Performance Measure Sensitive Congruences for Markovian Process Algebras

Marco Bernardo [1]

Università di Torino, Dipartimento di Informatica
Corso Svizzera 185, 10149 Torino, Italy

Mario Bravetti

Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy

Abstract

The modeling and analysis experience with process algebras has shown the necessity of extending them with priority, probabilistic internal/external choice, and time while preserving compositionality. The purpose of this paper is to make a further step by introducing a way to express performance measures, in order to allow the modeler to capture the QoS metrics of interest. We show that the standard technique of expressing stationary and transient performance measures as weighted sums of state probabilities and transition frequencies can be imported in the process algebra framework.

Technically speaking, if we denote by $n \in \mathbb{N}$ the number of performance measures of interest, in this paper we define a family of extended Markovian process algebras with generative master-reactive slaves synchronization mechanism called $\mathrm{EMPA}_{\mathrm{gr}_n}$ including probabilities, priorities, exponentially distributed durations, and sequences of rewards of length n. Then we show that the Markovian bisimulation equivalence $\sim_{\mathrm{MB}_n}$ is a congruence for $\mathrm{EMPA}_{\mathrm{gr}_n}$ which preserves the specified performance measures and we give a sound and complete axiomatization for finite $\mathrm{EMPA}_{\mathrm{gr}_n}$ terms. Finally, we present a case study conducted with the software tool TwoTowers in which we contrast the average performance of a selection of distributed algorithms for mutual exclusion modeled with $\mathrm{EMPA}_{\mathrm{gr}_n}$.

[1] Corresponding author. E-mail: `bernardo@di.unito.it`

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 [2] we consider the process algebra $\mathrm{EMPA}_{\mathrm{gr}}$ [8], because it includes probabilities, priorities, and exponentially distributed durations while preserving compositionality. $\mathrm{EMPA}_{\mathrm{gr}}$ is recalled in an incremental fashion. We start with a simple process algebra and we show how to introduce 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 that the resulting Markovian process algebra can be given semantics in the usual interleaving style thanks to the memoryless property of exponential distributions. We then extend the language with immediate actions, i.e. actions having duration zero, in order to be able to represent activities that are irrelevant from the timing viewpoint or just control the system behavior. We subsequently augment the language by attaching priorities and weights to immediate actions, to reflect the fact that it often happens in practice to encounter systems where different competing activities are scheduled according to some priority assignment and/or with a certain frequency. Finally we recall 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 [17], 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. 2).

Starting from $\mathrm{EMPA}_{\mathrm{gr}}$, the objective of this paper is to make a further step in the direction of expressivity by introducing a way to describe performance measures, in order to allow the modeler to capture the QoS metrics of interest.

[2] Full and revised version of [4] and [5] Chap. 7.

We achieve this by showing that the standard technique of expressing stationary and transient performance measures as weighted sums of state probabilities and transition frequencies can be imported in the process algebra framework. This is carried out by extending the action format to include sequences of $n \in \mathbb{N}$ yield and bonus rewards [23], thus allowing the specification of several instant-of-time performance measures (Sect. 3).

After introducing all the ingredients for our extended Markovian process algebra with generative-reactive synchronizations and rewards, called $\mathrm{EMPA}_{\mathrm{gr}_n}$, we formalize its syntax and we define its operational semantics as a mapping from terms to reward master-slaves transition systems of order n (Sect. 4).

We subsequently define a notion of equivalence in the bisimulation style, which equates $\mathrm{EMPA}_{\mathrm{gr}_n}$ terms possessing the same functional, probabilistic, prioritized and exponentially timed behavior as well as the same performance measure values. We then show that such an equivalence is a congruence, thus providing support for compositional manipulation while preserving the values of the specified performance measures, and we give a sound and complete axiomatization for nonrecursive process terms (Sect. 5).

Afterwards, we present a case study in which we model with $\mathrm{EMPA}_{\mathrm{gr}_n}$ a selection of mutual exclusion algorithms and we compute through the $\mathrm{EMPA}_{\mathrm{gr}_n}$ based software tool TwoTowers their average performance, based on indices such as the mean numbers of accesses per time unit to the critical section and to the shared control variables (Sect. 6).

The paper concludes with a discussion of related work (Sect. 7).

2 An Overview of $\mathbf{EMPA}_{\mathrm{gr}}$

In this section we recall $\mathrm{EMPA}_{\mathrm{gr}}$ in an incremental fashion as in [8].

2.1 Markovian Process Algebras

Process algebras (see, e.g., [28,22]) are compositional languages for the high level specification of concurrent systems. The main operators to build up system specifications are:

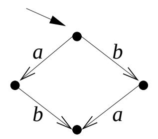
- The *action prefix operator*: $a.E$ is a system that can perform action a and then behaves as described by E.
- The *alternative composition operator*: $E_1 + E_2$ is a system that behaves as either E_1 or E_2 depending on whether an action of E_1 or an action of E_2 is

executed. The choice above is nondeterministic.

- The *parallel composition operator*: $E_1 \parallel_S E_2$ 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 $A \triangleq E$, which allows repetitive behaviors to be described.

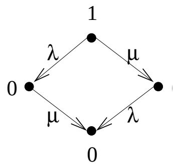
The semantics for process algebra terms is given by means of *rooted labeled transition systems* (LTSs for short) in which states correspond to process terms and transitions are labeled with actions. Such LTSs are defined by following the *interleaving approach*, i.e. parallel executions are serialized 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 $a.\underline{0} \parallel_{\emptyset} b.\underline{0}$ 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* [32] (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 $\lambda_k / \sum_{i=1}^{n} \lambda_i$.

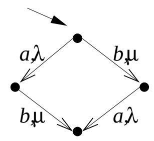
The two properties above essentially stem from the fact that the transitions leaving the same state are thought of as being in a race: the fastest one is the one that is executed. Such a *race policy* naturally applies also to the case in which two actions, whose durations are exponentially distributed with rate λ and μ 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 λ is terminated before the action with rate μ, the leftmost state is reached and its outgoing transition is labeled with μ because, when entering that state, the time to the completion of the action with rate μ is still exponentially distributed with rate μ. 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 determines the rate of the synchronization, while all the other actions must be passive. We shall return on this master-slaves synchronization mechanism in Sect. 2.5.

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 of the system. The latter, instead, is called the *Markovian semantics* as it expresses the CTMC governing the stochastic behavior of the system.

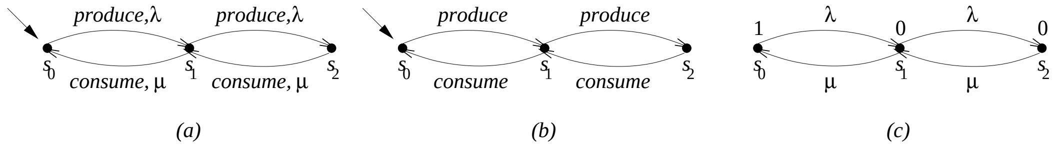


Fig. 1. Interleaving semantic models of $PCSystem_2$

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:

$$PCSystem_2 \triangleq Producer \parallel_{\{produce\}} Buffer_0 \parallel_{\{consume\}} Consumer$$

Assuming that the item production process and the item consumption process are Markovian with rate λ and μ, 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):

$$Buffer_0 \triangleq \ <produce, *>.Buffer_1$$

$$Buffer_1 \triangleq \ <produce, *>.Buffer_2 + \ <consume, *>.Buffer_0$$

$$Buffer_2 \triangleq \ <consume, *>.Buffer_1$$

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

In Fig. 1(a) we show the integrated interleaving semantic model of $PCSystem_2$. The initial state s_0 corresponds to $PCSystem_2$, state s_1 to $Producer \parallel_{\{produce\}} Buffer_1 \parallel_{\{consume\}} Consumer$, state s_2 to $Producer \parallel_{\{produce\}} Buffer_2 \parallel_{\{consume\}} Consumer$. 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. ■

2.2 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 ∞. 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 duration, 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 actually executed. If instead E represents an open system, then the execution of action $<b, \infty>$ may be disabled by the environment. For instance, $E \,\|_{\{b\}}\, \underline{0}$ 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.

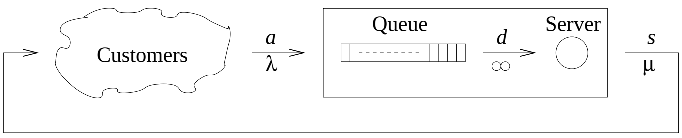


Fig. 2. Structure of a queueing system $M/M/1/q$

Example 2.2 From now on we shall exemplify each feature added to our language by means of *queueing systems* [24] (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 λ and service rate μ, which are defined as follows:

(1) The customer arrival process is Markovian with rate λ.
(2) The customer service process is Markovian with rate μ.
(3) There are n independent servers.
(4) There is a FIFO queue with $q - n$ seats. When missing, parameter q denotes an unbounded queue.
(5) 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 λ and service rate μ, 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:

$$
\begin{aligned}
QS_{M/M/1/q} &\triangleq \mathit{Arrivals} \,\|_{\{a\}} (\mathit{Queue}_0 \,\|_{\{d\}}\, \mathit{Server}) \\
\mathit{Arrivals} &\triangleq <a, \lambda>.\mathit{Arrivals} \\
\mathit{Queue}_0 &\triangleq <a, *>.\mathit{Queue}_1 \\
\mathit{Queue}_h &\triangleq <a, *>.\mathit{Queue}_{h+1} + \\
&\ <d, *>.\mathit{Queue}_{h-1}, \quad 0 < h < q - 1 \\
\mathit{Queue}_{q-1} &\triangleq <d, *>.\mathit{Queue}_{q-2} \\
\mathit{Server} &\triangleq <d, \infty>.<s, \mu>.\mathit{Server}
\end{aligned}
$$

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 $\mathit{Arrivals} \,\|_{\{a\}} (\mathit{Queue}_h \,\|_{\{d\}}\, \mathit{Server})$, AQ_hS' stands for $\mathit{Arrivals} \,\|_{\{a\}} (\mathit{Queue}_h \,\|_{\{d\}} <s, \mu>.\mathit{Server})$, and $0 \le h \le q - 1$. 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. ■

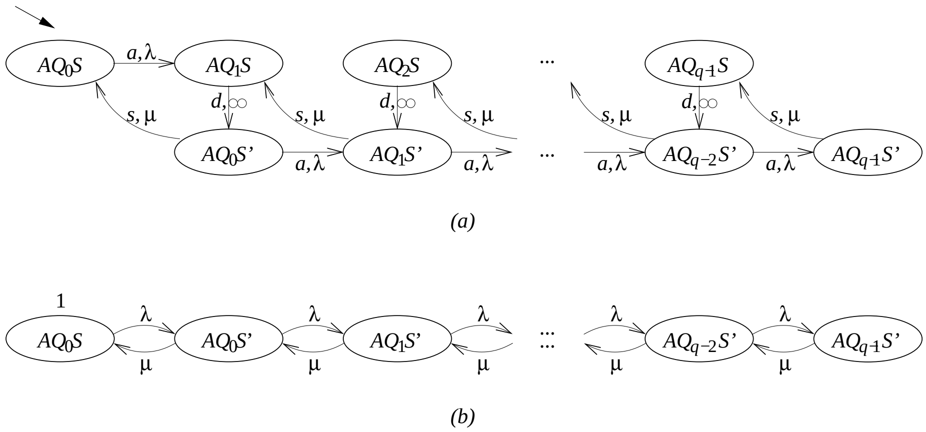


Fig. 3. Integrated and Markovian semantic models of $QS_{M/M/1/q}$

2.3 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 (∞_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:

$$\begin{tikzcd} & E \arrow[dl, "{a,\infty_1}"'] \arrow[dr, "{b,\infty_2}"] & \\ E_1 & & E_2 \end{tikzcd}$$

If E represents a closed system, then only the transition labeled with action $<b, \infty_2>$ can be actually executed. If instead E represents an open system, then the execution of action $<b, \infty_2>$ may be disabled by the environment. For instance, $E \parallel_{\{b\}} \underline{0}$ 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 2.3 Let us consider a variant of the QS of Ex. 2.2 in which there are two different classes of customers, reds and blacks, with two different arrival rates, λ_r and λ_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:

$$QS_{prio} \stackrel{\Delta}{=} (Arrivals_r \|_{\emptyset} Arrivals_b) \|_{\{a_r,a_b\}} ((Queue_{r,0} \|_{\emptyset} Queue_{b,0}) \|_{\{d_r,d_b\}} Server)$$

$$\begin{aligned} Server \stackrel{\Delta}{=}\ & <d_r, \infty_r>.<s, \mu>.Server + \\ & <d_b, \infty_b>.<s, \mu>.Server \end{aligned}$$

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:

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

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 μ. 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. ■

2.4 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:

$$\begin{tikzcd} & E \arrow[dl, "{a, \infty_{1,2}}"'] \arrow[dr, "{b, \infty_{1,3}}"] & \\ E_1 & & E_2 \end{tikzcd}$$

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, $E \parallel_{\{b\}} \underline{0}$ 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 2.4 Let us consider a variant of the QS of Ex. 2.3 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:

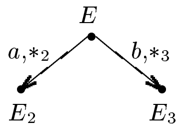
$$\begin{aligned} \mathit{Server} \triangleq\ & <d_r, \infty_{l,r}>.<s, \mu>.\mathit{Server}\ + \\ & <d_b, \infty_{l,b}>.<s, \mu>.\mathit{Server} \end{aligned}$$ ■

2.5 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 [22] 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 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. Since the role of the reactive priorities is to realize a choice mechanism among passive actions of the same type, the choice among passive actions of different types is nondeterministic, i.e. it is guided by the type of the selected active action. Thus, in the example above, the choice between $<a, *_2>$ and $<b, *_3>$ is nondeterministic.

