Electronic Notes in Theoretical Computer Science 39 No. 3 (2000)
URL: http://www.elsevier.nl/locate/entcs/volume39.html 34 pages

# Compositional Asymmetric Cooperations for Process Algebras with Probabilities, Priorities, and Time

Mario Bravetti

*Università di Bologna, Dipartimento di Scienze dell'Informazione*
*Mura Anteo Zamboni 7, 40127 Bologna, Italy*
*Email:* bravetti@cs.unibo.it

Marco Bernardo

*Università di Torino, Dipartimento di Informatica*
*Corso Svizzera 185, 10149 Torino, Italy*
*Email:* bernardo@di.unito.it

**Abstract**

The modeling and analysis experience with process algebras has shown the necessity of extending them with priority, probabilistic internal/external choice, and time in order to be able to faithfully model the behavior of real systems and capture the properties of interest. An important open problem in this scenario is how to obtain semantic compositionality in the presence of all these mechanisms, to allow for an efficient analysis.

In this paper we argue that, when abandoning the classical nondeterministic setting by considering the mechanisms above, a natural solution is to break the symmetry of the roles of the processes participating in a synchronization. We accomplish this by distinguishing between master actions – the choice among which is carried out generatively according to their priorities/probabilities or exponentially distributed durations – and slave actions – the choice among which is carried out reactively according to their priorities/probabilities – and by imposing that a master action can synchronize with slave actions only.

Technically speaking, in this paper we define a process algebra called $\mathrm{EMPA_{gr}}$ including probabilities, priorities, exponentially distributed durations, and the generative master-reactive slaves synchronization mechanism. Then, we prove that the synchronization mechanism in $\mathrm{EMPA_{gr}}$ is correct w.r.t. the novel cooperation structure model, we show that the Markovian bisimulation equivalence is a congruence for $\mathrm{EMPA_{gr}}$, and we present a sound and complete axiomatization for finite terms.

# 1 Introduction

The experience of the past twenty years with process algebras has shown that several expressive features are necessary to be able to model real world systems. Moreover, to be hopefully able to analyze such systems, the expressive features must be introduced in such a way that semantic compositionality is achieved, i.e. in such a way that it is possible to define a congruence that can be exploited to compositionally minimize the state space before applying the analysis techniques.

In this paper we start with a simple process algebra and we show step by step how its expressive power can be greatly enhanced while preserving semantic compositionality. At the beginning, we introduce in the process algebra the concept of time through the capability of expressing exponentially timed actions and passive actions (whose duration becomes specified only upon synchronization with exponentially timed actions of the same type) and we show that the resulting Markovian process algebra can be given semantics in the usual interleaving style thanks to the memoryless property of exponential distributions (Sect. 2).

We then argue that we need a way of representing actions that are irrelevant from the timing viewpoint or just control the system behavior. We thus extend our Markovian process algebra with immediate actions, i.e. actions having duration zero (Sect. 3).

We subsequently observe that it often happens in practice to encounter systems where different competing actions are scheduled according to some priority assignment and/or with a certain frequency. Therefore, we extend our Markovian process algebra by attaching priorities and weights to immediate actions (Sect. 4 and 5).

Afterwards, we note that, when abandoning the classical nondeterministic setting by considering the expressive features above, a natural solution to the problem of achieving semantic compositionality is to break the symmetry of the roles of the processes participating in a synchronization. We accomplish this by distinguishing between master actions (exponentially timed and prioritized-weighted immediate actions) and slave actions (passive actions enriched with priorities and weights enforced only among passive actions of the same type) and by imposing that a master action can synchronize with slave actions only. Following the terminology of [11], the choice among master actions is carried out generatively according to their priorities/weights or exponentially distributed durations, while the choice among slave actions of the same type is carried out reactively according to their priorities/weights (Sect. 6).

After introducing all the ingredients for our extended Markovian process algebra with generative-reactive synchronizations, called $\mathrm{EMPA}_{\mathrm{gr}}$, we formalize its syntax and then we define its operational semantics as a mapping from terms to master-slaves transition systems, which is proved to be correct w.r.t.

the novel cooperation structure model (Sect. 7).

We subsequently define a notion of equivalence in the bisimulation style, which equates $\mathrm{EMPA_{gr}}$ terms possessing the same functional, probabilistic, prioritized and exponentially timed behavior. We then show that such an equivalence is a congruence, thus providing support for compositional manipulation, and we give a sound and complete axiomatization for nonrecursive process terms (Sect. 8).

The paper concludes with a discussion of related work and future research directions to further increase the expressiveness of process algebras, their usability, and their efficient analysis (Sect. 9).

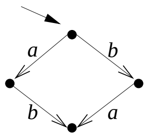
## 2 Markovian Process Algebras

*Process algebras* (see, e.g., [20,17]) are compositional languages for the high level specification of concurrent systems. The main operators to build up system specifications are:

- The *action prefix operator*: $a.E$ is a system that can perform action $a$ and then behaves as described by $E$.
- The *alternative composition operator*: $E_1 + E_2$ is a system that behaves as either $E_1$ or $E_2$ depending on whether an action of $E_1$ or an action of $E_2$ is executed. The choice above is nondeterministic.
- The *parallel composition operator*: $E_1 \,\|_S\, E_2$ is a system that asynchronously executes actions of $E_1$ or $E_2$ not belonging to $S$, and synchronously executes actions of $E_1$ and $E_2$ belonging to the synchronization set $S$ if they are of the same type, which becomes the type of the resulting action.

The syntax of a process algebra is then integrated with other operators. For the time being, we consider the null term $\underline{0}$, which represents a system that cannot execute any action, and the mechanism of constant defining equation $A \stackrel{\Delta}{=} E$, which allows repetitive behaviors to be described.

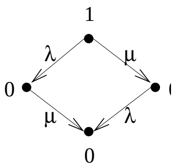
The semantics for process algebra terms is given by means of *rooted labeled transition systems* (LTSs for short) in which states correspond to process terms and transitions are labeled with actions. Such LTSs are defined by following the *interleaving approach*, i.e. parallel executions are serialized by representing each of them through the set of all the possible sequential executions obtained by interleaving the actions executed by the parallel components. A consequence of the interleaving approach is that the two different systems $a.\underline{0} \,\|_\emptyset\, b.\underline{0}$ and $a.b.\underline{0} + b.a.\underline{0}$ are assigned isomorphic LTSs:




In the field of performance evaluation, a model largely used to compute efficiency measures is that of *Markov chains* [24] (MCs for short). In their continuous time variant, MCs are essentially LTSs where the initial state is replaced by a probability mass function, which expresses for each state the probability that it is the initial one, and the transitions are labeled by positive real numbers, which are the rates of the exponentially distributed random variables describing transition durations. Two important properties of continuous time Markov chains (CTMCs for short, as opposed to DTMCs where the first letter stands for discrete) are the following. Given a state $s$ with $n$ outgoing transitions labeled with $\lambda_1, \ldots, \lambda_n$, respectively, we have that:

- The average sojourn time in $s$ is exponentially distributed with rate $\sum_{i=1}^{n} \lambda_i$.
- The probability of executing the $k$-th outgoing transition of $s$ is $\lambda_k / \sum_{i=1}^{n} \lambda_i$.

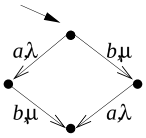
The two properties above essentially stem from the fact that the transitions leaving the same state are thought of as being in a race: the fastest one is the one that is executed. Such a *race policy* naturally applies also to the case in which two actions, whose durations are exponentially distributed with rate $\lambda$ and $\mu$ respectively, are executed in parallel:




We point out that the CTMC above correctly depicts the aforementioned scenario thanks to the *memoryless property* of the exponential distribution, because an action can be regarded as being initiated in the same state in which it terminates its execution. For instance, if in the initial state of the CTMC above (i.e., the state labeled with initial probability 1) the action with rate $\lambda$ is terminated before the action with rate $\mu$, the leftmost state is reached and its outgoing transition is labeled with $\mu$ because, when entering that state, the time to the completion of the action with rate $\mu$ is still exponentially distributed with rate $\mu$. We also observe that no transition is possible from the initial state to the absorbing one as the probability that the two actions terminate simultaneously is zero.

When merged together, the specification languages and the stochastic models above give rise to *Markovian process algebras*. From the syntactical viewpoint, we describe each action as a pair $<a, \tilde{\lambda}>$, where $a$ is the type of the action and $\tilde{\lambda}$ is the rate of the action. If $\tilde{\lambda} \in \mathbb{R}_+$, then the action is called *exponentially timed* as its duration is assumed to be exponentially distributed with rate $\tilde{\lambda}$. If instead $\tilde{\lambda} = *$, then the action is called *passive* and its duration is unspecified. As for the binary operators, the alternative composition operator is governed by the race policy as long as a choice among exponentially timed actions is concerned. In the case of the parallel composition operator, instead, a synchronization between $<a, \tilde{\lambda}>$ and $<a, \tilde{\mu}>$, with $a$ in the synchronization set, is possible only if at least one of $\tilde{\lambda}$ and $\tilde{\mu}$ is $*$, and the resulting rate is given by the other rate. This entails that, in a multiway synchronization, at most one exponentially timed action can be involved (which plays the role of the master, hence determines the rate of the synchronization), while all the other actions (which play the role of slaves) must be passive. We shall return on this master-slaves synchronization mechanism in Sect. 6.

From the semantic viewpoint, we observe that the interleaving approach of process algebras and the memoryless property of exponential distributions fit together well, so that the interleaving approach can be followed also in the case of Markovian process algebras. As an example, the two different systems $<a, \lambda>.\underline{0} \,\|_{\emptyset}\, <b, \mu>.\underline{0}$ and $<a, \lambda>.<b, \mu>.\underline{0} + <b, \mu>.<a, \lambda>.\underline{0}$ are assigned isomorphic LTSs:




The LTS above is called the *integrated interleaving semantics* of the process terms at hand, because each transition is labeled with both the type and the rate of the corresponding action. From such an integrated model two projected semantic models can be derived by discarding action rates or action types, respectively. The former is called the *functional semantics* as its transitions are not decorated with performance related information, thus representing only the functional behavior the system. The latter, instead, is called the *Markovian semantics* as it expresses the CTMC governing the stochastic behavior of the system.

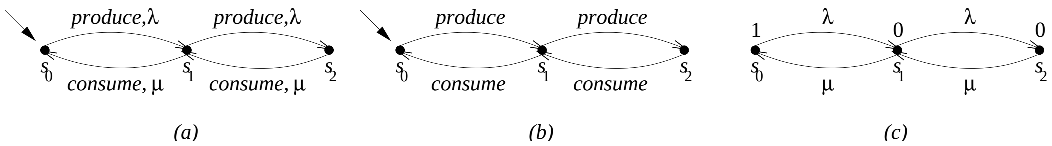



Fig. 1. Interleaving semantic models of $PCSystem_2$

**Example 2.1** A producer/consumer system is a system composed of a producer, a buffer, and a consumer. The producer repeatedly produces new items at a certain speed and puts them into the buffer until the buffer is empty, while the consumer withdraws items from the buffer at a certain rate unless the buffer is empty. Assuming for simplicity a buffer of capacity two, the architecture of this system can be modeled with our Markovian process algebra as follows:

$$PCSystem_2 \triangleq Producer \,\|_{\{produce\}}\, Buffer_0 \,\|_{\{consume\}}\, Consumer$$

Assuming that the item production process and the item consumption process are Markovian with rate $\lambda$ and $\mu$, respectively, the producer and the consumer can be modeled as follows:

$$Producer \triangleq <produce, \lambda>.Producer$$

$$Consumer \triangleq <consume, \mu>.Consumer$$

The buffer, instead, is at any time ready to accept new incoming items (if not full) and to deliver previously produced items (if not empty):

$$\begin{aligned}
\mathit{Buffer}_0 &\stackrel{\Delta}{=} <\mathit{produce}, *>.\mathit{Buffer}_1 \\
\mathit{Buffer}_1 &\stackrel{\Delta}{=} <\mathit{produce}, *>.\mathit{Buffer}_2 + \\
&\qquad <\mathit{consume}, *>.\mathit{Buffer}_0 \\
\mathit{Buffer}_2 &\stackrel{\Delta}{=} <\mathit{consume}, *>.\mathit{Buffer}_1
\end{aligned}$$

Note that only passive actions occur in *Buffer*, to reflect the fact that the interactions established by the two synchronization sets $\{\mathit{produce}\}$ and $\{\mathit{consume}\}$ are guided by the exponentially timed actions of the producer and the consumer.

