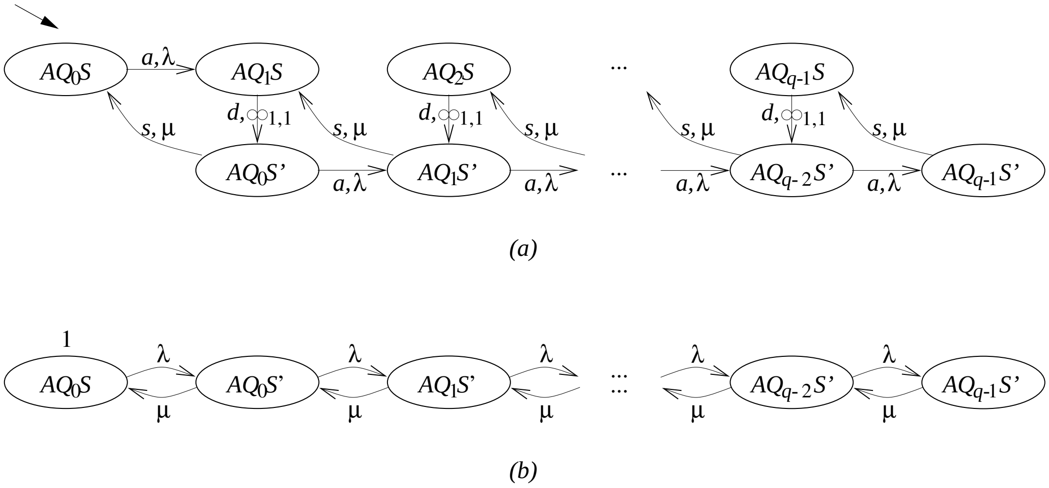


Figure 8: Semantic models of $QS_{M/M/1/q}$

We conclude by showing the HCTMC $\mathcal{M}[\![QS_{M/M/1/q}]\!]$ in Fig. 8*(b)*, which is obtained from the LTS $\mathcal{I}[\![QS_{M/M/1/q}]\!]$ in Fig. 8*(a)*, where AQ_hS stands for $Arrivals \,\|_{\{a\}} (Queue_h \,\|_{\{d\}}\, Server)$, AQ_hS' stands for $Arrivals \,\|_{\{a\}} (Queue_h \,\|_{\{d\}} \lt s, \mu\gt.Server)$, and $0 \leq h \leq q-1$. We observe that $\mathcal{M}[\![QS_{M/M/1/q}]\!]$ is p-isomorphic to the HCTMC underlying a queueing system $M/M/1/q$ [Kle75]. In [BDG94b] we proved that this result holds for each QS of the class M/M up to ordinary lumping, thus supporting our claim that we have captured the correct Markovian semantics.

5.3 Queueing Systems M/M/1/q with Scalable Service Rate

Assume that a QS $M/M/1/q$ with arrival rate λ provides service at a speed depending on the number of customers in the queue. Let us denote by μ the basic service rate, and by $sf : \mathbb{N}_+ \longrightarrow \mathbb{R}_+$ the function describing the scaling factor. This QS can be modeled as follows:

- $SSRQS_{M/M/1/q} \triangleq Arrivals \,\|_{\{a\}} (Queue_0 \,\|_{\{d_h \mid 1 \leq h \leq q-1\}}\, Server)$:

  – $Arrivals \triangleq \lt a, \lambda\gt.Arrivals$;

  – $Queue_0 \triangleq \lt a, *\gt.Queue_1$,
  $Queue_h \triangleq \lt a, *\gt.Queue_{h+1} + \lt d_h, *\gt.Queue_{h-1}, \quad 0 < h < q-1$,
  $Queue_{q-1} \triangleq \lt d_{q-1}, *\gt.Queue_{q-2}$;

- $Server \stackrel{\Delta}{=} <d_1, \infty_{1,1}>.Server_1 + \ldots + <d_{q-1}, \infty_{1,1}>.Server_{q-1}$:
  - $Server_h \stackrel{\Delta}{=} <s, sf(h) \cdot \mu>.Server, \quad 1 \leq h \leq q-1.$

It is worth noting that the structure of $SSRQS_{M/M/1/q}$ is the same as that of $QS_{M/M/1/q}$. Only component *Server* has been significantly modified in order to be able to provide service at a rate depending on the queue length.

5.4 Queueing Systems M/M/1/q with Different Service Rates

Assume that a QS $M/M/1/q$ must serve two different types of customers. Both types are characterized by the same arrival rate λ, but red customers require a service rate μ_r whereas black customers require a service rate μ_b. Such a situation can arise, e.g., in a computer system where the central unit can be viewed as the server and the various devices can be viewed as the customers. In this case, service requests can arrive from several different points and may require different service rates; the type associated with each request singles out the routing of the request itself. This QS can be modeled as follows:

- $DSRQS_{M/M/1/q} \stackrel{\Delta}{=} Arrivals \,\|_{\{a_r, a_b\}} (Queue_{\epsilon} \,\|_{\{d_r, d_b\}} \, Server)$:
  - $Arrivals \stackrel{\Delta}{=} <a_r, \lambda>.Arrivals + <a_b, \lambda>.Arrivals;$
  - $Queue_{\epsilon} \stackrel{\Delta}{=} <a_r, *>.Queue_r + <a_b, *>.Queue_b,$
    $Queue_{rw} \stackrel{\Delta}{=} <a_r, *>.Queue_{rwr} + <a_b, *>.Queue_{rwb} + <d_r, *>.Queue_w, \quad 0 \leq |w| < q-2,$
    $Queue_{bw} \stackrel{\Delta}{=} <a_r, *>.Queue_{bwr} + <a_b, *>.Queue_{bwb} + <d_b, *>.Queue_w, \quad 0 \leq |w| < q-2,$
    $Queue_{rw} \stackrel{\Delta}{=} <d_r, *>.Queue_w, \quad |w| = q-2,$
    $Queue_{bw} \stackrel{\Delta}{=} <d_b, *>.Queue_w, \quad |w| = q-2;$
  - $Server \stackrel{\Delta}{=} <d_r, \infty_{1,1}>.<s_r, \mu_r>.Server + <d_b, \infty_{1,1}>.<s_b, \mu_b>.Server.$

Again, note that the structure of $DSRQS_{M/M/1/q}$ is the same as that of $QS_{M/M/1/q}$. Only the components have been locally modified in order to be able to treat the two types of customers.

5.5 Queueing Systems M/M/1/q with Different Priorities

Assume that a QS $M/M/1/q$ must serve two different types of customers characterized by the same arrival rate λ and the same service rate μ. Red customers are assigned a priority level $r > b$, where b is the priority level assigned to black customers. There are two cases.

In the first case, we assume that the priority mechanism only affects the queueing discipline, i.e. we assume that possible preemption on the customer being served cannot be exercised. This QS can be modeled as follows:

- $PQS_{M/M/1/q} \stackrel{\Delta}{=} Arrivals \,\|_{\{a_r, a_b\}} (Queue_{0,0} \,\|_{\{d_r, d_b\}} \, Server)$:
  - $Arrivals \stackrel{\Delta}{=} <a_r, \lambda>.Arrivals + <a_b, \lambda>.Arrivals;$
  - $Queue_{0,0} \stackrel{\Delta}{=} <a_r, *>.Queue_{1,0} + <a_b, *>.Queue_{0,1},$
    $Queue_{i,0} \stackrel{\Delta}{=} <a_r, *>.Queue_{i+1,0} + <a_b, *>.Queue_{i,1} + <d_r, *>.Queue_{i-1,0}, \quad 0 < i < q-1,$
    $Queue_{0,j} \stackrel{\Delta}{=} <a_r, *>.Queue_{1,j} + <a_b, *>.Queue_{0,j+1} + <d_b, *>.Queue_{0,j-1}, \quad 0 < j < q-1,$
    $Queue_{i,j} \stackrel{\Delta}{=} <a_r, *>.Queue_{i+1,j} + <a_b, *>.Queue_{i,j+1} +$
    $\qquad <d_r, *>.Queue_{i-1,j} + <d_b, *>.Queue_{i,j-1}, \quad 0 < i \wedge 0 < j \wedge i+j < q-1,$
    $Queue_{q-1,0} \stackrel{\Delta}{=} <d_r, *>.Queue_{q-2,0},$
    $Queue_{0,q-1} \stackrel{\Delta}{=} <d_b, *>.Queue_{0,q-2},$
    $Queue_{i,j} \stackrel{\Delta}{=} <d_r, *>.Queue_{i-1,j} + <d_b, *>.Queue_{i,j-1}, \quad 0 < i \wedge 0 < j \wedge i+j = q-1;$

- $Server \triangleq \langle d_r, \infty_{r,1}\rangle.\langle s,\mu\rangle.Server + \langle d_b, \infty_{b,1}\rangle.\langle s,\mu\rangle.Server$.

Note that the precedence of red customers over black ones has been enforced by means of the two immediate actions with different priority levels in *Server*.

In the second case, we assume that preemption on a black customer being served can be exercised by red customers. This QS can be modeled as follows (*Arrivals* and $Queue_{i,j}$ are omitted since they stay the same):

- $PPQS_{M/M/1/q} \triangleq Arrivals \,\|_{\{a_r,a_b\}} (Queue_{0,0} \,\|_{\{d_r,d_b\}}\, Server)$:
  - $Server \triangleq \langle d_r, \infty_{r,1}\rangle.Server_r + \langle d_b, \infty_{b,1}\rangle.Server_b$:
    - $Server_r \triangleq \langle s,\mu\rangle.Server$;
    - $Server_b \triangleq \langle s,\mu\rangle.Server + \langle d_r, \infty_{r,1}\rangle.\langle s,\mu\rangle.Server_b$.

Note that, due to the memoryless property of the exponential distribution, there is no difference between the preemptive restart policy (i.e., the preempted customer restarts from the beginning) and the preemptive resume policy (i.e., the preempted customer resumes from the point at which it has been interrupted).

5.6 Queueing Systems with Forks and Joins

In this section we want to model a QS with a fork and a join that is composed of n QSs $M/M/1/q$ with the same service rate μ operating in parallel. The service request r of each customer arrived at the QS is divided by the fork into n subrequests sr_i, $1 \leq i \leq n$, that are then sent to the n QSs $M/M/1/q$. After being served, the n subrequests sr'_i, $1 \leq i \leq n$, are delivered to the join; here they are merged together in r' and the whole request is considered fulfilled. This QS can be modeled as follows: [4]

- $FJQS \triangleq In \,\|_{\{r\}} (Fork \,\|_{\{sr_i \mid 1 \leq i \leq n\}}\, Center \,\|_{\{sr'_i \mid 1 \leq i \leq n\}}\, Join) \,\|_{\{r'\}}\, Out$:
  - $In \triangleq \langle r, \lambda\rangle.In$;
  - $Fork \triangleq F[\varphi_1] \,\|_{\{r\}}\, F[\varphi_2] \,\|_{\{r\}} \ldots \|_{\{r\}}\, F[\varphi_n]$:
    - $F \triangleq \langle r, *\rangle.\langle sr, \infty_{1,1}\rangle.F$;
    - $\varphi_i = \{(sr, sr_i)\} \cup Id_{AType-\{sr\}}, \;\; 1 \leq i \leq n$;
  - $Center \triangleq C[\varphi'_1] \,\|_{\emptyset}\, C[\varphi'_2] \,\|_{\emptyset} \ldots \|_{\emptyset}\, C[\varphi'_n]$:
    - $C \triangleq Queue_0 \,\|_{\{d\}}\, Server$:
      - $Queue_0 \triangleq \langle sr, *\rangle.Queue_1$,
        $Queue_h \triangleq \langle sr, *\rangle.Queue_{h+1} + \langle d, *\rangle.Queue_{h-1}, \;\; 0 < h < q-1$,
        $Queue_{q-1} \triangleq \langle d, *\rangle.Queue_{q-2}$;
      - $Server \triangleq \langle d, \infty_{1,1}\rangle.\langle s,\mu\rangle.\langle sr', \infty_{1,1}\rangle.Server$;
    - $\varphi'_i = \{(sr, sr_i), (sr', sr'_i)\} \cup Id_{AType-\{sr,sr'\}}, \;\; 1 \leq i \leq n$;
  - $Join \triangleq J_0[\varphi''_1] \,\|_{\{r'\}}\, J_0[\varphi''_2] \,\|_{\{r'\}} \ldots \|_{\{r'\}}\, J_0[\varphi''_n]$:
    - $J_0 \triangleq \langle sr', *\rangle.J_1$,
      $J_h \triangleq \langle sr', *\rangle.J_{h+1} + \langle r', *\rangle.J_{h-1}, \;\; h > 0$;
    - $\varphi''_i = \{(sr', sr'_i)\} \cup Id_{AType-\{sr'\}}, \;\; 1 \leq i \leq n$;
  - $Out \triangleq \langle r', \infty_{1,1}\rangle.Out$.

[4] We denote by Id_S the identity function over set S.

Note that the availability of the functional relabeling operator has allowed us to obtain more compact algebraic representations of components having the same structure but differring for some action types only. Moreover, note the n-way synchronization over r among the fork components, and the n-way synchronization over r' among the join components.

5.7 Queueing Networks

A queueing network (QN) is composed of a set of QSs linked to each other. In general, every QS can receive customers from the outside (external sources), from the other QSs in the network, and from itself (feedback paths). The case of open QNs, where interactions with the outside are allowed, is particularly interesting because this kind of QN can be used to describe store and forward packet switched communication networks.

Let us focus our attention on an open QN composed of n QSs $M/M/1/q$ with service rates $\mu_1, \mu_2, \ldots, \mu_n$, respectively. Assume that there are n external sources of customers with rates $\lambda_1, \lambda_2, \ldots, \lambda_n$, respectively. Let us denote by $r_{i,j}$ and $p_{i,j}$ the routing action type and the routing probability, respectively, from QS i to QS j or the outside ($j = n+1$). This QN can be modeled as follows:

- $QN \stackrel{\Delta}{=} QS_1 \|_{R_2} QS_2 \|_{R_3} \ldots \|_{R_n} QS_n$:
  - $QS_i \stackrel{\Delta}{=} Arrivals_i \|_{\{a_i\}} (Queue_{i,0} \|_{\{d_i, r_{i,i}\}} Server_i), \;\; 1 \leq i \leq n$:
    - $Arrivals_i \stackrel{\Delta}{=} <a_i, \lambda_i>.Arrivals_i$;
    - $Queue_{i,0} \stackrel{\Delta}{=} <a_i, *>.Queue_{i,1} + <r_{1,i}, *>.Queue_{i,1} + \ldots + <r_{n,i}, *>.Queue_{i,1}$,

      $Queue_{i,h} \stackrel{\Delta}{=} <a_i, *>.Queue_{i,h+1} + <r_{1,i}, *>.Queue_{i,h+1} + \ldots + <r_{n,i}, *>.Queue_{i,h+1} + <d_i, *>.Queue_{i,h-1}, \;\; 0 < h < q-1$,

      $Queue_{i,q-1} \stackrel{\Delta}{=} <d_i, *>.Queue_{i,q-2}$:
    - $Server_i \stackrel{\Delta}{=} <d_i, \infty_{1,1}>.<s_i, \mu_i>.Router_i$:
      - $Router_i \stackrel{\Delta}{=} <r_{i,1}, \infty_{1,p_{i,1}}>.Server_i + \ldots + <r_{i,n+1}, \infty_{1,p_{i,n+1}}>.Server_i$;
  - $R_j = \{r_{i,j}, r_{j,i} \mid 1 \leq i < j\}, \;\; 2 \leq j \leq n$.

Observe that the description of the QN has been obtained by simply composing the descriptions of the single QSs. Furthermore, routing probabilities have been easily specified by means of the weights of immediate actions.

6 A Notion of Integrated Equivalence for EMPA

In order to complete the implementation of the first phase of the integrated approach of Fig. 1, we need to equip EMPA with a notion of integrated equivalence in order to achieve (i) the capability of performing an integrated analysis, i.e. without building projected semantic models, (ii) semantic compositionality, i.e. the possibility of studying separately the various system components thanks to the congruence property, and (iii) consistency with respect to the notion of ordinary lumping and its mathematical properties. Note that the integrated equivalence allows for a qualitative analysis, because it tells us whether two terms represent two concurrent systems with the same functional and performance properties regardless of their values. In order to know whether a functional property holds, or the value of a performance measure, we have to study the projected semantic models of (the simplest) one of the two terms.

The purpose of the notion of integrated equivalence is to relate terms describing systems that are indistinguishable from the point of view of an external observer, i.e. having the same functional and performance properties. As it turns out, it is straightforward to define two projected equivalences on the two projected semantic models in the following way.