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:

$$\begin{tikzcd}[column sep=small]
 & E \arrow[dl, "{a,*_{1,2}}"'] \arrow[d, "{a,*_{1,3}}"] \arrow[dr, "{b,*_{1,4}}"] & \\
E_1 & E_2 & E_3
\end{tikzcd}$$

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 $4/4 = 1$. Note that the sum of such probabilities is greater than 1. This is a consequence of the fact that the role of the reactive weights is to realize a choice mechanism among passive actions of the same type and priority level. The choice among passive actions of different types is nondeterministic, i.e. it is guided by the type of the selected active action. Thus, in the example above, the choice between a passive action of type a and a passive action of type b is nondeterministic.

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.

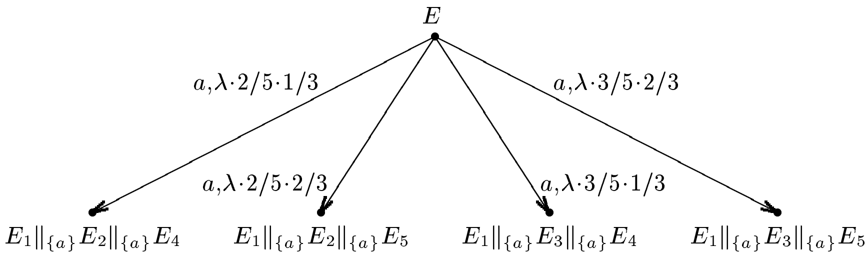
We are now in a position of explaining how the rate of an action resulting from a master-slaves synchronization is determined. In the case of synchronization between an exponentially timed action of rate λ and a passive action of the same type, the resulting rate is $\lambda \cdot p$ where p is the execution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

$$\begin{tikzcd}[column sep=small]
 & E \arrow[dl, "{a,\lambda\cdot 2/5}"'] \arrow[dr, "{a,\lambda\cdot 3/5}"] & \\
E_1 \parallel_{\{a\}} E_2 & & E_1 \parallel_{\{a\}} E_3
\end{tikzcd}$$

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 $<a, \infty_{1,4}>.E_1 \parallel_{\{a\}} (<a, *_{1,2}>.E_2 + <a, *_{1,3}>.E_3)$ has the two following initial transitions:

$$\begin{tikzcd}[column sep=small]
 & E \arrow[dl, "{a, \infty_{1,4\cdot 2/5}}"'] \arrow[dr, "{a, \infty_{1,4\cdot 3/5}}"] & \\
E_1 \parallel_{\{a\}} E_2 & & E_1 \parallel_{\{a\}} E_3
\end{tikzcd}$$

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 λ ($\infty_{l,w}$) 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 [7] based on a mixture of the generative and reactive models of probabilistic processes of [17].

We now briefly recall the two models above by resorting to the terminology of [28], 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 [36,7] 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 [31], 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 [36,7], 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* [21], 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 λ, 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 λ and μ, respectively, whose duration is the *maximum* of the two durations [20]. If we denote by τ 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. 2.1.

Example 2.5 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. 2.3 and 2.4 in which there are n classes of customers, with class i, $1 \leq i \leq n$, having arrival rate λ_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:

$$QS_n \triangleq \prod_{i=1}^{n} Arrivals_i \parallel_{\{a_i \mid 1 \leq i \leq n\}} (\prod_{i=1}^{n} Queue_{i,0} \parallel_{\{d\}} Server)$$

$$Arrivals_i \triangleq <a_i, \lambda_i>.Arrivals_i$$

$$
\begin{aligned}
Queue_{i,0} &\triangleq {<}a_i, *_{1,1}{>}.Queue_{i,1} \\
Queue_{i,h} &\triangleq {<}a_i, *_{1,1}{>}.Queue_{i,h+1} + \\
&\quad\; {<}d, *_{l_i,w_i}{>}.Queue_{i,h-1}, \;\; 0 < h < q-1 \\
Queue_{i,q-1} &\triangleq {<}d, *_{l_i,w_i}{>}.Queue_{i,q-2} \\
Server &\triangleq {<}d, \infty_{1,1}{>}.{<}s, \mu{>}.Server
\end{aligned}
$$

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. ■

3 Reward Structures

In the performance evaluation area the technique of rewards is frequently used to specify and derive measures for system models whose underlying stochastic process is a MC. According to [23], a *reward structure* for a MC is composed of:

- A *yield function* $y_{i,j}(t)$ expressing the rate at which reward is accumulated at state i t time units after i was entered when the successor state is j.
- A *bonus function* $b_{i,j}(t)$ expressing the reward awarded upon exit from state i and subsequent entry into state j given that the holding time in state i was t time units.

Since the generality of this structure is difficult to fully exploit due to the complexity of the resulting solution, the analysis is usually simplified by considering yield functions that do not depend on the time nor the successor state, as well as bonus functions that do not depend on the holding time of the previously occupied state: $y_{i,j}(t) = y_i$ and $b_{i,j}(t) = b_{i,j}$.

Several performance measures can be calculated by exploiting rewards. According to the classifications proposed in [30,18], we have *instant-of-time measures*, expressing the gain received at a particular time instant, and *interval-of-time (or cumulative) measures*, expressing the overall gain received over some time interval. Both kinds of measures can refer to stationary or transient state. In the following, we shall concentrate on instant-of-time performance measures.

In the stationary case, instant-of-time performance measures quantify the long run gain received per unit of time. Given yield rewards y_i and bonus rewards $b_{i,j}$ for a certain MC, the corresponding stationary performance measure is computed as:

$$\sum_i y_i \cdot \pi_i + \sum_i \sum_j b_{i,j} \cdot \phi_{i,j} \tag{1}$$

where π_i is the stationary probability of state i and $\phi_{i,j}$ is the stationary frequency with which the transition from state i to state j is traversed. Since $\phi_{i,j}$ is given by the stationary frequency with which state i is entered (i.e. the ratio of its stationary probability to its average holding time) multiplied by the probability with which the transition from state i to state j is traversed given that the current state is i, in the case of a CTMC we have

$$\phi_{i,j} = \pi_i \cdot q_{i,j}$$

while in the case of a DTMC we have

$$\phi_{i,j} = \pi_i \cdot p_{i,j}$$

In the transient case, instant-of-time performance measures quantify the gain received at a specific time instant. Given yield rewards y_i and bonus rewards $b_{i,j}$ for a certain MC, the corresponding transient state performance measure is computed as:

$$\sum_i y_i \cdot \pi_i(t) + \sum_i \sum_j b_{i,j} \cdot \phi_{i,j}(t) \tag{2}$$

where $\pi_i(t)$ is the probability of being in state i at time t and $\phi_{i,j}(t)$ is the transient frequency with which the transition from state i to state j is traversed at time t, which is computed in the same way as $\phi_{i,j}$ with $\pi_i(t)$ in place of π_i.

When using a formal description technique to represent the performance aspects of a system, the stochastic process associated with the underlying performance model is not directly provided by the modeler but automatically derived from the more abstract formal description of the system in order to ease the task of the modeler. As a consequence, rewards should not be defined at the level of the stochastic process but at the level of the formal description, and then automatically inherited by the stochastic process.

This is exactly what happens for well known and tool supported extensions of the Petri net formalism such as reward generalized stochastic Petri nets [9] and stochastic activity networks with rewards [29]. In both cases, yield rewards (also called rate rewards) are naturally associated with net markings, while bonus rewards (also called impulse rewards) are naturally associated with net transitions/activities.

The method we propose in this paper for specifying instant-of-time performance measures for process algebras consists of attaching sequences of pairs of the form (yield_reward, bonus_reward) to process algebra actions, where rewards are unspecified in the case of passive actions. As far as yield rewards are concerned, we assume that the yield reward earned by a state is the sum of the yield rewards of the actions it can execute *(additivity assumption)*. Since rewards are specified in the process algebraic description, we call the proposed method *algebra based*. We now assess its adequacy w.r.t. the following criteria: expressive power, ease of use, computational cost, and equational characterization.

As far as the first two criteria (expressive power and ease of use) are concerned, we observe that the algebra based method achieves a reasonable balance in that it allows many of the more frequent performance measures to be specified in a relatively easy way, which in particular does not require the knowledge of any extra formalism to describe reward structures. As an example, we show how to specify for a QS $M/M/n/n$ with arrival rate λ and service rate μ several stationary performance measures frequently occurring in practice such as those identified in [10]: rate type (e.g. throughput of a service center), counting type (e.g. mean number of customers waiting in a service center), delay type (e.g. mean response time experienced by customers in a service center), and percentage type (e.g. utilization of a service center). The QS at hand can be given two different descriptions: a state oriented description, where the focus is on the state of the set of servers (intended as the number of servers that are currently busy), and a resource oriented description, where the servers (i.e. the resources) are modeled separately [35]. The state oriented description is given by:

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

whereas the resource oriented description is given by:

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

where a stands for arrival of a customer and s stands for service of a customer.

These two different descriptions represent the same system, as it can easily be shown that they are Markovian bisimulation equivalent (see Sect. 5).

Let us compute for the QS above the mean number of customers in the system, which is the sum of the numbers of customers over all the states with each number weighted by the stationary probability of the corresponding state. According to formula (1), every state of the CTMC underlying each of the two terms above must then be given a yield reward equal to the number of customers in that state. Such a number is the number of s actions executable in that state. Therefore, in the case of $QS^{so}_{M/M/n/n}$ we must replace every action of the form $<s, h \cdot \mu>$ with $<s, h \cdot \mu, (h, 0)>$, while in the case of $QS^{ro}_{M/M/n/n}$ every action of the form $<s, \mu>$ must be replaced with $<s, \mu, (1, 0)>$ by virtue of the additivity assumption for yield rewards. All the other actions must be given zero or unspecified rewards. More precisely, unspecified rewards must be assigned to all and only the passive actions; in case of synchronization of an active action and a passive action, the resulting action essentially inherits the rewards of the original active action.

If we want to compute the throughput of the QS, defined as the mean number of customers served per time unit, we have to take into account the rate of actions having type s. In fact, the throughput is given by the service rate multiplied by the stationary probability of being in a state where service can be provided. As a consequence, in the case of $QS^{so}_{M/M/n/n}$ we must replace every action of the form $<s, h \cdot \mu>$ with $<s, h \cdot \mu, (h \cdot \mu, 0)>$ or equivalently $<s, h \cdot \mu, (0, 1)>$, while in the case of $QS^{ro}_{M/M/n/n}$ we must replace every action of the form $<s, \mu>$ with $<s, \mu, (\mu, 0)>$ or equivalently $<s, \mu, (0, 1)>$. [3]

If we want to compute instead the mean response time of the QS, defined as the mean time spent by one customer in the service center, we can exploit Little's law [24] which states that the mean response time experienced by a customer is equal to the mean number of customers in the service center divided by the customer arrival rate. Therefore, in the case of $QS^{so}_{M/M/n/n}$ we must replace every action of the form $<s, h \cdot \mu>$ with $<s, h \cdot \mu, (h/\lambda, 0)>$, while in the case of $QS^{ro}_{M/M/n/n}$ we must replace every action of the form $<s, \mu>$ with $<s, \mu, (1/\lambda, 0)>$.