In Fig. $1(a)$ we show the integrated interleaving semantics of $\mathit{PCSystem}_2$. The initial state $s_0$ corresponds to $\mathit{PCSystem}_2$, state $s_1$ to $\mathit{Producer} \parallel_{\{produce\}} \mathit{Buffer}_1 \parallel_{\{consume\}} \mathit{Consumer}$, state $s_2$ to $\mathit{Producer} \parallel_{\{produce\}} \mathit{Buffer}_2 \parallel_{\{consume\}} \mathit{Consumer}$. As reported in Fig. $1(b)$ and $(c)$, from such a LTS a functional LTS and a CTMC can be derived by dropping action rates or action types, respectively. ■

## 3 Immediate Actions

The first extension of our Markovian process algebra is concerned with the introduction of *immediate actions*. They are executed in zero time, hence their rate is denoted by $\infty$. Introducing immediate actions is necessary to model system activities which are several orders of magnitude faster than those relevant from the performance viewpoint, as well as system activities that control the system behavior.

Since immediate actions have zero durations, they take precedence over exponentially timed ones. To make this clear, let us consider a system $E$ that initially can perform either an exponentially timed action $a$ or an immediate action $b$: $<a, \lambda>.E_1 + <b, \infty>.E_2$. The integrated interleaving semantic model of $E$ has the two following initial transitions:

$$\begin{tikzcd}
 & E \arrow[dl, "{a, \lambda}"'] \arrow[dr, "{b, \infty}"] & \\
E_1 & & E_2
\end{tikzcd}$$

If $E$ represents a closed system, i.e. a system for which all the interactions with its environment have been described, then only the transition labeled with action $<b, \infty>$ can be executed. If instead $E$ represents an open system, then the execution of action $<b, \infty>$ may be disabled by the environment. For instance, $E \parallel_{\{b\}} \underline{0}$ has a single initial transition labeled with action $<a, \lambda>$, as $\underline{0}$ is not willing to perform any $b$ action hence no synchronization on $b$ can occur.

Similarly to exponentially timed actions, in a synchronization at most one immediate action can be involved while all the other actions must be passive. If an immediate action is involved, then the rate of the resulting action is immediate, otherwise it is passive.

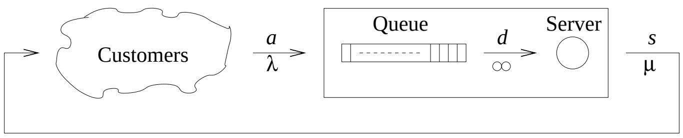



Fig. 2. Structure of a queueing system $M/M/1/q$

**Example 3.1** From now on we shall exemplify each feature added to our language by means of *queueing systems* [18] (QSs for short), which are abstract models largely used for evaluating the performance of computer and communication systems through the computation of measures such as system throughput, resource utilization, and user response time. A QS is a service center, composed of a waiting queue and a given number of servers, which provides a certain service to a population of customers according to a given discipline. In the following, we shall be concerned with QSs $M/M/n/q/m$ with arrival rate $\lambda$ and service rate $\mu$, which are defined as follows:

- (i) The customer arrival process is Markovian with rate $\lambda$.
- (ii) The customer service process is Markovian with rate $\mu$.
- (iii) There are $n$ independent servers.
- (iv) There is a FIFO queue with $q - n$ seats. When missing, parameter $q$ denotes an unbounded queue.
- (v) There are $m$ independent customers. When missing, parameter $m$ denotes an unbounded population of customers.

Let us consider a QS $M/M/1/q$ with arrival rate $\lambda$ and service rate $\mu$, whose structure is depicted in Fig. 2 where $a$ stands for arrive, $d$ for deliver, and $s$ for serve. To faithfully represent the fact that the buffer has capacity $q-1$, an immediate action is necessary to model the fact that the customer at the beginning of the queue is passed to the server as soon as it becomes free. Without such an immediate action, the capacity of the service center would be decreased by one.

The QS at hand can be modeled as follows:

$$\begin{aligned} QS_{M/M/1/q} &\triangleq Arrivals \,\|_{\{a\}} (Queue_0 \,\|_{\{d\}}\, Server) \\ Arrivals &\triangleq <a, \lambda>.Arrivals \end{aligned}$$

$$
\begin{aligned}
Queue_0 &\triangleq \,<\!a,*\!>.Queue_1 \\
Queue_h &\triangleq \,<\!a,*\!>.Queue_{h+1} + \\
&\quad\; <\!d,*\!>.Queue_{h-1}, \;\; 0 < h < q-1 \\
Queue_{q-1} &\triangleq \,<\!d,*\!>.Queue_{q-2} \\
Server &\triangleq \,<\!d,\infty\!>.<\!s,\mu\!>.Server
\end{aligned}
$$

where we note that all the actions describing the behavior of the queue are passive. We conclude by showing the Markovian semantic model of $QS_{M/M/1/q}$ in Fig. 3($b$), which is obtained from the integrated semantic model of $QS_{M/M/1/q}$ in Fig. 3($a$), where $AQ_hS$ stands for $Arrivals \parallel_{\{a\}} (Queue_h \parallel_{\{d\}} Server)$, $AQ_hS'$ stands for $Arrivals \parallel_{\{a\}} (Queue_h \parallel_{\{d\}} <\!s,\mu\!>.Server)$, and $0 \leq h \leq q-1$. We observe that, when deriving a CTMC from an integrated LTS, the immediate transitions and the related source states are removed. The reason is that the sojourn time in those states is zero, so they are irrelevant from the performance viewpoint. ■

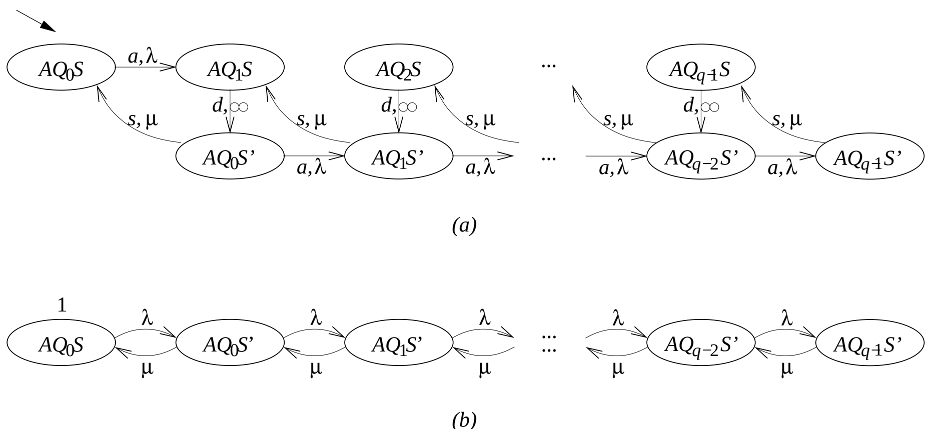
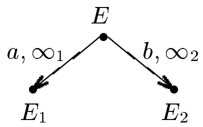



Fig. 3. Integrated and Markovian semantic models of $QS_{M/M/1/q}$

## 4 Prioritized Choices

The second extension of our Markovian process algebra is concerned with the introduction of *priorities*, which are expressed as positive natural numbers attached to immediate action rates ($\infty_l$). Introducing priorities is necessary to model prioritized choices and to improve the capability of expressing system control mechanisms, such as preemption.

Higher priority immediate actions take precedence over lower priority ones. To make this clear, let us consider a system $E$ that initially can perform either an immediate action $a$ with priority 1 or an immediate action $b$ with priority 2: $<\!a,\infty_1\!>.E_1 + <\!b,\infty_2\!>.E_2$. The integrated interleaving semantic model of $E$ has the two following initial transitions:




If $E$ represents a closed system, then only the transition labeled with action $<b, \infty_2>$ can be executed. If instead $E$ represents an open system, then the execution of action $<b, \infty_2>$ may be disabled by the environment. For instance, $E \,\|_{\{b\}}\, \underline{0}$ has a single initial transition labeled with action $<a, \infty_1>$.

In the case of synchronization of an immediate action and a passive action, the resulting immediate action inherits the priority of the original immediate action.

**Example 4.1** Let us consider a variant of the QS of Ex. 3.1 in which there are two different classes of customers, reds and blacks, with two different arrival rates, $\lambda_r$ and $\lambda_b$. The service center comprises two distinct queues of capacity $q-1$ for the two classes of customers. In the situation in which both queues are nonempty and the server is free, the first come red customer must be served, i.e. red customers take precedence over black customers. This can be easily modeled in our Markovian process algebra extended with priorities as follows:

$$\begin{aligned}
QS_{prio} &\triangleq (Arrivals_r \,\|_\emptyset\, Arrivals_b) \,\|_{\{a_r, a_b\}} ((Queue_{r,0} \,\|_\emptyset\, Queue_{b,0}) \,\|_{\{d_r, d_b\}}\, Server) \\
Server &\triangleq <d_r, \infty_r>.<s, \mu>.Server\; + \\
&\quad\; <d_b, \infty_b>.<s, \mu>.Server
\end{aligned}$$

where $Arrivals_r$ ($Arrivals_b$) is the same as $Arrivals$ in which every action type is given subscript $r$ ($b$), $Queue_{r,0}$ ($Queue_{b,0}$) is the same as $Queue_0$ in which every action type is given subscript $r$ ($b$), and $r > b$.

Note that in the model above, no preemption can be exercised on the black customer being served in the case a red customer arrives at the service center. To take this into account, it is sufficient to modify the model of the server as follows:

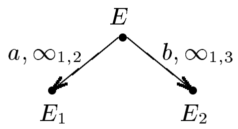
$$\begin{aligned}
Server &\triangleq <d_r, \infty_r>.<s, \mu>.Server_r\; + \\
&\quad\; <d_b, \infty_b>.<s, \mu>.Server_b \\
Server_r &\triangleq <s, \mu>.Server \\
Server_b &\triangleq <s, \mu>.Server\; + \\
&\quad\; <d_r, \infty_r>.<s, \mu>.Server_b
\end{aligned}$$

where the second summand of $Server_b$ describes the service of the newly arrived, preempting red customer. In such a model the memoryless property of exponential distributions guarantees that the remaining time to the completion of the service of a preempted black customer is still exponentially distributed with rate $\mu$. Therefore, the first summand of $Server_b$ is used to describe both the service of a black customer with no interruption and the service of a black customer which has been interrupted several times. ■

## 5 Probabilistic Choices

The third extension of our Markovian process algebra is concerned with the introduction of *weights*, which are expressed as positive real numbers attached to immediate action rates ($\infty_{l,w}$, thus resembling immediate transitions of generalized stochastic Petri nets [1]). Introducing weights is necessary to model probabilistic choices and to improve the capability of expressing system control mechanisms, such as probabilistic events.

The execution probability of immediate actions at the same priority level is proportional to their weights. To make this clear, let us consider a system $E$ that initially can perform either an immediate action $a$ with priority 1 and weight 2 or an immediate action $b$ with priority 1 and weight 3: $<a, \infty_{1,2}>.E_1 + <b, \infty_{1,3}>.E_2$. The integrated interleaving semantic model of $E$ has the two following initial transitions:




If $E$ represents a closed system, then the former transition is executed with probability $2/(2+3) = 0.4$, while the latter transition is executed with probability $3/(2+3) = 0.6$. If instead $E$ represents an open system, then the execution of one of its two actions may be disabled by the environment. For instance, $E \,\|_{\{b\}}\, \underline{0}$ has a single initial transition labeled with action $<a, \infty_{1,2}>$ which is executed with probability $2/2 = 1$.

In summary, the strategy adopted to choose among several alternative immediate actions is the *preselection policy*: the immediate actions having the highest priority level are singled out, then each of them is given an execution probability proportional to its weight.

In the case of synchronization of an immediate action and a passive action, the resulting immediate action inherits also the weight of the original immediate action.