Definition 6.1 Let $E_1, E_2 \in \mathcal{G}$. We say that E_1 is *functionally equivalent* to E_2, written $E_1 \sim_F E_2$, if and only if $\mathcal{F}[\![E_1]\!]$ is bisimilar to $\mathcal{F}[\![E_2]\!]$. ∎

Definition 6.2 Let $E_1, E_2 \in \mathcal{E}$. We say that E_1 is *performance equivalent* to E_2, written $E_1 \sim_P E_2$, if and only if $\mathcal{M}[\![E_1]\!]$ is p-bisimilar to $\mathcal{M}[\![E_2]\!]$. ■

As a consequence, a natural candidate notion of integrated equivalence would be $\sim_{FP} = \sim_F \cap \sim_P$. The problem is that $\sim_{FP}$ is not useful as it is not a congruence. As an example, if we consider terms

$$\begin{aligned} E_1 &\equiv <a, \lambda>.\underline{0} + <b, \mu>.\underline{0} \\ E_2 &\equiv <a, \mu>.\underline{0} + <b, \lambda>.\underline{0} \end{aligned}$$

where $\lambda \neq \mu$, it turns out that $E_1 \sim_{FP} E_2$ but $E_1 \,\|_{\{b\}}\, \underline{0} \not\sim_P E_2 \,\|_{\{b\}}\, \underline{0}$ because the left hand side term can execute only one action with rate λ while the right hand side term can execute only one action with rate μ. The example above shows that $\sim_{FP}$ is unable to keep track of the link between the functional part and the performance part of the actions. This means that to achieve semantic compositionality, it is *necessary* to define an equivalence based on the integrated semantic model. Incidentally, this is even *convenient* with respect to $\sim_{FP}$, since it avoids the need of building the two projected semantic models and checking them for bisimilarity and p-bisimilarity, respectively.

In order to define an integrated equivalence in the bisimulation style, we can follow the guideline below:

- Active actions should be treated according to the notion of *probabilistic bisimulation* proposed in [LS91], which consists of requiring a bisimulation to be an equivalence relation such that two bisimilar terms have the same aggregated probability to reach the same equivalence class by executing actions of the same type and priority level.
  - As far as exponentially timed actions are concerned, the notion of probabilistic bisimulation must be refined by requiring additionally that two bisimilar terms have identically distributed sojourn times. For example, if we consider terms $E_1 \equiv <a, \lambda>.F + <a, \mu>.G$ and $E_2 \equiv <a, 2 \cdot \lambda>.F + <a, 2 \cdot \mu>.G$, then both transitions labeled with a, λ and $a, 2 \cdot \lambda$ have execution probability $\lambda/(\lambda + \mu)$, and both transitions labeled with a, μ and $a, 2 \cdot \mu$ have execution probability $\mu/(\lambda + \mu)$, but the average sojourn time of E_1 is twice the average sojourn time of E_2. Due to the race policy, requiring that two bisimilar terms have identically distributed sojourn times and the same aggregated probability to reach the same equivalence class by executing exponentially timed actions of the same type amounts to requiring that two bisimilar terms have the same aggregated rate to reach the same equivalence class by executing exponentially timed actions of the same type [HR94, Hil96, Buc94].
  - As far as immediate actions are concerned, the notion of probabilistic bisimulation must be restated in terms of weights. As a consequence, two bisimilar terms are required to have the same aggregated weight to reach the same equivalence class by executing immediate actions of the same type and priority level [Tof94].
- Passive actions should be treated by following the classical notion of bisimulation [Mil89]. Thus, bisimilar terms are required to have the same passive actions reaching the same equivalence class, regardless of the actual number of these passive actions.
- Finally, priority levels should be treated carefully. It might seem useful to be able to write equations like $<c, \infty_{l,w}>.E + <d, \infty_{l',w'}>.F \sim <d, \infty_{l',w'}>.F$ if $l' > l$ or $<a, \lambda>.E + <b, \infty_{l,w}>.F \sim <b, \infty_{l,w}>.F$. The problem is that the applicability of such equations depends on the context: e.g., terms $E_1 \equiv (<a, \lambda>.E + <b, \infty_{l,w}>.F) \,\|_{\{b\}}\, \underline{0}$ and $E_2 \equiv (<b, \infty_{l,w}>.F) \,\|_{\{b\}}\, \underline{0}$ are not equivalent because E_1 can execute one action while E_2 cannot execute actions at all. To solve the problem, we follow the proposal of [BBK96] by introducing a *priority operator* "$\Theta(\_)$": priority levels are taken to be potential, and they become effective only within the scope of the priority operator. We thus consider the language $\mathcal{L}_\Theta$ generated by the following syntax

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

  whose semantic rules are those in Table 1 except that the rule in the first part is replaced by

  $$\frac{(<a, \tilde{\lambda}>, E') \in \mathit{Melt}(PM(E))}{E \xrightarrow{a, \tilde{\lambda}} E'}$$

  and the following rule for the priority operator is introduced in the second part

$$PM(\Theta(E)) = Select(PM(E))$$

It is easily seen that EMPA coincides with the set of terms $\{\Theta(E) \mid E \in \mathcal{L}\}$. As explained in [BG98], the priority operator is not part of EMPA in that useless from the modeling point of view. We have therefore preferred to develop the equivalence theory for a slightly changed language in order not to force the designer to unnecessarily burden the algebraic models of systems with priority operators.

All the conditions above that should be met in order for two terms to be considered equivalent can be subsumed by means of the following function expressing the aggregated rate with which a term can reach a class of terms by executing actions of a given type and priority level.

Definition 6.3 We define partial function $Rate : (\mathcal{G}_\Theta \times AType \times APLev \times \mathcal{P}(\mathcal{G}_\Theta)) \twoheadrightarrow ARate$ by

$$Rate(E, a, l, C) = Min\{\!| \tilde{\lambda} \mid E \xrightarrow{a,\tilde{\lambda}} E' \wedge PL(<a, \tilde{\lambda}>) = l \wedge E' \in C |\!\}$$ ■

Now we are in a position of defining the notion of integrated equivalence and showing its properties. Proofs of results reported in this section can be found in [BG98].

Definition 6.4 An equivalence relation $\mathcal{B} \subseteq \mathcal{G}_\Theta \times \mathcal{G}_\Theta$ is a *strong extended Markovian bisimulation (strong EMB)* if and only if, whenever $(E_1, E_2) \in \mathcal{B}$, then for all $a \in AType$, $l \in APLev$ and $C \in \mathcal{G}_\Theta/\mathcal{B}$

$$Rate(E_1, a, l, C) = Rate(E_2, a, l, C)$$

In this case we say that E_1 and E_2 are *strongly extended Markovian bisimilar (strongly EMB).* ■

Proposition 6.5 Let $\sim_{EMB}$ be the union of all the strong EMBs. Then $\sim_{EMB}$ is the largest strong EMB. ■

Definition 6.6 We call $\sim_{EMB}$ the *strong extended Markovian bisimulation equivalence (strong EMBE).* ■

Theorem 6.7 $\sim_{EMB}$ is a congruence for $\mathcal{G}_\Theta$. ■

Example 6.8 Consider a QS $M/M/n/n$ with arrival rate λ and service rate μ. The QS at hand can be given two different descriptions with EMPA: a *state oriented description* where the focus is on the state of the set of servers (intended as the number of servers that are currently busy), and a *resource oriented description* where the servers are modeled separately [VSSB91]. The state oriented description is given by

$$\begin{aligned}
QS^{so}_{M/M/n/n} &\stackrel{\Delta}{=} Arrivals \,\|_{\{a\}}\, Servers_0 \\
Arrivals &\stackrel{\Delta}{=} <a, \lambda>.Arrivals \\
Servers_0 &\stackrel{\Delta}{=} <a, *>.Servers_1 \\
Servers_h &\stackrel{\Delta}{=} <a, *>.Servers_{h+1} + <s, h \cdot \mu>.Servers_{h-1}, \quad 1 \leq h \leq n-1 \\
Servers_n &\stackrel{\Delta}{=} <s, n \cdot \mu>.Servers_{n-1}
\end{aligned}$$

whereas the resource oriented description is given by

$$\begin{aligned}
QS^{ro}_{M/M/n/n} &\stackrel{\Delta}{=} Arrivals \,\|_{\{a\}}\, Servers \\
Arrivals &\stackrel{\Delta}{=} <a, \lambda>.Arrivals \\
Servers &\stackrel{\Delta}{=} \underbrace{S \,\|_\emptyset\, S \,\|_\emptyset \ldots \|_\emptyset\, S}_{n} \\
S &\stackrel{\Delta}{=} <a, *>.<s, \mu>.S
\end{aligned}$$

Since in these representations immediate actions do not occur, we have that $\Theta(QS^{so}_{M/M/n/n}) \sim_{EMB} QS^{so}_{M/M/n/n}$ and $\Theta(QS^{ro}_{M/M/n/n}) \sim_{EMB} QS^{ro}_{M/M/n/n}$. We now take advantage of the fact that $\sim_{EMB}$ is a congruence: to prove $QS^{so}_{M/M/n/n} \sim_{EMB} QS^{ro}_{M/M/n/n}$, it suffices to prove $Servers_0 \sim_{EMB} Servers$. This is the case because of the strong EMB (up to $\sim_{EMB}$) given by the reflexive, symmetric, and transitive closure of the relation made out of the following pairs of terms:

$$\begin{aligned}
&Servers_0, && S \,\|_\emptyset\, S \,\|_\emptyset \ldots \|_\emptyset\, S \\
&Servers_1, && <s, \mu>.S \,\|_\emptyset\, S \,\|_\emptyset \ldots \|_\emptyset\, S \\
&Servers_2, && <s, \mu>.S \,\|_\emptyset\, <s, \mu>.S \,\|_\emptyset \ldots \|_\emptyset\, S \\
&\ldots, && \ldots \\
&Servers_n, && <s, \mu>.S \,\|_\emptyset\, <s, \mu>.S \,\|_\emptyset \ldots \|_\emptyset\, <s, \mu>.S
\end{aligned}$$ ■

Theorem 6.9 Let $E_1, E_2 \in \mathcal{G}$. If $E_1 \sim_{EMB} E_2$ then $E_1 \sim_F E_2$. ■

Theorem 6.10 Let $E_1, E_2 \in \mathcal{E}$. If $E_1 \sim_{EMB} E_2$ then $E_1 \sim_P E_2$. ■

Corollary 6.11 Let $E_1, E_2 \in \mathcal{E}$. If $E_1 \sim_{EMB} E_2$ then the coarsest ordinary lumping of $\mathcal{M}[\![E_1]\!]$ is p-isomorphic to the coarsest ordinary lumping of $\mathcal{M}[\![E_2]\!]$. ■

Theorem 6.12 Let $\mathcal{E}_{-\tau\infty}$ be the set of terms in $\mathcal{E}$ whose integrated interleaving semantic model does not contain internal immediate transitions, and let $E_1, E_2 \in \mathcal{E}_{-\tau\infty}$. Then $E_1 \sim_{EMB} E_2$ if and only if, for all $F \in \mathcal{G}$ and $S \subseteq AType - \{\tau\}$ such that $E_1 + F$, $E_2 + F$, $E_1 \,\|_S\, F$, $E_2 \,\|_S\, F \in \mathcal{E}_{-\tau\infty}$, it turns out that $E_1 + F \sim_{FP} E_2 + F$ and $E_1 \,\|_S\, F \sim_{FP} E_2 \,\|_S\, F$. ■

The first three results reveal the adequacy of $\sim_{EMB}$ from both the functional point of view and the performance point of view, and justify the fact that the notion of integrated equivalence has been developed according to the bisimulation style, the main reason being that a clear connection with the notion of ordinary lumping has been established. In fact, Corollary 6.11 states that whenever two terms are equivalent according to $\sim_{EMB}$, then their coarsest ordinarily lumped Markovian semantics are the same, which means that the two terms have exactly the same transient and steady state performance characteristics. The fourth result shows that $\sim_{EMB}$ is the coarsest congruence contained in $\sim_{FP}$ as far as terms whose integrated interleaving semantic model does not contain internal immediate transitions are concerned, thereby stressing the need to define the integrated equivalence directly on the integrated semantic model in order to allow for compositional reasoning.

We conclude by recalling that the interested reader can find in [BG98] a sound and complete axiomatization of $\sim_{EMB}$ for nonrecursive terms, as well as an $\sim_{EMB}$ checking algorithm (a variant of which can be used to calculate the coarsest ordinary lumping of a MC).

7 Integrated Net Semantics of EMPA Terms

In order to implement the second phase of the integrated approach of Fig. 1, we must provide each EMPA term with a net semantics accounting for both functional and performance aspects. As explained in Sect. 1, a good candidate for the integrated net model is the class of GSPNs, because they take into account performance aspects since the beginning of the design process, and are supported by tools for the analysis of projected models.

This section is organized as follows. In Sect. 7.1 we recall some notions about GSPNs and we focus our attention on an extension of them, acting as semantic model in this framework. In Sect. 7.2 we define the integrated net semantics for EMPA. The consistency of this semantics with respect to the integrated interleaving one is assessed in Sect. 7.3 by showing that it satisfies the functional and performance retrievability principles, while its completeness is evaluated in Sect. 7.4 by showing that it meets the concurrency principle.

7.1 Passive Generalized Stochastic Petri Nets

In this section we shall be concerned with the class of the GSPNs [ABC84, ABCC87]. They are essentially place/transition nets [Rei85] equipped with inhibitor arcs whose transitions are either exponentially timed or immediate (with priority levels and weights) and have rates that can depend on the current marking M_{curr} of the net. Since GSPNs do not admit passive transitions, and since we need passive transitions to carry out the translation of EMPA passive actions, we propose below an extension of GSPNs where passive transitions are included.

Definition 7.1 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*;
- $U = \hat{U}^{\mathcal{M}u_{fin}(P)}$ is a set whose elements are called *labels*;

- $T \subseteq \mathcal{M}u_{fin}(P) \times \mathcal{P}_{fin}(P) \times U \times \mathcal{M}u_{fin}(P)$ whose elements are called *transitions*;
- $M_0 \in \mathcal{M}u_{fin}(P)$ is called the *initial marking*;
- $L : T \longrightarrow APLev$, called *priority function*, is such that:
  - $L(t) = -1$ if t is passive;
  - $L(t) = 0$ if t is exponentially timed;
  - $L(t) \in \mathbb{N}_+$ is the priority level of t if t is immediate;
- $W : T \longrightarrow (\{*\} \cup \mathbb{R}_+^{\mathcal{M}u_{fin}(P)})$, called *weight function*, is such that:
  - $W(t) = *$ if $L(t) = -1$;
  - $W(t) \in \mathbb{R}_+^{\mathcal{M}u_{fin}(P)}$ is the rate of the exponential distribution associated with t if $L(t) = 0$;
  - $W(t) \in \mathbb{R}_+^{\mathcal{M}u_{fin}(P)}$ is the weight of t if $L(t) \in \mathbb{N}_+$. ■

In the graphical representation of a PGSPN, places are drawn as circles and transitions are drawn as either boxes (if exponentially timed), bars (if immediate), or black boxes (if passive), with the appropriate labels. If the current marking of the net is M_{curr}, we draw $M_{curr}(p)$ black dots (called *tokens*) in every place p: the current marking (i.e., the current state) of the net is then given a representation distributed among places. Each transition t can be written as $({}^\bullet t, {}^\circ t) \xrightarrow{u_t} t^\bullet$ where ${}^\bullet t$ is the *preset* of t (places where tokens are consumed), ${}^\circ t$ is the *inhibitor set* of t (places where tokens must be absent), u_t is the *label* of t, and $t^\bullet$ is the *postset* of t (places where tokens are produced). Places and transitions are linked as follows: given a transition t, there is an arrow headed arc from each place in ${}^\bullet t$ to t, a circle headed arc from each place in ${}^\circ t$ to t, and an arrow headed arc from t to each place in $t^\bullet$.

Definition 7.2 Let $N = (P, U, T, M_0, L, W)$ be a PGSPN.

- A *marking* of N is an element of $\mathcal{M}u_{fin}(P)$.
- Transition t is *enabled* at marking M if and only if ${}^\bullet t \subseteq M$ and $dom(M) \cap {}^\circ t = \emptyset$. We denote by $E(M)$ the set of transitions enabled at marking M.
- Transition $t \in E(M)$ can *fire* if and only if either $L(t) = -1$ or $L(t)$ is the highest priority level among the transitions in $E(M)$. The firing of t produces marking $M' = (M \ominus {}^\bullet t) \oplus t^\bullet$, written $M\,[u_t\rangle\, M'$.
- The *reachability set* $R(M)$ of marking M is the least subset of $\mathcal{M}u_{fin}(P)$ such that:
  - $M \in R(M)$;
  - if $M_1 \in R(M)$ and $M_1\,[u_t\rangle\, M_2$, then $M_2 \in R(M)$.
- The *reachability graph (or interleaving marking graph)* of N is the LTS
$$\mathcal{RG}[\![N]\!] = (R(M_0), \hat{U}, [\rangle, M_0)$$
■

If $\hat{U} = Act$, then from $\mathcal{RG}[\![N]\!]$ we can extract the functional semantics $\mathcal{F}[\![N]\!]$ and, provided that $\mathcal{RG}[\![N]\!]$ has no passive transitions, also the Markovian semantics $\mathcal{M}[\![N]\!]$. Since in the following inhibitor arcs will not come into play, i.e. inhibitor sets will be empty, each transition t will be written as ${}^\bullet t \xrightarrow{u_t} t^\bullet$.

7.2 Integrated Location Oriented Net Semantics

The integrated net semantics of a term $E \in \mathcal{G}$ is obtained by resorting to a suitable extension of the approach followed in Sect. 3.2. The idea [DDM88, Old91] consists of associating with every term E a net such that:

1. Net places correspond to the sequential subterms of E and its derivatives.
2. Net transitions are defined by induction on the syntactical structure of the sets of sequential terms.
3. Net markings correspond roughly to E and its derivatives.

This approach is called *location oriented* because all the information about the syntactical structure of terms is encoded within places.

In this section we adapt the proposal of [Old91] to our stochastically timed framework. To be more precise, we first introduce the syntax of net places, then we inductively define net transitions, and finally we present nets associated with EMPA terms.