Finally, if we want to compute the utilization of the QS, defined as the fraction of time during which servers are busy, we have to single out those states having an outgoing transition labeled with s, because the utilization is the sum of the stationary probabilities of such states. Thus, in the case of $QS^{so}_{M/M/n/n}$ we must replace every action of the form $<s, h \cdot \mu>$ with $<s, h \cdot \mu, (1, 0)>$. However, in the case of $QS^{ro}_{M/M/n/n}$ the algebra based method fails to determine the utilization due to the additivity assumption: the yield reward to

[3] In the continuous time case, yield rewards and bonus rewards can be used interchangeably.

associate with actions of the form <s, μ> would be the reciprocal of the number of transitions labeled with s leaving the same state. Since one of the two main objectives of the algebra based method is its ease of use, we prefer to keep the specification of rewards as simple as possible, i.e. just by means of numbers. Thus we avoid the introduction of arithmetical expressions involving particular functions such as the one determining the number of transitions of a given type leaving the same state. Incidentally, the inability to compute the utilization in the case of the resource oriented description should not come as a surprise, since this description is more suited to the determination of performance indices concerning a single server instead of the whole set of servers. As it turns out, it is quite easy to measure the utilization of a given server specified in $QS^{ro}_{M/M/n/n}$, whereas this is not possible for $QS^{so}_{M/M/n/n}$. This means that the style [35] used to describe a given system through an algebraic term is strongly related to the possibility of specifying certain performance measures through the algebra based method.

For the considered QS, the algebra based method also allows transient measures to be expressed according to formula (2). As an example, yield rewards can be used to measure the mean number of customers in the system at a given instant or the probability that a certain server is in use at a given instant, whereas bonus rewards can be employed to assess the frequency with which customers arrive or are served at a given instant.

The third criterion (computational cost) requires associating rewards with states and transitions to be not exceedingly expensive: in particular, a full scan of the state space should be avoided. As we shall see in Sect. 4, the algebra based method satisfies this requirement because rewards can be computed and assigned to states and transitions at semantic model construction time.

Finally, the fourth criterion (equational characterization) requires the method to allow process terms to be compositionally manipulated without altering their performance measures. This is an important feature. As an example, if one uses a measure insensitive equivalence to reduce the state space before evaluating the performance, there is the risk to merge together states which are different w.r.t. the measures of interest, thus resulting in wrong performance figures. In Sect. 5 we shall see that the algebra based method permits the definition of performance measure sensitive congruences over process terms. This is the other main objective of the algebra based method and constitutes its major advantage.

4 Syntax and Semantics for $\text{EMPA}_{\text{gr}_n}$

In this section we formalize the syntax and the semantics for the process algebra informally presented in the previous sections. More precisely, denoted by $n \in \mathbb{N}$ the number of performance measures of interest, we define the syntax of a family of extended Markovian process algebras with generative-reactive synchronizations $\text{EMPA}_{\text{gr}_n}$, where each action is extended to accommodate a sequence of n pairs of rewards. Then we introduce the semantic model constituted by the reward master-slaves transition system. Finally we present an operational semantics that maps $\text{EMPA}_{\text{gr}_n}$ terms onto reward master-slaves transition systems of order n.

4.1 Syntax and Informal Semantics

The main ingredients of our calculus are the actions, each composed of a type, a rate, and a sequence of pairs of yield and bonus rewards, 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 τ, as usual.

Definition 4.1 Let *AType* be the set of *action types* including the invisible type τ, $ARate = \mathbb{R}_+ \cup \{\infty_{l,w} \mid l \in \mathbb{N}_+ \wedge w \in \mathbb{R}_+\} \cup \{*_{l,w} \mid l \in \mathbb{N}_+ \wedge w \in \mathbb{R}_+\}$ be the set of *action rates*, $ARew = \mathbb{R} \cup \{*\}$ be the set of *action rewards*. We use a to range over *AType*, $\tilde{\lambda}$ to range over *ARate*, λ to range over exponentially timed rates, $\bar{\lambda}$ to range over nonpassive rates, $\tilde{y}$ to range over yield rewards (y if not $*$), and $\tilde{b}$ to range over bonus rewards (b if not $*$). The set of *actions* with $n \in \mathbb{N}$ pairs of rewards is defined by

$$\begin{aligned} Act_n = \{ &<a, \tilde{\lambda}, (\tilde{y}_1, \tilde{b}_1) \dots (\tilde{y}_n, \tilde{b}_n)> \in AType \times ARate \times (ARew \times ARew)^n \mid \\ &(\tilde{\lambda} \in \{*_{l,w} \mid l \in \mathbb{N}_+ \wedge w \in \mathbb{R}_+\} \wedge \forall i \in \{1, \dots, n\}.\, \tilde{y}_i = \tilde{b}_i = *) \vee \\ &(\tilde{\lambda} \in \mathbb{R}_+ \cup \{\infty_{l,w} \mid l \in \mathbb{N}_+ \wedge w \in \mathbb{R}_+\} \wedge \forall i \in \{1, \dots, n\}.\, \tilde{y}_i, \tilde{b}_i \in \mathbb{R})\} \end{aligned}$$

■

Definition 4.2 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 φ. The set $\mathcal{L}_n$ of *process terms* of $\text{EMPA}_{\text{gr}_n}$ is generated by the following syntax

$$E ::= \underline{0} \mid <a, \tilde{\lambda}, (\tilde{y}_1, \tilde{b}_1) \dots (\tilde{y}_n, \tilde{b}_n)>.E \mid E/L \mid E[\varphi] \mid E + E \mid E \parallel_S E \mid A$$

where $L, S \subseteq AType - \{\tau\}$. ■

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

The *action prefix operator* “$<a, \tilde{\lambda}, (\tilde{y}_1, \tilde{b}_1) \dots (\tilde{y}_n, \tilde{b}_n)>.\_$” denotes the sequential composition of an action and a term. Term $<a, \tilde{\lambda}, (\tilde{y}_1, \tilde{b}_1) \dots (\tilde{y}_n, \tilde{b}_n)>.E$ can execute an action with type a and rate $\tilde{\lambda}$, thus making the corresponding state earn additional yield rewards $\tilde{y}_1 \dots \tilde{y}_n$ and the related transition gain bonus rewards $\tilde{b}_1 \dots \tilde{b}_n$, 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 τ 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. 2.5. 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 complies with the bounded capacity assumption, hence makes the structure of synchronizations in a system state easier to understand, as formally shown in [8]. As far as action rewards are concerned, since only the rewards of active actions are specified, in case of synchronization they are handled as follows. The yield rewards of an active action are treated exactly as the rate of that action, i.e. they are multiplied by the execution probabilities of the passive actions involved in the synchronization. Instead, the bonus rewards of an active action are just inherited, as multiplying them by the execution probabilities of the aforementioned passive actions would lead to an underestimation of the performance measures. The reason is that, in the calculation of the performance measures, each bonus reward of a transition is multiplied by a factor which is proportional to the rate of the transition itself, hence multiplying the rates by the execution probabilities of passive actions is all we have to do. In the case of synchronization between two passive actions, the rewards of the resulting passive actions are still unspecified.

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

4.2 Reward Master-Slaves Transition Systems

The semantic model of $\mathrm{EMPA}_{\mathrm{gr}_n}$ is a special kind of LTS we call *master-slaves transition system of order* n (RMSTS_n for short), whose transitions are labeled with elements of Act_n. Recalling that active actions play the role of the masters while passive actions play the role of the slaves, each state of a RMSTS_n 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 $\mathrm{EMPA}_{\mathrm{gr}_n}$ will be defined in such a way that lower priority active transitions are not pruned (see the congruence related motivation in Sect. 5) while lower priority passive transitions of a given type are, all the passive transitions belonging to the same slave bundle of a generated RMSTS_n have the same priority level. For the sake of simplicity, in the rest of this section we shall deal with reward pair sequences of length one.

Definition 4.3 A reward master-slaves transition system of order 1 (RMSTS_1) is a triple

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

where: [4]

- S is a set of states;
- $AType$ is a set of action types;
- $\longrightarrow \in \mathcal{M}(S \times Act_1 \times S)$ is a multiset of transitions such that for all $s \in S$ and $a \in AType$:

[4] 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.

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

A rooted reward master-slaves transition system of order 1 (RRMSTS_1) is a quadruple

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

where $(S, AType, \longrightarrow)$ is a RMSTS_1 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 $<a, \lambda, (y, b)>$, using sets instead of multisets would reduce the two transitions into a single one with rate λ, thus erroneously altering the average sojourn time in the state.

The choice among the bundles of transitions enabled in a state is nondeterministic. 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 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 $<a_i, \lambda_i, (y_i, b_i)>$, $1 \le i \le n$, then the time to choose one of such actions is exponentially distributed with rate $\sum_{1 \le i \le n} \lambda_i$ and the probability of choosing a_k is given by $\lambda_k / \sum_{1 \le i \le n} \lambda_i$. 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 $<a_i, \infty_{l_{max},w_i}, (y_i, b_i)>$, $1 \le i \le n$, then the probability of choosing a_k is given by $w_k / \sum_{1 \le i \le n} w_i$.

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 $<a_i, *_{l,w_i}, (*, *)>$, $1 \le i \le n$, then the probability of choosing a_k is given by $w_k / \sum_{1 \le i \le n} w_i$. 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 reward DTMC or 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 [5] Chap. 4.

4.3 Operational Semantics

The formal semantics for $\mathrm{EMPA}_{\mathrm{gr}_n}$ maps terms onto RRMSTS_n. In the following, we shall consider $\mathrm{EMPA}_{\mathrm{gr}_1}$ for the sake of simplicity. We also provide the following shorthands to make the definition of the operational semantic rules easier.

Definition 4.4 Given a RMSTS_1 $M = (S, \mathit{AType}, \longrightarrow)$, $s \in S$, and $a \in \mathit{AType}$, 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 $\mathrm{EMPA}_{\mathrm{gr}}$ is the least RMSTS_1 $(\mathcal{G}_1, \mathit{AType}, \longrightarrow_1)$ 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. Similarly to [21], we consider the operational rules as generating a multiset of transitions (consistently with the definition of RMSTS_1), where a transition has arity m if and only if it can be derived in m possible ways from the operational rules.

Some explanations are now in order. First of all, the operational rules give rise to an interleaving semantics, which is made possible by the memoryless property of exponential distributions. The removal of lower priority passive transitions of the same type is carried out in rules $(Ch2_l)$ and $(Ch2_r)$ for the

$$(\boldsymbol{Pr})\quad <a,\tilde{\lambda},(\tilde{y},\tilde{b})>.E \xrightarrow{a,\tilde{\lambda},(\tilde{y},\tilde{b})}_1 E$$

$$(\boldsymbol{Hi1})\quad \frac{E \xrightarrow{a,\tilde{\lambda},(\tilde{y},\tilde{b})}_1 E'}{E/L \xrightarrow{a,\tilde{\lambda},(\tilde{y},\tilde{b})}_1 E'/L}\ a \notin L \qquad\qquad (\boldsymbol{Hi2})\quad \frac{E \xrightarrow{a,\tilde{\lambda},(\tilde{y},\tilde{b})}_1 E'}{E/L \xrightarrow{\tau,\tilde{\lambda},(\tilde{y},\tilde{b})}_1 E'/L}\ a \in L$$

$$(\boldsymbol{Re})\quad \frac{E \xrightarrow{a,\tilde{\lambda},(\tilde{y},\tilde{b})}_1 E'}{E[\varphi] \xrightarrow{\varphi(a),\tilde{\lambda},(\tilde{y},\tilde{b})}_1 E'[\varphi]}$$

$$(\boldsymbol{Ch1}_l)\quad \frac{E_1 \xrightarrow{a,\bar{\lambda},(y,b)}_1 E'_1}{E_1 + E_2 \xrightarrow{a,\bar{\lambda},(y,b)}_1 E'_1} \qquad\qquad (\boldsymbol{Ch2}_l)\quad \frac{E_1 \xrightarrow{a,*_{l,w},(*,*)}_1 E'_1 \quad l \geq L_a(E_2)}{E_1 + E_2 \xrightarrow{a,*_{l,w},(*,*)}_1 E'_1}$$

$$(\boldsymbol{Pa1}_l)\quad \frac{E_1 \xrightarrow{a,\bar{\lambda},(y,b)}_1 E'_1}{E_1 \parallel_S E_2 \xrightarrow{a,\bar{\lambda},(y,b)}_1 E'_1 \parallel_S E_2}\ a \notin S$$

$$(\boldsymbol{Pa2}_l)\quad \frac{E_1 \xrightarrow{a,*_{l,w},(*,*)}_1 E'_1 \quad l \geq L_a(E_2)}{E_1 \parallel_S E_2 \xrightarrow{a,*_{l,w},(*,*)}_1 E'_1 \parallel_S E_2}\ a \notin S$$

$$(\boldsymbol{Sy1}_l)\quad \frac{E_1 \xrightarrow{a,\bar{\lambda},(y,b)}_1 E'_1 \qquad E_2 \xrightarrow{a,*_{l,w},(*,*)}_1 E'_2}{E_1 \parallel_S E_2 \xrightarrow{a,\bar{\lambda}\cdot\frac{w}{W_a(E_2)},(y\cdot\frac{w}{W_a(E_2)},b)}_1 E'_1 \parallel_S E'_2}\ a \in S$$

$$(\boldsymbol{Sy2})\quad \frac{E_1 \xrightarrow{a,*_{l_1,w_1},(*,*)}_1 E'_1 \qquad E_2 \xrightarrow{a,*_{l_2,w_2},(*,*)}_1 E'_2}{E_1 \parallel_S E_2 \xrightarrow{a,*_{\max(l_1,l_2),p\cdot N},(*,*)}_1 E'_1 \parallel_S E'_2}\ a \in S$$

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

$$(\boldsymbol{Co})\quad \frac{E \xrightarrow{a,\tilde{\lambda},(\tilde{y},\tilde{b})}_1 E'}{A \xrightarrow{a,\tilde{\lambda},(\tilde{y},\tilde{b})}_1 E'}\ A \triangleq E$$

Table 1
$\mathrm{EMPA}_{\mathrm{gr}_1}$ operational semantics

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. 5.4, 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 [13], 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. The yield reward of the active action undergoes the same treatment, while the bonus reward is just inherited. 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. As explained in Sect. 4.1, 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 $W_a(E)$.