**Example 5.1** Let us consider a variant of the QS of Ex. 4.1 such that, in the situation in which both queues are nonempty and the server is free, the first come red customer and the first come black customer have the same priority but different frequencies with which they are served, say $r/(r+b)$ and $b/(r+b)$, respectively. This can be taken into account with our Markovian process algebra extended with weights by simply modifying the model of the server as follows:

$$\begin{aligned} Server \triangleq\ & <d_r, \infty_{l,r}>.<s, \mu>.Server\ + \\ & <d_b, \infty_{l,b}>.<s, \mu>.Server \end{aligned}$$

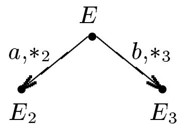
■

## 6 Master-Slaves Synchronization

Our extended Markovian process algebra employs an *asymmetric master-slaves synchronization mechanism*, where exponentially timed and immediate actions (also called active actions) play the role of the masters, in the sense that they determine the rate of the resulting action, while passive actions play the role of the slaves. Such a mechanism is enforced by imposing that, in case of multiway synchronization, at most one active action can be involved while all the other actions must be passive. More formally, we adopt a *CSP [17] like parallel composition operator*, which allows for multiway synchronizations by assuming that the result of the synchronization of two actions with type $a$ is again an action with type $a$. In addition, we impose that a synchronization between two actions of type $a$ may occur only if either they are both passive actions (and the result is a passive action of type $a$), or one of them is an active action and the other one is a passive action (and the result is an active action of type $a$).

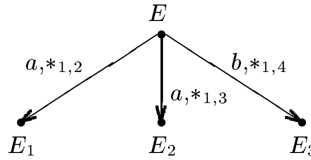
So far we have considered particular kinds of binary synchronizations in which an active action of a process could synchronize with a single passive action of another process only. However, if several alternative passive actions of a given type may synchronize with the same active action of that type, it remains to establish how we choose among those passive actions. This is accomplished in two steps.

First of all, we endow passive actions with positive natural numbers acting as *reactive priorities* ($*_l$). Unlike priorities of immediate actions, reactive priorities are enforced only among passive actions of the same type, which makes it safe to discard lower priority passive actions of a given type. To make this clear, let us consider a system $E$ that initially can perform a passive action $a$ with priority 1, a passive action $a$ with priority 2, or a passive action $b$ with priority 3: $<a, *_1>.E_1 + <a, *_2>.E_2 + <b, *_3>.E_3$. The integrated interleaving semantic model of $E$ has the two following initial transitions:




As it can be noted, a transition labeled with action $<a, *_2>$ is in the model above because the highest priority transition has a different type, whereas there is no transition labeled with action $<a, *_1>$ because of the presence of a higher priority transition of the same type. Due to the reactive meaning ascribed to priorities of passive actions, the environment cannot disable the higher priority passive $a$ action and enable the lower priority passive $a$ action at the same time, so it is safe to neglect lower priority passive actions of a given type.

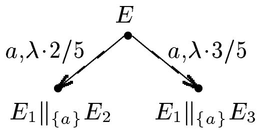
Second, we endow passive actions with positive real numbers acting as *reactive weights* ($*_{l,w}$). Unlike weights of immediate actions, reactive weights determine the choice only among passive actions of the same type. To make this clear, let us consider a system $E$ that initially can perform a passive action $a$ with priority 1 and weight 2, a passive action $a$ with priority 1 and weight 3, or a passive action $b$ with priority 1 and weight 4: $<a, *_{1,2}>.E_1 + <a, *_{1,3}>.E_2 + <b, *_{1,4}>.E_3$. The integrated interleaving semantic model of $E$ has the three following initial transitions:



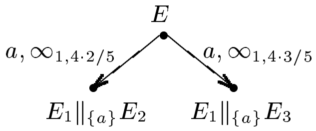

Because of the reactive interpretation of passive action weights, the first transition is executed with probability $2/(2+3) = 0.4$, the second with probability $3/(2+3) = 0.6$, and the third with probability $4/4 = 1$.

In summary, the strategy adopted to choose among several alternative passive actions is the *reactive preselection policy*: for a given type, the passive actions of that type having the highest priority level are singled out, then each of them is given an execution probability proportional to its weight.

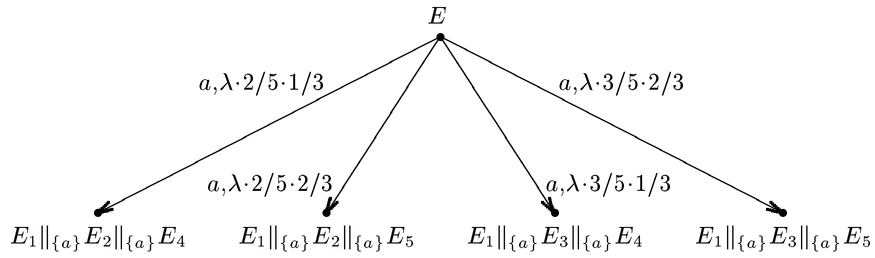
We are now in a position of explaining how the rate of an action resulting from a master-slaves synchronization is determined. In the case of synchronization between an exponentially timed action of rate $\lambda$ and a passive action of the same type, the resulting rate is $\lambda \cdot p$ where $p$ is the execution probability of the passive action. As an example, term $E$ defined by $<a, \lambda>.E_1 \,\|_{\{a\}} (<a, *_{1,2}>.E_2 + <a, *_{1,3}>.E_3)$ has the two following initial transitions:




In the case of synchronization between an immediate action of rate $\infty_{l,w}$ and a passive action of the same type, the resulting rate is $\infty_{l,w \cdot p}$ where $p$ is the execution probability of the passive action. As an example, term $E$ defined by $<a, \infty_{1,4}>.E_1 \,\|_{\{a\}} (<a, *_{1,2}>.E_2 + <a, *_{1,3}>.E_3)$ has the two following initial transitions:




In the case of a multiway synchronization where an active action is synchronized with several passive actions, each passive action is chosen by performing an *independent* choice. That is, if an exponentially timed (immediate) action with rate $\lambda$ ($\infty_{l,w}$) synchronizes with $n$ passive actions of the same type, the resulting rate is $\lambda \cdot \prod_{i=1}^{n} p_i$ ($\infty_{l,w \cdot \prod_{i=1}^{n} p_i}$) where $p_i$ is the execution probability of the $i$-th passive action involved in the synchronization. As an example, term $E$ defined by $<a, \lambda>.E_1 \,\|_{\{a\}} (<a, *_{1,2}>.E_2 + <a, *_{1,3}>.E_3) \,\|_{\{a\}} (<a, *_{1,1}>.E_4 + <a, *_{1,2}>.E_5)$ has the four following initial transitions:




In general our master-slaves synchronization mechanism can be interpreted as an extension to priorities and exponential time of the probabilistic synchronization mechanism presented in [6] based on a mixture of the generative and reactive models of probabilistic processes of [11].

We now briefly recall the two models above by resorting to the terminology of [20], where an action type based synchronization is described in terms of *button pushing experiments*. In this view, the environment experiments on a process by pushing one of several buttons, where a button represents an action type. According to the *reactive model* of probability, a process reacts internally to a button push performed by its environment on the basis of a probability distribution which depends on the button which is pushed. According to the *generative model* of probability, instead, the process itself autonomously decides, on the basis of a probability distribution, which button will go down and how to behave after such an event.

When two processes behaving in a reactive way synchronize on an action $a$, each of them reacts internally to the synchronization according to the probability distribution associated with the actions of type $a$ it can perform. Whenever the two processes can synchronize on more than one action type, each of them leaves the decision to the environment, hence the choice of the synchronizing action type turns out to be nondeterministic. This kind of synchronization is simple and natural, but does not make it possible to express a mechanism for the choice of the button to be pushed (external choice), thus leaving the system, in a sense, underspecified.

On the other hand, two processes behaving in a generative way independently decide the action type on which they want to synchronize, hence there may be no agreement on the action type.

A solution to this problem proposed in [26,6] is to adopt a *mixed generative-reactive approach* based on an *asymmetric form of synchronization*, where a process which behaves generatively may synchronize only with processes which behave reactively. The intuition behind this solution, suggested also in [22], is that the process which behaves generatively decides which button will go down (and how to behave afterwards) and the process which behaves reactively just reacts to the button push of the other process. In [26,6], the integration of the generative and reactive approaches is naturally obtained by designating some actions (the master actions) as behaving generatively and the other actions (the slave actions) as behaving reactively, and by imposing (as we do in this paper) that master actions can synchronize with slave actions only.

According to the master-slaves synchronization mechanism of our extended Markovian process algebra, which extends the generative-reactive mechanism explained above, we have that, in a system state, first a *master choice* is generatively made according to the rates of the master actions. Then, if the chosen master action must synchronize, a *slave choice* is reactively made among the slave actions that can synchronize with the selected master action according to their reactive rates.

We conclude by observing that our generative master-reactive slaves synchronization mechanism complies with the *bounded capacity assumption* [16], which establishes that the rate of an action cannot be arbitrarily increased/decreased when synchronizing it with several actions. This assumption, which imposes a safe modeling methodology from the stochastic viewpoint, is satisfied because it can be easily shown that our mechanism preserves the average sojourn times. For instance, in the example depicted in the last encountered figure, we have that the rates of the four transitions sum up to $\lambda$, which is exactly the rate of the only active action present in $E$. Additionally, we point out that in our Markovian framework extended with immediate actions it is possible to simulate a synchronization between two $a$ actions with rate $\lambda$ and $\mu$, respectively, whose duration is the *maximum* of the two durations [13]. If we denote by $\tau$ an action type representing an invisible activity, this is easily achieved by means of a term like $<\tau, \lambda>.<a, \infty_{l,w}>.\underline{0} \parallel_{\{a\}} <\tau, \mu>.<a, *_{l',w'}>.\underline{0}$, as it gives rise to the first CTMC depicted in Sect. 2.

**Example 6.1** Attaching reactive priorities and weights to passive actions turns out to be advantageous from the modeling viewpoint as it allows more compact process algebraic descriptions to be obtained. As an example, let us consider a variant of the QSs of Ex. 4.1 and 5.1 in which there are $n$ classes of customers, with class $i$, $1 \leq i \leq n$, having arrival rate $\lambda_i$, service priority $l_i$, and service frequency $w_i / \sum_{j=1}^{n} w_j$. If we denote by $\prod$ the parallel composition of several terms which do not synchronize on any action, the QS above can be modeled in our Markovian process algebra extended with reactive priorities and weights as follows:

$$\begin{aligned} QS_n &\triangleq \prod_{i=1}^{n} Arrivals_i \parallel_{\{a_i \mid 1 \leq i \leq n\}} (\prod_{i=1}^{n} Queue_{i,0} \parallel_{\{d\}} Server) \\ Arrivals_i &\triangleq <a_i, \lambda_i>.Arrivals_i \end{aligned}$$

$$
\begin{aligned}
Queue_{i,0} &\stackrel{\Delta}{=} \lt a_i, *_{1,1}\gt .Queue_{i,1}\\
Queue_{i,h} &\stackrel{\Delta}{=} \lt a_i, *_{1,1}\gt .Queue_{i,h+1} +\\
&\quad\ \lt d, *_{l_i,w_i}\gt .Queue_{i,h-1}, \ \ 0 < h < q-1\\
Queue_{i,q-1} &\stackrel{\Delta}{=} \lt d, *_{l_i,w_i}\gt .Queue_{i,q-2}\\
Server &\stackrel{\Delta}{=} \lt d, \infty_{1,1}\gt .\lt s, \mu\gt .Server
\end{aligned}
$$

It is worth observing that the model above is scalable w.r.t. the number of classes, in the sense that the description of the server does not need to be modified when adding/removing a class of customers. This is made possible by the fact that the information about the service priority and frequency of each class must not necessarily be described within the server (as it would be if priorities and weights could not be attached to passive actions), but can be described in the model for the queue corresponding to the class. ■

## 7 Syntax and Semantics for EMPA$_{\mathrm{gr}}$

In this section we formalize the syntax and the semantics for the process algebra informally presented in the previous section. More precisely, we define the syntax of an extended Markovian process algebra with generative-reactive synchronizations called EMPA$_{\mathrm{gr}}$. Then we introduce the master-slaves transition system model and we present an operational semantics that maps EMPA$_{\mathrm{gr}}$ terms onto such a model. Finally, the semantics is proven correct by means of the novel cooperation structure model.