7.2.1 Net Places

The first step in the definition of the integrated net semantics consists of establishing a correspondence between net places and sequential terms, thereby inducing a correspondence between net markings and terms.

Definition 7.3 The set $\mathcal{V}$ of places is generated by the following syntax

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

where $L, S \subseteq AType - \{\tau\}$. We use $V, V', \dots$ as metavariables for $\mathcal{V}$, and $Q, Q', \dots$ as metavariables for $\mathcal{M}u_{fin}(\mathcal{V})$. ■

The main difference with respect to the syntax of EMPA terms (Def. 2.1) is that the binary operator "$\_ \,\|_S\, \_$" has been replaced by the two unary operators "$\_ \,\|_S\, id$" and "$id \,\|_S\, \_$". This is the means whereby it is possible to express the decomposition of terms into sequential terms mapped onto places.

Definition 7.4 The *decomposition function* $dec : \mathcal{G} \longrightarrow \mathcal{M}u_{fin}(\mathcal{V})$ is defined by induction on the syntactical structure of the terms in $\mathcal{G}$ as follows:

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

where $Q \in \mathcal{M}u_{fin}(\mathcal{V})$ is *complete* if and only if there exists $E \in \mathcal{G}$ such that $dec(E) = Q$. ■

The decomposition function is well defined because we consider only guardedly closed terms. It is injective as well if we identify each constant with the right hand side term of its defining equation, and it assigns place sets, rather than multisets, to terms. Note that the decomposition function embeds the syntactical structure of terms into places.

$$\frac{(norm(<a,\tilde{\lambda}>,V,f),Q') \in melt_2(melt_1(PM(Q)))}{Q \xrightarrow{norm(<a,\tilde{\lambda}>,V,f)} Q'}$$

$PM(\{\!|\, <a,\tilde{\lambda}>.E \,|\!\}) = \{\!|\, (norm(<a,\tilde{\lambda}>, <a,\tilde{\lambda}>.E, 1), dec(E)) \,|\!\}$

$PM(Q/L) = \{\!|\, (norm(<a,\tilde{\lambda}>, V/L, f), Q'/L) \mid (norm(<a,\tilde{\lambda}>, V, f), Q') \in PM(Q) \wedge a \notin L \,|\!\} \oplus$
$\qquad \{\!|\, (norm(<\tau,\tilde{\lambda}>, V/L, f), Q'/L) \mid (norm(<a,\tilde{\lambda}>, V, f), Q') \in PM(Q) \wedge a \in L \,|\!\}$

$PM(Q[\varphi]) = \{\!|\, (norm(<\varphi(a),\tilde{\lambda}>, V[\varphi], f), Q'[\varphi]) \mid (norm(<a,\tilde{\lambda}>, V, f), Q') \in PM(Q) \,|\!\}$

$PM((Q_1 + Q_2) \oplus Q_3) = \{\!|\, (norm(<a,\tilde{\lambda}>, V + id, f), Q') \mid (norm(<a,\tilde{\lambda}>, V, f), Q') \in PM(Q_1 \oplus Q_3) \,|\!\}$
if Q_1 not complete $\wedge\, Q_2$ complete $\wedge\, dom(Q_1) \cap dom(Q_3) = \emptyset$

$PM((Q_1 + Q_2) \oplus Q_3) = \{\!|\, (norm(<a,\tilde{\lambda}>, id + V, f), Q') \mid (norm(<a,\tilde{\lambda}>, V, f), Q') \in PM(Q_2 \oplus Q_3) \,|\!\}$
if Q_1 complete $\wedge\, Q_2$ not complete $\wedge\, dom(Q_2) \cap dom(Q_3) = \emptyset$

$PM(Q_1 + Q_2) = \{\!|\, (norm(<a,\tilde{\lambda}>, V + id, f), Q') \mid (norm(<a,\tilde{\lambda}>, V, f), Q') \in PM(Q_1) \,|\!\} \oplus$
$\qquad \{\!|\, (norm(<a,\tilde{\lambda}>, id + V, f), Q') \mid (norm(<a,\tilde{\lambda}>, V, f), Q') \in PM(Q_2) \,|\!\}$
if Q_1 complete $\wedge\, Q_2$ complete

$PM(Q \,\|_S\, id) = \{\!|\, (norm(<a,\tilde{\lambda}>, V \,\|_S\, id, f), Q' \,\|_S\, id) \mid (norm(<a,\tilde{\lambda}>, V, f), Q') \in PM(Q) \wedge a \notin S \,|\!\}$

$PM(id \,\|_S\, Q) = \{\!|\, (norm(<a,\tilde{\lambda}>, id \,\|_S\, V, f), id \,\|_S\, Q') \mid (norm(<a,\tilde{\lambda}>, V, f), Q') \in PM(Q) \wedge a \notin S \,|\!\}$

$PM(Q_1 \,\|_S\, id \oplus id \,\|_S\, Q_2) = \{\!|\, (norm(<a, \max(\tilde{\lambda},\tilde{\mu})>, V, f), Q'_1 \,\|_S\, id \oplus id \,\|_S\, Q'_2) \mid$
$\quad a \in S \wedge \min(\tilde{\lambda},\tilde{\mu}) = * \,\wedge$
$\quad (norm(<a,\tilde{\lambda}>, V_1, f_1), Q'_1) \in PM(Q_1) \,\wedge$
$\quad (norm(<a,\tilde{\mu}>, V_2, f_2), Q'_2) \in PM(Q_2) \,\wedge$
$\quad ((\tilde{\lambda} = \tilde{\mu} = * \wedge V \equiv V_1 \,\|_S\, id \wedge f = f_1 \cdot f_2) \,\vee$
$\quad (\tilde{\lambda} \in \mathbb{R}_+ \cup Inf \wedge V \equiv V_1 \,\|_S\, id \wedge f = f_2) \,\vee$
$\quad (\tilde{\mu} \in \mathbb{R}_+ \cup Inf \wedge V \equiv id \,\|_S\, V_2 \wedge f = f_1)) \,|\!\}$

$norm(<a,\tilde{\lambda}>, V, f) = <a, Split(\tilde{\lambda}, f / \sum\{\!|\, f' \mid Q_1 \xrightarrow{norm(<a,\tilde{\lambda}>,V,f')} Q_2 \wedge Q_1 \subseteq M_{curr} \,|\!\})>$

$melt_1(PM) = \{(norm(<a,\tilde{\lambda}>, V, f), Q) \mid \exists f' \in \mathbb{N}_+ .\, (norm(<a,\tilde{\lambda}>, V, f'), Q) \in PM \,\wedge$
$\qquad f = \sum\{\!|\, f'' \mid (norm(<a,\tilde{\lambda}>, V, f''), Q) \in PM \,|\!\}\}$

$melt_2(PM) = \{(norm(<a,\tilde{\lambda}>, V, f), Q) \mid$
$\quad \exists \tilde{\mu} \in ARate .\, \exists V' \in \mathcal{V}' .\, (norm(<a,\tilde{\mu}>, V', f), Q) \in PM \,\wedge$
$\quad \tilde{\lambda} = Min\{\!|\, \tilde{\gamma} \mid (norm(<a,\tilde{\gamma}>, V'', f), Q) \in PM \wedge PL(<a,\tilde{\gamma}>) = PL(<a,\tilde{\mu}>) \,|\!\} \,\wedge$
$\quad V = inner_{\_+id}(\{V'' \mid norm(<a,\tilde{\gamma}>, V'', f), Q) \in PM \wedge PL(<a,\tilde{\gamma}>) = PL(<a,\tilde{\mu}>)\})\}$

Table 2: Inductive rules for EMPA integrated location oriented net semantics

7.2.2 Net Transitions

The second step in the definition of the integrated net semantics consists of introducing an appropriate relation over net places whereby net transitions are constructed. Following the guideline of Sect. 3.2, we define the transition relation $\longrightarrow$ as the least subset of $\mathcal{M}u_{fin}(\mathcal{V}) \times Act^{\mathcal{M}u_{fin}(\mathcal{V})} \times \mathcal{M}u_{fin}(\mathcal{V})$ generated by the inference rule reported in the first part of Table 2, which in turn is based on the multiset $PM(Q) \in \mathcal{M}u_{fin}(Act^{\mathcal{M}u_{fin}(\mathcal{V})} \times \mathcal{M}u_{fin}(\mathcal{V}))$ of potential moves of $Q \in \mathcal{M}u_{fin}(\mathcal{V})$ defined by structural induction in the second part of Table 2.

These rules are strictly related to those in Table 1 for the integrated interleaving semantics of EMPA terms. The major differences are listed below and are clarified by the corresponding upcoming examples:

1. There are three rules for the alternative composition operator, instead of one. In the first two rules only a part of the sequential terms needs to have an alternative, and such a part is not complete whereas its alternative is. This guarantees that none of the sequential terms in the complete alternative has been previously involved in an execution, so the noncomplete alternative has not been discarded yet due to an action previously executed by a sequential term in the complete alternative (see Ex. 7.5).

2. There are three rules for the parallel composition operator, instead of one. This is a consequence of the distributed notion of state typical of nets (see Ex. 7.6).

3. There are no rules for constants. The treatment of constants has been already embodied in function *dec* (see Def. 7.4), which is used in the rule for the prefix operator.

4. Function *Select* does not appear because it is unnecessary, since the race policy is included in the net firing rule, as well as difficult to implement, due to the distributed notion of state (see Ex. 7.7).

5. Rate normalization is carried out through function $norm : (Act \times \mathcal{V}' \times \mathbb{N}_+) \longrightarrow Act^{\mathcal{M}u_{fin}(\mathcal{V})}$ defined in the third part of Table 2, where $\mathcal{V}'$ is generated by the same syntax as $\mathcal{V}$ except that $V + V$ is replaced by $V + id$ and $id + V$. In order to determine the correct rate of transitions deriving from the synchronization of the same active action with several independent or alternative passive actions of the same type, function *norm* considers for each transition three parameters: the basic action, the basic place and the passive contribution. The basic action is the action that will label the transition after the normalization of its rate. The basic place is the place contributing with the basic action to the transition (see Ex. 7.8). The passive contribution is the product of the number of alternative passive actions of places contributing to the transition with such actions (see Ex. 7.9). These three parameters are initialized by the rule for the prefix operator and then modified by the third rule for the parallel composition operator: the second parameter is modified by every rule. The normalizing factor for a given transition is the ratio of its passive contribution to the sum of the passive contributions of the enabled (see Ex. 7.10) transitions having the same basic action and the same basic place as the transition at hand. Unlike function *Norm*, function *norm* comes into play not only in the case of a synchronization. Again, this is a consequence of the distributed notion of state.

6. Potential move merging is carried out through functions $melt_1 : \mathcal{M}u_{fin}(Act^{\mathcal{M}u_{fin}(\mathcal{V})} \times \mathcal{M}u_{fin}(\mathcal{V})) \longrightarrow \mathcal{P}_{fin}(Act^{\mathcal{M}u_{fin}(\mathcal{V})} \times \mathcal{M}u_{fin}(\mathcal{V}))$ and $melt_2 : \mathcal{P}_{fin}(Act^{\mathcal{M}u_{fin}(\mathcal{V})} \times \mathcal{M}u_{fin}(\mathcal{V})) \longrightarrow \mathcal{P}_{fin}(Act^{\mathcal{M}u_{fin}(\mathcal{V})} \times \mathcal{M}u_{fin}(\mathcal{V}))$ defined in the fourth part of Table 2. Function $melt_1$ merges the potential moves having the same basic action, the same basic place and the same postset by summing their passive contributions (see Ex. 7.9). Function $melt_2$ merges the potential moves having the same basic action type, the same priority level, the same passive contribution and the same postset by applying operation *Min* to their basic action rates: since the basic places of these potential moves can differ only due to "$\_ + id$" or "$id + \_$" operators, and since the basic place of the resulting potential move must be uniquely defined in order for function *norm* to work correctly, the choice is made by taking the basic place having the innermost "$\_ + id$" operator (see Ex. 7.11).

Example 7.5 Consider term

$$E \equiv (<a, \lambda>.\underline{0} \,\|_{\emptyset}\, <b, \mu>.\underline{0}) + <c, \gamma>.\underline{0}$$

whose decomposition is given by

$$dec(E) = \{\!|\, (<a,\lambda>.\underline{0} \,\|_\emptyset\, id) + <c,\gamma>.\underline{0}, (id \,\|_\emptyset\, <b,\mu>.\underline{0}) + <c,\gamma>.\underline{0} \,|\!\}$$

By applying the rules in Table 2, we get the following transitions

$$\begin{array}{rcl} \{\!|\, (<a,\lambda>.\underline{0} \,\|_\emptyset\, id) + <c,\gamma>.\underline{0} \,|\!\} & \xrightarrow{norm(<a,\lambda>,(<a,\lambda>.\underline{0} \,\|_\emptyset\, id)+id,1)} & \{\!|\, \underline{0} \,\|_\emptyset\, id \,|\!\} \\ \{\!|\, (id \,\|_\emptyset\, <b,\mu>.\underline{0}) + <c,\gamma>.\underline{0} \,|\!\} & \xrightarrow{norm(<b,\mu>,(id \,\|_\emptyset <b,\mu>.\underline{0})+id,1)} & \{\!|\, id \,\|_\emptyset\, \underline{0} \,|\!\} \\ dec(E) & \xrightarrow{norm(<c,\gamma>,id+<c,\gamma>.\underline{0},1)} & \{\!|\, \underline{0} \,|\!\} \end{array}$$

If $dec(E)$ is the current marking then all the transitions above are enabled and firing the first transition results in marking $\{\!|\, \underline{0} \,\|_\emptyset\, id, (id \,\|_\emptyset\, <b,\mu>.\underline{0}) + <c,\gamma>.\underline{0} \,|\!\}$ which cannot be the preset of any transition labeled with action type c, because the execution of either $<a,\lambda>$ or $<b,\mu>$ prevents $<c,\gamma>$ from being executed according to the intended meaning of E. This fact is detected by the rules in Table 2, i.e. they generate no transition labeled with action type c for the marking above, since the alternative $id \,\|_\emptyset\, <b,\mu>.\underline{0}$ of $<c,\gamma>.\underline{0}$ is not complete.

To understand the presence of Q_3 in the first two rules for the alternative composition operator, let us now slightly modify term E in the following way

$$E' \equiv (<a,\lambda>.<b,*>.\underline{0} \,\|_{\{b\}}\, <b,\mu>.\underline{0}) + <c,\gamma>.\underline{0}$$

where

$$dec(E') = \{\!|\, (<a,\lambda>.<b,*>.\underline{0} \,\|_{\{b\}}\, id) + <c,\gamma>.\underline{0}, (id \,\|_{\{b\}}\, <b,\mu>.\underline{0}) + <c,\gamma>.\underline{0} \,|\!\}$$

By applying the rules in Table 2, we get the two transitions

$$\begin{array}{rcl} \{\!|\, (<a,\lambda>.<b,*>.\underline{0} \,\|_{\{b\}}\, id) + <c,\gamma>.\underline{0} \,|\!\} & \xrightarrow{norm(<a,\lambda>,(<a,\lambda>.<b,*>.\underline{0} \,\|_{\{b\}}\, id)+id,1)} & \{\!|\, <b,*>.\underline{0} \,\|_{\{b\}}\, id \,|\!\} \\ dec(E') & \xrightarrow{norm(<c,\gamma>,id+<c,\gamma>.\underline{0},1)} & \{\!|\, \underline{0} \,|\!\} \end{array}$$

If $dec(E')$ is the current marking then all the transitions above are enabled and firing the first transition results in marking $\{\!|\, <b,*>.\underline{0} \,\|_{\{b\}}\, id, (id \,\|_{\{b\}}\, <b,\mu>.\underline{0}) + <c,\gamma>.\underline{0} \,|\!\}$ which is the preset of the following transition

$$\{\!|\, <b,*>.\underline{0} \,\|_{\{b\}}\, id, (id \,\|_{\{b\}}\, <b,\mu>.\underline{0}) + <c,\gamma>.\underline{0} \,|\!\} \xrightarrow{norm(<b,\mu>,(id \,\|_{\{b\}} <b,\mu>.\underline{0})+id,1)} \{\!|\, \underline{0} \,\|_{\{b\}}\, id, id \,\|_{\{b\}}\, \underline{0} \,|\!\}$$

If Q_3 were not taken into account, then the transition above would not be constructed. ■

Example 7.6 Consider term

$$E \equiv <a,\tilde{\lambda}>.\underline{0} \,\|_\emptyset\, <b,\tilde{\mu}>.\underline{0}$$

whose decomposition is given by

$$dec(E) = \{\!|\, <a,\tilde{\lambda}>.\underline{0} \,\|_\emptyset\, id, id \,\|_\emptyset\, <b,\tilde{\mu}>.\underline{0} \,|\!\}$$