Definition 4.5 The *integrated semantics* of $E \in \mathcal{G}_1$ is the RRMSTS_1

$$\mathcal{I}_1[\![E]\!] = (\mathcal{G}_{1,E}, AType, \longrightarrow_{1,E}, E)$$

where $\mathcal{G}_{1,E}$ is the set of terms reachable from E according to the RMSTS_1 $(\mathcal{G}_1, AType, \longrightarrow_1)$ and $\longrightarrow_{1,E}$ is the restriction of $\longrightarrow_1$ to transitions between terms in $\mathcal{G}_{1,E}$. We say that $E \in \mathcal{G}_1$ is *performance closed* if and only if $\mathcal{I}_1[\![E]\!]$ does not contain passive transitions. ■

We conclude by recalling that from $\mathcal{I}_1[\![E]\!]$ two projected semantic models can be obtained by essentially dropping action rates or action types, respectively. Before applying such a transformation to $\mathcal{I}_1[\![E]\!]$, 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* $\mathcal{F}_1[\![E]\!]$ is a standard LTS whose transitions are decorated with action types only. The *Markovian semantics* $\mathcal{M}_1[\![E]\!]$ is instead a reward CTMC or DTMC, as seen in Sect. 4.2, which is well defined only if E is performance closed.

5 Markovian Bisimulation Equivalence $\sim_{\mathrm{MB}_n}$

In this section we equip $\mathrm{EMPA}_{\mathrm{gr}_n}$ with a Markovian bisimulation equivalence $\sim_{\mathrm{MB}_n}$, which relates systems having the same functional, probabilistic, prioritized and exponentially timed behavior as well as the same performance measure values. We then show that such an equivalence is a congruence, thus providing support for compositional manipulation while preserving the values of the specified performance measures, and we give a sound and complete axiomatization for nonrecursive process terms. Finally, we present an example of application of the Markovian bisimulation equivalence to the performability analysis of a QS.

5.1 Definition, Congruence Property, and Axiomatization

Our Markovian bisimulation equivalence $\sim_{\mathrm{MB}_n}$ extends the Markovian bisimulation equivalence for $\mathrm{EMPA}_{\mathrm{gr}}$ [8]. The latter is in turn inspired by the probabilistic bisimulation equivalence of [25], 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.

Let us consider $\mathrm{EMPA}_{\mathrm{gr}_0}$ for the moment. 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. 2.1), 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

$$<a, \lambda_1>.E + <a, \lambda_2>.E \sim_{\mathrm{MB}_0} <a, \lambda_1 + \lambda_2>.E$$

This treatment of rates, originally proposed in [21], is basically the same as that of the exact aggregation for MCs known as *ordinary lumping* [32], 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

$$<a, \infty_{l,w_1}>.E + <a, \infty_{l,w_2}>.E \sim_{\mathrm{MB}_0} <a, \infty_{l,w_1+w_2}>.E$$

$$<a, *_{l,w_1}>.E + <a, *_{l,w_2}>.E \sim_{\mathrm{MB}_0} <a, *_{l,w_1+w_2}>.E$$

This treatment of weights was originally proposed in [33].

As far as rewards are concerned, according to the formulas of Sect. 3 we have that yield rewards must be handled in the same way as rates, because of the additivity assumption. Bonus rewards of actions of the same type and priority level, instead, cannot be summed up, as this would result in an overestimation of the specified performance measures. The reason is that, in the calculation of the performance measures, the bonus reward of a transition is multiplied by a factor which is proportional to the rate of the transition itself, hence summing rates up is all we have to do. As an example, in $\mathrm{EMPA}_{\mathrm{gr}_1}$ it must hold that

$$<a, \lambda_1, (y_1, b)>.E + <a, \lambda_2, (y_2, b)>.E \sim_{\mathrm{MB}_1} <a, \lambda_1 + \lambda_2, (y_1 + y_2, b)>.E$$

We are now in a position of defining a Markovian bisimulation equivalence that is sensitive to performance measures. For the sake of simplicity, we shall be working with $\mathrm{EMPA}_{\mathrm{gr}_1}$, with the understanding that, when working with arbitrarily long sequences of pairs of rewards, all the yield rewards must be treated in the same way and that all the bonus rewards must be treated in the same way.

Definition 5.1 We define function priority level $PL : ARate \longrightarrow \mathbb{Z}$ by:

$$\begin{aligned} PL(*_{l,w}) &= -l \\ PL(\lambda) &= 0 \\ PL(\infty_{l,w}) &= l \end{aligned}$$

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

$$\begin{aligned} *_{l,w_1} + *_{l,w_2} &= *_{l,w_1+w_2} \\ \infty_{l,w_1} + \infty_{l,w_2} &= \infty_{l,w_1+w_2} \\ * + * &= * \end{aligned}$$

We then define partial function aggregated rate-yield $RY_1 : \mathcal{G}_1 \times AType \times \mathbb{Z} \times ARew \times \mathcal{P}(\mathcal{G}_1) \longrightarrow\!\!\!\!\!\!\circ\;\; ARate \times ARew$ by:

$$\begin{aligned} RY_1(E, a, l, \tilde{b}, C) = (&\textstyle\sum\{\!| \, \tilde{\lambda} \mid \exists \tilde{y}.\, \exists E' \in C.\, E \xrightarrow{a,\tilde{\lambda},(\tilde{y},\tilde{b})}_1 E' \wedge PL(\tilde{\lambda}) = l \, |\!\}, \\ &\textstyle\sum\{\!| \, \tilde{y} \mid \exists \tilde{\lambda}.\, \exists E' \in C.\, E \xrightarrow{a,\tilde{\lambda},(\tilde{y},\tilde{b})}_1 E' \wedge PL(\tilde{\lambda}) = l \, |\!\}) \end{aligned}$$

with $RY_1(E, a, l, \tilde{b}, C) = \perp$ whenever the multisets above are empty. ■

Definition 5.2 An equivalence relation $\mathcal{B} \subseteq \mathcal{G}_1 \times \mathcal{G}_1$ is a *Markovian bisimulation of order 1* if and only if, whenever $(E_1, E_2) \in \mathcal{B}$, then for all $a \in AType$, $l \in \mathbb{Z}$, $\tilde{b} \in ARew$, and equivalence classes $C \in \mathcal{G}_1/\mathcal{B}$

$$RY_1(E_1, a, l, \tilde{b}, C) = RY_1(E_2, a, l, \tilde{b}, C)$$

■

It is easy to see that the union of all the Markovian bisimulations of order 1 is a Markovian bisimulation of order 1.

Definition 5.3 We call $\sim_{\mathrm{MB}_1}$, defined as the union of all the Markovian bisimulations of order 1, the *Markovian bisimulation equivalence of order 1.*

■

Theorem 5.4 Let $E_1, E_2 \in \mathcal{G}_1$. If $E_1 \sim_{\mathrm{MB}_1} E_2$ then:

(1) For all $<a, \tilde{\lambda}, (\tilde{y}, \tilde{b})> \in Act_1$, $<a, \tilde{\lambda}, (\tilde{y}, \tilde{b})>.E_1 \sim_{\mathrm{MB}_1} <a, \tilde{\lambda}, (\tilde{y}, \tilde{b})>.E_2$.
(2) For all $L \subseteq AType - \{\tau\}$, $E_1/L \sim_{\mathrm{MB}_1} E_2/L$.
(3) For all $\varphi \in ATRFun$, $E_1[\varphi] \sim_{\mathrm{MB}_1} E_2[\varphi]$.
(4) For all $F \in \mathcal{G}_1$, $E_1 + F \sim_{\mathrm{MB}_1} E_2 + F$ and $F + E_1 \sim_{\mathrm{MB}_1} F + E_2$.
(5) For all $F \in \mathcal{G}_1$ and $S \subseteq AType - \{\tau\}$, $E_1 \parallel_S F \sim_{\mathrm{MB}_1} E_2 \parallel_S F$ and $F \parallel_S E_1 \sim_{\mathrm{MB}_1} F \parallel_S E_2$.

Additionally, $\sim_{\mathrm{MB}_1}$ is closed w.r.t. recursive constant definitions.

Proof The proof is similar to that of the corresponding theorem of [8] with some changes in the case of the alternative and parallel composition operators that we now show. In the following, the extended real number summation and multiplication of Def. 5.1 and 4.4, respectively, are also used to express the componentwise, extended real number vector summation and multiplication. We also denote by $Rate_1$ the partial function obtained from RY_1 by returning only the first component of the result vector.

- Let $\mathcal{B} \subseteq \mathcal{G}_1 \times \mathcal{G}_1$ be a Markovian bisimulation of order 1 such that $(E_1, E_2) \in \mathcal{B}$. Given $F \in \mathcal{G}_1$, we prove that $\mathcal{B}' = (\mathcal{B} \cup \{(E_1 + F, E_2 + F), (E_2 + F, E_1 + F)\})^+$ is a Markovian bisimulation of order 1. 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
  $$\mathcal{G}_1/\mathcal{B}' = (\mathcal{G}_1/\mathcal{B} - \{[E_1 + F]_{\mathcal{B}}, [E_2 + F]_{\mathcal{B}}\}) \cup \{[E_1 + F]_{\mathcal{B}} \cup [E_2 + F]_{\mathcal{B}}\}$$
  let $(F_1, F_2) \in \mathcal{B}'$, $a \in AType$, $l \in \mathbb{Z}$, $\tilde{b} \in ARew$, and $C \in \mathcal{G}_1/\mathcal{B}'$.
    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 $RY_1(F_1, a, l, \tilde{b}, C) = RY_1(F_2, a, l, \tilde{b}, C)$.
    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 $RY_1(F_j, a, l, \tilde{b}, C) = RY_1(F_j, a, l, \tilde{b}, [E_1 + F]_{\mathcal{B}}) + RY_1(F_j, a, l, \tilde{b}, [E_2 + F]_{\mathcal{B}})$ so $RY_1(F_1, a, l, \tilde{b}, C) = RY_1(F_2, a, l, \tilde{b}, C)$.
    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
    $$RY_1(F_j, a, l, \tilde{b}, C) = \begin{cases} RY_1(E_j, a, l, \tilde{b}, C) + RY_1(F, a, l, \tilde{b}, C) \\ RY_1(E_j, a, l, \tilde{b}, C) \\ RY_1(F, a, l, \tilde{b}, C) \\ \bot \end{cases}$$
    depending on whether $RY_1(E_j, a, l, \tilde{b}, C) \neq \bot \wedge RY_1(F, a, l, \tilde{b}, C) \neq \bot$ or $RY_1(E_j, a, l, \tilde{b}, C) \neq \bot \wedge ((l \geq 0 \wedge RY_1(F, a, l, \tilde{b}, C) = \bot) \vee (l < 0 \wedge \forall l' \in \mathbb{Z}_- . \, l' \leq l \Longrightarrow RY_1(F, a, l', *, \mathcal{G}_1) = \bot))$ or $RY_1(F, a, l, \tilde{b}, C) \neq \bot \wedge ((l \geq 0 \wedge RY_1(E_j, a, l, \tilde{b}, C) = \bot) \vee (l < 0 \wedge \forall l' \in \mathbb{Z}_- . \, l' \leq l \Longrightarrow RY_1(E_j, a, l', *, \mathcal{G}_1) = \bot))$ or none of the previous clauses holds.