### *7.1 Syntax and Informal Semantics*

The main ingredients of our calculus are the actions, each composed of a type and a rate, and the algebraic operators. As far as actions are concerned, based on their rates they are classified into exponentially timed, immediate, and passive, as already seen. Moreover, based on their types they are classified into visible and invisible depending on whether they are different or equal to $\tau$, as usual.

**Definition 7.1** Let *AType* be the set of *action types*, including the invisible type $\tau$, and $ARate = \mathbb{R}_+ \cup \{\infty_{l,w} \mid l \in \mathbb{N}_+ \wedge w \in \mathbb{R}_+\} \cup \{*_{l,w} \mid l \in \mathbb{N}_+ \wedge w \in \mathbb{R}_+\}$ be the set of *action rates*. We use $a$ to range over *AType*, $\tilde{\lambda}$ to range over *ARate*, $\lambda$ to range over exponentially timed rates, and $\bar{\lambda}$ to range over non-passive rates. The set of *actions* is defined by

$$Act = AType \times ARate$$

■

**Definition 7.2** Let *Const* be a set of *constants* ranged over by $A$ and let $ATRFun = \{\varphi : AType \longrightarrow AType \mid \varphi^{-1}(\tau) = \{\tau\}\}$ be a set of *action type* *relabeling functions* ranged over by $\varphi$. The set $\mathcal{L}$ of *process terms* of $\mathrm{EMPA_{gr}}$ 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 *null term* "$\underline{0}$" is the term that cannot execute any action.

The *action prefix operator* "$<a, \tilde{\lambda}>.\_$" denotes the sequential composition of an action and a term. Term $<a, \tilde{\lambda}>.E$ can execute an action with type $a$ and rate $\tilde{\lambda}$ 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 $a$ of each executed action is turned into $\tau$ whenever $a \in L$.

The *functional relabeling operator* "$\_[\varphi]$" changes the type of the actions. Term $E[\varphi]$ behaves as term $E$ except that the type $a$ of each executed action becomes $\varphi(a)$.

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. As we have already seen, the choice is solved according to the race policy in case of exponentially timed actions, the preselection policy in case of immediate actions, and the reactive preselection policy in case of passive actions.

The *parallel composition operator* "$\_ \,\|_S\, \_$" expresses the concurrent execution of two terms. Term $E_1 \,\|_S\, E_2$ asynchronously executes actions of $E_1$ or $E_2$ not belonging to $S$ and synchronously executes actions of $E_1$ and $E_2$ belonging to $S$ according to the two following synchronization disciplines. The synchronization discipline on action types establishes that two actions can synchronize if and only if they have the same observable type in $S$, which becomes the resulting type. The synchronization discipline on action rates is the generative master-reactive slaves mechanism explained in Sect. 6. In case of synchronization of an active action $a$ having rate $\tilde{\lambda}$ executed by $E_1$ ($E_2$) with a passive action $a$ having rate $*_{l,w}$ executed by $E_2$ ($E_1$), the resulting active action $a$ has rate/weight given by the original rate/weight multiplied by the probability that $E_2$ ($E_1$) chooses the passive action at hand among its passive actions of type $a$. Instead, in case of synchronization of two passive actions $a$ having rate $*_{l_1,w_1}$ and $*_{l_2,w_2}$ executed by $E_1$ and $E_2$, respectively, the resulting passive action of type $a$ has priority level given by the maximum $l_{max}$ between $l_1$ and $l_2$ and weight given by the probability that $E_1$ and $E_2$ independently choose the two actions, multiplied by a normalization factor given by the overall weight of the passive actions of type $a$ executable by $E_1$ and $E_2$ at the priority level $l_{max}$. The choice of such a normalization factor and of the priority level of the resulting passive action makes the structure of synchronizations in a system state easier to understand, as will be shown in Sect. 7.4.

Finally, let partial function $Def : Const \multimap \mathcal{L}$ be a set of *constant defining* *equations* of the form $A \stackrel{\Delta}{=} E$. In order to guarantee the correctness of recursive definitions, as usual we restrict ourselves to the set $\mathcal{G}$ of terms that are closed and guarded w.r.t. *Def*.

### *7.2 Master-Slaves Transition Systems*

The semantic model of $\mathrm{EMPA_{gr}}$ is a special kind of LTS we call *master-slaves transition system* (MSTS for short), whose transitions are labeled with elements of *Act*. Recalling that active actions play the role of the masters while passive actions play the role of the slaves, each state of a MSTS has a single *master bundle* composed of all the transitions labeled with an active action and, for each action type $a$, a single *slave bundle* of type $a$ composed of all the transitions labeled with a passive action of type $a$. Since the operational semantics for $\mathrm{EMPA_{gr}}$ will be defined in such a way that lower priority active transitions are not pruned (see the congruence related motivation in Sect. 8) while lower priority passive transitions of a given type are, all the passive transitions belonging to the same slave bundle of a generated MSTS have the same priority level.

**Definition 7.3** A master-slaves transition system (MSTS) is a triple

$$(S, AType, \longrightarrow)$$

where: [1]

- $S$ is a set of states;
- $AType$ is a set of action types;
- $\longrightarrow \in \mathcal{M}(S \times Act \times S)$ is a multiset of transitions such that for all $s \in S$ and $a \in AType$

$$(s \xrightarrow{a,*_{l',w'}} s' \wedge s \xrightarrow{a,*_{l'',w''}} s'') \implies l' = l''$$

A rooted master-slaves transition system (RMSTS) is a quadruple

$$(S, AType, \longrightarrow, s_0)$$

where $(S, AType, \longrightarrow)$ is a MSTS and $s_0 \in S$ is the initial state. ■

We point out that the transition relation is a multiset, not a set. This allows the multiplicity of identically labeled transitions to be taken into account, which is necessary from the stochastic point of view. As an example, if a state has two transitions both labeled with $<a, \lambda>$, using sets instead of multisets would reduce the two transitions into a single one labeled with $<a, \lambda>$, thus erroneously altering the average sojourn time in the state.

The choice of a transition within the master bundle of a state is made according to the race policy, i.e. the transition sampling the least duration succeeds, with immediate transitions taking precedence over exponentially

[1] We use "$\{\!|$" and "$|\!\}$" as brackets for multisets and $\mathcal{M}(S)$ ($\mathcal{P}(S)$) to denote the collection of multisets over (subsets of) set $S$.

timed transitions. We consider the transitions composing a master bundle as grouped according to their priority level. The level zero is composed of all the transitions labeled with exponentially timed actions and, for each $l \in \mathbb{N}_+$, the level $l$ is composed of all the transitions labeled with an immediate action with priority $l$. If all the transitions composing the master bundle are labeled with exponentially timed actions, then the master bundle includes the group of transitions at level zero only. Supposed that such a group is composed of $n$ transitions labeled with active actions $<a_i, \lambda_i>$, $1 \leq i \leq n$, then the time to choose one of such actions is exponentially distributed with rate $\sum_{1 \leq i \leq n} \lambda_i$ and the probability of choosing $a_k$ is given by $\lambda_k / \sum_{1 \leq i \leq n} \lambda_i$. Otherwise, if there is some transition labeled with an immediate action, the preselection policy is applied, which means that a probabilistic choice is made in zero time according to the weights of the immediate actions labeling the group of transitions at the maximum priority level $l_{max}$. Supposed that such a group is composed of $n$ transitions labeled with active actions $<a_i, \infty_{l_{max}, w_i}>$, $1 \leq i \leq n$, then the probability of choosing $a_k$ is given by $w_k / \sum_{1 \leq i \leq n} w_i$.

The choice within a slave bundle of type $a$ is governed by the prelection policy: each transition of the bundle is chosen with probability proportional to its weight. Supposed that such a bundle is composed of $n$ transitions labeled with passive actions $<a_i, *_{l, w_i}>$, $1 \leq i \leq n$, then the probability of choosing $a_k$ is given by $w_k / \sum_{1 \leq i \leq n} w_i$. Since the duration of passive actions is unspecified, the time to choose one of the actions above is unspecified.

We conclude by recalling that passive actions are seen as *incomplete* actions which must synchronize with active actions of the same type of another system component in order to form a complete system. Therefore a fully specified system is *performance closed*, in the sense that it gives rise to a fully probabilistic transition system which does not include slave bundles. If in such a transition system we keep for each state only the highest priority transitions, then we can easily derive a performance model in the form of a DTMC or a CTMC, depending on whether only immediate transitions occur or not. Should exponentially timed and immediate transitions coexist (in different states), a CTMC is derived by eliminating the immediate transitions and the related source states and by suitably splitting the exponentially timed transitions entering the removed source states in such a way that they are caused to reach the target states of the removed immediate transitions. The reader interested in the details of this procedure is referred to [3] Chap. 4.

### *7.3 Operational Semantics*

The formal semantics for $\text{EMPA}_{\text{gr}}$ maps terms onto RMSTSs. We preliminarily provide the following shorthands to make the definition of the operational semantic rules easier.

**Definition 7.4** Given a MSTS $M = (S, AType, \longrightarrow)$, $s \in S$, and $a \in AType$, we denote by $L_a(s)$ the priority level of the slave transitions of type $a$

executable at $s$ ($L_a(s) = 0$ if the slave bundle $a$ of $s$ is empty) and we denote by $W_a(s)$ the overall weight of the slave transitions of type $a$ executable at $s$:

$$W_a(s) = \sum \{\!| \, w \mid \exists s' \in S. \, s \xrightarrow{a, *_{L_a(s), w}} s' \, |\!\}$$

Furthermore, we extend the real number multiplication to immediate rates as follows:

$$\infty_{l,w} \cdot p = \infty_{l,w \cdot p}$$

■

The operational semantics for $\text{EMPA}_{\text{gr}}$ is the least MSTS $(\mathcal{G}, AType, \longrightarrow)$ satisfying the inference rules of Table 1, where in addition to the rules $(Ch1_l)$, $(Ch2_l)$, $(Pa1_l)$, $(Pa2_l)$, $(Sy1_l)$ referring to a move of the lefthand process $E_1$, we consider also the symmetrical rules $(Ch1_r)$, $(Ch2_r)$, $(Pa1_r)$, $(Pa2_r)$, $(Sy1_r)$ taking into account the moves of the righthand process $E_2$, obtained by exchanging the roles of terms $E_1$ and $E_2$.

Some explanations are now in order. First of all, the operational rule give rise to an interleaving semantics, which is made possible by the memoryless property of exponential distributions. Moreover, similarly to [16], we consider the operational rules as generating a multiset of transitions (consistently with the definition of a MSTS), where a transition has arity $m$ if and only if it can be derived in $m$ possible ways from the operational rules.

The removal of lower priority passive transitions of the same type is carried out in rules $(Ch2_l)$ and $(Ch2_r)$ for the alternative composition operator and rules $(Pa1_l)$ and $(Pa1_r)$ for the parallel composition operator by using $L_a(E)$. As we shall see in Thm. 8.4, discarding lower priority passive transitions does not compromise the achievement of the congruence property for the Markovian bisimulation equivalence. While higher priority active transitions can be prevented by a context which does not prevent lower priority active transitions (because of their different types), this cannot happen for passive transitions as their priorities are reactive, i.e. imposed only among passive transitions of the same type. We also note that the priorities are intepreted as being global according to the classification of [10], as their scope is not limited to sequential terms but includes terms composed in parallel.