By applying the rules in Table 2, we get the two independent transitions

$$\begin{array}{rcl} \{\!|\, <a,\tilde{\lambda}>.\underline{0} \,\|_\emptyset\, id \,|\!\} & \xrightarrow{norm(<a,\tilde{\lambda}>,<a,\tilde{\lambda}>.\underline{0} \,\|_\emptyset\, id,1)} & \{\!|\, \underline{0} \,\|_\emptyset\, id \,|\!\} \\ \{\!|\, id \,\|_\emptyset\, <b,\tilde{\mu}>.\underline{0} \,|\!\} & \xrightarrow{norm(<b,\tilde{\mu}>,id \,\|_\emptyset <b,\tilde{\mu}>.\underline{0},1)} & \{\!|\, id \,\|_\emptyset\, \underline{0} \,|\!\} \end{array}$$

as expected. If we replaced the three rules for the parallel composition operator with a single rule similar to that in Table 1, then we would get instead the two alternative transitions

$$\begin{array}{rcl} dec(E) & \xrightarrow{norm(<a,\tilde{\lambda}>,<a,\tilde{\lambda}>.\underline{0} \,\|_\emptyset\, id,1)} & \{\!|\, \underline{0} \,\|_\emptyset\, id, id \,\|_\emptyset\, <b,\tilde{\mu}>.\underline{0} \,|\!\} \\ dec(E) & \xrightarrow{norm(<b,\tilde{\mu}>,id \,\|_\emptyset <b,\tilde{\mu}>.\underline{0},1)} & \{\!|\, <a,\tilde{\lambda}>.\underline{0} \,\|_\emptyset\, id, id \,\|_\emptyset\, \underline{0} \,|\!\} \end{array}$$

which are not consistent with the fact that the two subterms of E are independent, thereby resulting in a violation of the concurrency principle (see Sect. 7.4). ■

Example 7.7 Consider term

$$E \equiv (<a,\lambda>.\underline{0} + <c,\infty_{1,1}>.\underline{0}) \,\|_{\{c\}} (<b,\mu>.\underline{0} + <c,*>.\underline{0})$$

whose decomposition comprises places $V_1 \,\|_{\{c\}}\, id$ and $id \,\|_{\{c\}}\, V_2$ where

$$\begin{array}{rcl} V_1 & \equiv & <a,\lambda>.\underline{0} + <c,\infty_{1,1}>.\underline{0} \\ V_2 & \equiv & <b,\mu>.\underline{0} + <c,*>.\underline{0} \end{array}$$

By applying the rules in Table 2, we get the three transitions

$$\begin{array}{lcl}
\{\!| V_1 \,||_{\{c\}}\, id \,|\!\} & \xrightarrow{norm(<a,\lambda>,(<a,\lambda>.\underline{0}+id)\,||_{\{c\}}\, id,1)} & \{\!| \underline{0} \,||_{\{c\}}\, id \,|\!\} \\
\{\!| id \,||_{\{c\}}\, V_2 \,|\!\} & \xrightarrow{norm(<b,\mu>,id\,||_{\{c\}}\,(<b,\mu>.\underline{0}+id),1)} & \{\!| id \,||_{\{c\}}\, \underline{0} \,|\!\} \\
dec(E) & \xrightarrow{norm(<c,\infty_{1,1}>,(id+<c,\infty_{1,1}>.\underline{0})\,||_{\{c\}}\, id,1)} & \{\!| \underline{0} \,||_{\{c\}}\, id, id \,||_{\{c\}}\, \underline{0} \,|\!\}
\end{array}$$

If $dec(E)$ is the current marking then all the transitions above are enabled, but the third transition prevents both the first one and the second one from firing: this could not be caught by means of a function similar to *Select* because the three transitions have different presets. ■

Example 7.8 Consider term

$$E \equiv (<a,\lambda>.\underline{0} \,||_{\{a\}}\, <a,*>.(\underline{0}+\underline{0})) + (<a,\lambda>.\underline{0} \,||_{\{a\}}\, <a,*>.\underline{0})$$

whose decomposition comprises places $(V_1 \,||_{\{a\}}\, id) + (V_1 \,||_{\{a\}}\, id)$, $(V_1 \,||_{\{a\}}\, id) + (id \,||_{\{a\}}\, V_3)$, $(id \,||_{\{a\}}\, V_2) + (V_1 \,||_{\{a\}}\, id)$ and $(id \,||_{\{a\}}\, V_2) + (id \,||_{\{a\}}\, V_3)$ where

$$\begin{array}{lcl}
V_1 & \equiv & <a,\lambda>.\underline{0} \\
V_2 & \equiv & <a,*>.(\underline{0}+\underline{0}) \\
V_3 & \equiv & <a,*>.\underline{0}
\end{array}$$

By applying the rules in Table 2, we get the following two transitions

$$\begin{array}{lcl}
dec(E) & \xrightarrow{norm(<a,\lambda>,(V_1\,||_{\{a\}}\, id)+id,1)} & \{\!| \underline{0} \,||_{\{a\}}\, id, id \,||_{\{a\}}(\underline{0}+\underline{0}) \,|\!\} \\
dec(E) & \xrightarrow{norm(<a,\lambda>,id+(V_1\,||_{\{a\}}\, id),1)} & \{\!| \underline{0} \,||_{\{a\}}\, id, id \,||_{\{a\}}\, \underline{0} \,|\!\}
\end{array}$$

If $dec(E)$ is the current marking then both transitions are enabled and the normalizing factor is 1 for both transitions, as expected. This example motivates the use of $\mathcal{V}'$ instead of $\mathcal{V}$ for expressing the basic place: if $\mathcal{V}$ were used, then the two transitions above would have the same basic place (beside the same basic action), so they would be given the wrong normalizing factor 1/2 by function *norm*. ■

Example 7.9 Consider term

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

whose decomposition comprises places $V_1 \,||_{\{a\}}\, id$, $id \,||_{\{a\}}(V_2 \,||_{\emptyset}\, id)$ and $id \,||_{\{a\}}(id \,||_{\emptyset}\, V_3)$ where

$$\begin{array}{lcl}
V_1 & \equiv & <a,\lambda>.\underline{0} \\
V_2 & \equiv & <a,*>.\underline{0} + <a,*>.\underline{0} \\
V_3 & \equiv & <a,*>.\underline{0}
\end{array}$$

By applying the rules in Table 2, we get the following two transitions

$$\begin{array}{lcl}
\{\!| V_1 \,||_{\{a\}}\, id, id \,||_{\{a\}}(V_2 \,||_{\emptyset}\, id) \,|\!\} & \xrightarrow{norm(<a,\lambda>,V_1\,||_{\{a\}}\, id,2)} & \{\!| \underline{0} \,||_{\{a\}}\, id, id \,||_{\{a\}}(\underline{0} \,||_{\emptyset}\, id) \,|\!\} \\
\{\!| V_1 \,||_{\{a\}}\, id, id \,||_{\{a\}}(id \,||_{\emptyset}\, V_3) \,|\!\} & \xrightarrow{norm(<a,\lambda>,V_1\,||_{\{a\}}\, id,1)} & \{\!| \underline{0} \,||_{\{a\}}\, id, id \,||_{\{a\}}(id \,||_{\emptyset}\, \underline{0}) \,|\!\}
\end{array}$$

where value 2 for the passive contribution of the first transition is determined by function $melt_1$. If $dec(E)$ is the current marking then both transitions are enabled and the normalizing factor is 2/3 for the first transition, and 1/3 for the second transition, as expected. This example motivates the use of passive contributions: if the normalizing factor were computed as the inverse of the number of enabled transitions having the same basic action and the same basic place as the transition at hand, then we would obtain the wrong normalizing factor 1/2 for the two transitions above. ■

Example 7.10 Consider term

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

whose decomposition comprises places $V_1 \,||_{\{a\}}\, id$, $id \,||_{\{a\}}(V_2 \,||_{\emptyset}\, id)$ and $id \,||_{\{a\}}(id \,||_{\emptyset}\, V_3)$ where

$$\begin{array}{lcl}
V_1 & \equiv & <a,\lambda>.\underline{0} \\
V_2 & \equiv & <a,*>.<a,*>.\underline{0} \\
V_3 & \equiv & <a,*>.\underline{0}
\end{array}$$

By applying the rules in Table 2, we get the following three transitions

$$\begin{array}{lcl}
\{\!| V_1 \,||_{\{a\}}\, id, id \,||_{\{a\}}(V_2 \,||_{\emptyset}\, id) \,|\!\} & \xrightarrow{norm(<a,\lambda>,V_1\,||_{\{a\}}\, id,1)} & \{\!| \underline{0} \,||_{\{a\}}\, id, id \,||_{\{a\}}(V_3 \,||_{\emptyset}\, id) \,|\!\} \\
\{\!| V_1 \,||_{\{a\}}\, id, id \,||_{\{a\}}(id \,||_{\emptyset}\, V_3) \,|\!\} & \xrightarrow{norm(<a,\lambda>,V_1\,||_{\{a\}}\, id,1)} & \{\!| \underline{0} \,||_{\{a\}}\, id, id \,||_{\{a\}}(id \,||_{\emptyset}\, \underline{0}) \,|\!\} \\
\{\!| V_1 \,||_{\{a\}}\, id, id \,||_{\{a\}}(V_3 \,||_{\emptyset}\, id) \,|\!\} & \xrightarrow{norm(<a,\lambda>,V_1\,||_{\{a\}}\, id,1)} & \{\!| \underline{0} \,||_{\{a\}}\, id, id \,||_{\{a\}}(\underline{0} \,||_{\emptyset}\, id) \,|\!\}
\end{array}$$

If $dec(E)$ is the current marking then only the first and the second transitions are enabled and their normalizing factor computed by $norm$ is 1/2 as expected. This example motivates the use of marking dependent rates: if also the third transition were taken into account though not enabled at $dec(E)$, then we would obtain the wrong normalizing factor 1/3 for the first two transitions. ■

Example 7.11 Consider term

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

whose decomposition comprises places $V_1 \,||_{\{a\}}\, id$, $id \,||_{\{a\}} (V_2 \,||_{\emptyset}\, id)$ and $id \,||_{\{a\}} (id \,||_{\emptyset}\, V_2)$ where

$$\begin{array}{rcl} V_1 & \equiv & <a,\lambda>.\underline{0} + <a,\lambda>.\underline{0} \\ V_2 & \equiv & <a,*>.\underline{0} \end{array}$$

By applying the rules in Table 2, we get the following two transitions

$$\begin{array}{ccc} \{\!| V_1 \,||_{\{a\}}\, id, id \,||_{\{a\}} (V_2 \,||_{\emptyset}\, id) |\!\} & \xrightarrow{norm(<a,2\cdot\lambda>,(<a,\lambda>.\underline{0}+id)\,||_{\{a\}}\, id),1)} & \{\!| \underline{0} \,||_{\{a\}}\, id, id \,||_{\{a\}} (\underline{0} \,||_{\emptyset}\, id) |\!\} \\ \{\!| V_1 \,||_{\{a\}}\, id, id \,||_{\{a\}} (id \,||_{\emptyset}\, V_2) |\!\} & \xrightarrow{norm(<a,2\cdot\lambda>,(<a,\lambda>.\underline{0}+id)\,||_{\{a\}}\, id),1)} & \{\!| \underline{0} \,||_{\{a\}}\, id, id \,||_{\{a\}} (id \,||_{\emptyset}\, \underline{0}) |\!\} \end{array}$$

each of which is obtained by applying function $melt_2$ to two potential moves having as a basic place $(<a,\lambda>.\underline{0} + id) \,||_{\{a\}}\, id$ and $(id + <a,\lambda>.\underline{0}) \,||_{\{a\}}\, id$, respectively. If $dec(E)$ is the current marking then both transitions are enabled and the normalizing factor is 1/2 for both transitions, as expected. This example motivates that fact that, if two potential moves having different basic places are merged by function $melt_2$, the basic place of the resulting potential move must be uniquely identified: if the two transitions above had as a basic place $(<a,\lambda>.\underline{0} + id) \,||_{\{a\}}\, id$ and $(id + <a,\lambda>.\underline{0}) \,||_{\{a\}}\, id$, respectively, then we would obtain the wrong normalizing factor 1 for them. ■

7.2.3 Nets Associated with Terms

The third step in the definition of the integrated net semantics consists of associating with each term an appropriate PGSPN by exploiting the previous two steps.

Definition 7.12 The *integrated location oriented net semantics* of a term $E \in \mathcal{G}$ is the PGSPN

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

where:

- P is the least subset of $\mathcal{V}$ such that:
  - $dom(dec(E)) \subseteq P$;
  - if $dom(Q_1) \subseteq P$ and $Q_1 \xrightarrow{norm(<a,\tilde{\lambda}>,V,f)} Q_2$, then $dom(Q_2) \subseteq P$;
- $U = Act^{\mathcal{M}u_{fin}(P)}$;
- T is the restriction of $\longrightarrow$ to $\mathcal{M}u_{fin}(P) \times Act^{\mathcal{M}u_{fin}(P)} \times \mathcal{M}u_{fin}(P)$;
- $M_0 = dec(E)$;
- $L : T \longrightarrow APLev$ such that:
  - $L(Q_1, norm(<a,*>,V,f), Q_2) = -1$;
  - $L(Q_1, norm(<a,\lambda>,V,f), Q_2) = 0$;
  - $L(Q_1, norm(<a,\infty_{l,w}>,V,f), Q_2) = l$;
- $W : T \longrightarrow (\{*\} \cup \mathbb{R}_+^{\mathcal{M}u_{fin}(P)})$ such that:
  - $W(Q_1, norm(<a,*>,V,f), Q_2) = *$;
  - $W(Q_1, norm(<a,\lambda>,V,f), Q_2) = \lambda'$ if $norm(<a,\lambda>,V,f) = <a,\lambda'>$;
  - $W(Q_1, norm(<a,\infty_{l,w}>,V,f), Q_2) = w'$ if $norm(<a,\infty_{l,w}>,V,f) = <a,\infty_{l,w'}>$. ■

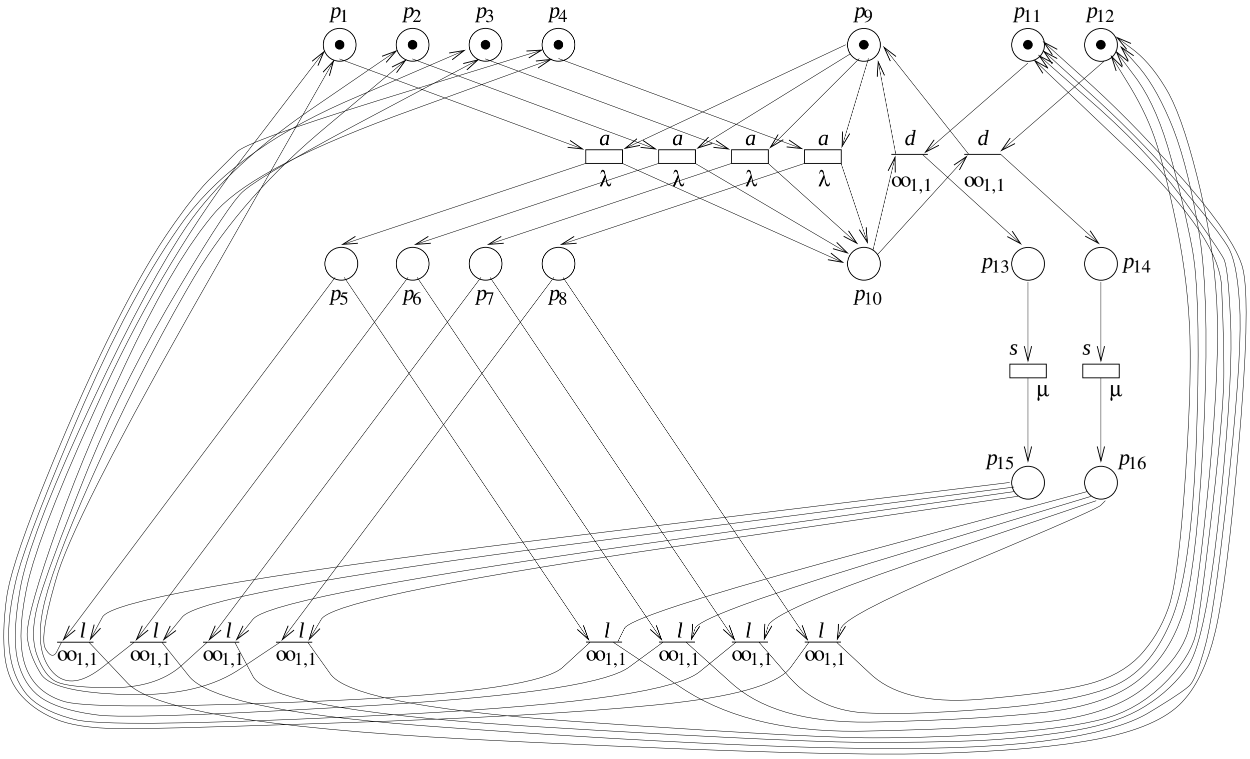


Figure 9: Integrated net semantics of $QS_{M/M/2/3/4}$

Example 7.13 Let us consider a QS $M/M/2/3/4$ with arrival rate λ and service rate μ. Once the action type "a customer leaves the service center" is denoted by l, its resource oriented representation is the following:

- $QS_{M/M/2/3/4} \triangleq Customers_4 \|_{\{a,l\}} (Queue \|_{\{d\}} Servers_2)$:
  - $Customers_4 \triangleq C \|_{\emptyset} C \|_{\emptyset} C \|_{\emptyset} C$:
    - $C \triangleq {<}a, \lambda{>}.{<}l, *{>}.C$;
  - $Queue \triangleq {<}a, *{>}.{<}d, *{>}.Queue$;
  - $Servers_2 \triangleq S \|_{\emptyset} S$:
    - $S \triangleq {<}d, \infty_{1,1}{>}.{<}s, \mu{>}.{<}l, \infty_{1,1}{>}.S$.