If $C \in \mathcal{G}_1/\mathcal{B} - \{[E_1 + F]_{\mathcal{B}}, [E_2 + F]_{\mathcal{B}}\}$, then from $(E_1, E_2) \in \mathcal{B}$ we derive $RY_1(E_1, a, l, \tilde{b}, C) = RY_1(E_2, a, l, \tilde{b}, C)$ so $RY_1(F_1, a, l, \tilde{b}, C) = RY_1(F_2, a, l, \tilde{b}, C)$.

If $C = [E_1 + F]_{\mathcal{B}} \cup [E_2 + F]_{\mathcal{B}}$, then for $j \in \{1, 2\}$ we have $RY_1(E_j, a, l, \tilde{b}, C) = RY_1(E_j, a, l, \tilde{b}, [E_1 + F]_{\mathcal{B}}) + RY_1(E_j, a, l, \tilde{b}, [E_2 + F]_{\mathcal{B}})$. Since $(E_1, E_2) \in \mathcal{B}$, it turns out that $RY_1(E_1, a, l, \tilde{b}, C) = RY_1(E_2, a, l, \tilde{b}, C)$ so $RY_1(F_1, a, l, \tilde{b}, C) = RY_1(F_2, a, l, \tilde{b}, C)$.

- Given $F \in \mathcal{G}_1$ and $S \subseteq AType - \{\tau\}$, we prove that $\mathcal{B}' = \mathcal{B} \cup Id_{\mathcal{G}_1}$, where $\mathcal{B} = \{(E_1 \,\|_S\, F, E_2 \,\|_S\, F) \mid E_1 \sim_{\mathrm{MB}_1} E_2\}$ and $Id_{\mathcal{G}_1}$ is the identity relation over $\mathcal{G}_1$, is a Markovian bisimulation of order 1. 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 $(F_1, F_2) \in \mathcal{B}'$, $a \in AType$, $l \in \mathbb{Z}$, $\tilde{b} \in ARew$, and $C \in \mathcal{G}_1/\mathcal{B}'$.

· If $(F_1, F_2) \in Id_{\mathcal{G}_1}$, then trivially $RY_1(F_1, a, l, \tilde{b}, C) = RY_1(F_2, a, l, \tilde{b}, C)$.

· 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 $E_1 \sim_{\mathrm{MB}_1} E_2$.

If none of the terms in C has "$\_ \,\|_S\, F$" as outermost operator, then trivially $RY_1(F_1, a, l, \tilde{b}, C) = \bot = RY_1(F_2, a, l, \tilde{b}, C)$.

If each of the terms in C has "$\_ \,\|_S\, F$" as outermost operator, given $E \,\|_S\, G \in C$ it turns out that $C = \{E' \,\|_S\, G \mid E' \in [E]_{\sim_{\mathrm{MB}_1}}\}$.

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

$$RY_1(F_j,a,l,\tilde{b},C) = \begin{cases} RY_1(E_j, a, l, \tilde{b}, [E]_{\sim_{\mathrm{MB}_1}}) + RY_1(F, a, l, \tilde{b}, \{G\}) \\ RY_1(E_j, a, l, \tilde{b}, [E]_{\sim_{\mathrm{MB}_1}}) \\ RY_1(F, a, l, \tilde{b}, \{G\}) \\ \bot \end{cases}$$

depending on whether $E_j \in [E]_{\sim_{\mathrm{MB}_1}} \wedge F \equiv G \wedge RY_1(E_j, a, l, \tilde{b}, [E]_{\sim_{\mathrm{MB}_1}}) \neq \bot \wedge RY_1(F, a, l, \tilde{b}, \{G\}) \neq \bot$ or $E_j \notin [E]_{\sim_{\mathrm{MB}_1}} \wedge F \equiv G \wedge RY_1(E_j, a, l, \tilde{b}, [E]_{\sim_{\mathrm{MB}_1}}) \neq \bot \wedge (l \geq 0 \vee (l < 0 \wedge \forall l' \in \mathbb{Z}_-.\, l' < l \Longrightarrow RY_1(F, a, l', *, \mathcal{G}_1) = \bot))$ or $E_j \in [E]_{\sim_{\mathrm{MB}_1}} \wedge F \not\equiv G \wedge RY_1(F, a, l, \tilde{b}, \{G\}) \neq \bot \wedge (l \geq 0 \vee (l < 0 \wedge \forall l' \in \mathbb{Z}_-.\, l' < l \Longrightarrow RY_1(E_j, a, l', *, [E]_{\sim_{\mathrm{MB}_1}}) = \bot$ or none of the previous clauses holds. Since $E_1 \sim_{\mathrm{MB}_1} E_2$, it follows that $RY_1(F_1, a, l, \tilde{b}, C) = RY_1(F_2, a, l, \tilde{b}, C)$.

If $a \in S$ then, supposed

$$\begin{aligned} *_{-l, w_{E_j,l}} &= Rate_1(E_j, a, l, *, [E]_{\sim_{\mathrm{MB}_1}}) \\ *_{-l, w_{E_j,l,\mathrm{tot}}} &= Rate_1(E_j, a, l, *, \mathcal{G}_1) \\ *_{-l, w_{F,l}} &= Rate_1(F, a, l, *, \{G\}) \\ *_{-l, w_{F,l,\mathrm{tot}}} &= Rate_1(F, a, l, *, \mathcal{G}_1) \end{aligned}$$

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

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

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

$$RY_1(F_j, a, l, \tilde{b}, C) = \begin{cases} RY_1(E_j, a, l, \tilde{b}, [E]_{\sim_{\mathrm{MB}_1}}) \cdot w_{F,l'}/w_{F,l',\mathrm{tot}} + \\ \qquad RY_1(F, a, l, \tilde{b}, \{G\}) \cdot w_{E_j,l''}/w_{E_j,l'',\mathrm{tot}} \\ RY_1(E_j, a, l, \tilde{b}, [E]_{\sim_{\mathrm{MB}_1}}) \cdot w_{F,l'}/w_{F,l',\mathrm{tot}} \\ RY_1(F, a, l, \tilde{b}, \{G\}) \cdot w_{E_j,l''}/w_{E_j,l'',\mathrm{tot}} \\ *_{-l, w_{E_j,l_1}/w_{E_j,l_1,\mathrm{tot}} \cdot w_{F,l_2}/w_{F,l_2,\mathrm{tot}} \cdot N} \\ \perp \end{cases}$$

depending on whether $l \geq 0 \wedge RY_1(E_j, a, l, \tilde{b}, [E]_{\sim_{\mathrm{MB}_1}}) \neq \perp \wedge \exists l' \in \mathbb{Z}_-.\ RY_1(F, a, l', *, \{G\}) \neq \perp \wedge RY_1(F, a, l, \tilde{b}, \{G\}) \neq \perp \wedge \exists l'' \in \mathbb{Z}_-.\ RY_1(E_j, a, l'', *, [E]_{\sim_{\mathrm{MB}_1}}) \neq \perp$ or $l \geq 0 \wedge RY_1(E_j, a, l, \tilde{b}, [E]_{\sim_{\mathrm{MB}_1}}) \neq \perp \wedge \exists l' \in \mathbb{Z}_-.\ RY_1(F, a, l', *, \{G\}) \neq \perp \wedge (RY_1(F, a, l, \tilde{b}, \{G\}) = \perp \vee \forall l''' \in \mathbb{Z}_-.\ RY_1(E_j, a, l''', *, [E]_{\sim_{\mathrm{MB}_1}}) = \perp)$ or $l \geq 0 \wedge RY_1(F, a, l, \tilde{b}, \{G\}) \neq \perp \wedge \exists l'' \in \mathbb{Z}_-.\ RY_1(E_j, a, l'', *, [E]_{\sim_{\mathrm{MB}_1}}) \neq \perp \wedge (RY_1(E_j, a, l, \tilde{b}, [E]_{\sim_{\mathrm{MB}_1}}) = \perp \vee \forall l''' \in \mathbb{Z}_-.\ RY_1(F, a, l''', *, \{G\}) = \perp)$ or $l < 0 \wedge \exists l_1, l_2 \in \mathbb{Z}_-.\ RY_1(E_j, a, l_1, \tilde{b}, [E]_{\sim_{\mathrm{MB}_1}}) \neq \perp \wedge RY_1(F, a, l_2, \tilde{b}, \{G\}) \neq \perp \wedge -l = \max(-l_1, -l_2)$ or none of the previous clauses holds. From $E_1 \sim_{\mathrm{MB}_1} E_2$, it follows that $RY_1(F_1, a, l, \tilde{b}, C) = RY_1(F_2, a, l, \tilde{b}, C)$. ■

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. $<a_1, \lambda, (y_1, b_1)>.\underline{0} + <a_2, \infty_{l,w}, (y_2, b_2)>.\underline{0} \sim_{\mathrm{MB}_1} <a_2, \infty_{l,w}, (y_2, b_2)>.\underline{0}$ as both terms would have only one transition labeled with $<a_2, \infty_{l,w}, (y_2, b_2)>$, but $(<a_1, \lambda, (y_1, b_1)>.\underline{0} + <a_2, \infty_{l,w}, (y_2, b_2)>.\underline{0}) \parallel_{\{a_2\}} \underline{0} \not\sim_{\mathrm{MB}_1} <a_2, \infty_{l,w}, (y_2, b_2)>.\underline{0} \parallel_{\{a_2\}} \underline{0}$ because the first term has a transition labeled with action $<a_1, \lambda, (y_1, b_1)>$ 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 5.5 Let $\mathcal{A}_1$ be the set of axioms in Table 2. The deductive system $Ded(\mathcal{A}_1)$ is sound and complete for $\sim_{\mathrm{MB}_1}$ over the set of nonrecursive terms of $\mathcal{G}_1$.

Proof The proof is similar to that of the corresponding theorem of [8] with the difference that a nonrecursive term $E \in \mathcal{G}_1$ is defined to be in sum normal form (snf) if and only if E is $\underline{0}$ or $\sum_{i \in I} <a_i, \tilde{\lambda}_i, (\tilde{y}_i, \tilde{b}_i)>.E_i$ with every E_i in snf,

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

$$
\begin{array}{ll}
(\mathcal{A}_{13})_1 & \sum\limits_{i \in I_0} <a_i, \tilde{\lambda}_i, (\tilde{y}_i, \tilde{b}_i)>.E_i \parallel_S \sum\limits_{i \in I_1} <a_i, \tilde{\lambda}_i, (\tilde{y}_i, \tilde{b}_i)>.E_i \; = \\
 & \quad \sum\limits_{j \in I_0, a_j \notin S} <a_j, \tilde{\lambda}_j, (\tilde{y}_j, \tilde{b}_j)>.(E_j \parallel_S \sum\limits_{i \in I_1} <a_i, \tilde{\lambda}_i, (\tilde{y}_i, \tilde{b}_i)>.E_i) \; + \\
 & \quad \sum\limits_{j \in I_1, a_j \notin S} <a_j, \tilde{\lambda}_j, (\tilde{y}_j, \tilde{b}_j)>.(\sum\limits_{i \in I_0} <a_i, \tilde{\lambda}_i, (\tilde{y}_i, \tilde{b}_i)>.E_i \parallel_S E_j) \; + \\
 & \quad \sum\limits_{k \in K_0} \sum\limits_{h \in P_{1,a_k}} <a_k, \tilde{\lambda}_k \cdot (w_h / W_{1,a_k}), (\tilde{y}_k \cdot (w_h / W_{1,a_k}), \tilde{b}_k)>.(E_k \parallel_S E_h) \; + \\
 & \quad \sum\limits_{k \in K_1} \sum\limits_{h \in P_{0,a_k}} <a_k, \tilde{\lambda}_k \cdot (w_h / W_{0,a_k}), (\tilde{y}_k \cdot (w_h / W_{0,a_k}), \tilde{b}_k)>.(E_h \parallel_S E_k) \; + \\
 & \quad \sum\limits_{k \in P'_0} \sum\limits_{h \in P_{1,a_k}} <a_k, *_{\max(l_k, l_h), (w_k / W_{0,a_k}) \cdot (w_h / W_{0,a_k}) \cdot N_{a_k}}, (*,*)>.(E_k \parallel_S E_h)
\end{array}
$$

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 $j \in \{0,1\}$

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

Table 2
Axiomatization of $\sim_{\mathrm{MB}_1}$

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 observe that axiom $(\mathcal{A}_4)_1$ is exactly the rule we wanted our equivalence to satisfy, while axiom $(\mathcal{A}_5)_1$ establishes that lower priority passive actions of a given type can be left out.