In the case of a synchronization, the evaluation of the rate of the resulting action is carried out by rules $(Sy1_l)$, $(Sy1_r)$, and $(Sy2)$ as follows. Whenever an active action synchronizes with a passive action of the same type, the rate of the resulting active action is evaluated in rules $(Sy1_l)$ and $(Sy1_r)$ by multiplying the rate of the active action by the probability of choosing the passive action. Whenever two passive actions of type $a$ synchronize, instead, the priority level and the weight of the resulting passive action are computed as described by rule $(Sy2)$. In particular, the weight is computed by multiplying the probability $p$ of independently choosing the two original actions by the normalization factor $N$. $N$ is given by the overall weight of the passive transitions of type $a$ with maximum priority level executable by $E_1$ and $E_2$, computed by using $W_a(E)$.

$$(\textbf{\textit{Pr}}) \quad <a,\tilde{\lambda}>.E \xrightarrow{a,\tilde{\lambda}} E$$

$$(\textbf{\textit{Hi}1}) \quad \frac{E \xrightarrow{a,\tilde{\lambda}} E'}{E/L \xrightarrow{a,\tilde{\lambda}} E'/L} \quad a \notin L \qquad\qquad (\textbf{\textit{Hi}2}) \quad \frac{E \xrightarrow{a,\tilde{\lambda}} E'}{E/L \xrightarrow{\tau,\tilde{\lambda}} E'/L} \quad a \in L$$

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

$$(\textbf{\textit{Ch}1}_l) \quad \frac{E_1 \xrightarrow{a,\bar{\lambda}} E_1'}{E_1 + E_2 \xrightarrow{a,\bar{\lambda}} E_1'} \qquad\qquad (\textbf{\textit{Ch}2}_l) \quad \frac{E_1 \xrightarrow{a,*_{l,w}} E_1' \quad l \geq L_a(E_2)}{E_1 + E_2 \xrightarrow{a,*_{l,w}} E_1'}$$

$$(\textbf{\textit{Pa}1}_l) \quad \frac{E_1 \xrightarrow{a,\bar{\lambda}} E_1'}{E_1 \,\|_S\, E_2 \xrightarrow{a,\bar{\lambda}} E_1' \,\|_S\, E_2} \quad a \notin S$$

$$(\textbf{\textit{Pa}2}_l) \quad \frac{E_1 \xrightarrow{a,*_{l,w}} E_1' \quad l \geq L_a(E_2)}{E_1 \,\|_S\, E_2 \xrightarrow{a,*_{l,w}} E_1' \,\|_S\, E_2} \quad a \notin S$$

$$(\textbf{\textit{Sy}1}_l) \quad \frac{E_1 \xrightarrow{a,\bar{\lambda}} E_1' \quad E_2 \xrightarrow{a,*_{l,w}} E_2'}{E_1 \,\|_S\, E_2 \xrightarrow{a,\bar{\lambda}\cdot\frac{w}{W_a(E_2)}} E_1' \,\|_S\, E_2'} \quad a \in S$$

$$(\textbf{\textit{Sy}2}) \quad \frac{E_1 \xrightarrow{a,*_{l_1,w_1}} E_1' \quad E_2 \xrightarrow{a,*_{l_2,w_2}} E_2'}{E_1 \,\|_S\, E_2 \xrightarrow{a,*_{\max(l_1,l_2),p\cdot N}} E_1' \,\|_S\, E_2'} \quad a \in S$$

$$\text{where:} \quad p = \frac{w_1}{W_a(E_1)} \cdot \frac{w_2}{W_a(E_2)} \qquad N = \begin{cases} W_a(E_1) + W_a(E_2) & \text{if } l_1 = l_2 \\ W_a(E_1) & \text{if } l_1 > l_2 \\ W_a(E_2) & \text{if } l_2 > l_1 \end{cases}$$

$$(\textbf{\textit{Co}}) \quad \frac{E \xrightarrow{a,\tilde{\lambda}} E'}{A \xrightarrow{a,\tilde{\lambda}} E'} \quad A \triangleq E$$

Table 1
$\mathrm{EMPA_{gr}}$ operational semantics

**Definition 7.5** The *integrated interleaving semantics* of $E \in \mathcal{G}$ is the RMSTS

$$\mathcal{I}[\![E]\!] = (\mathcal{G}_E, AType, \longrightarrow_E, E)$$

where $\mathcal{G}_E$ is the set of terms reachable from $E$ according to MSTS $(\mathcal{G}, AType, \longrightarrow)$ and $\longrightarrow_E$ is the restriction of $\longrightarrow$ to transitions between terms in $\mathcal{G}_E$. We say that $E \in \mathcal{G}$ is *performance closed* if and only if $\mathcal{I}[\![E]\!]$ does not contain passive transitions. ■

We conclude by recalling that from $\mathcal{I}[\![E]\!]$ two projected semantic models can be obtained by essentially dropping action rates or action types, respectively. Before applying such a transformation to $\mathcal{I}[\![E]\!]$, lower priority active transitions are pruned because $E$ is no longer to be composed with other terms as it describes the whole system we are interested in. The *functional semantics* $\mathcal{F}[\![E]\!]$ is a standard LTS whose transitions are decorated with action types only. The *Markovian semantics* $\mathcal{M}[\![E]\!]$ is instead a CTMC or a DTMC, as seen in Sect. 7.2, which is well defined only if $E$ is performance closed.

### *7.4 The Cooperation Structure Model*

In this section we briefly present the model of *cooperation structures* formalized in [8], which allows us to represent the structure of master-slaves synchronizations in a system state. The importance of such a model is that it justifies the choice of the priority level and the weight normalization factor of the passive action resulting from the synchronization of two passive actions.

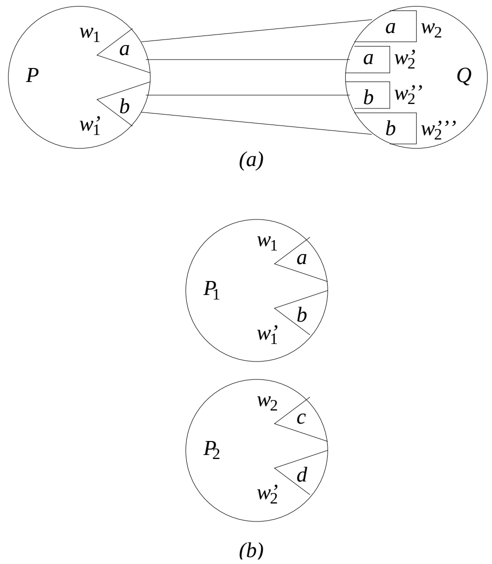



Fig. 4. Probabilistic cooperation scenarios

As already explained, the synchronization mechanism of $\mathrm{EMPA_{gr}}$ is based on a master-slaves discipline, which establishes that, in a system state, first a choice among the master actions is generatively performed, then a choice among the passive actions with which the selected master action can synchronize is reactively made. As an example, let us consider Fig. 4$(a)$, which depicts a state of system $P \,\|_{\{a,b\}}\, Q$. In the picture, each circle describes the current state of a sequential process. The alternative actions currently enabled by a process are represented as labeled triangles if masters and labeled boxes if slaves, with each label being the action type. Each process chooses an action according to its attached rate. [2] For instance, in process $P$ the intended meaning of weights $w_1$ and $w_1'$ is that the probability of choosing $a$ is proportional to $w_1$ while the probability of choosing $b$ is proportional to $w_1'$. Linked triangles/boxes denote cooperating actions. [3] According to the master-slaves approach, there are two steps. In the first step we probabilistically choose, locally to $P$, between the two master actions $a$ and $b$ according to weights $w_1$ and $w_1'$ (master choice). In the second step, we probabilistically choose, locally to $Q$, between the two slave actions $a$ according to weights $w_2$ and $w_2'$ or between the two slave actions $b$ according to weights $w_2''$ and $w_2'''$, depending on whether master action $a$ or $b$ wins in the first step (slave choice).

As far as the master choice is concerned, several processes may be involved in the choice of the master action. For example, consider Fig. 4($b$), which depicts a state of system $P_1 \,\|_{\emptyset}\, P_2$. Conceptually, the choice of the master action can be seen as being performed in two rounds (*interprocess choice* and *intraprocess choice*). In the first round, the interprocess choice is made according to the overall rate of each process. The overall rate of a process state is obtained by summing: the rates of the enabled exponentially timed actions if no immediate action is executable, the weights of the enabled immediate actions at the maximum priority level otherwise. Therefore, in Fig. 4($b$) the interprocess choice is made according to weights $w_1 + w_1'$ and $w_2 + w_2'$. In the second round, a local choice is performed in the winning process according to the rates of its master actions.

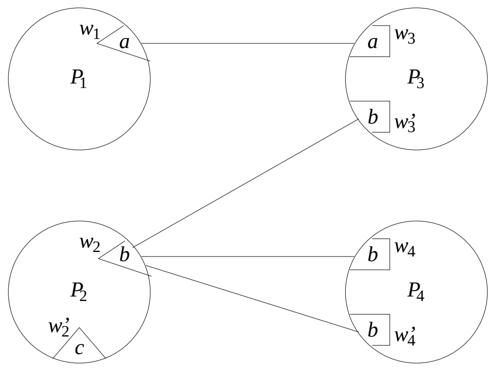



Fig. 5. A master-slaves cooperation scenario

In general, the selection of the action to be executed in a system state is

[2] In Fig. 4 and Fig. 5 we abbreviate rates simply through weights because we assume that all the occurring master actions are immediate with the same priority level and that all the occurring slave actions are passive with the same priority level.

[3] This dynamic system representation is not to be confused with the static system representation via flow graphs of [20].

conceptually carried out in two steps (master choice and slave choice) each composed of two rounds (interprocess choice and intraprocess choice). As an example, look at the scenario depicted in Fig. 5 which represents a state of system $(P_1 \parallel_\emptyset P_2) \parallel_{\{a,b\}} (P_3 \parallel_\emptyset P_4)$. In the first step a master action is chosen in two rounds. In the first round an interprocess choice takes place among the processes enabling master actions, hence we probabilistically choose between $P_1$ and $P_2$ according to weights $w_1$ and $w_2 + w_2'$. In the second round a local choice among the master actions of the process that wins the first round occurs. If $P_1$ wins, a second round is not necessary because $P_1$ can do one master action only. If $P_2$ wins, instead, a local choice between the two actions $b$ (weight $w_2$) and $c$ (weight $w_2'$) occurs in the second round. If the winning master action must not cooperate (action $c$ of $P_2$) or can be engaged in a single cooperation (action $a$ of $P_1$), then we are done. If this is not the case (action $b$ of $P_2$), a second step must be undertaken, where a slave action is chosen in two rounds. In the first round an interprocess choice takes place among the processes enabling slave actions which can cooperate with the winning master action, hence we probabilistically choose between $P_3$ and $P_4$ according to weights $w_3'$ and $w_4 + w_4'$. In the second round a local choice among the slave actions of the process that wins the first round occurs. If $P_3$ wins a second round is not necessary. If $P_4$ wins, instead, we locally choose between the two slave actions $b$ of $P_4$, according to weights $w_4$ and $w_4'$. To be more precise, since a master action may be synchronized with several slave actions performed by different processes, once the master action has been chosen (through an interprocess choice followed by an intraprocess choice), we have an interprocess choice among groups of processes cooperating to perform the slave actions, followed by an intraprocess choice which is conducted independently within each process of the selected group.

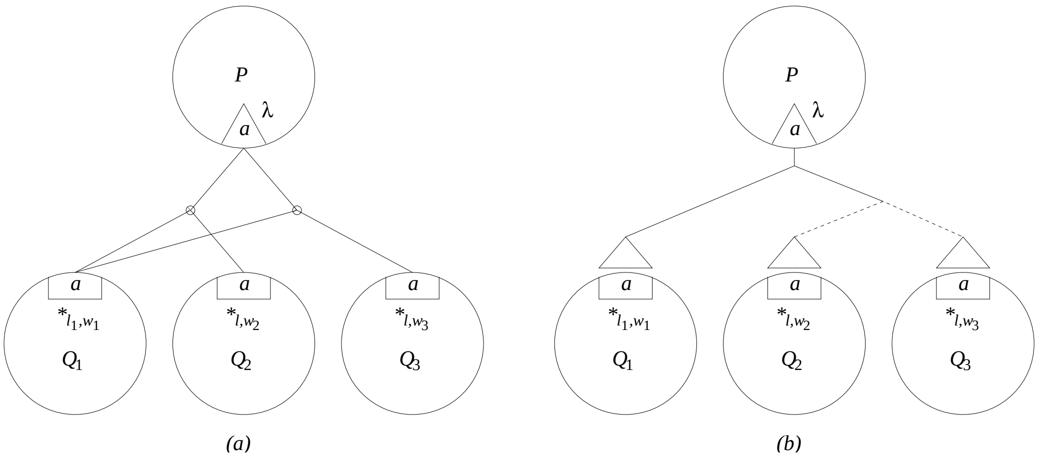



Fig. 6. A cooperation structure

In order to introduce cooperation structures, let us consider the scenario of Fig. 6($a$) where we have two alternative actions of $P \parallel_{\{a\}} (Q_1 \parallel_{\{a\}} (Q_2 \parallel_\emptyset Q_3))$ each formed by a three-way cooperation including the winning master action and two cooperating slave actions. Here we have to choose which is the pair of processes which cooperate with the master action $a$. In the scenario of Fig. 6($a$) we simply have to choose which process between $Q_2$ and $Q_3$ cooperates with process $Q_1$. We may describe this scenario as in Fig. 6($b$), where we have represented the two possible system actions that can originate from the master action of $P$ as a single *cooperation tree* indicating that process $Q_1$ may cooperate with either process $Q_2$ or process $Q_3$.