Its integrated net semantics $\mathcal{N}_{loc}[\![QS_{M/M/2/3/4}]\!]$ is the GSPN in Fig. 9, where the following shorthands have been used:

- $p_1 = (((C \|_{\emptyset} id) \|_{\emptyset} id) \|_{\emptyset} id) \|_{\{a,l\}} id$,
  $p_2 = (((id \|_{\emptyset} C) \|_{\emptyset} id) \|_{\emptyset} id) \|_{\{a,l\}} id$,
  $p_3 = ((id \|_{\emptyset} C) \|_{\emptyset} id) \|_{\{a,l\}} id$,
  $p_4 = (id \|_{\emptyset} C) \|_{\{a,l\}} id$,
  $p_5 = ((({<}l, *{>}.C \|_{\emptyset} id) \|_{\emptyset} id) \|_{\emptyset} id) \|_{\{a,l\}} id$,
  $p_6 = (((id \|_{\emptyset} {<}l, *{>}.C) \|_{\emptyset} id) \|_{\emptyset} id) \|_{\{a,l\}} id$,
  $p_7 = ((id \|_{\emptyset} {<}l, *{>}.C) \|_{\emptyset} id) \|_{\{a,l\}} id$,
  $p_8 = (id \|_{\emptyset} {<}l, *{>}.C) \|_{\{a,l\}} id$;

- $p_9 = id\|_{\{a,l\}}(Queue\|_{\{d\}} id)$,
  $p_{10} = id\|_{\{a,l\}}(<d, *>.Queue\|_{\{d\}} id)$;

- $p_{11} = id\|_{\{a,l\}}(id\|_{\{d\}}(S\|_{\emptyset} id))$,
  $p_{12} = id\|_{\{a,l\}}(id\|_{\{d\}}(id\|_{\emptyset} S))$,
  $p_{13} = id\|_{\{a,l\}}(id\|_{\{d\}}(<s, \mu>.<l, \infty_{1,1}>.S\|_{\emptyset} id))$,
  $p_{14} = id\|_{\{a,l\}}(id\|_{\{d\}}(id\|_{\emptyset}<s, \mu>.<l, \infty_{1,1}>.S))$,
  $p_{15} = id\|_{\{a,l\}}(id\|_{\{d\}}(<l, \infty_{1,1}>.S\|_{\emptyset} id))$,
  $p_{16} = id\|_{\{a,l\}}(id\|_{\{d\}}(id\|_{\emptyset}<l, \infty_{1,1}>.S))$. ■

Now we show two properties of the integrated net semantics, which can be demonstrated with a proof similar to that provided in [Old91].

Theorem 7.14 Let $E \in \mathcal{G}$.

(i) $\mathcal{N}_{loc}[\![E]\!]$ is safe, i.e. every marking reachable from the initial one is a set.

(ii) $\mathcal{N}_{loc}[\![E]\!]$ is finite if each subterm of E of the form $E'/L, E'[\varphi], E_1 \|_S E_2$ 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 $\mathcal{N}_{loc}[\![E]\!]$ is a GSPN. As we can expect, the above class is given by $\mathcal{E}$ and this will be proved later.

7.3 Retrievability Principles

In this section we assess the soundness of the integrated net semantics with respect to the integrated interleaving semantics. To this aim, we adapt the proposal in [Old91] to our stochastically timed framework by resorting to the following two principles:

- *Functional retrievability principle*: the functional semantics of each term should be retrievable from its integrated net semantics. Such a principle can be formalized by requiring that, for each term, its functional semantics is isomorphic or bisimilar to the functional semantics of its integrated net semantics.

- *Performance retrievability principle*: the performance semantics of each term should be retrievable from its integrated net semantics. Such a principle can be formalized by requiring that, for each term, its Markovian semantics is p-isomorphic or p-bisimilar to the Markovian semantics of its integrated net semantics.

These two principles guarantee that each term and its integrated net semantics describe the same system both from the functional and the performance point of view.

Theorem 7.15 Let $E \in \mathcal{G}$. Then $\mathcal{RG}[\![\mathcal{N}_{loc}[\![E]\!]]\!]$ is isomorphic to $\mathcal{I}[\![E]\!]$.

Proof The proof is divided into three parts.

(1st part) Suppose that priority levels are taken into account neither in EMPA nor in PGSPNs. More accurately, assume that the active transitions of PGSPNs are not divided into different priority levels, and consider $\mathcal{I}'[\![E]\!]$ instead of $\mathcal{I}[\![E]\!]$, i.e. consider the LTS (whose set of states is denoted by $\uparrow' E$) representing the integrated interleaving semantics of E if function *Select* were not applied. Then we can demonstrate, by following the proof developed in [Old91] Thm. 3.7.18, that $\mathcal{RG}[\![\mathcal{N}_{loc}[\![E]\!]]\!]$ is bisimilar to $\mathcal{I}'[\![E]\!]$ through relation

$$\mathcal{B} = \{(F, Q) \in \uparrow' E \times R(dec(E)) \mid Q \,\text{swf} \wedge dec(F) = upd(Q)\}$$

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 and $Q[\varphi]$.

  - If $Q_1 \oplus Q_3$ is swf, $dom(Q_1) \cap dom(Q_3) = \emptyset$, either $Q_3 = \emptyset$ or not all components in Q_3 contain "+" as their topmost operator, and Q_2 is complete, then $(Q_1 + Q_2) \oplus Q_3$ and $(Q_2 + Q_1) \oplus Q_3$ are swf.
  - If Q_1 and Q_2 are swf, then so is $Q_1 \,\|_S\, id \cup id \,\|_S\, Q_2$.

  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 $upd(Q) = Q$.
  - If $Q \equiv Q'/L$ is incomplete, then $upd(Q) = upd(Q')/L$.
  - If $Q \equiv Q'[\varphi]$ is incomplete, then $upd(Q) = upd(Q')[\varphi]$.
  - If $Q \equiv (Q_1 + Q_2) \oplus Q_3$ or $Q \equiv (Q_2 + Q_1) \oplus Q_3$ is incomplete, and Q_2 is complete, then $upd(Q) = upd(Q_1 \oplus Q_3)$.
  - If $Q \equiv Q_1 \,\|_S\, id \cup id \,\|_S\, Q_2$ is incomplete, then $upd(Q) = upd(Q_1) \,\|_S\, id \cup id \,\|_S\, upd(Q_2)$.

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

(2nd part) Now we want to prove, under the same assumption made at the beginning of the previous part, that bisimulation $\mathcal{B}$ is actually an isomorphism between $\mathcal{RG}[\![\mathcal{N}_{loc}[\![E]\!]]\!]$ and $\mathcal{I}'[\![E]\!]$.

First, 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 bisimulation, there must exist $Q \in R(dec(E))$ such that $(F, Q) \in \mathcal{B}$, i.e. $dec(F) = upd(Q)$. It remains to prove the uniqueness of such a swf reachable marking Q. Suppose that there exist $Q_1, Q_2 \in R(dec(E))$ swf and different from each other such that $upd(Q_1) = upd(Q_2) = dec(F)$. 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 that 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 alternative composition 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 alternative composition operator disappears and only the alternative involved remains after it has been transformed (see the rules for the alternative composition operator).

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

Third, we have to prove that $\mathcal{B}$ is surjective. This is true because given $Q \in R(dec(E))$, since Q is reachable from $dec(E)$ and $\mathcal{B}$ is a 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 part) Now let us take into account the priority levels. Since the priority mechanism for EMPA actions is exactly the same as the priority mechanism for PGSPN transitions, from the previous step it follows that $\mathcal{RG}[\![\mathcal{N}_{loc}[\![E]\!]]\!]$ is isomorphic to $\mathcal{I}[\![E]\!]$. ∎

Corollary 7.16 Let $E \in \mathcal{G}$. Then $\mathcal{F}[\![\mathcal{N}_{loc}[\![E]\!]]\!]$ is isomorphic to $\mathcal{F}[\![E]\!]$. ∎

Corollary 7.17 Let $E \in \mathcal{E}$. Then $\mathcal{M}[\![\mathcal{N}_{loc}[\![E]\!]]\!]$ is p-isomorphic to $\mathcal{M}[\![E]\!]$. ∎

From retrievability, the following result immediately follows.

Theorem 7.18 Let $E \in \mathcal{G}$. Then $\mathcal{N}_{loc}[\![E]\!]$ is a GSPN if and only if $E \in \mathcal{E}$.

Proof ($\Longrightarrow$) Suppose that $\mathcal{N}_{loc}[\![E]\!]$ is a GSPN, i.e. suppose that $\mathcal{N}_{loc}[\![E]\!]$ has no passive transitions. Then $\mathcal{RG}[\![\mathcal{N}_{loc}[\![E]\!]]\!]$ has no passive transitions hence, by virtue of Thm. 7.15, $\mathcal{I}[\![E]\!]$ has no passive transitions. Thus $E \in \mathcal{E}$.

($\Longleftarrow$) Suppose that $E \in \mathcal{E}$, i.e. suppose that $\mathcal{I}[\![E]\!]$ has no passive transitions. We prove that $\mathcal{N}_{loc}[\![E]\!]$ is a GSPN by proceeding by induction on the syntactical structure of E:

- If $E \equiv \underline{0}$ then $\mathcal{N}_{loc}[\![E]\!]$ is obviously a GSPN.
- Let $E \equiv <a, \tilde{\lambda}>.E'$. From $E \in \mathcal{E}$ it follows that $\tilde{\lambda} \neq *$ and $E' \in \mathcal{E}$, so by the induction hypothesis we have that $\mathcal{N}_{loc}[\![E']\!]$ is a GSPN hence $\mathcal{N}_{loc}[\![E]\!]$ is a GSPN too.
- Let $E \equiv E'/L$. From $E \in \mathcal{E}$ it follows that $E' \in \mathcal{E}$, so by the induction hypothesis we have that $\mathcal{N}_{loc}[\![E']\!]$ is a GSPN hence $\mathcal{N}_{loc}[\![E]\!]$ is a GSPN too.
- Let $E \equiv E'[\varphi]$. From $E \in \mathcal{E}$ it follows that $E' \in \mathcal{E}$, so by the induction hypothesis we have that $\mathcal{N}_{loc}[\![E']\!]$ is a GSPN hence $\mathcal{N}_{loc}[\![E]\!]$ is a GSPN too.
- Let $E \equiv E_1 + E_2$. From $E \in \mathcal{E}$ it follows that $E_1 \in \mathcal{E}$ and $E_2 \in \mathcal{E}$, so by the induction hypothesis we have that $\mathcal{N}_{loc}[\![E_1]\!]$ and $\mathcal{N}_{loc}[\![E_2]\!]$ are two GSPNs hence $\mathcal{N}_{loc}[\![E]\!]$ is a GSPN too.
- Let $E \equiv E_1 \,\|_S\, E_2$. There are two cases:
  - If $E_1 \in \mathcal{E} \wedge E_2 \in \mathcal{E}$ then, by the induction hypothesis, we have that $\mathcal{N}_{loc}[\![E_1]\!]$ and $\mathcal{N}_{loc}[\![E_2]\!]$ are two GSPNs hence $\mathcal{N}_{loc}[\![E]\!]$ is a GSPN too.
  - If $E_1 \notin \mathcal{E} \vee E_2 \notin \mathcal{E}$ then E_1 or E_2 can execute some passive actions which, due to the fact that $E \in \mathcal{E}$, have types in S and either do not synchronize at all or synchronize with active actions of the same type present in the other subterm. By the rules for the parallel composition operator, the passive transitions present in $\mathcal{N}_{loc}[\![E_1]\!]$ or in $\mathcal{N}_{loc}[\![E_2]\!]$ cannot be present in $\mathcal{N}_{loc}[\![E]\!]$; hence $\mathcal{N}_{loc}[\![E]\!]$ is a GSPN. ■

7.4 Concurrency Principle

In this section we assess the completeness of the integrated net semantics by resorting to the *concurrency principle* [Old91], which requires that the intended concurrency of each term should be represented by its integrated net semantics. 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 stochastically timed 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., [Old91]), we develop an *integrated denotational net semantics* for EMPA and then we investigate whether the integrated net semantics admits the same concurrent computations as the denotational one.

The operators on safe PGSPNs with no inhibitor arcs are presented below: the definitions of the set of labels, the priority function and the weight function for the resulting net of each operator are omitted because they are similar to those reported in Def. 7.12.

- $\underline{0} = (\{p\}, U, \emptyset, \{\!|\, p \,|\!\}, \emptyset, \emptyset)$;
- $<a, \tilde{\lambda}>.(P, U, T, M_0, L, W) = (P', U, T', M'_0, L', W')$ where:
  - $P' = P \cup \{p\}$, $p \notin P$;
  - $T' = T \cup \{(\{\!|\, p \,|\!\}, norm_d(<a, \tilde{\lambda}>, p), M_0)\}$;
  - $M'_0 = \{\!|\, p \,|\!\}$;
- $(P, U, T, M_0, L', W)/L = (P, U, T', M_0, L'', W')$ where:
  - $T' = \{(M_1, norm_d(<a, \tilde{\lambda}>, p), M_2) \in T \mid a \notin L\} \;\cup$
    $\{(M_1, norm_d(<\tau, \tilde{\lambda}>, p), M_2) \mid \exists a \in L.\, (M_1, norm_d(<a, \tilde{\lambda}>, p), M_2) \in T\}$;
- $(P, U, T, M_0, L, W)[\varphi] = (P, U, T', M_0, L', W')$ where:
  - $T' = \{(M_1, norm(<\varphi(a), \tilde{\lambda}>, p), M_2) \mid (M_1, norm(<a, \tilde{\lambda}>, p), M_2) \in T\}$;

- $(P_1, U, T_1, M_{01}, L_1, W_1) + (P_2, U, T_2, M_{02}, L_2, W_2) = (P, U, T, M_0, L, W)$ where:
  - $P = (dom(M_{01}) \times dom(M_{02})) \cup P_1 \cup P_2,\ P_1 \cap P_2 = \emptyset$;
  - $T = \{((M_1 \otimes M_{02}) \oplus M'_1, norm_d(<a, \tilde{\lambda}>, p), M_2) \mid M_1 \subseteq M_{01} \ \wedge$
    $dom(M_1) \cap dom(M'_1) = \emptyset \ \wedge$
    $(M_1 \oplus M'_1, norm_d(<a, \tilde{\lambda}>, p), M_2) \in T_1\} \ \cup$
    $\{((M_{01} \otimes M_1) \oplus M'_1, norm_d(<a, \tilde{\lambda}>, p), M_2) \mid M_1 \subseteq M_{02} \ \wedge$
    $dom(M_1) \cap dom(M'_1) = \emptyset \ \wedge$
    $(M_1 \oplus M'_1, norm_d(<a, \tilde{\lambda}>, p), M_2) \in T_2\}$;
  - $M_0 = M_{01} \otimes M_{02}$;
- $(P_1, U, T_1, M_{01}, L_1, W_1) \|_S (P_2, U, T_2, M_{02}, L_2, W_2) = (P, U, T, M_0, L, W)$ where:
  - $P = P_1 \cup P_2,\ P_1 \cap P_2 = \emptyset$;
  - $T = \{(M_1, norm_d(<a, \tilde{\lambda}>, p), M_2) \in T_1 \cup T_2 \mid a \notin S\} \cup$
    $\{(M_1 \oplus M'_1, norm_d(<a, \max(\tilde{\lambda}, \tilde{\mu})>, p), M_2 \oplus M'_2) \mid a \in S \wedge \min(\tilde{\lambda}, \tilde{\mu}) = * \ \wedge$
    $(M_1, norm_d(<a, \tilde{\lambda}>, p'), M_2) \in T_1 \ \wedge$
    $(M'_1, norm_d(<a, \tilde{\mu}>, p''), M'_2) \in T_2 \ \wedge$
    $((\tilde{\lambda} = \tilde{\mu} = * \wedge p = p') \ \vee$
    $(\tilde{\lambda} \in \mathbb{R}_+ \cup Inf \wedge p = p') \ \vee$
    $(\tilde{\mu} \in \mathbb{R}_+ \cup Inf \wedge p = p''))\}$;
  - $M_0 = M_{01} \oplus M_{02}$,

where function $norm_d : (Act \times P) \longrightarrow Act^{\mathcal{M}u_{fin}(P)}$ is defined by

$$norm_d(<a, \tilde{\lambda}>, p) = <a, Split(\tilde{\lambda}, 1/|\{(M_1, norm_d(<a, \tilde{\lambda}>, p), M_2) \in T \mid M_1 \subseteq M_{curr}\}|)>$$

The effect of these net operators should be easy to understand, except for the alternative composition one. It 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 then uses the Cartesian product to introduce choices between all the pairs of initial transitions of the two nets to which it is applied. Root unwinding allows the correct interplay of alternative composition and recursion to be implemented.

Example 7.19 Consider terms

$$A \ \stackrel{\Delta}{=} \ <a, \lambda>.A$$
$$B \ \stackrel{\Delta}{=} \ <b, \mu>.B$$

The integrated denotational net semantics of A is a net with one place p_A and one transition

$$\{\!|\, p_A \,|\!\} \xrightarrow{norm_d(<a,\lambda>,p_A)} \{\!|\, p_A \,|\!\}$$

The integrated denotational net semantics of B is a net with one place p_B and one transition

$$\{\!|\, p_B \,|\!\} \xrightarrow{norm_d(<b,\mu>,p_B)} \{\!|\, p_B \,|\!\}$$

Consider now term

$$E \equiv A + B$$

If we used the Cartesian product construction without root unwinding, then the integrated denotational net semantics of E would be a net with one place (p_A, p_B) and two transitions

$$\{\!|\, (p_A, p_B) \,|\!\} \xrightarrow{norm_d(<a,\lambda>,p_A)} \{\!|\, (p_A, p_B) \,|\!\}$$
$$\{\!|\, (p_A, p_B) \,|\!\} \xrightarrow{norm_d(<b,\mu>,p_B)} \{\!|\, (p_A, p_B) \,|\!\}$$

This net is not the right integrated denotational net semantics of E since E can perform either infinitely many actions $<a, \lambda>$ or infinitely many actions $<b, \mu>$, whereas the net above allows the two different actions to be arbitrarily interleaved. ■

Function $norm_d$ plays the same role as function $norm$. The main difference between them is that function $norm_d$ does not consider passive contributions. This is due to the fact that the integrated denotational net semantics generates a new place whenever $\underline{0}$ or a prefix operator is encountered. As a consequence, it is not possible that two or more transitions constructed by the alternative composition operator with the same preset, have the same postset. This is reflected by the definition of the normalizing factor: it is simply the inverse of the number of enabled transitions having the same basic actions and the same basic place as the transition at hand.