We conclude by pointing out that all the results above smoothly extend to an arbitrary length n of the sequences of pairs of rewards within the actions. In particular, the set of axioms $\mathcal{A}_n$ is a trivial extension of $\mathcal{A}_1$.

5.2 Modeling the Performability of a Queueing System

We now report an example of application of the Markovian bisimulation equivalence of order 1. The performance of computing and communicating systems is often degradable because internal or external faults can reduce the quality of the delivered service even though that service remains proper according to its specification. It is therefore important to measure their ability to perform, or *performability*, at different accomplishment levels specifying the extent to which a system is faulty, i.e. which resources are faulty and, among them, which ones have failed, which ones are being recovered, and which ones contain latent faults [27]. From the modeling point of view, we would like to be able to describe both performance and dependability within a single model. On the other hand, this results in problems from the analysis standpoint such as largeness, caused by the presence of several resources working in parallel possibly at different operational levels, and stiffness, originated from the large difference of performance related event rates and rare failure related event rates implying numerical instability. As recognized in [34], this leads to a natural hierarchy of models: a higher level dependability model and a set of as many lower level performance models as there are states in the higher level model. This stems from the fact that the rate of occurrence of failure and repair events is smaller than the rate of occurrence of performance related events, hence the system achieves a quasi steady state w.r.t. the performance related events between successive occurrences of failure or repair events [15]. This means that the system can be characterized by weighing these quasi steady state performance measures by the probabilities of the corresponding states of the higher level model.

We show that $\sim_{\mathrm{MB}_1}$ can be used while building the hierarchy of models of [34] to correctly manipulate the lower level models, so that they are translated into equivalent models whose solution is well known. Let us consider a QS $M/M/n/n+q$ whose servers can fail and be repaired, where the arrival rate is λ, the service rates are μ_i $(1 \leq i \leq n)$, the failure rates are ϕ_i $(1 \leq i \leq n)$, and the repair rates are ρ_i $(1 \leq i \leq n)$, with ϕ_i and ρ_i much smaller than λ and μ_i:

$$\begin{aligned}
FRQS_{M/M/n/n+q} &\triangleq Arrivals \parallel_{\{a\}} (Queue_0 \parallel_D Servers) \\
D &= \{d_i \mid 1 \leq i \leq n\} \\
Arrivals &\triangleq \,<a, \lambda, (0,0)>.Arrivals \\
Queue_0 &\triangleq \,<a, *_{1,1}, (*,*)>.Queue_1 \\
Queue_h &\triangleq \,<a, *_{1,1}, (*,*)>.Queue_{h+1} + \\
&\quad \sum_{i=1}^{n} <d_i, *_{1,1}, (*,*)>.Queue_{h-1}, \;\; 1 \leq h \leq q-1 \\
Queue_q &\triangleq \sum_{i=1}^{n} <d_i, *_{1,1}, (*,*)>.Queue_{q-1} \\
Servers &\triangleq S_1 \parallel_{\emptyset} S_2 \parallel_{\emptyset} \ldots \parallel_{\emptyset} S_n \\
S_i &\triangleq \,<d_i, \infty_{1,1}, (0,0)>.S'_i, \;\; 1 \leq i \leq n \\
S'_i &\triangleq \,<s_i, \mu_i, (0,1)>.S_i + \\
&\quad <f_i, \phi_i, (0,0)>.<r_i, \rho_i, (0,0)>.S'_i, \;\; 1 \leq i \leq n
\end{aligned}$$

where a stands for arrival, d_i stands for delivery, s_i stands for service, f_i stands for failure, and r_i stands for repair. Note that in terms S_i and S'_i actions d_i have been modeled as immediate, in that irrelevant from the performance standpoint, and actions s_i have been given bonus reward 1, since we are interested in computing the throughput of the system, i.e. the number of customers served per unit of time.

Now, since the monolithic model above causes largeness and stiffness problems during its analysis, we build the hierarchy of models proposed in [34] to facilitate the analysis. First we recognize that the higher level dependability model, i.e. the failure-repair model, can be represented as follows:

$$\begin{aligned}
FR &\triangleq FR_1 \parallel_{\emptyset} FR_2 \parallel_{\emptyset} \ldots \parallel_{\emptyset} FR_n \\
FR_i &\triangleq \,<f_i, \phi_i, (0,0)>.<r_i, \rho_i, (0,0)>.FR_i, \;\; 1 \leq i \leq n
\end{aligned}$$

and can be efficiently studied since it trivially admits a product form solution, i.e. the stationary probability of a given state of $\mathcal{M}[\![FR]\!]$ is the product of the stationary probabilities of the related states of $\mathcal{M}[\![FR_i]\!]$ for $1 \leq i \leq n$. Each state of FR determines the set I of operational servers and the set J of failed servers, with $I \cup J = \{1, \ldots, n\}$ and $I \cap J = \emptyset$, so that the corresponding lower level performance model is given by:

$$\begin{aligned}
FRQS_{M/M/n/n+q,I,J} &\triangleq FRQS_{M/M/n/n+q} \parallel_{D_J \cup F_I} \underline{0} \\
D_J &= \{d_j \mid j \in J\} \\
F_I &= \{f_i \mid i \in I\}
\end{aligned}$$

The effect of the synchronization with $\underline{0}$ is that only operational servers can receive customers (D_J) and these servers cannot fail (F_I). It is easily seen that $FRQS_{M/M/n/n+q,I,J}$ is equivalent via $\sim_{\mathrm{MB}_1}$ to $QS_{M/M/|I|/|I|+q,I}$ described below:

$$
\begin{aligned}
QS_{M/M/|I|/|I|+q,I} &\triangleq \mathit{Arrivals} \parallel_{\{a\}} (\mathit{Queue}_{I,0} \parallel_{D_I} \mathit{Servers}_I) \\
D_I &= \{d_i \mid i \in I\} \\
\mathit{Arrivals} &\triangleq <a, \lambda, (0,0)>.\mathit{Arrivals} \\
\mathit{Queue}_{I,0} &\triangleq <a, *_{1,1}, (*,*)>.\mathit{Queue}_{I,1} \\
\mathit{Queue}_{I,h} &\triangleq <a, *_{1,1}, (*,*)>.\mathit{Queue}_{I,h+1} + \\
&\quad \sum_{i \in I} <d_i, *_{1,1}, (*,*)>.\mathit{Queue}_{I,h-1}, \;\; 1 \leq h \leq q-1 \\
\mathit{Queue}_{I,q} &\triangleq \sum_{i \in I} <d_i, *_{1,1}, (*,*)>.\mathit{Queue}_{I,q-1} \\
\mathit{Servers}_I &\triangleq S_{i_1} \parallel_\emptyset S_{i_2} \parallel_\emptyset \ldots \parallel_\emptyset S_{i_{|I|}}, \;\; \{i_1, i_2, \ldots, i_{|I|}\} = I \\
S_i &\triangleq <d_i, \infty_{1,1}, (0,0)>.<s_i, \mu_i, (0,1)>.S_i, \;\; i \in I
\end{aligned}
$$

Since the servers of $QS_{M/M/|I|/|I|+q,I}$ are subject neither to failures nor to repairs, the manipulations above preserve the properties of the system under study and give rise to a model whose solution is well known in the literature [24]. The overall throughput is finally obtained as the weighted sum of the throughputs of every lower level model, where the product form stationary probabilities of the higher level model are used as weights.

6 Comparing Mutual Exclusion Algorithms

The mutual exclusion problem is the problem of managing access to a single indivisible resource that can only support one user at a time. Alternatively, it can be viewed as the problem of ensuring that certain portions of program code are executed within critical sections, where no two programs are permitted to be in critical sections at the same time.

In this section we consider six mutual exclusion algorithms taken from [26]: Dijkstra, Peterson, tournament, Lamport, Burns, and ticket, and we compare their performance by evaluating the corresponding mean numbers of accesses per time unit to the critical section and to the shared control variables. The contribution of this case study is to show that $\mathrm{EMPA}_{\mathrm{gr}_n}$ constitutes a valid support to the analysis of the performance of distributed algorithms in the average case. This is important because in the literature only lower and upper bounds are usually provided.

In the rest of this section, first we briefly describe the software tool used to conduct the case study, then we provide the $\mathrm{EMPA}_{\mathrm{gr}_n}$ model of the Dijkstra algorithm only, and finally we show the results of the comparison. The structure of the $\text{EMPA}_{\text{gr}_n}$ models of the other five algorithms are similar to that of the Dijkstra algorithm; the interested reader is referred to [5], where each passive action should be read as having priority level and weight equal to one.

6.1 An Overview of TwoTowers

The case study has been conducted with TwoTowers [6,5], a software tool for modeling and analyzing functional and performance properties of computer, communication and software systems described in $\text{EMPA}_{\text{gr}_n}$. As shown in Fig. 4, TwoTowers is composed of a graphical user interface, a compiler, a functional analyzer, a performance analyzer, and an integrated analyzer.

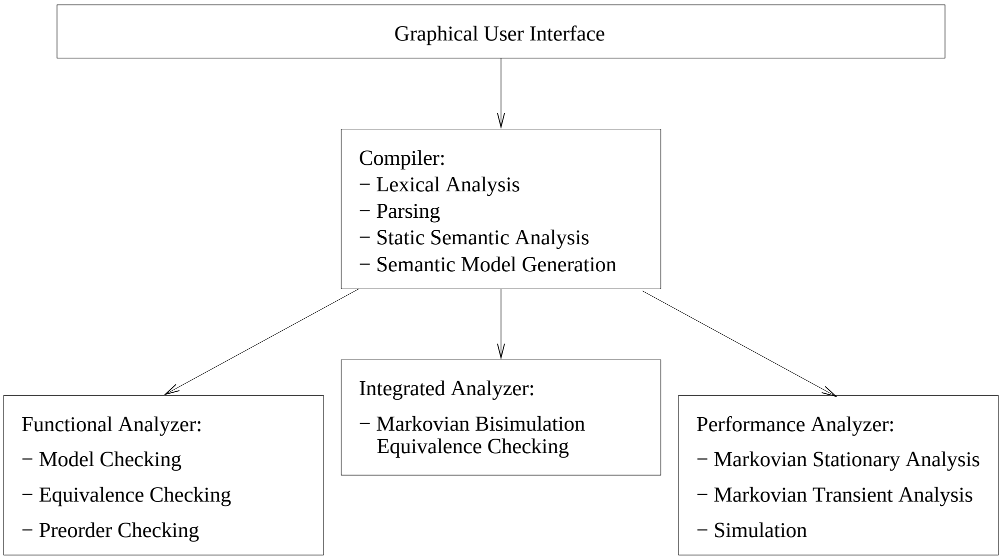


Fig. 4. Architecture of TwoTowers

The graphical user interface allows the user to edit the specifications of the systems in $\text{EMPA}_{\text{gr}_n}$, compile them, and run the various analysis routines. Additionally, it permits to edit the specifications of functional requirements and QoS metrics for the systems under investigation.

The compiler is in charge of parsing $\text{EMPA}_{\text{gr}_n}$ specifications and pinpointing lexical, syntactical and static semantical errors. If a specification is correct, the compiler can produce the semantic model (integrated LTS, functional LTS, CTMC/DTMC) on which further analysis is based.

The integrated analyzer conducts those investigations that require both functional and performance information. It thus contains a routine to check two correct $\text{EMPA}_{\text{gr}_n}$ specifications for Markovian bisimulation equivalence of order n.

The functional analyzer takes care of verifying that certain functional requirements are satisfied by the functional LTS derived from a correct $\mathrm{EMPA}_{\mathrm{gr}_n}$ specification. This is achieved by interfacing TwoTowers with the Concurrency Workbench of New Century (CWB-NC) [14], thereby providing support for model checking in the μ-calculus or CTL [12], equivalence checking (strong and weak bisimulation equivalences [28] and may and must testing equivalences [16]), and preorder checking (may and must testing preorders [16]).