In general a cooperation tree represents a choice among a set of system actions that originate from a given master action. A cooperation tree is a binary tree with two kinds of nodes:

- The nodes with outgoing dashed lines, which are called *choice nodes*, represent the fact that either actions of their lefthand subtree or actions of their righthand subtree can take part in the final system action.
- The nodes with outgoing solid lines, which are called *cooperation nodes*, represent the fact that actions which are possible according to their lefthand subtree must cooperate with actions which are possible according to their righthand subtree to take part in the final system action.

The leaves of the tree are represented by triangles. Each leaf is adjacent to a different process, where the base of the triangle singles out a subset of the slave actions of such a process (the slave actions just underneath the triangle base). The intended meaning of the leaf is that one and only one slave action in this set can take part in the final system action.

The root of a cooperation tree is connected with a solid line to the master action it refers to. The pair composed of a master action and the connected cooperation tree is called a cooperation structure.

As shown in [8], in the case of system states representable by $\mathrm{EMPA_{gr}}$ terms, the set of system actions that originate from the winning master action can always be represented by a cooperation tree. This is due to the fact that in $\mathrm{EMPA_{gr}}$ the synchronization is based on action types, so, if a slave synchronization between some processes is required for the action type of the chosen master action, then all the slave actions with that action type of the processes can engage in the cooperation. As a consequence of the fact that system actions originating from a master action can always be represented by a cooperation tree, it is possible to make independent local choices in each process of the group chosen according to the cooperation tree. Therefore, it is not necessary to break again the symmetry of cooperation at the level of the slave actions, e.g. by designating some actions as submaster actions.

A cooperation structure represents a prioritized-probabilistic choice among the groups of processes that can cooperate with the selected master action. To this purpose, each choice node is intepreted as representing a prioritized-probabilistic choice, where we consider, for each of its two alternative subtrees, as priority level the maximum priority level of the processes adjacent to its leaves, and as weight the overall weight at that level of the processes adjacent to its leaves.

In [8] we have proved the correctness of the operational semantics for $\mathrm{EMPA_{gr}}$ – in particular of the choice of the priority level and the weight normalization factor of the passive action resulting from the synchronization of two passive actions – by showing that the calculations of the rates of the synchronizations are consistent with those resulting from cooperation structures.

## 8 Markovian Bisimulation Equivalence

In this section we equip $\mathrm{EMPA_{gr}}$ with a Markovian bisimulation equivalence, which relates systems having the same functional, probabilistic, prioritized and exponentially timed behavior. We then show that such an equivalence is a congruence, thus providing support for compositional manipulation, and we give a sound and complete axiomatization for nonrecursive process terms.

Our Markovian bisimulation equivalence is inspired by the *probabilistic bisimulation equivalence* of [19], according to which two equivalent terms have the same aggregated probability to reach the same equivalence class of terms by executing actions of the same type and priority level.

In the case of exponentially timed actions, we have to take into account not only the transition probabilities but also the state sojourn times. Because of the adoption of the race policy (see Sect. 2), this can be easily accomplished by considering the aggregated rate with which an equivalence class is reached by a term by executing actions of the same type. As an example, it must hold that

$$<a, \lambda_1>.E + <a, \lambda_2>.E \sim_{\mathrm{MB}} <a, \lambda_1 + \lambda_2>.E$$

This treatment of rates, originally proposed in [16], is basically the same as that of the exact aggregation for CTMCs known as *ordinary lumping* [24], thus establishing a clear connection between the Markovian bisimulation equivalence and the ordinary lumping. In the case of immediate and passive actions, instead, the probabilistic bisimulation equivalence must be rephrased in terms of weights. As an example, it must hold that

$$<a, \infty_{l,w_1}>.E + <a, \infty_{l,w_2}>.E \sim_{\mathrm{MB}} <a, \infty_{l,w_1+w_2}>.E$$

$$<a, *_{l,w_1}>.E + <a, *_{l,w_2}>.E \sim_{\mathrm{MB}} <a, *_{l,w_1+w_2}>.E$$

This treatment of weights was originally proposed in [25].

We are now in a position of defining our Markovian bisimulation equivalence.

**Definition 8.1** We define function priority level $PL : ARate \longrightarrow \mathbb{Z}$ by:

$$PL(*_{l,w}) = -l$$

$$PL(\lambda) = 0$$

$$PL(\infty_{l,w}) = l$$

$$
\begin{array}{lrcll}
(\mathcal{A}_1) & (E_1 + E_2) + E_3 & = & E_1 + (E_2 + E_3) & \\
(\mathcal{A}_2) & E_1 + E_2 & = & E_2 + E_1 & \\
(\mathcal{A}_3) & E + \underline{0} & = & E & \\
(\mathcal{A}_4) & <a, \tilde{\lambda}_1>.E + <a, \tilde{\lambda}_2>.E & = & <a, \tilde{\lambda}_1 + \tilde{\lambda}_2>.E & \text{if } PL(\tilde{\lambda}_1) = PL(\tilde{\lambda}_2) \\
(\mathcal{A}_5) & <a, *_{l_1,w_1}>.E_1 + <a, *_{l_2,w_2}>.E_2 & = & <a, *_{l_1,w_1}>.E_1 & \text{if } l_1 > l_2 \\
 & & & & \\
(\mathcal{A}_6) & \underline{0}/L & = & \underline{0} & \\
(\mathcal{A}_7) & (<a, \tilde{\lambda}>.E)/L & = & <a, \tilde{\lambda}>.(E/L) & \text{if } a \notin L \\
(\mathcal{A}_8) & (<a, \tilde{\lambda}>.E)/L & = & <\tau, \tilde{\lambda}>.(E/L) & \text{if } a \in L \\
(\mathcal{A}_9) & (E_1 + E_2)/L & = & E_1/L + E_2/L & \\
 & & & & \\
(\mathcal{A}_{10}) & \underline{0}[\varphi] & = & \underline{0} & \\
(\mathcal{A}_{11}) & (<a, \tilde{\lambda}>.E)[\varphi] & = & <\varphi(a), \tilde{\lambda}>.(E[\varphi]) & \\
(\mathcal{A}_{12}) & (E_1 + E_2)[\varphi] & = & E_1[\varphi] + E_2[\varphi] &
\end{array}
$$

$$
\begin{aligned}
(\mathcal{A}_{13})\quad & \sum_{i \in I_0} <a_i, \tilde{\lambda}_i>.E_i \parallel_S \sum_{i \in I_1} <a_i, \tilde{\lambda}_i>.E_i \; = \\
& \quad \sum_{j \in I_0, a_j \notin S} <a_j, \tilde{\lambda}_j>.(E_j \parallel_S \sum_{i \in I_1} <a_i, \tilde{\lambda}_i>.E_i) \; + \\
& \quad \sum_{j \in I_1, a_j \notin S} <a_j, \tilde{\lambda}_j>.(\sum_{i \in I_0} <a_i, \tilde{\lambda}_i>.E_i \parallel_S E_j) \; + \\
& \quad \sum_{k \in K_0} \sum_{h \in P_{1,a_k}} <a_k, \tilde{\lambda}_k \cdot (w_h / W_{1,a_k})>.(E_k \parallel_S E_h) \; + \\
& \quad \sum_{k \in K_1} \sum_{h \in P_{0,a_k}} <a_k, \tilde{\lambda}_k \cdot (w_h / W_{0,a_k})>.(E_h \parallel_S E_k) \; + \\
& \quad \sum_{k \in P'_0} \sum_{h \in P_{1,a_k}} <a_k, *_{\max(l_k, l_h), (w_k / W_{0,a_k}) \cdot (w_h / W_{0,a_k}) \cdot N_{a_k}}>.(E_k \parallel_S E_h)
\end{aligned}
$$

where $I_0 \cap I_1 = \emptyset, \; \tilde{\lambda}_i = *_{l_i,w_i}$ for $i \in I_0 \cup I_1 .\, PL(\tilde{\lambda}_i) < 0,$ and for $j \in \{0, 1\}$

$$
\begin{aligned}
L_{j,a} &= \max\{l_k \mid k \in I_j \wedge a_k = a \wedge \tilde{\lambda}_k = *_{l_k,w_k}\} \\
P_{j,a} &= \{k \in I_j \mid a_k = a \wedge \tilde{\lambda}_k = *_{l_k,w_k} \wedge l_k = L_{j,a}\} \\
K_j &= \{k \in I_j \mid a_k \in S \wedge PL(\tilde{\lambda}_k) \geq 0 \wedge P_{1-j,a_k} \neq \emptyset\} \\
P'_0 &= \{k \in I_0 \mid \exists a \in S.\, k \in P_{0,a} \wedge P_{1,a} \neq \emptyset\} \\
W_{j,a} &= \sum \{\!| \, w_k \mid k \in P_{j,a} \wedge \tilde{\lambda}_k = *_{l_k,w_k} \, |\!\} \\
N_a &= \begin{cases} W_{0,a} + W_{1,a} & \text{if } L_{0,a} = L_{1,a} \\ W_{0,a} & \text{if } L_{0,a} > L_{1,a} \\ W_{1,a} & \text{if } L_{1,a} > L_{0,a} \end{cases}
\end{aligned}
$$

Table 2
Axiomatization of $\sim_{\mathrm{MB}}$

and we extend the real number summation to rates of the same priority level as follows:

$$*_{l,w_1} + *_{l,w_2} \;= *_{l,w_1+w_2}$$
$$\infty_{l,w_1} + \infty_{l,w_2} = \infty_{l,w_1+w_2}$$

We then define partial function aggregated rate $Rate : \mathcal{G} \times AType \times \mathbb{Z} \times \mathcal{P}(\mathcal{G}) \rightarrow\!\!\!\!\!\circ\;\, ARate$ by:

$$Rate(E, a, l, C) = \sum\{\!|\, \tilde{\lambda} \mid \exists E' \in C.\, E \xrightarrow{a,\tilde{\lambda}} E' \wedge PL(\tilde{\lambda}) = l \,|\!\}$$

with $Rate(E, a, l, C) = \perp$ whenever the multiset above is empty. ■

**Definition 8.2** An equivalence relation $\mathcal{B} \subseteq \mathcal{G} \times \mathcal{G}$ is a *Markovian bisimulation* if and only if, whenever $(E_1, E_2) \in \mathcal{B}$, then for all $a \in AType$, $l \in \mathbb{Z}$, and equivalence classes $C \in \mathcal{G}/\mathcal{B}$

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

**Definition 8.3** We call $\sim_{\mathrm{MB}}$, defined as the union of all the Markovian bisimulations over $\mathcal{G}$, the *Markovian bisimulation equivalence.* ■

**Theorem 8.4** *Let $E_1, E_2 \in \mathcal{G}$. If $E_1 \sim_{\mathrm{MB}} E_2$ then:*

(i) *For all $<a, \tilde{\lambda}> \in Act$, $<a, \tilde{\lambda}>.E_1 \sim_{\mathrm{MB}} <a, \tilde{\lambda}>.E_2$.*

(ii) *For all $L \subseteq AType - \{\tau\}$, $E_1/L \sim_{\mathrm{MB}} E_2/L$.*

(iii) *For all $\varphi \in ATRFun$, $E_1[\varphi] \sim_{\mathrm{MB}} E_2[\varphi]$.*

(iv) *For all $F \in \mathcal{G}$, $E_1 + F \sim_{\mathrm{MB}} E_2 + F$ and $F + E_1 \sim_{\mathrm{MB}} F + E_2$.*

(v) *For all $F \in \mathcal{G}$ and $S \subseteq AType - \{\tau\}$, $E_1 \parallel_S F \sim_{\mathrm{MB}} E_2 \parallel_S F$ and $F \parallel_S E_1 \sim_{\mathrm{MB}} F \parallel_S E_2$.*

*Additionally, $\sim_{\mathrm{MB}}$ is closed w.r.t. recursive constant definitions.*

**Proof.** The proof is similar to that of the corresponding theorem of [3] Chap. 5, with some changes in the case of the alternative and parallel composition operators that we now show.