Example 7.20 Consider term E of Ex. 7.9 and let $E_1 \equiv V_1$, $E_2 \equiv V_2$, $E_3 \equiv V_3$. The integrated denotational net semantics of E_1 is a net with two places $p_{1,1}$, $p_{1,2}$ and one transition

$$\{\!|\, p_{1,1} \,|\!\} \xrightarrow{norm_d(<a,\lambda>,p_{1,1})} \{\!|\, p_{1,2} \,|\!\}$$

The integrated denotational net semantics of E_2 is a net with three places $(p_{2,1}, p_{2,2})$, $p_{2,3}$, $p_{2,4}$ and two transitions

$$\begin{array}{ccc} \{\!|\, (p_{2,1}, p_{2,2}) \,|\!\} & \xrightarrow{norm_d(<a,*>,p_{2,1})} & \{\!|\, p_{2,3} \,|\!\} \\ \{\!|\, (p_{2,1}, p_{2,2}) \,|\!\} & \xrightarrow{norm_d(<a,*>,p_{2,2})} & \{\!|\, p_{2,4} \,|\!\} \end{array}$$

The integrated denotational net semantics of E_3 is a net with two places $p_{3,1}$, $p_{3,2}$ and one transition

$$\{\!|\, p_{3,1} \,|\!\} \xrightarrow{norm_d(<a,*>,p_{3,1})} \{\!|\, p_{3,2} \,|\!\}$$

Finally, the integrated denotational net semantics of E is a net having the same places as the previous nets and three transitions

$$\begin{array}{rcl} \{\!|\, p_{1,1}, (p_{2,1}, p_{2,2}) \,|\!\} & \xrightarrow{norm_d(<a,\lambda>,p_{1,1})} & \{\!|\, p_{1,2}, p_{2,3} \,|\!\} \\ \{\!|\, p_{1,1}, (p_{2,1}, p_{2,2}) \,|\!\} & \xrightarrow{norm_d(<a,\lambda>,p_{1,1})} & \{\!|\, p_{1,2}, p_{2,4} \,|\!\} \\ \{\!|\, p_{1,1}, p_{3,1} \,|\!\} & \xrightarrow{norm_d(<a,\lambda>,p_{1,1})} & \{\!|\, p_{1,2}, p_{3,2} \,|\!\} \end{array}$$

In the initial marking all the transitions above are enabled, and their normalizing factor is 1/3 as expected. This example motivates the fact that passive contributions are unnecessary. ■

Using the notion of *place based bisimilarity (pl-bisimilarity)* on safe nets of [Old91] Def. 2.3.8 suitably modified in order to take into account aggregated rates, and following a demonstration similar to that of [Old91] Thm. 3.8.3, we can now prove that for each n-ary operator op we have that $\mathcal{N}_{loc}[\![op_{EMPA}(E_1, \ldots, E_n)]\!]$ is pl-bisimilar to $op_{PGSPN}(\mathcal{N}_{loc}[\![E_1]\!], \ldots, \mathcal{N}_{loc}[\![E_n]\!])$. By virtue of [Old91] Thm. 2.3.10, this means that the two nets have the same causal semantics, i.e. they have the same concurrent computations.

Theorem 7.21 It turns out that:

(*i*) For every $E \in \mathcal{G}$ and $<a, \tilde{\lambda}> \in Act$, $\mathcal{N}_{loc}[\![<a, \tilde{\lambda}>.E]\!]$ is pl-bisimilar to $<a, \tilde{\lambda}>.\mathcal{N}_{loc}[\![E]\!]$.

(*ii*) For every $E \in \mathcal{G}$ and $L \subseteq AType - \{\tau\}$, $\mathcal{N}_{loc}[\![E/L]\!]$ is pl-bisimilar to $\mathcal{N}_{loc}[\![E]\!]/L$.

(*iii*) For every $E \in \mathcal{G}$ and $\varphi \in ARFun$, $\mathcal{N}_{loc}[\![E[\varphi]]\!]$ is pl-bisimilar to $\mathcal{N}_{loc}[\![E]\!][\varphi]$.

(*iv*) For every $E_1, E_2 \in \mathcal{G}$, $\mathcal{N}_{loc}[\![E_1 + E_2]\!]$ is pl-bisimilar to $\mathcal{N}_{loc}[\![E_1]\!] + \mathcal{N}_{loc}[\![E_2]\!]$.

(*v*) For every $E_1, E_2 \in \mathcal{G}$ and $S \subseteq AType - \{\tau\}$, $\mathcal{N}_{loc}[\![E_1 \,\|_S\, E_2]\!]$ is pl-bisimilar to $\mathcal{N}_{loc}[\![E_1]\!] \,\|_S\, \mathcal{N}_{loc}[\![E_2]\!]$. ■

8 The Alternating Bit Protocol

In this section we illustrate the application of the integrated approach of Fig. 1 to the alternating bit protocol. The protocol is modeled by means of an EMPA term and then analyzed by studying the semantic models associated with the term. The reason why we have chosen the alternating bit protocol as a case study to illustrate the integrated approach is that such a protocol has become a standard example in the literature (see, e.g., [Mil89, CPS93, Mol82, NY85, HMR94, ABCSV94]), so it can be used to compare the EMPA model with other models.

8.1 Informal Specification

The *alternating bit protocol* [BSW69] is a data link level communication protocol that establishes a means whereby two stations, one acting as a sender and the other acting as a receiver, connected by a full duplex communication channel that may lose messages, can cope with message loss. The name of the protocol stems from the fact that each message is augmented with an additional bit: since consecutive messages that are not lost are tagged with additional bits that are pairwise complementary, it is easy to distinguish between an original message and its possible duplicates. Initially, if the sender obtains a message from the upper level, it augments the message with an additional bit set to 0, sends the tagged message to the receiver, and starts a timer: if an acknowledgement tagged with 0 is received before the timeout expires, then the subsequent message obtained from the upper level will be sent with an additional bit set to 1, otherwise the current tagged message is sent again. On the other side, the receiver waits for a message tagged with 0: if it receives such a tagged message for the first time, then it passes the message to the upper level, sends an acknowledgement tagged with 0 to the sender, and waits for a message tagged with 1, whereas if it receives a duplicate tagged message (due to message loss, acknowledgement loss, or propagation taking an arbitrarily long time), then it sends an acknowledgement tagged with the same additional bit to the sender.

8.2 Formal Description with EMPA

Since it is helpful to take advantage from compositionality, we figure out how to deal with three interacting entities: *Sender*, *Receiver*, *Channel*. The interaction between *Sender* and *Channel* is described by action types tm_i, $i \in \{0,1\}$, standing for "transmit message tagged with i", and da_i, $i \in \{0,1\}$, standing for "deliver acknowledgement tagged with i". The interaction between *Receiver* and *Channel* is described by action types dm_i, $i \in \{0,1\}$, standing for "deliver message tagged with i", and ta_i, $i \in \{0,1\}$, standing for "transmit acknowledgement tagged with i". The scenario can be modeled as follows:

$$\begin{aligned} ABP &\stackrel{\Delta}{=} Sender_0 \,\|_S\, Channel \,\|_R\, Receiver_0 \\ S &= \{tm_0, tm_1, da_0, da_1\} \\ R &= \{dm_0, dm_1, ta_0, ta_1\} \end{aligned}$$

Thanks to compositionality, we can now focus our attention on the single entities separately. *Channel* is composed of two independent half duplex lines $Line_m$ and $Line_a$. The local activities of *Channel* are described by action types pm_i, $i \in \{0,1\}$, standing for "propagate message tagged with i", and pa_i, $i \in \{0,1\}$, standing for "propagate acknowledgement tagged with i". Additionally, there are other two activities local to *Channel* that are described by action type τ and represent the fact that a message or an acknowledgement is lost or not. As far as the timing of the actions in which *Channel* is involved is concerned, we assume that the length of a message and the length of an acknowledgement are exponentially distributed, so that message/acknowledgement transmission, propagation, and delivery times are exponentially distributed. However, the three phases given by message/acknowledgement transmission, propagation and delivery are temporally overlapped, i.e. they constitute a pipeline. As a consequence, in order to correctly determine the time taken by a message/acknowledgement to reach *Receiver*/*Sender*, we model actions related to transmission and delivery as immediate and we associate the actual timing (i.e., the duration of the slowest stage of the pipeline) with actions related to propagation. We thus assume that the message propagation time is exponentially distributed with rate δ, the acknowledgement propagation time is exponentially distributed with rate γ, and the loss probability is $p \in \mathbb{R}_{]0,1[}$. *Channel* can be modeled as follows:

$$\begin{aligned} Channel &\stackrel{\Delta}{=} Line_m \,\|_\emptyset\, Line_a \\ Line_m &\stackrel{\Delta}{=} <tm_0,*>.<pm_0,\delta>.(<\tau,\infty_{1,1-p}>.<dm_0,\infty_{1,1}>.Line_m + <\tau,\infty_{1,p}>.Line_m) + \\ &\phantom{\stackrel{\Delta}{=}} <tm_1,*>.<pm_1,\delta>.(<\tau,\infty_{1,1-p}>.<dm_1,\infty_{1,1}>.Line_m + <\tau,\infty_{1,p}>.Line_m) \\ Line_a &\stackrel{\Delta}{=} <ta_0,*>.<pa_0,\gamma>.(<\tau,\infty_{1,1-p}>.<da_0,\infty_{1,1}>.Line_a + <\tau,\infty_{1,p}>.Line_a) + \\ &\phantom{\stackrel{\Delta}{=}} <ta_1,*>.<pa_1,\gamma>.(<\tau,\infty_{1,1-p}>.<da_1,\infty_{1,1}>.Line_a + <\tau,\infty_{1,p}>.Line_a) \end{aligned}$$

Observe that the probabilistic choice between the reception and the loss of a message/acknowledgement has been easily represented by means of the weights associated with the two immediate actions $<\tau,\infty_{1,1-p}>$ and $<\tau,\infty_{1,p}>$.

The local activities of *Sender* are described by action types gm standing for "generate message" and to standing for "timeout". We assume that the message generation time is exponentially distributed with rate

λ and that the timeout period is exponentially distributed with rate θ. Of course, this is not realistic, but EMPA does not enable us to express deterministic durations, and a Markovian analysis would not be possible otherwise. A good approximation would consist of describing the deterministic duration of the timeout period by means of a sequence of exponentially timed actions with the same rate (thereby implementing an Erlang distribution, which is a special case of the hypoexponential distribution of Sect. 5.1) as done in [HMR94], but the underlying semantic model would be much bigger than the one in Fig. 10. *Sender* can be modeled as follows:

$$\begin{array}{rcl}
Sender_0 & \triangleq & <gm, \lambda>.<tm_0, \infty_{1,1}>.Sender'_0 \\
Sender'_0 & \triangleq & <da_0, *>.Sender_1 + <da_1, *>.Sender'_0 + <to, \theta>.Sender''_0 \\
Sender''_0 & \triangleq & <tm_0, \infty_{1,1}>.Sender'_0 + <da_0, *>.Sender_1 + <da_1, *>.Sender''_0, \\
Sender_1 & \triangleq & <gm, \lambda>.<tm_1, \infty_{1,1}>.Sender'_1 \\
Sender'_1 & \triangleq & <da_1, *>.Sender_0 + <da_0, *>.Sender'_1 + <to, \theta>.Sender''_1 \\
Sender''_1 & \triangleq & <tm_1, \infty_{1,1}>.Sender'_1 + <da_1, *>.Sender_0 + <da_0, *>.Sender''_1
\end{array}$$

An important observation (similar to the one reported in [CPS93] for a CCS model of the same protocol) concerns terms $Sender''_0$ and $Sender''_1$. Since they model the situation after a timeout expiration, they should comprise the retransmission action only in order to be consistent with the definition of the protocol. The problem is that a deadlock may occur whenever, after a sequence of premature timeouts (i.e. timeouts expired although nothing is lost), the sender is waiting to be able to retransmit the message, the receiver is waiting to be able to retransmit the corresponding acknowledgement, the message line is waiting to be able to deliver a previous copy of the message, and the acknowledgement line is waiting to be able to deliver a previous copy of the acknowledgement. To destroy deadlock, $Sender''_0$ and $Sender''_1$ are allowed to receive possible acknowledgements, thereby avoiding unnecessary retransmissions.

The only local activity of *Receiver* is described by action type *cm* standing for "consume message" which is taken to be immediate in that it is irrelevant from the performance viewpoint. *Receiver* can be modeled as follows:

$$\begin{array}{rcl}
Receiver_0 & \triangleq & <dm_0, *>.<cm, \infty_{1,1}>.<ta_0, \infty_{1,1}>.Receiver_1 + <dm_1, *>.<ta_1, \infty_{1,1}>.Receiver_0 \\
Receiver_1 & \triangleq & <dm_1, *>.<cm, \infty_{1,1}>.<ta_1, \infty_{1,1}>.Receiver_0 + <dm_0, *>.<ta_0, \infty_{1,1}>.Receiver_1
\end{array}$$

8.3 Comparison with Other Formal Descriptions

At the beginning of this section we said that we have chosen the alternating bit protocol in order to compare its EMPA model with others expressed with different formalisms. For example, it turns out that in [Mil89, CPS93] performance aspects are completely neglected because a classical process algebra is used, while in [Mol82] a stochastically timed Petri net model is adopted but the unrealistic assumption that the timeout expires only if a loss actually occurs is made. In [HMR94] a stochastically timed process algebraic description is given, where the deterministic duration of the timeout period has been better approximated by means of an Erlang distribution. However, this description does not accurately take into account the division into three temporally overlapped phases (like in [NY85, ABCSV94]), and represents the probabilistic choice between the reception and the loss of a message/acknowledgement by giving a context dependent meaning to the rate of the actions.

8.4 Functional Analysis

The integrated interleaving semantics of *ABP* is presented in Fig. 10. The LTS $\mathcal{I}[\![ABP]\!]$ has 302 states (76 tangible, 226 vanishing) and 464 transitions (284 observable, 180 invisible; 140 exponentially timed, 324 immediate). Due to the symmetry of the protocol, only half of the state space has been drawn (dashed transitions depict the link with the remaining states). Whenever neither losses nor premature timeouts occur, the states visited by the protocol are 1, 3, 5, 9, 15, 21, 25, 45, 51, 55 and the corresponding symmetric ones, i.e. 2, 4, 6, 10, 16, 22, 26, 46, 52, 56. Following the proposed approach, we can use the LTS $\mathcal{F}[\![ABP]\!]$ (obtained from $\mathcal{I}[\![ABP]\!]$ 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: this means that the