Finally, the performance analyzer computes certain performance measures on the CTMC/DTMC derived from a correct $\mathrm{EMPA}_{\mathrm{gr}_n}$ specification. This can be done via numerical analysis through the Markov Chain Analyzer (MarCA) [32] or via simulation. As far as the attachment of rewards is concerned, we observe that in TwoTowers it is separated from the system specification for operational convenience. In other words, two distinguished files must be prepared: one with the $\mathrm{EMPA}_{\mathrm{gr}_0}$ specification of the system and one with the specification of the performance measures, each under the form $id = (reward\_list)$ where id is the measure identifier and $reward\_list$ is a list of reward assignments of the form "$a\, y\, b$", which means that every nonpassive action with type a must be given yield reward y and bonus reward b for the measure under specification. This way of specifying rewards, which is equivalent to attaching sequences of reward pairs to actions, has two advantages. On the one hand, the system specification is not obfuscated by performance measure related details. On the other hand, the specification of performance measures can be easily updated without changing the specification of the system they refer to and can be reused for other system specifications.

6.2 Dijkstra Algorithm

This algorithm makes use of two shared variables for controlling the access to the critical section. Variable $turn$ (an integer in $\{1, \ldots, n\}$) indicates which program owns the turn to access the critical section, while $flag(i)$, $1 \leq i \leq n$, denotes the stage (an integer in $\{1, 2, 3\}$) of program i in accessing the critical section.

The Dijkstra algorithm works as follows. In the first stage, program i starts by setting $flag(i)$ to 1 and then repeatedly checks $turn$ until it is equal to i. If not, and if the current owner of the turn is seen not to be currently active ($flag(turn) = 0$), program i sets $turn$ to i. Once having seen $turn = i$, program i moves on to the second stage. In this stage, program i sets $flag(i)$ to 2 and then checks to see that no other program j has $flag(j) = 2$. If the check completes successfully, program i goes to its critical section, otherwise it returns to the first stage. Upon leaving the critical section, program i lowers $flag(i)$ to 0.

The Dijkstra algorithm can be modeled with $\text{EMPA}_{\text{gr}_0}$ as follows:

$$\begin{aligned}
\mathit{DijkstraME}_n &\stackrel{\Delta}{=} (\mathit{Program}_1 \parallel_\emptyset \ldots \parallel_\emptyset \mathit{Program}_n) \parallel_S \\
&\quad ((\mathit{Flag0}_1 \parallel_\emptyset \ldots \parallel_\emptyset \mathit{Flag0}_n) \parallel_R \\
&\quad \mathit{Turn1}) \\
S &= \{\mathit{set_flag_to_0}_i, \mathit{set_flag_to_1}_i, \mathit{set_flag_to_2}_i, \mathit{modify_turn}_i, \\
&\quad \mathit{read_turn_eq_i}, \mathit{turn_eq_i}, \mathit{turn_neq_i}, \\
&\quad \mathit{read_flag_turn_eq_0}_i, \mathit{flag_turn_eq_0}_i, \mathit{flag_turn_neq_0}_i, \\
&\quad \mathit{read_flag_eq_2}_i, \mathit{flag_eq_2}_i, \mathit{flag_neq_2}_i \mid 1 \le i \le n\} \\
R &= \{\mathit{read_flag_eq_0}_i, \mathit{flag_eq_0}_i, \mathit{flag_neq_0}_i \mid 1 \le i \le n\}
\end{aligned}$$

Let us denote by exec_i ($\mathit{exec\_cs}_i$) the action type describing the fact that $\mathit{Program}_i$ is executing outside (inside) the critical section. These actions are assumed to be exponentially timed with rate λ_i and δ_i, respectively. All the other actions related to reading or writing shared control variables are assumed to have the same duration for every program (the convention is that they are exponentially timed with rate 1).

$\mathit{Program}_i$ can be modeled as follows:

$$\begin{aligned}
\mathit{Program}_i &\stackrel{\Delta}{=} <\mathit{exec}_i, \lambda_i>.\mathit{SetFlag1}_i \\
\mathit{SetFlag1}_i &\stackrel{\Delta}{=} <\mathit{set_flag_to_1}_i, 1>.\mathit{TestTurn}_i \\
\mathit{TestTurn}_i &\stackrel{\Delta}{=} <\mathit{read_turn_eq_i}, 1>. \\
&\qquad (<\mathit{turn_eq_i}, *_{1,1}>.\mathit{SetFlag2}_i + \\
&\qquad <\mathit{turn_neq_i}, *_{1,1}>.\mathit{TestFlag}_i) \\
\mathit{TestFlag}_i &\stackrel{\Delta}{=} <\mathit{read_flag_turn_eq_0}_i, 1>. \\
&\qquad (<\mathit{flag_turn_eq_0}_i, *_{1,1}>.<\mathit{modify_turn}_i, 1>.\mathit{SetFlag2}_i + \\
&\qquad <\mathit{flag_turn_neq_0}_i, *_{1,1}>.\mathit{TestTurn}_i) \\
\mathit{SetFlag2}_i &\stackrel{\Delta}{=} <\mathit{set_flag_to_2}_i, 1>.\mathit{TestFlag2}_{\{1,\ldots,n\}-\{i\},i}
\end{aligned}$$

$$
\begin{aligned}
\mathit{TestFlag2}_{K,i} \triangleq \sum_{k \in K} &<\mathit{read\_flag\_eq\_2}_k, 1>.\\
&(<\mathit{flag\_eq\_2}_k, *_{1,1}>.\mathit{SetFlag1}_i +\\
&\ <\mathit{flag\_neq\_2}_k, *_{1,1}>.\mathit{TestFlag2}_{K-\{k\},i}),\ 1 < |K| < n\\
\mathit{TestFlag2}_{\{k\},i} \triangleq &<\mathit{read\_flag\_eq\_2}_k, 1>.\\
&(<\mathit{flag\_eq\_2}_k, *_{1,1}>.\mathit{SetFlag1}_i +\\
&\ <\mathit{flag\_neq\_2}_k, *_{1,1}>.\mathit{CriticalSection}_i),\ 1 \le k \le n\\
\mathit{CriticalSection}_i \triangleq &<\mathit{exec\_cs}_i, \delta_i>.<\mathit{set\_flag\_to\_0}_i, 1>.\mathit{Program}_i
\end{aligned}
$$

FlagV_i, where V denotes the value of the flag, can be modeled as follows:

$$
\begin{aligned}
\mathit{Flag0}_i \triangleq &<\mathit{set\_flag\_to\_1}_i, *_{1,1}>.\mathit{Flag1}_i +\\
&<\mathit{read\_flag\_eq\_0}_i, *_{1,1}>.<\mathit{flag\_eq\_0}_i, \infty_{1,1}>.\mathit{Flag0}_i +\\
&<\mathit{read\_flag\_eq\_2}_i, *_{1,1}>.<\mathit{flag\_neq\_2}_i, \infty_{1,1}>.\mathit{Flag0}_i\\
\mathit{Flag1}_i \triangleq &<\mathit{set\_flag\_to\_2}_i, *_{1,1}>.\mathit{Flag2}_i +\\
&<\mathit{read\_flag\_eq\_0}_i, *_{1,1}>.<\mathit{flag\_neq\_0}_i, \infty_{1,1}>.\mathit{Flag1}_i +\\
&<\mathit{read\_flag\_eq\_2}_i, *_{1,1}>.<\mathit{flag\_neq\_2}_i, \infty_{1,1}>.\mathit{Flag1}_i\\
\mathit{Flag2}_i \triangleq &<\mathit{set\_flag\_to\_0}_i, *_{1,1}>.\mathit{Flag0}_i +\\
&<\mathit{set\_flag\_to\_1}_i, *_{1,1}>.\mathit{Flag1}_i +\\
&<\mathit{read\_flag\_eq\_0}_i, *_{1,1}>.<\mathit{flag\_neq\_0}_i, \infty_{1,1}>.\mathit{Flag2}_i +\\
&<\mathit{read\_flag\_eq\_2}_i, *_{1,1}>.<\mathit{flag\_eq\_2}_i, \infty_{1,1}>.\mathit{Flag2}_i
\end{aligned}
$$

Note that testing shared control variables is modeled through two actions: reading and outcome (either true or false). The duration of the testing operation is associated with the former action, while the latter is described as immediate.

Finally, *Turn i* can be modeled as follows:

$$
\begin{aligned}
\mathit{Turn}\ i \triangleq &\sum_{j=1}^{n} <\mathit{modify\_turn}_j, *_{1,1}>.\mathit{Turn}\ j +\\
&\sum_{j=1}^{n} <\mathit{read\_flag\_turn\_eq\_0}_j, *_{1,1}>.<\mathit{read\_flag\_eq\_0}_i, \infty_{1,1}>.\\
&\quad (<\mathit{flag\_eq\_0}_i, *_{1,1}>.<\mathit{flag\_turn\_eq\_0}_j, \infty_{1,1}>.\mathit{Turn}\ i +\\
&\quad\ <\mathit{flag\_neq\_0}_i, *_{1,1}>.<\mathit{flag\_turn\_neq\_0}_j, \infty_{1,1}>.\mathit{Turn}\ i) +\\
&<\mathit{read\_turn\_eq\_i}, *_{1,1}>.<\mathit{turn\_eq\_i}, \infty_{1,1}>.\mathit{Turn}\ i +\\
&\sum_{j=1 \land j \neq i}^{n} <\mathit{read\_turn\_eq\_j}, *_{1,1}>.<\mathit{turn\_neq\_j}, \infty_{1,1}>.\mathit{Turn}\ i
\end{aligned}
$$

We now compare the performance of the six mutual exclusion algorithms we have selected. The performance measures we are interested in are the mean numbers of accesses per time unit to the critical section and to the shared variables. They are computed on the Markovian semantic model of each algorithm; the size of such models in the case of two programs is shown in Table 3. The former performance index represents the throughput of the algorithm and has been specified by assigning bonus reward 1 to every action with type $exec\_cs_i$. The latter performance index represents instead the delay introduced by the algorithm in order to guarantee the mutual exclusive access to the critical section and has been specified by assigning bonus reward 1 to every action related to reading or writing shared control variables. As an example, for $DijkstraME_2$ the following reward specification file has been provided:

$$
\begin{array}{lll}
\hat{}\,critical\_section\_accesses = (& exec\_cs_1 & 0\ 1 \\
 & exec\_cs_2 & 0\ 1\ ) \\
\hat{}\,shared\_variable\_accesses = (& set\_flag\_to\_0_1 & 0\ 1 \\
 & set\_flag\_to\_1_1 & 0\ 1 \\
 & set\_flag\_to\_2_1 & 0\ 1 \\
 & read\_turn\_eq\_1 & 0\ 1 \\
 & read\_flag\_turn\_eq\_0_1 & 0\ 1 \\
 & modify\_turn_1 & 0\ 1 \\
 & read\_flag\_eq\_2_1 & 0\ 1 \\
 & set\_flag\_to\_0_2 & 0\ 1 \\
 & set\_flag\_to\_1_2 & 0\ 1 \\
 & set\_flag\_to\_2_2 & 0\ 1 \\
 & read\_turn\_eq\_2 & 0\ 1 \\
 & read\_flag\_turn\_eq\_0_2 & 0\ 1 \\
 & modify\_turn_2 & 0\ 1 \\
 & read\_flag\_eq\_2_2 & 0\ 1\ )
\end{array}
$$

We report in Fig. 5 and 6 the curves concerning the mean number of accesses per time unit to the critical section and to the shared variables, respectively, for each of the six algorithms in the case of two programs. The curves are plotted for different values of the ratio $\delta_i^{-1}/\lambda_i^{-1}$ of the average time spent inside the critical section to the average time spent outside the critical section.