- Let $\mathcal{B} \subseteq \mathcal{G} \times \mathcal{G}$ be a Markovian bisimulation such that $(E_1, E_2) \in \mathcal{B}$. Given $F \in \mathcal{G}$, we prove that $\mathcal{B}' = (\mathcal{B} \cup \{(E_1 + F, E_2 + F), (E_2 + F, E_1 + F)\})^+$ is a Markovian bisimulation. Observed that $\mathcal{B}'$ is an equivalence relation, we have two cases:
  - If $(E_1 + F, E_2 + F) \in \mathcal{B}$, then $\mathcal{B}' = \mathcal{B}$ and the result trivially follows.
  - Assume that $(E_1 + F, E_2 + F) \notin \mathcal{B}$. Observed that
  $$\mathcal{G}/\mathcal{B}' = (\mathcal{G}/\mathcal{B} - \{[E_1 + F]_{\mathcal{B}}, [E_2 + F]_{\mathcal{B}}\}) \cup \{[E_1 + F]_{\mathcal{B}} \cup [E_2 + F]_{\mathcal{B}}\}$$
  let $(F_1, F_2) \in \mathcal{B}'$, $a \in AType$, $l \in \mathbb{Z}$, and $C \in \mathcal{G}/\mathcal{B}'$.
  If $(F_1, F_2) \in \mathcal{B}$ and $C \in \mathcal{G}/\mathcal{B} - \{[E_1 + F]_{\mathcal{B}}, [E_2 + F]_{\mathcal{B}}\}$, then trivially $Rate(F_1, a, l, C) = Rate(F_2, a, l, C)$.
  If $(F_1, F_2) \in \mathcal{B}$ and $C = [E_1 + F]_{\mathcal{B}} \cup [E_2 + F]_{\mathcal{B}}$, then for $j \in \{1, 2\}$ we have $Rate(F_j, a, l, C) = Rate(F_j, a, l, [E_1 + F]_{\mathcal{B}}) + Rate(F_j, a, l, [E_2 + F]_{\mathcal{B}})$ so $Rate(F_1, a, l, C) = Rate(F_2, a, l, C)$.

If $(F_1, F_2) \in \mathcal{B}' - \mathcal{B}$, i.e. $F_1 \in [E_1 + F]_{\mathcal{B}}$ and $F_2 \in [E_2 + F]_{\mathcal{B}}$, then for $j \in \{1, 2\}$ we have

$$Rate(F_j, a, l, C) = \begin{cases} Rate(E_j, a, l, C) + Rate(F, a, l, C) \\ Rate(E_j, a, l, C) \\ Rate(F, a, l, C) \\ \perp \end{cases}$$

depending on whether $Rate(E_j, a, l, C) \neq \perp \wedge Rate(F, a, l, C) \neq \perp$ or $Rate(E_j, a, l, C) \neq \perp \wedge ((l \geq 0 \wedge Rate(F, a, l, C) = \perp) \vee (l < 0 \wedge \forall l' \in \mathbb{Z}_-. \, l' \leq l \Longrightarrow Rate(F, a, l', \mathcal{G}) = \perp))$ or $Rate(F, a, l, C) \neq \perp \wedge ((l \geq 0 \wedge Rate(E_j, a, l, C) = \perp) \vee (l < 0 \wedge \forall l' \in \mathbb{Z}_-. \, l' \leq l \Longrightarrow Rate(E_j, a, l', \mathcal{G}) = \perp))$ or none of the previous clauses holds.

If $C \in \mathcal{G}/\mathcal{B} - \{[E_1 + F]_{\mathcal{B}}, [E_2 + F]_{\mathcal{B}}\}$, then from $(E_1, E_2) \in \mathcal{B}$ we derive $Rate(E_1, a, l, C) = Rate(E_2, a, l, C)$ so $Rate(F_1, a, l, C) = Rate(F_2, a, l, C)$.

If $C = [E_1 + F]_{\mathcal{B}} \cup [E_2 + F]_{\mathcal{B}}$, then for $j \in \{1, 2\}$ we have $Rate(E_j, a, l, C) = Rate(E_j, a, l, [E_1 + F]_{\mathcal{B}}) + Rate(E_j, a, l, [E_2 + F]_{\mathcal{B}})$. Since $(E_1, E_2) \in \mathcal{B}$, it turns out that $Rate(E_1, a, l, C) = Rate(E_2, a, l, C)$ so $Rate(F_1, a, l, C) = Rate(F_2, a, l, C)$.

- Given $F \in \mathcal{G}$ and $S \subseteq AType - \{\tau\}$, we prove that $\mathcal{B}' = \mathcal{B} \cup Id_{\mathcal{G}}$, where $\mathcal{B} = \{(E_1 \|_S F, E_2 \|_S F) \mid E_1 \sim_{\mathrm{MB}} E_2\}$ and $Id_{\mathcal{G}}$ is the identity relation over $\mathcal{G}$, is a Markovian bisimulation. Observed that $\mathcal{B}'$ is an equivalence relation and that either each of the terms of an equivalence class has "$\_ \|_S F$" as outermost operator or none of them has, let $(F_1, F_2) \in \mathcal{B}'$, $a \in AType$, $l \in \mathbb{Z}$, and $C \in \mathcal{G}/\mathcal{B}'$.
  - If $(F_1, F_2) \in Id_{\mathcal{G}}$, then trivially $Rate(F_1, a, l, C) = Rate(F_2, a, l, C)$.
  - If $(F_1, F_2) \in \mathcal{B}$, then $F_1 \equiv E_1 \|_S F$ and $F_2 \equiv E_2 \|_S F$ where $E_1 \sim_{\mathrm{MB}} E_2$.

    If none of the terms in $C$ has "$\_ \|_S F$" as outermost operator, then trivially $Rate(F_1, a, l, C) = \perp = Rate(F_2, a, l, C)$.

    If each of the terms in $C$ has "$\_ \|_S F$" as outermost operator, given $E \|_S G \in C$ it turns out that $C = \{E' \|_S G \mid E' \in [E]_{\sim_{\mathrm{MB}}}\}$.

    If $a \notin S$, then for $j \in \{1, 2\}$ we have that

$$Rate(F_j, a, l, C) = \begin{cases} Rate(E_j, a, l, [E]_{\sim_{\mathrm{MB}}}) + Rate(F, a, l, \{G\}) \\ Rate(E_j, a, l, [E]_{\sim_{\mathrm{MB}}}) \\ Rate(F, a, l, \{G\}) \\ \perp \end{cases}$$

depending on whether $E_j \in [E]_{\sim_{\mathrm{MB}}} \wedge F \equiv G \wedge Rate(E_j, a, l, [E]_{\sim_{\mathrm{MB}}}) \neq \perp \wedge Rate(F, a, l, \{G\}) \neq \perp$ or $E_j \notin [E]_{\sim_{\mathrm{MB}}} \wedge F \equiv G \wedge Rate(E_j, a, l, [E]_{\sim_{\mathrm{MB}}}) \neq \perp \wedge (l \geq 0 \vee (l < 0 \wedge \forall l' \in \mathbb{Z}_-. \, l' < l \Longrightarrow Rate(F, a, l', \mathcal{G}) = \perp))$ or $E_j \in [E]_{\sim_{\mathrm{MB}}} \wedge F \not\equiv G \wedge Rate(F, a, l, \tilde{b}, \{G\}) \neq \perp \wedge (l \geq 0 \vee (l < 0 \wedge \forall l' \in \mathbb{Z}_-. \, l' < l \Longrightarrow Rate(E_j, a, l', [E]_{\sim_{\mathrm{MB}}}) = \perp$ or none of the previous clauses holds. Since $E_1 \sim_{\mathrm{MB}} E_2$, it follows that $Rate(F_1, a, l, C) = Rate(F_2, a, l, C)$.

If $a \in S$ and we pose

$$\begin{aligned}
*_{-l,w_{E_j,l}} &= Rate(E_j, a, l, [E]_{\sim_{\mathrm{MB}}}) \\
*_{-l,w_{E_j,l,\mathrm{tot}}} &= Rate(E_j, a, l, \mathcal{G}) \\
*_{-l,w_{F,l}} &= Rate(F, a, l, \{G\}) \\
*_{-l,w_{F,l,\mathrm{tot}}} &= Rate(F, a, l, \mathcal{G})
\end{aligned}$$

for $l \in \mathbb{Z}_-$ and

$$N = \begin{cases} w_{E_j,l_1,\mathrm{tot}} + w_{F,l_2,\mathrm{tot}} & \text{if } l_1 = l_2 \\ w_{E_j,l_1,\mathrm{tot}} & \text{if } l_1 > l_2 \\ w_{F,l_2,\mathrm{tot}} & \text{if } l_2 > l_1 \end{cases}$$

then for $j \in \{1, 2\}$ we have that

$$Rate(F_j, a, l, C) = \begin{cases} Rate(E_j, a, l, [E]_{\sim_{\mathrm{MB}}}) \cdot w_{F,l'}/w_{F,l',\mathrm{tot}} + \\ \quad Rate(F, a, l, \{G\}) \cdot w_{E_j,l''}/w_{E_j,l'',\mathrm{tot}} \\ Rate(E_j, a, l, [E]_{\sim_{\mathrm{MB}}}) \cdot w_{F,l'}/w_{F,l',\mathrm{tot}} \\ Rate(F, a, l, \{G\}) \cdot w_{E_j,l''}/w_{E_j,l'',\mathrm{tot}} \\ *_{-l,w_{E_j,l_1}/w_{E_j,l_1,\mathrm{tot}} \cdot w_{F,l_2}/w_{F,l_2,\mathrm{tot}} \cdot N} \\ \perp \end{cases}$$

depending on whether $l \geq 0 \wedge Rate(E_j, a, l, [E]_{\sim_{\mathrm{MB}}}) \neq \perp \wedge \exists l' \in \mathbb{Z}_-. \, Rate(F, a, l', \{G\}) \neq \perp \wedge Rate(F, a, l, \{G\}) \neq \perp \wedge \exists l'' \in \mathbb{Z}_-. \, Rate(E_j, a, l'', [E]_{\sim_{\mathrm{MB}}}) \neq \perp$ or $l \geq 0 \wedge Rate(E_j, a, l, [E]_{\sim_{\mathrm{MB}}}) \neq \perp \wedge \exists l' \in \mathbb{Z}_-. \, Rate(F, a, l', \{G\}) \neq \perp \wedge (Rate(F, a, l, \{G\}) = \perp \vee \forall l''' \in \mathbb{Z}_-. \, Rate(E_j, a, l''', [E]_{\sim_{\mathrm{MB}}}) = \perp)$ or $l \geq 0 \wedge Rate(F, a, l, \{G\}) \neq \perp \wedge \exists l'' \in \mathbb{Z}_-. \, Rate(E_j, a, l'', [E]_{\sim_{\mathrm{MB}}}) \neq \perp \wedge (Rate(E_j, a, l, [E]_{\sim_{\mathrm{MB}}}) = \perp \vee \forall l''' \in \mathbb{Z}_-. \, Rate(F, a, l''', \{G\}) = \perp)$ or $l < 0 \wedge \exists l_1, l_2 \in \mathbb{Z}_-. \, Rate(E_j, a, l_1, [E]_{\sim_{\mathrm{MB}}}) \neq \perp \wedge Rate(F, a, l_2, \{G\}) \neq \perp \wedge -l = \max(-l_1, -l_2)$ or none of the previous clauses holds. From $E_1 \sim_{\mathrm{MB}} E_2$, it follows that $Rate(F_1, a, l, C) = Rate(F_2, a, l, C)$.