Figure 10: Integrated interleaving semantics of ABP

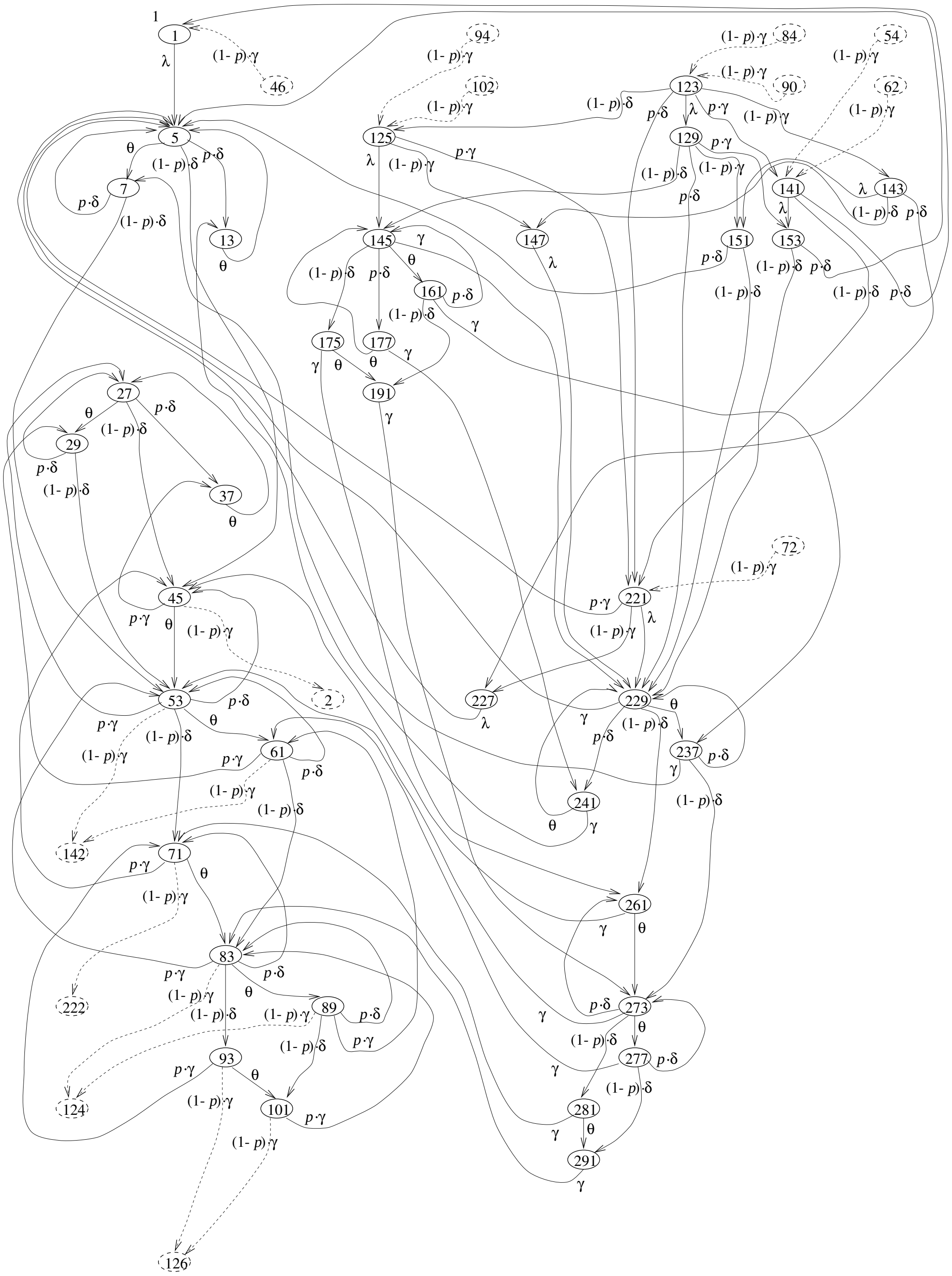


Figure 11: Markovian semantics of *ABP*

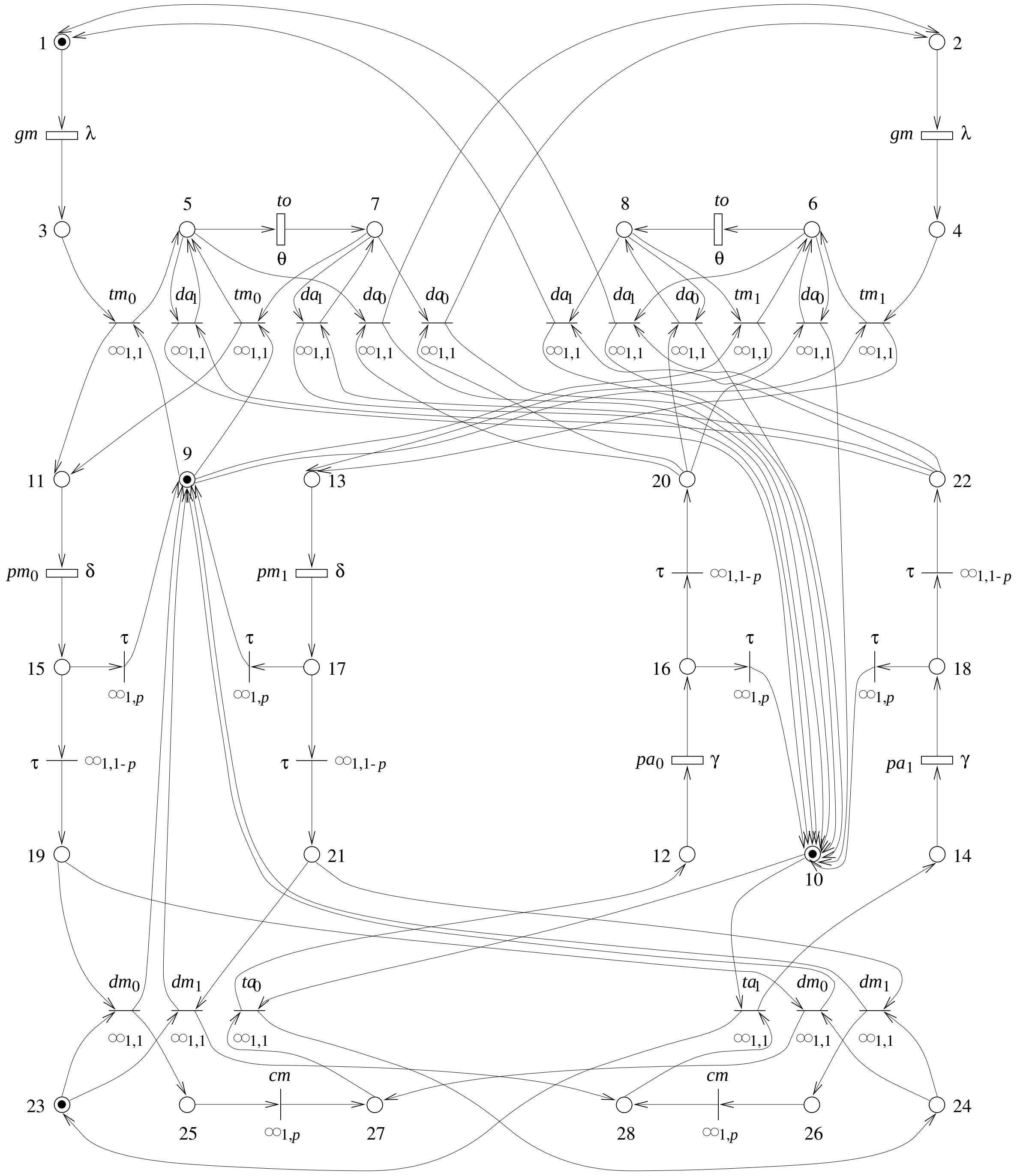


Figure 12: Integrated net semantics of *ABP*

protocol is deadlock free. If $Sender''_0$ and $Sender''_1$ had not been carefully designed (as explained in Sect. 8.2), then we would have obtained eight deadlock states: 113, 197, 213, 301 and the corresponding symmetric ones. As another example, by resorting to equivalence checking we have proved that, whenever all the action types occurring in ABP are hidden except for gm and cm, then ABP behaves as a buffer with capacity one that can engage in a sequence of alternating actions with the two observable types.

8.5 Performance Analysis

The Markovian semantics of ABP is presented in Fig. 11. The p-LTS $\mathcal{M}[\![ABP]\!]$, which has 76 states and 204 transitions, has been obtained from $\mathcal{I}[\![ABP]\!]$ by applying the algorithm in Sect. 4.3. Since $\mathcal{M}[\![ABP]\!]$ is finite and strongly connected, it represents a HCTMC for which the steady state probability distribution function exists. Following the proposed approach, we can exploit such a HCTMC for assessing some performance indices. For example, the throughput of the protocol is given by the number λ of messages per second that arrive at the $Sender$ multiplied by the probability that the $Sender$ can accept a new message to send: this probability is given by the sum of the steady state probabilities of the states having an outgoing transition labeled with λ. In the table below we report the value of the actual throughput for different values of the offered load λ. We assume that the protocol uses two 9600 baud lines and that the (mean) length of the packets is 1024 bits, so that the propagation rate is $\delta = \gamma = 9.375$ packets per second: we finally assume that the timeout period is 1 second ($\theta = 1$) and that the loss probability is $p = 0.05$.

load (msg/sec)	throughput (msg/sec)	load (msg/sec)	throughput (msg/sec)
5	1.106630	30	1.588230
10	1.356460	35	1.607200
15	1.464435	40	1.621760
20	1.524200	45	1.633095
25	1.562150	50	1.642400

8.6 The Equivalent Net Description

The integrated net semantics of ABP is presented in Fig. 12. The GSPN $\mathcal{N}_{loc}[\![ABP]\!]$ comprises 28 places and 36 transitions. Since the integrated net semantics for EMPA meets the retrievability principles, ABP and $\mathcal{N}_{loc}[\![ABP]\!]$ model exactly the same protocol in two different ways: the algebraic description is compositional and more readable, the net description is more concrete and provides a means to detect dependencies, conflicts, and synchronizations among activities which cannot be discovered in an interleaving model like $\mathcal{I}[\![ABP]\!]$. Also notice that $\mathcal{N}_{loc}[\![ABP]\!]$ is considerably more compact than $\mathcal{I}[\![ABP]\!]$: this fact may turn out to be helpful in order to carry out a more efficient assessment of system properties.

9 Conclusion

In this paper we have proposed an integrated approach for modeling and analyzing functional and performance properties of concurrent systems. In order to implement the integrated approach in the exponential case, we have developed a new stochastically timed process algebra called EMPA.

Related work: The idea underlying the integrated approach comes from [Old91], where complementary views of concurrent systems, each one describing the systems at a different level of abstraction, are brought together in one uniform framework by establishing the appropriate semantic links. This realizes the stepwise development of complex systems through various levels of abstraction, which is good practice in software and hardware design. We have then extended the proposal of [Old91] by considering an orthogonal form of integration that relates functional and performance aspects of concurrent systems.

The development of EMPA, instead, has been influenced by the stochastically timed process algebras MTIPP [GHR93b] and PEPA [Hil96], and by the formalism of GSPNs [ABC84, ABCC87]. While designing EMPA, emphasis has been placed on expressiveness.

In EMPA, action durations are mainly given by means of exponentially distributed random variables like in MTIPP and PEPA, but it is also possible to express immediate actions each of which is assigned a priority level and a weight like GSPN transitions. Immediate actions permit to model activities associated with logical events (see, e.g., the delivery of a customer to the server in Sect. 5.2) as well as activities that are irrelevant from the performance viewpoint (see, e.g., message consumption in Sect. 8), thereby providing a mechanism for performance abstraction in the same way as action type τ provides a mechanism for functional abstraction: they also supply the designer with a good degree of flexibility (see, e.g., the description of the pipeline in Sect. 8). Furthermore, immediate actions allow to model concurrent systems whose activities may have different priorities (see, e.g., the QS in Sect. 5.5) and can be used to describe explicitly probabilistic choices avoiding the need of a new operator (see, e.g., message and acknowledgement loss in Sect. 8). Finally, the interplay of exponentially timed and immediate actions makes it possible, though not atomically, the description of activities whose durations follow a phase type distribution (see Sect. 5.1). It is worth noting that, e.g., hyperexponential distributions cannot be represented without weighted immediate actions, since there is no term in which only exponentially timed actions occur such that its Markovian semantics is p-isomorphic to the HCTMC reported in Fig. 7(c). Actually, like weighted immediate transitions in GSPNs, weighted immediate actions are essential in order to model HCTMCs where more than one state can be initial.

EMPA is also endowed with passive actions somewhat different from those of MTIPP and PEPA. Passive actions play a prominent role in EMPA because they allow for nondeterministic choices, and are essential in the synchronization discipline on action rates since it requires that at most one active action is involved. MTIPP and PEPA allow for more general kinds of synchronization [Hil94], but we think that our discipline leads to a more intuitive treatment of the interaction among processes, and in [BG98] we have shown that it is not so restrictive.

As recognized in [BG98], the resulting expressive power of EMPA is considerable: basically, it can be viewed as the union of a classical process algebra, a prioritized process algebra, a probabilistic process algebra, and an exponentially timed process algebra. On the other hand, this has required a great care in the definition of the integrated interleaving semantics (reflected by the use of functions $Melt$, $Select$, and $Norm$ and the related computation of all the potential moves of a term at once), in the definition of the Markovian semantics (because of the possible coexistence of exponentially timed and immediate transitions), and in the definition of the integrated net semantics (witnessed by the handling of marking dependent rates).

Finally, the notion of integrated equivalence $\sim_{EMB}$ has been set up by assembling complementary proposals [LS91, HR94, Hil96, Buc94, Tof94, Mil89] in an elegant and compact way, and it has turned out to be the coarsest congruence contained in $\sim_{FP}$ for a large class of terms, thereby allowing for compositional reasoning and highlighting the necessity (beside the convenience) of defining a notion of equivalence directly on the integrated semantic model.

Tool support: As the various semantics for EMPA can be fully mechanized, we are currently designing a software tool called TwoTowers [BCSS98] which implements the integrated approach of Fig. 1 in the exponential case. The tool is composed of a graphical user interface written in Tcl/Tk [Ous94], a tool driver, an integrated kernel, a functional kernel and a Markovian kernel. The tool driver, which is written in C [KR88] and uses Lex [Les75] and YACC [Joh75], includes routines for parsing EMPA specifications and performing lexical, syntactic, and static semantic (closure, guardedness, finiteness) checks on the specifications. The integrated kernel, which is implemented in C, currently contains only the routines to generate the integrated interleaving semantic model of EMPA specifications according to the rules of Table 1: this kernel will be extended by implementing a $\sim_{EMB}$ checking algorithm. The functional kernel, which is written in C, is based on a version of CWB-NC [CS96] that was retargeted for EMPA using PAC-NC [CMS95]. The Markovian kernel, which is written in C, is in turn based on MarCA [Ste94].

The current version of TwoTowers has been used to study the alternating bit protocol in Sect. 8. In the future, we plan to add a net kernel which compiles EMPA terms to the corresponding integrated net models and analyzes such nets by means of GreatSPN [Chi91].

Future research: Finally we outline several open problems left for future research, some of which have already been addressed since the first version of this paper was prepared, while others are currently being studied.

1. The Markovian semantics for EMPA is defined in the case of coexistence of exponentially timed and immediate transitions by means of an algorithm that manipulates the LTS produced by the integrated interleaving semantics. It would be useful in this case to find a compositional definition for the Markovian semantics by considering the syntactical structure of the term itself, in order not to be forced to scan the whole state space. Also, to tackle state space explosion, efficient aggregation and solution techniques for the MCs underlying terms must be found, possibly exploiting compositionality of terms themselves. Some work in this direction has been done in [Buc94, RS94, HM95, Ser95, HH95, MS96]. Additionally, for the time being the Markovian semantics is defined only for performance closed terms. However it could be extended to all the terms, provided that passive transitions are treated as parametric active transitions. As a consequence, we would obtain parametric MCs suitable for sensitivity analysis of performance.

2. The integrated equivalence $\sim_{EMB}$ is strong, which means that it does not abstract from internal immediate actions, i.e. those actions which are not observable and take no time. From the state space reduction standpoint, it would be profitable to define an integrated equivalence which does abstract from those actions, as it has already been done for some extensions of MTIPP [Ret95, HRW95, HR96]. A somewhat different kind of weak integrated equivalence based on insensitivity results has been instead developed for PEPA [Hil96].
   Moreover, it would be useful to introduce a notion of preorder for EMPA which sorts systems according to their performance.

3. The integrated net semantics for EMPA is developed according to the location oriented approach, i.e. the syntactical structure of terms is encoded within places. From the applicative viewpoint, its major drawback is that the resulting nets are safe, hence huge. In [BBG95] this problem has been solved by resorting to the label oriented approach: terms are decomposed into places that ignore the syntactical structure of terms themselves, notably the presence of parallel composition operators, so that e.g. term $<a, \tilde{\lambda}>.\underline{0} \,\|_{\emptyset} <a, \tilde{\lambda}>.\underline{0}$ needs only one place $<a, \tilde{\lambda}>.\underline{0}$ marked with two tokens instead of two places $<a, \tilde{\lambda}>.\underline{0} \,\|_{\emptyset}\, id$ and $id \,\|_{\emptyset} <a, \tilde{\lambda}>.\underline{0}$. Another optimization concerns choices: alternative compositions are translated by linear constructions instead of Cartesian product constructions. Given a term E, its integrated label oriented net semantics $\mathcal{N}_{lab}[\![E]\!]$ is in general smaller than $\mathcal{N}_{loc}[\![E]\!]$, and sometimes even finite instead of infinite. For instance, while $\mathcal{N}_{loc}[\![QS_{M/M/2/3/4}]\!]$ in Fig. 9 has 16 places, 16 transitions, and 60 arcs, $\mathcal{N}_{lab}[\![QS_{M/M/2/3/4}]\!]$ in [BBG95] has only 7 places, 4 transitions, and 14 arcs. The price to pay is that inhibitor arcs come into play, except for terms in which all the choices are guarded. A different approach for obtaining smaller net representations is proposed in [Rib95], where an integrated denotational net semantics based on colored stochastically timed Petri nets is outlined.

4. A commonly used method to specify steady state performance measures for Markovian models is based on rewards [How71]. The basic idea is that a number describing a reward (or weight) is attached to every state of the Markovian model, and the performance index is defined as the weighted sum of the steady state probabilities of the states of the Markovian model. In order to specify rewards without having to manually scan the whole state space underlying a term of a stochastically timed process algebra, in [Cla96] a method has been proposed which requires expressing a reward structure by means of a logical formula and an arithmetical expression, such that every state satisfying the formula is assigned the reward given by the arithmetical expression. In [Ber97b] we have proposed an alternative method based on the idea of specifying rewards directly in the algebraic model of systems by suitably extending the structure of actions, so that there is no need to resort to a logical formalism and an algebraic treatment of terms which preserves performance measures by means of an extension of $\sim_{EMB}$ is possible. Besides, it is worth noting that the specification of rewards in the algebraic terms prevents their Markovian models from being ordinarily lumped too much.

5. EMPA cannot be used to deal with those systems where data play a fundamental role. To achieve this, in [Ber97a] we have enriched EMPA with value passing features, and the proposal of [HL95]

relying on symbolic LTSs and symbolic bisimulations has been adapted to our framework by providing suitable semantic rules based on lookahead in order to benefit as much as possible from the inherent parametricity of value passing. As an example, the symbolic LTS underlying an EMPA value passing description of the alternating bit protocol has only half as many states as the LTS in Fig. 10.
We also point out that with EMPA it is not possible to model mobility features. Such a topic has been addressed in [Pri95].