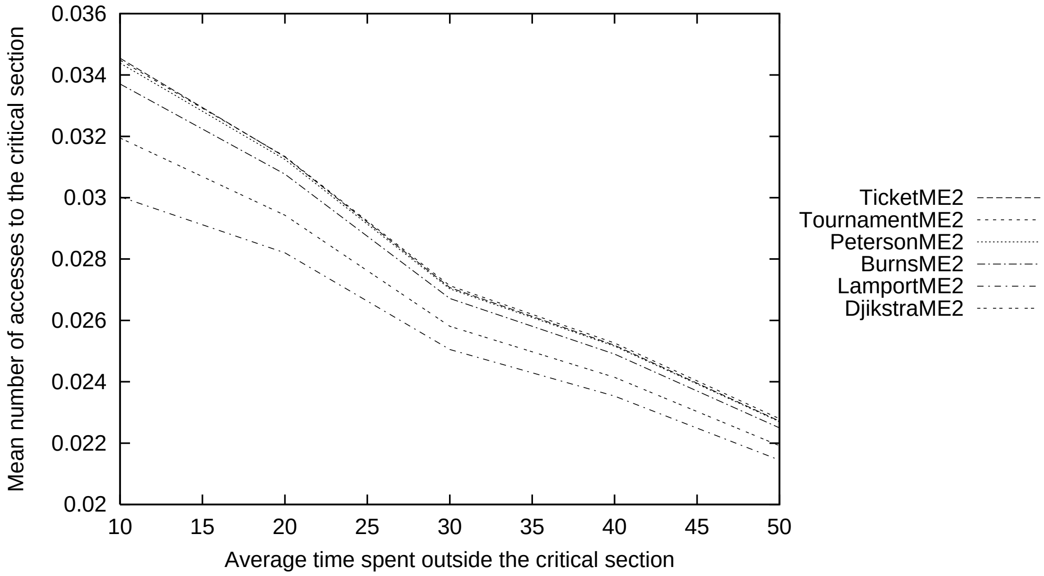


Fig. 5. Mean number of accesses per time unit to the critical section

Such values are those obtained by letting $\delta_i = 0.04$ (corresponding to 25 time units) and varying λ_i from 0.1 (10 time units) to 0.02 (50 time units). The two figures show that the throughput and the delay of each algorithm decrease as the above mentioned ratio decreases, i.e. as the frequency with which programs want to access the critical section decreases. Moreover, the two figures confirms that the Peterson algorithm and the tournament algorithm behave the same if two programs only are involved. Finally, the figures indicate that, in a scenario with only two programs, the six algorithms have comparable performance, with the ticket algorithm achieving the higher throughput and introducing the lower delay. Note that such a comparison is not based on lower or upper bounds for the performance of the algorithms, as usually happens, but on average values, so that it provides additional information to choose the most appropriate algorithm for a specific case.

algorithm	states	transitions
$DjikstraME_2$	148	296
$PetersonME_2$	49	98
$TournamentME_2$	49	98
$LamportME_2$	346	692
$BurnsME_2$	44	88
$TicketME_2$	72	144

Table 3
Size of the Markovian semantic models of the six mutual exclusion algorithms

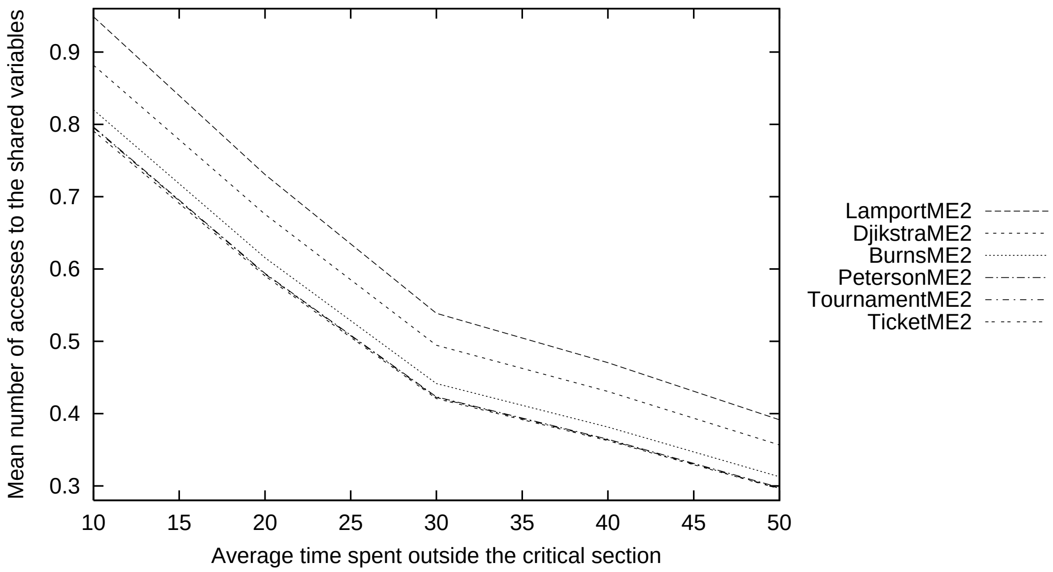


Fig. 6. Mean number of accesses per time unit to the shared variables

7 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 made a further step by introducing a way to express performance measures, in order to allow the modeler to capture the QoS metrics of interest. The proposed method consists of specifying performance measures by attaching sequences of yield and bonus reward pairs to process algebra actions, thus resulting in a family of process algebras $\mathrm{EMPA}_{\mathrm{gr}_n}$. We have shown that this method achieves an acceptable expressive power and ease of use and, most importantly, allows performance measure sensitive congruences to be defined.

Prior to our algebra based method, a different method was proposed in [10] to specify reward based performance measure in a process algebraic framework. Such a method was inspired by the preliminary work in [19], where it is proposed to use a temporal logic formula to partition the semantic model of a Markovian process algebra description in such a way that each part exhibits or not a particular behavior formalized through the logic formula itself. The idea is to define a reward structure as a function of such a partition, which associates a unique (yield) reward to all the states of the same class.

In [10] this logic based method is further elaborated on. The process of specifying performance measures is split into two stages. The first stage consists of defining a reward specification, which is a pair composed of a Hennessy-Milner logic formula [28] and an expression: every state satisfying the modal logic formula is assigned as a yield reward the value of the expression, which may consist of the usual arithmetic operators applied to real numbers, action rates, and special variables storing previously or currently assigned rewards. The second stage, instead, consists of defining a reward attachment that determines at which process derivatives a particular reward specification is evaluated. Such a method addresses only yield rewards and stationary measures.

If we compare the algebra based method and the logic based method w.r.t. the four criteria of Sect. 3, we see that in general the algebra based method is less powerful than the logic based method, as rewards are simply expressed as real numbers in the former method and particular behaviors formalizable through logic formulas cannot be captured (see e.g. the specification of the utilization for $QS^{ro}_{M/M/n/n}$ in Sect. 3), but easier to learn and use, as it does not require the knowledge of any extra formalism to specify rewards (consider e.g. the logic formula necessary to specify the mean number of customers for $QS^{ro}_{M/M/n/n}$ in Sect. 3). The logic based method is more time consuming than the algebra based method, as it would require in principle an additional scan of the state space in order to check states against the modal logic formulas in order to attach yield rewards: fortunately, model checking on the fly should be possible. Finally, an equational characterization is possible in the case of the algebra based method but not in the case of the logic based method, hence with the latter method a compositional, performance measure preserving term manipulation cannot be conducted. Because of the results about the relationship between bisimulation equivalence and Hennessy-Milner logic formula satisfiability [28], the logic based method only guarantees that if two terms are related by $\sim_{\mathrm{MB}_0}$ then equivalent states get the same yield reward, hence the performance index under study has the same value for the two terms. The converse does not hold: if two terms satisfy a given set of Hennessy-Milner logic formulas, then the two terms may be bisimulation equivalent but not necessarily Markovian bisimulation equivalent, which means that the value of the specified performance measures for the two terms may be different.

Recently, in [11] the lack of equational characterization for the logic based method has been remedied. This has been accomplished by using a Markovian modal logic inspired by the probabilistic modal logic of [25], instead of the Hennessy-Milner logic, and by showing that two terms satisfy the same Markovian modal logic formulas if and only if they are Markovian bisimulation equivalent. As a consequence, Markovian bisimulation equivalent states get the same reward according to the approach of [11]. In this respect, the algebra based method turns out to be more flexible, as it allows different rewards to be associated with Markovian bisimulation equivalent states, hence the need for the previously presented family of Markovian bisimulation equivalences that take rewards into account. Additionally, in [11] the ease of use of the logic based method has been enhanced by proposing a high level language for enquiring about the stationary performance characteristics possessed by a process term. Such a language, whose formal underpinning is constituted by the Markovian modal logic (which thus becomes transparent to the user), is based on the combination of the standard mathematical notation (arithmetical, relational and logical operators as well as probability), a notation based on the Markovian bisimulation equivalence which is useful to focus queries directly on states, and a notation expressing the potential to perform an action of a given type.

We conclude by mentioning that, as far as the problem of specifying performance measures is concerned, more recently a different logic based approach has been proposed in [3], which has a relationship with the Markovian bisimulation equivalence. Unlike [10,11], where the logic is used to single out those states to which a certain yield reward must be attached, this new approach relies on the logic CSL, a continuous stochastic time variant of CTL [12], to inquiry (similarly to the high level language of [11]) about the value of stationary and transient performability measures of a process term. Based on the observation that the progress of time can be regarded as the earning of reward, a variant of CSL called CRL has been subsequently proposed in [2], where yield rewards are assumed to be attached to the states. A drawback of this approach is that the way in which yield rewards should be specified and attached to the states is not provided. We thus envision that this novel approach may be profitably integrated with the algebra based method of this paper or the logic based method of [11].

References

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

[2] C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen, *"On the Logical Characterisation of Performability Properties"*, in Proc. of the *27th Int. Coll. on Automata, Languages and Programming (ICALP '00)*, LNCS 1853:780-792, Geneve (Switzerland), 2000

[3] C. Baier, J.-P. Katoen, H. Hermanns, *"Approximate Symbolic Model Checking of Continuous Time Markov Chains"*, in Proc. of the *10th Int. Conf. on Concurrency Theory (CONCUR '00)*, LNCS 1664:146-162, Eindhoven (The Netherlands), 1999

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

[5] M. Bernardo, *"Theory and Application of Extended Markovian Process Algebra"*, Ph.D. Thesis, University of Bologna (Italy), 1999 (`http://www.di.unito.it/~bernardo/`)

[6] M. Bernardo, W.R. Cleaveland, S.T. Sims, W.J. Stewart, *"TwoTowers: A Tool Integrating Functional and Performance Analysis of Concurrent Systems"*, in Proc. of the *IFIP Joint Int. Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing and Verification (FORTE/PSTV '98)*, Kluwer, 457-467, Paris (France), 1998

[7] 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)

[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 (extended abstract in Proc. of the *1st Int. Workshop on Models for Time Critical Systems (MTCS '00)*, Electronic Notes in Theoretical Computer Science 39(3), State College (PA), 2000)

[9] G. Ciardo, J. Muppala, K.S. Trivedi, *"On the Solution of GSPN Reward Models"*, in Performance Evaluation 12:237-253, 1991

[10] G. Clark, *"Formalising the Specification of Rewards with PEPA"*, in Proc. of the *4th Workshop on Process Algebras and Performance Modelling (PAPM '96)*, CLUT, pp. 139-160, Torino (Italy), 1996

[11] G. Clark, S. Gilmore, J. Hillston, *"Specifying Performance Measures for PEPA"*, in Proc. of the *5th AMAST Int. Workshop on Formal Methods for Real Time and Probabilistic Systems (ARTS '99)*, LNCS 1601:211-227, Bamberg (Germany), 1999

[12] E.M. Clarke, O. Grumberg, D.A. Peled, *"Model Checking"*, MIT Press, 1999

[13] W.R. Cleaveland, G. Lüttgen, V. Natarajan, *"Priority in Process Algebras"*, in *"Handbook of Process Algebra"*, Elsevier, 2001

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

[15] P.J. Courtois, *"Decomposability: Queueing and Computer System Applications"*, Academic Press, 1977

[16] R. De Nicola, M.C.B. Hennessy, *"Testing Equivalences for Processes"*, in Theoretical Computer Science 34:83-133, 1983

[17] 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

[18] B.R. Haverkort, K.S. Trivedi, *"Specification Techniques for Markov Reward Models"*, in Discrete Event Dynamic Systems: Theory and Applications 3:219-247, 1993

[19] H. Hermanns, *"Leistungsvorhersage von Verhaltensbeschreibungen mittels Temporaler Logik"*, contribution to the *GI/ITG Fachgespraech '95: Formale Beschreibungstechniken fuer Verteilte Systeme*, 1995

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

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

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

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

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

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

[26] N.A. Lynch, *"Distributed Algorithms"*, Morgan Kaufmann Publishers, 1996

[27] J.F. Meyer, *"Performability: A Retrospective and some Pointers to the Future"*, in Performance Evaluation 14:139-156, 1992

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

[29] M.A. Qureshi, W.H. Sanders, *"Reward Model Solution Methods with Impulse and Rate Rewards: An Algorithm and Numerical Results"*, in Performance Evaluation 20:413-436, 1994

[30] W.H. Sanders, J.F. Meyer, *"A Unified Approach for Specifying Measures of Performance, Dependability, and Performability"*, in Dependable Computing and Fault Tolerant Systems 4:215-237, 1991

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

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

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

[34] K.S. Trivedi, J.K. Muppala, S.P. Woolet, B.R. Haverkort, *"Composite Performance and Dependability Analysis"*, in Performance Evaluation 14:197-215, 1992

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

[36] 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