□

We observe that the congruence result above holds because the operational semantics is defined in such a way that lower priority active transitions are not pruned. If this were not the case, we would have e.g. $<a_1, \lambda>.\underline{0} + <a_2, \infty_{l,w}>.\underline{0} \sim_{\mathrm{MB}} <a_2, \infty_{l,w}>.\underline{0}$ as both terms would have only one transition labeled with $<a_2, \infty_{l,w}>$, but $(<a_1, \lambda>.\underline{0} + <a_2, \infty_{l,w}>.\underline{0}) \,\|_{\{a_2\}}\, \underline{0} \not\sim_{\mathrm{MB}} <a_2, \infty_{l,w}>.\underline{0} \,\|_{\{a_2\}}\, \underline{0}$ because the first term has a transition labeled with action $<a_1, \lambda>$ while the second term has no transitions at all. On the contrary, the removal of lower priority passive actions of a given type does not cause any problem.

**Theorem 8.5** *Let $\mathcal{A}$ be the set of axioms in Table 2. The deductive system $Ded(\mathcal{A})$ is sound and complete for $\sim_{\mathrm{MB}}$ over the set of nonrecursive terms of*

$\mathcal{G}$.

**Proof.** The proof is similar to that of the corresponding theorem of [3] Chap. 5, with the difference that a nonrecursive term $E \in \mathcal{G}$ is defined to be in sum normal form (snf) if and only if $E$ is $\underline{0}$ or $\sum_{i \in I} <a_i, \tilde{\lambda}_i>.E_i$ with every $E_i$ in snf, where the nonempty finite set $I$ is such that there are no $i, i' \in I$ for which $a_i = a_{i'} \wedge \tilde{\lambda}_i = *_{l_i, w_i} \wedge \tilde{\lambda}_{i'} = *_{l_{i'}, w_{i'}} \wedge l_i \neq l_{i'}$. □

We conclude by observing that axiom $(\mathcal{A}_4)$ is exactly the rule we wanted our equivalence to satisfy, while axiom $(\mathcal{A}_5)$ establishes that lower priority passive actions of a given type can be left out.

## 9 Conclusion

The experience with process algebras has shown the necessity of mechanisms like priority, probabilistic internal/external choice, and time to model the behavior of real systems, as well as the necessity of compositionality for efficient system analysis. In this paper we have developed a new process algebra called $\mathrm{EMPA_{gr}}$ that has a considerable expressive power, because it includes all the above mentioned mechanisms, and achieves semantic compositionality thanks to a suitable asymmetric synchronization mechanism, because Markovian bisimulation equivalence turns out to be a congruence for $\mathrm{EMPA_{gr}}$.

### *9.1 Related Work*

The starting point for our work has been the process algebra EMPA [3]. However, it is worth observing that this paper does not only provide an improvement over EMPA, for which Markovian bisimulation equivalence is not a congruence in general. In fact, the main contribution of this paper is the proposal of a natural and intuitive solution to an important open problem in the literature, which is independent of the considered formalism: how to obtain semantic compositionality w.r.t. an asynchronous [4] CSP like parallel composition operator in the presence of priority, probabilistic internal/external choice, and time. Our solution essentially consists of an extension to priorities and time of the generative-reactive approach of [6]. Such an approach realizes an integration of an asymmetric form of cooperation inspired by probabilistic I/O automata [26] with an approach to interprocess selection inspired by probabilistic ACP [2]. In [26] output actions behave generatively while input actions behave reactively according to the terminology of [11]. In [2] a probabilistic mechanism is provided for choosing (in the context of an asynchronous parallel composition operator) which of two processes in parallel must make the next move in a system state, by associating a probabilistic advancing speed to each process.

[4] In the sense that the computation does not proceed in locksteps; e.g., in $E_1 \,\|_\emptyset\, E_2$ we have to choose whether the next move is made by $E_1$ or $E_2$.

More precisely, the aforementioned contribution is twofold:

- We have extended the generative-reactive approach of [6] to deal with priorities. This has been accomplished by endowing master actions with generative priorities in addition to generative probabilities and slave actions with reactive priorities in addition to reactive weights.
- We have extended the generative-reactive approach of [6] to a timed setting, where a master action may have either an exponentially distributed duration or a zero duration (in which case it is given a priority and a weight), while slave actions have unspecified duration (and are given reactive priorities and weights). This extension encompasses a synchronization mechanism similar to that of PEPA [16], where master-slaves cooperations are realized through cooperations between exponentially timed actions and weighted passive actions.

If we compare $\mathrm{EMPA_{gr}}$ with the other Markovian process algebras appeared in the literature, we observe that none of them reaches the expressive power of $\mathrm{EMPA_{gr}}$. In particular, the expressiveness of M-TIPP [14] and MPA [9] is limited because they consider exponentially timed actions only. Passive actions with reactive weights are instead introduced in PEPA [16], which however does not include immediate actions. This reduces the capability of modeling real systems and does not allow discrete time systems to be described. On the contrary, in IM-TIPP [15] and IMC [13] immediate actions can be expressed but neither generative nor reactive probabilistic choices can be represented. It is however worth noting that in the approach of [15,13] the fact that immediate actions do not bear probabilistic information makes it ease and natural the definition of a weak Markovian bisimulation congruence which abstracts from internal immediate actions. Such an approach is further extended in PM-TIPP [21] with reactive (but not generative) probabilistic choices. This capability, however, is not exploited like in PEPA to enforce a master-slaves synchronization policy. We conclude by observing that, unlike $\mathrm{EMPA_{gr}}$, none of the previously considered Markovian process algebras handles (generative or reactive) priorities.

### *9.2 Future Work*

The research activity we have been conducting in the field of process algebras aims at developing a methodology for the specification and analysis of computer, communication and software systems that achieves a reasonable balance among formality, expressivity, usability and efficiency. In the future, we intend to make other steps towards the achievement of our goal.

**Expressivity**

$\mathrm{EMPA_{gr}}$ has already a considerable expressive power, as it includes priorities, probabilistic internal/external choices, and exponentially distributed and zero durations. However, such an expressive power can be further enhanced by considering generally distributed durations. Here the problem is that the semantics can no longer be defined in the interleaving style. In fact, since the memoryless property is lost, an action can no longer be considered as started in the same state in which it terminates its execution. In other words, one has to keep track of both the state in which the execution of an action starts and the state in which the execution of the same action finishes. In [7] we have shown that the right semantic approach in this scenario is that of the ST semantics [12] and we have defined and studied a process algebra with general distributions called GSMPA whose semantics is defined according to the ST approach.

**Usability**

In order for a process algebra to be helpful in practice, its usability should be considered to ease the task of the designer. In this respect, it may be convenient to take the view of software architecture [23], which has emerged in the last decade as a discipline within software engineering to cope with the increasing size and complexity of software systems during the early stage of their development. To achieve this, the focus is turned from algorithmic and data structure related issues to the overall architecture of the system, meant to be a collection of computational components together with a description of their connectors, i.e. the interactions between these components. From the process algebra perspective, it is desirable to force the designer to model systems in a way that elucidates the basic architectural concepts of component and connector. In [4] we have followed this view by defining an $\text{EMPA}_{\text{gr}}$ based architectural description language called ÆMPA, which allows the modular and hierarchical representation of the functional and performance behavior of families of systems through the specification of their sequential components and connectors and the attachments between them. For the sake of usability, ÆMPA is not only based on a textual notation, but also provides the designer with a graphical notation inspired by flow graphs [20]. Finally, ÆMPA enforces a controlled way of modeling by means of compatibility, interoperability and conformity architectural checks that essentially aim at establishing whether the specified components and connectors fit well together or give rise to architectural mismatches.

**Efficiency**

The efficiency of the analysis of $\text{EMPA}_{\text{gr}}$ specifications can be improved in several ways. First of all, we would like to develop in our expressive framework a Markovian congruence that abstracts from internal immediate actions, i.e. those actions which are unobservable and take no time. Unlike $\sim_{\text{MB}}$, such a weak Markovian congruence could be used to compositionally minimize in an effective way the state space of an $\text{EMPA}_{\text{gr}}$ specification. We believe that a promising starting point for this research is the Markovian testing equivalence introduced in [5]. Secondly, we would like to exploit the compositional nature of process algebras for efficiently calculating probability distributions. In particular, we are thinking of importing in the $\mathrm{EMPA_{gr}}$ framework tensor based methods as well as distributed simulation techniques.

## References


[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, *"Modelling with Generalized Stochastic Petri Nets"*, John Wiley & Sons, 1995

[2] J.C.M. Baeten, J.A. Bergstra, S.A. Smolka, *"Axiomatizing Probabilistic Processes: ACP with Generative Probabilities"*, in Information and Computation 121:234-255, 1995

[3] M. Bernardo, *"Theory and Application of Extended Markovian Process Algebra"*, Ph.D. Thesis, University of Bologna (Italy), 1999

[4] M. Bernardo, P. Ciancarini, L. Donatiello, *"ÆMPA: A Process Algebraic Description Language for the Performance Analysis of Software Architectures"*, in Proc. of the *2nd Int. Workshop on Software and Performance (WOSP '00)*, ACM Press, pp. 1-11, Ottawa (Canada), 2000

[5] M. Bernardo, W.R. Cleaveland, *"A Theory of Testing for Markovian Processes"*, in Proc. of the *11th Int. Conf. on Concurrency Theory (CONCUR '00)*, LNCS 1877:305-319, State College (PA), 2000

[6] M. Bravetti, A. Aldini, *"An Asynchronous Calculus for Generative-Reactive Probabilistic Systems"*, Tech. Rep. UBLCS-2000-03, University of Bologna (Italy), 2000 (extended abstract in Proc. of the *8th Int. Workshop on Process Algebra and Performance Modelling (PAPM '00)*, Carleton Scientific, pp. 591-605, Geneva (Switzerland), 2000)

[7] M. Bravetti, M. Bernardo, R. Gorrieri, *"Towards Performance Evaluation with General Distributions in Process Algebras"*, in Proc. of the *9th Int. Conf. on Concurrency Theory (CONCUR '98)*, LNCS 1466:405-422, Nice (France), 1998

[8] M Bravetti, M. Bernardo, *"Compositional Asymmetric Cooperations for Process Algebras with Probabilities, Priorities, and Time"*, Tech. Rep. UBLCS-2000-01, University of Bologna (Italy), 2000

[9] P. Buchholz, *"Markovian Process Algebra: Composition and Equivalence"*, in Proc. of the *2nd Int. Workshop on Process Algebra and Performance Modelling (PAPM '94)*, pp. 11-30, Erlangen (Germany), 1994

[10] W.R. Cleaveland, G. Lüttgen, V. Natarajan, *"Priority in Process Algebras"*, to appear in *"Handbook of Process Algebra"*, Elsevier, 2000

[11] R.J. van Glabbeek, S.A. Smolka, B. Steffen, *"Reactive, Generative and Stratified Models of Probabilistic Processes"*, in Information and Computation 121:59-80, 1995

[12] R.J. van Glabbeek, F.W. Vaandrager, *"Petri Net Models for Algebraic Theories of Concurrency"*, in Proc. of the *Conf. on Parallel Architectures and Languages Europe (PARLE '87)*, LNCS 259:224-242, Eindhoven (The Netherlands), 1987

[13] H. Hermanns, *"Interactive Markov Chains"*, Ph.D. Thesis, University of Erlangen-Nürnberg (Germany), 1998

[14] H. Hermanns, M. Rettelbach, *"Syntax, Semantics, Equivalences, and Axioms for MTIPP"*, in Proc. of the *2nd Int. Workshop on Process Algebra and Performance Modelling (PAPM '94)*, pp. 71-87, Erlangen (Germany), 1994

[15] H. Hermanns, M. Rettelbach, T. Weiß, *"Formal Characterisation of Immediate Actions in SPA with Nondeterministic Branching"*, in Computer Journal 38:530-541, 1995

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

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

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

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

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

[21] M. Rettelbach, *"Probabilistic Branching in Markovian Process Algebras"*, in Computer Journal 38:590-599, 1995

[22] R. Segala, *"Modeling and Verification of Randomized Distributed Real-Time Systems"*, Ph.D. Thesis, MIT, Boston (MA), 1995

[23] M. Shaw, D. Garlan, *"Software Architecture: Perspectives on an Emerging Discipline"*, Prentice Hall, 1996

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

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

[26] S.-H. Wu, S.A. Smolka, E.W. Stark, *"Composition and Behaviors of Probabilistic I/O Automata"*, in Theoretical Computer Science 176:1-38, 1997