6. In [BBCC92] it has been shown that GSPNs can be used to assess both the correctness and the performance of concurrent algorithms, provided that translation rules are given in order to derive a GSPN model from the code of the algorithm. Of course, the same idea can be applied to EMPA. In particular, the translation rules may be set up by following the guideline in [Mil89], where an imperative concurrent programming language is defined and its semantics is given by translation into CCS.

7. The integrated approach of Fig. 1 allows for the simulative analysis of concurrent systems by means of GSPNs, which is quite helpful whenever the state space is huge or even infinite. In [Ber96] it has been argued that this can be done directly with EMPA since its integrated interleaving semantics has been defined in an operational way, thereby making it possible to build the state space on a by need basis (a routine for the simulative analysis of EMPA terms is going to be introduced in TwoTowers). Former algebraic approaches to discrete event simulation can be found in [HS95, KBLL96].

8. How can the integrated approach of Fig. 1 be scaled to general distributions? Although the combined use of exponentially timed and immediate actions allows us to model or approximate many frequently occurring distributions, from the modeling point of view it is advantageous to be able to directly express any distribution, hence this question needs to be answered. Probably, this is the most challenging open problem because we can no longer exploit the memoryless property of exponential distributions, which allowed us to obtain MCs as performance models and to smoothly define the integrated semantics in the interleaving style. Several proposals have been elaborated throughout these years, which can be found in [GHR93a, HS95, ABCSV94, BKLL95, Her96, Pri96]. Our proposal [BBG97], in particular, retains the interleaving approach by adding suitable information to LTSs, and relies on generalized semi Markov processes as performance models since these can be always analyzed via simulation and sometimes by solving the corresponding HCTMCs whenever insensitivity conditions are met which allow for the substitution of exponential distributions for general distributions with the same mean. The purpose is to be able to integrate deterministic and probabilistic durations since they both often occur in the description of systems, and to manage the simultaneous termination of actions whose duration is expressed by noncontinuous distributions. A comparison among the different proposals can be found in [BBG97].

Acknowledgements

We thank Rance Cleaveland, Alessandro Fabbri, Roberto Segala, and Rick Sheldon for the valuable discussions, and Roberto Segala for his careful proofreading of an earlier version of this paper. Furthermore, we are especially grateful to Nadia Busi for her suggestions about the presentation of the integrated interleaving semantics and Mario Bravetti for his remarks about rate normalization in the integrated net semantics. We finally thank the anonymous referees for their helpful suggestions. This research has been partially funded by MURST, CNR, and ESPRIT BRA 8130 LOMAPS.

References

[ABBCCC89] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, A. Cumani, *"The Effect of Execution Policies on the Semantics and Analysis of Stochastic Petri Nets"*, in IEEE Trans. on Software Engineering 15:832-846, 1989

[ABC84] M. Ajmone 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 2:143-172, 1984

[ABCC87] M. Ajmone Marsan, G. Balbo, G. Chiola, G. Conte, *"Generalized Stochastic Petri Nets Revisited: Random Switches and Priorities"*, in Proc. of the *2nd Int. Workshop on Petri Nets and Performance Models (PNPM '87)*, IEEE-CS Press, Madison (WI), 1987

[ABCSV94] M. Ajmone Marsan, A. Bianco, L. Ciminiera, R. Sisto, A. Valenzano, *"A LOTOS Extension for the Performance Analysis of Distributed Systems"*, in IEEE/ACM Trans. on Networking 2:151-164, 1994

[Ajm90] M. Ajmone Marsan, *"Stochastic Petri Nets: An Elementary Introduction"*, in *Advances in Petri Nets '89*, LNCS 424:1-29, 1990

[BBK96] J.C.M. Baeten, J.A. Bergstra, J.W. Klop, *"Syntax and Defining Equations for an Interrupt Mechanism in Process Algebra"*, in Fundamentae Informatica IX:127-168, 1986

[BBCC92] G. Balbo, S.C. Bruell, P. Chen, G. Chiola, *"An Example of Modeling and Evaluation of a Concurrent Program Using Colored Stochastic Petri Nets: Lamport's Fast Mutual Exclusion Algorithm"*, in IEEE Trans. on Parallel and Distributed Systems 3:221-240, 1992

[BSW69] K.A. Bartlett, R.A. Scantlebury, P.T. Wilkinson, *"A Note on Reliable Full-Duplex Transmission over Half-Duplex Links"*, in Comm. of the ACM 12:260-261, 1969

[BBG95] M. Bernardo, N. Busi, R. Gorrieri, *"A Distributed Semantics for EMPA Based on Stochastic Contextual Nets"*, in [PAPM95], pp. 492-509

[BCSS98] M. Bernardo, R. Cleaveland, S. Sims, W. Stewart, *"TwoTowers: A Tool Integrating Functional and Performance Analysis of Concurrent Systems"*, submitted for publication, 1998

[BDG94a] M. Bernardo, L. Donatiello, R. Gorrieri, *"MPA: A Stochastic Process Algebra"*, Technical Report UBLCS-94-10, University of Bologna (Italy), 1994

[BDG94b] M. Bernardo, L. Donatiello, R. Gorrieri, *"Describing Queueing Systems with MPA"*, Technical Report UBLCS-94-11, University of Bologna (Italy), 1994

[BDG94c] M. Bernardo, L. Donatiello, R. Gorrieri, *"Operational GSPN Semantics of MPA"*, Technical Report UBLCS-94-12, University of Bologna (Italy), 1994

[BDG94d] M. Bernardo, L. Donatiello, R. Gorrieri, *"Modeling and Analyzing Concurrent Systems with MPA"*, in [PAPM94], pp. 175-189

[BDG94e] M. Bernardo, L. Donatiello, R. Gorrieri, *"Integrated Analysis of Concurrent Distributed Systems using Markovian Process Algebra"*, in Proc. of the *7th Int. Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE '94)*, Chapman & Hall, pp. 455-457, Berne (Switzerland), 1994

[BDG95] M. Bernardo, L. Donatiello, R. Gorrieri, *"Giving a Net Semantics to Markovian Process Algebra"*, in Proc. of the *6th Int. Workshop on Petri Nets and Performance Models (PNPM '95)*, IEEE-CS Press, pp. 169-178, Durham (NC), 1995

[BDG96] M. Bernardo, L. Donatiello, R. Gorrieri, *"A Stochastic Process Algebra Model for the Analysis of the Alternating Bit Protocol"*, in Proc. of the *11th Int. Symp. on Computer and Information Sciences (ISCIS XI)*, METU, pp. 375-384, Antalya (Turkey), 1996

[Ber96] M. Bernardo, *"A Methodology Based on EMPA for Modeling and Simulating Concurrent Systems"*, in Proc. of the *Annual Conf. of the Italian Society for Computer Simulation (ISCS '96)*, pp. 146-151, Rome (Italy), 1996

[Ber97a] M. Bernardo, *"Enriching EMPA with Value Passing: A Symbolic Approach based on Lookahead"*, in [PAPM97], pp. 35-49

[Ber97b] M. Bernardo, *"An Algebra-Based Method to Associate Rewards with EMPA Terms"*, in Proc. of the *24th Int. Coll. on Automata, Languages and Programming (ICALP '97)*, LNCS 1256:358-368, Bologna (Italy), 1997

[BG98] M. Bernardo, R. Gorrieri, *"A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time"*, to appear in Theoretical Computer Science, 1998

[BV88] G.V. Bochmann, J. Vaucher, *"Adding Performance Aspects to Specification Languages"*, in Proc. of the *8th Int. Conf. on Protocol Specification, Testing and Verification (PSTV VIII)*, North Holland, pp. 19-31, Atlantic City (NJ), 1988

[BBG97] M. Bravetti, M. Bernardo, R. Gorrieri, *"From EMPA to GSMPA: Allowing for General Distributions"*, in [PAPM97], pp. 17-33

[BKLL95] E. Brinksma, J.-P. Katoen, R. Langerak, D. Latella, *"A Stochastic Causality-Based Process Algebra"*, in [PAPM95], pp. 553-565

[Buc94] P. Buchholz, *"Markovian Process Algebra: Composition and Equivalence"*, in [PAPM94], pp. 11-30

[Chi91] G. Chiola, *"GreatSPN 1.5 Software Architecture"*, in Proc. of the *5th Int. Conf. on Modeling Techniques and Tools for Computer Performance Evaluation*, Elsevier, pp. 121-136, Torino (Italy), 1991

[Cla96] G. Clark, *"Formalising the Specification of Rewards with PEPA"*, in [PAPM96], pp. 139-160

[CMS95] W.R. Cleaveland, E. Madelaine, S. Sims, *"A Front-End Generator for Verification Tools"*, in Proc. of the *1st Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '95)*, LNCS 1019:153-173, Aarhus (Denmark), 1995

[CPS93] W.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 15:36-72, 1993

[CS96] W.R. Cleaveland, S. Sims, *"The NCSU Concurrency Workbench"*, in Proc. of the *8th Int. Conf. on Computer Aided Verification (CAV '96)*, LNCS 1102:394-397, New Brunswick (NJ), 1996

[DDM88] P. Degano, R. De Nicola, U. Montanari, *"A Distributed Operational Semantics for CCS Based on Condition/Event Systems"*, in Acta Informatica 26:59-91, 1988

[Fer86] D. Ferrari, *"Considerations on the Insularity of Performance Evaluation"*, in IEEE Trans. on Software Engineering 12:678-683, 1986

[GHR93a] N. Götz, U. Herzog, M. Rettelbach, *"TIPP - A Stochastic Process Algebra"*, in [PAPM93], pp. 31-36

[GHR93b] N. Götz, U. Herzog, M. Rettelbach, *"Multiprocessor and Distributed System Design: The Integration of Functional Specification and Performance Analysis Using Stochastic Process Algebras"*, in Proc. of the *Int. Symp. on Computer Performance Modelling, Measurement and Evaluation (PERFORMANCE '93)*, LNCS 729:121-146, Rome (Italy), 1993

[Har86] C. Harvey, *"Performance Engineering as an Integral Part of System Design"*, in BT Technology Journal 4:143-147, 1986

[Her96] U. Herzog, *"A Concept for Graph-Based Stochastic Process Algebras, Generally Distributed Activity Times, and Hierarchical Modelling"*, in [PAPM96], pp. 1-20

[HH95] P.G. Harrison, J. Hillston, *"Exploiting Quasi-Reversible Structures in Markovian Process Algebra Models"*, in [PAPM95], pp. 510-520

[Hil94] J. Hillston, *"The Nature of Synchronisation"*, in [PAPM94], pp. 51-70

[Hil96] J. Hillston, *"A Compositional Approach to Performance Modelling"*, Cambridge University Press, 1996

[HL95] M. Hennessy, H. Lin, *"Symbolic Bisimulations"*, in Theoretical Computer Science 138:353-389, 1995

[HM95] J. Hillston, V. Mertsiotakis, *"A Simple Time Scale Decomposition Technique for Stochastic Process Algebras"*, in [PAPM95], pp. 566-577

[HMR94] H. Hermanns, V. Mertsiotakis, M. Rettelbach, *"Performance Analysis of Distributed Systems Using TIPP - A Case Study"*, in Proc. of the *10th UK Performance Engineering Workshop for Computer and Telecommunication Systems*, Edinburgh (UK), 1994

[Hoa85] C.A.R. Hoare, *"Communicating Sequential Processes"*, Prentice Hall, 1985

[How71] R.A. Howard, *"Dynamic Probabilistic Systems"*, John Wiley & Sons, 1971

[HR94] H. Hermanns, M. Rettelbach, *"Syntax, Semantics, Equivalences, and Axioms for MTIPP"*, in [PAPM94], pp. 71-87

[HR96] H. Hermanns, M. Rettelbach, *"Towards a Superset of Basic LOTOS for Performance Prediction"*, in [PAPM96], pp. 77-94

[HRW95] H. Hermanns, M. Rettelbach, T. Weiß, *"Formal Characterisation of Immediate Actions in SPA with Nondeterministic Branching"*, in [PAPM95], pp. 530-541

[HS95] P.G. Harrison, B. Strulo, *"Stochastic Process Algebra for Discrete Event Simulation"*, in *Quantitative Methods in Parallel Systems*, Springer, pp. 18-37, 1995

[Joh75] S.C. Johnson, *"YACC - Yet Another Compiler Compiler"*, Computing Science Technical Report 32, AT&T Bell Labs, Murray Hill (NJ), 1975

[KBLL96] J.-P. Katoen, E. Brinksma, R. Langerak, D. Latella, *"Stochastic Simulation of Event Structures"*, in [PAPM96], pp. 21-40

[Kle75] L. Kleinrock, *"Queueing Systems"*, John Wiley & Sons, 1975

[KR88] B.W. Kernighan, D.M. Ritchie, *"The C Programming Language"*, Prentice Hall, 1988

[Les75] M.E. Lesk, *"Lex - A Lexical Analyzer Generator"*, Computing Science Technical Report 39, AT&T Bell Labs, Murray Hill (NJ), 1975

[LS91] K.G. Larsen, A. Skou, *"Bisimulation through Probabilistic Testing"*, in Information and Computation 94:1-28, 1991

[Mil89] R. Milner, *"Communication and Concurrency"*, Prentice Hall, 1989

[Mol82] M.K. Molloy, *"Performance Analysis using Stochastic Petri Nets"*, in IEEE Trans. on Computers 31:913-917, 1982

[MS96] V. Mertsiotakis, M. Silva, *"A Throughput Approximation Algorithm for Decision Free Processes"*, in [PAPM96], pp. 161-178

[Mur89] T. Murata, *"Petri Nets: Properties, Analysis and Applications"*, in Proc. of the IEEE 77:541-580, 1989

[Neu81] M.F. Neuts, *"Matrix-Geometric Solutions in Stochastic Models - An Algorithmic Approach"*, John Hopkins University Press, 1981

[NY85] N. Nounou, Y. Yemini, *"Algebraic Specification-Based Performance Analysis of Communication Protocols"*, in Proc. of the *5th Int. Conf. on Protocol Specification, Testing and Verification (PSTV V)*, North Holland, pp. 541-560, 1985

[Old91] E.-R. Olderog, *"Nets, Terms and Formulas"*, Cambridge University Press, 1991

[Ous94] J.K. Ousterhout, *"Tcl and the Tk Toolkit"*, Addison-Wesley, 1994

[PAPM93] *Proc. of the 1st Workshop on Process Algebras and Performance Modelling (PAPM '93)*, J. Hillston and F. Moller editors, Edinburgh (UK), 1993

[PAPM94] *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling (PAPM '94)*, U. Herzog and M. Rettelbach editors, Erlangen (Germany), 1994

[PAPM95] *Proc. of the 3rd Workshop on Process Algebras and Performance Modelling (PAPM '95)*, S. Gilmore and J. Hillston editors, Computer Journal 38(7), Edinburgh (UK), 1995

[PAPM96] *Proc. of the 4th Workshop on Process Algebras and Performance Modelling (PAPM '96)*, M. Ribaudo editor, CLUT, Torino (Italy), 1996

[PAPM97] *Proc. of the 5th Workshop on Process Algebras and Performance Modelling (PAPM '97)*, E. Brinksma and A. Nymeyer editors, Enschede (The Netherlands), 1997

[Par81] D. Park, *"Concurrency and Automata on Infinite Sequences"*, in Proc. of the *5th GI-Conf. on Theoretical Computer Science*, LNCS 104:167-183, 1981

[Pri95] C. Priami, *"Stochastic π-Calculus"*, in [PAPM95], pp. 578-589

[Pri96] C. Priami, *"Stochastic π-Calculus with General Distributions"*, in [PAPM96], pp. 41-57

[Rei85] W. Reisig, *"Petri Nets: An Introduction"*, Springer-Verlag, 1985

[Ret95] M. Rettelbach, *"Probabilistic Branching in Markovian Process Algebras"*, in [PAPM95], pp. 590-599

[Rib95] M. Ribaudo, *"On the Relationship between Stochastic Process Algebras and Stochastic Petri Nets"*, Ph.D. Thesis, University of Torino (Italy), 1995

[RS94] M. Rettelbach, M. Siegle, *"Compositional Minimal Semantics for the Stochastic Process Algebra TIPP"*, in [PAPM94], pp. 89-105

[Sch84] P. Schweitzer, *"Aggregation Methods for Large Markov Chains"*, in Mathematical Computer Performance and Reliability, pp. 275-286, North Holland, 1984

[Ser95] M. Sereno, *"Towards a Product Form Solution for Stochastic Process Algebras"*, in [PAPM95], pp. 622-632

[Ste94] W.J. Stewart, *"Introduction to the Numerical Solution of Markov Chains"*, Princeton University Press, 1994

[Tof94] C. Tofts, *"Processes with Probabilities, Priority and Time"*, in Formal Aspects of Computing 6:536-564, 1994

[VSSB91] C.A. Vissers, G. Scollo, M. van Sinderen, E. Brinksma, *"Specification Styles in Distributed Systems Design and Verification"*, in Theoretical Computer Science 89:179-206, 1991

[YK82] Y. Yemini, J. Kurose, *"Towards the Unification of the Functional and Performance Analysis of Protocols, or Is the Alternating-Bit Protocol Really Correct?"*, in Proc. of the *2nd Int. Conf. on Protocol Specification, Testing and Verification (PSTV II)*, North Holland, 1982