An Algebra-Based Method to Associate Rewards with EMPA Terms

Marco Bernardo

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

Abstract. We present a simple method to associate rewards with terms of the stochastic process algebra EMPA in order to make the specification and the computation of performance measures easier. The basic idea behind this method is to specify rewards within actions of EMPA terms, so it substantially differs from methods based on modal logic. The main motivations of this method are its ease of use as well as the possibility of defining a notion of equivalence which relates terms having the same reward, thus allowing for simplification without altering the performance index. We prove that such an equivalence is a congruence finer than the strong extended Markovian bisimulation equivalence, and we present its axiomatization.

1 Introduction

A commonly used method to specify steady-state performance measures for Markovian models is based on *rewards* [6]. The basic idea is that a number describing a reward (or weight) is attached to every state of the Markovian model, and the performance index is defined as the weighted sum of the steady-state probabilities of the states of the Markovian model.

So far the specification of performance measures in the field of stochastic process algebras has received a scarce attention. The main negative consequence is that the whole Markovian model underlying a given term has to be manually scanned by the designer in order to assign rewards to states.

Recently, in [3] a technique to formally specify rewards for the stochastic process algebra PEPA [5] has been proposed. The idea is to express rewards by means of the Hennessy-Milner logic [4]: a logical formula is specified together with an arithmetical expression, and every state satisfying the formula is assigned the reward specified by means of the arithmetical expression. We shall call such a method *logic-based*.

The idea of describing rewards through a modal logic seems to be quite adequate because modal logic formulae make assertions about changing state, hence they constitute an adequate link between algebraic terms, which describe the behavior of concurrent systems, and rewards, which are associated with states.

In this paper we propose a different way to associate rewards with terms of stochastic process algebras. The idea is not to use a separate formalism in order

to specify rewards: they are directly described within the actions forming the algebraic terms. This method, which we shall call *algebra-based*, closely resembles the manual method consisting of associating rewards while scanning the state space of the Markovian model: the difference is that in the algebra-based method the algebraic term, which is much more compact than its underlying state space, is scanned and the appropriate actions are assigned a reward. The algebra-based method could be convenient due to its ease of use, since the designer is not forced to know the modal logic formalism, its low computational cost, as rewards are associated with states during the construction of the semantic models without the need to check for a modal logic formula, and the possibility of defining a congruence which equates terms having the same reward, thereby allowing for simplification without altering the performance measure.

The purpose of this paper is to extend the theory developed for the stochastic process algebra EMPA [1, 2] in order to deal with rewards according to the algebra-based method. In Sect. 2 we show that several performance measures can be derived using the algebra-based method. In Sect. 3 we introduce the syntax and the semantics for EMPA augmented with rewards. In Sect. 4 we define an equivalence which relates two terms if they have the same reward, we prove that such an equivalence is a congruence strictly contained in the strong extended Markovian bisimulation equivalence, and we present its axiomatization. In Sect. 5 we report some concluding remarks.

2 Deriving Performance Measures

In this section we show by means of an example that the algebra-based method we are going to introduce, though less powerful in general than the logic-based method proposed in [3], allows the designer to easily specify several steady-state performance measures frequently occurring in practice such as those identified in [3]: 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. the fraction of time during which a server is busy).

The example we consider is taken from queueing theory, and concerns a queueing system $M/M/n/n$ with arrival rate λ and service rate μ [7]. Such a queueing system represents a service center composed of n independent servers, such that the customer interarrival time is exponentially distributed with rate λ and the service time of each server is exponentially distributed with rate μ. The queueing system at hand can be given two different descriptions with EMPA: a state-oriented description where the focus is on the state of the set of servers (intended as the number of servers that are currently busy), and a resource-oriented description where the servers are modeled separately [9]. Recalling that "$<a, \tilde{\lambda}>._-$" is the prefix operator where a is the action type and $\tilde{\lambda}$ is the action rate (a positive real number in the case of exponentially timed actions, $\infty_{l,w}$ in the case of prioritized weighted immediate actions, and $*$ in the case of passive actions), "$_- + _-$" is the alternative composition operator, and "$_- \|_S _-$" is the

parallel composition operator with synchronization set S, the state-oriented description is given by

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

whereas the resource-oriented description is given by

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

In order to highlight the difference between the logic-based method and the algebra-based method for assigning rewards to stochastic process algebra terms, we compute for the queueing system above the mean number of customers in the system. Since every state must be given a reward equal to the number of customers in that state, we proceed as follows:

- In the case of $System^{so}_{M/M/n/n}$, the reward specification used in the logic-based method is $\langle s \rangle tt \Longrightarrow rate(s)/\mu$, i.e. every state having an outgoing transition with type s is given a reward equal to the rate of that transition divided by μ. Using the algebra-based method, every action of the form $<s, h \cdot \mu>$ must be replaced by $<s, h \cdot \mu, h>$ (and any other action must be replaced by a triple with zero reward). Thus, in such a case the two methods are equally simple.
- In the case of $System^{ro}_{M/M/n/n}$, the logic-based method turns out to be more complex because the modal logic formula must somehow count the number of possible consecutive actions with type s that can be executed: as a consequence, the rewards can be specified through the set composed of $\langle s \rangle \neg \langle s \rangle tt \Longrightarrow 1$, $\langle s \rangle \langle s \rangle \neg \langle s \rangle tt \Longrightarrow 2$, ..., $\langle s \rangle \langle s \rangle \ldots \langle s \rangle \neg \langle s \rangle tt \Longrightarrow n$. If we use instead the algebra-based method, all we have to do is to replace every action of the form $<s, \mu>$ with $<s, \mu, 1>$ as we assume that rewards are additive (by analogy with rates of exponentially timed actions and weights of immediate actions), i.e. the reward gained by a state is the sum of the rewards labeling its outgoing transitions. Therefore, in such a case the ease of use of the algebra-based method becomes evident, and it would be even more evident if we considered e.g. a queueing system similar to the previous one where a FIFO queue with a given capacity is introduced in front of the set of servers: since the delivery of a customer from the queue to the server has to be modeled by means of an action, and since actions of type s are interleaved with actions of this kind, the formalization of modal logic formulae that capture the number of customers in the system is really difficult.

To conclude this section, we show that other performance measures for the queueing system above can be easily specified with the algebra-based method, and that this capability depends on the style used to represent the system:

- If we want to compute the throughput of the system, defined as the mean number of customers served per time unit, we have to take into account the rate of actions having type s. As a consequence, in the case of $System^{so}_{M/M/n/n}$ we must replace every action of the form $<s, h \cdot \mu>$ with $<s, h \cdot \mu, h \cdot \mu>$, while in the case of $System^{ro}_{M/M/n/n}$ we must replace every action of the form $<s, \mu>$ with $<s, \mu, \mu>$.
- If we want to compute the mean response time of the system, defined as the mean time spent by the customers in the system, we can exploit Little's law [7] which states that the mean response time of the system is equal to the mean number of customers in the system divided by the customer arrival rate. Therefore, in the case of $System^{so}_{M/M/n/n}$ we must replace every action of the form $<s, h{\cdot}\mu>$ with $<s, h{\cdot}\mu, h/\lambda>$, while in the case of $System^{ro}_{M/M/n/n}$ we must replace every action of the form $<s, \mu>$ with $<s, \mu, 1/\lambda>$.
- If we want to compute the utilization of the system, defined as the fraction of time during which servers are busy, we have to single out states having an outgoing transition labeled with s. Thus, in the case of $System^{so}_{M/M/n/n}$ we must replace every action of the form $<s, h \cdot \mu>$ with $<s, h \cdot \mu, 1>$. We observe that, unlike the logic-based method, in the case of $System^{ro}_{M/M/n/n}$ the algebra-based method cannot be used to determine the utilization of the system due to the additivity assumption: the rate to associate with actions of the form $<s, \mu>$ would be the reciprocal of the number of transitions labeled with s exiting from the same state. Since the main objective 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 as well as particular functions such as the one determining the number of transitions of a given type exiting from 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 $System^{ro}_{M/M/n/n}$, whereas this is not possible for $System^{so}_{M/M/n/n}$. This means that the style [9] used to describe a given system through an algebraic term is strongly related to the possibility of deriving certain performance measures through the algebra-based method.

3 Syntax and Semantics for EMPA$_r$

In this section we extend the syntax and the semantics for EMPA [1, 2] in order to cope with the presence of rewards treated according to the algebra-based method outlined in the previous section: the resulting stochastic process algebra is called $EMPA_r$.

As usual, the building blocks of EMPA$_r$ are actions. Each action is a triple $<a, \tilde{\lambda}, r>$ consisting of the type of the action, the rate of the action and the reward of the action: the third component is new with respect to the structure of EMPA actions. Like in EMPA, actions are divided into external and internal (τ) according to types, while they are classified as exponentially timed, immediate or passive according to rates. Since exponentially timed actions model activities that are relevant from the performance standpoint, nonzero rewards can be assigned only to them. We denote by $AType$ the set of types, by $ARate = \mathbb{R}_+ \cup Inf \cup \{*\}$, with $Inf = \{\infty_{l,w} \mid l \in \mathbb{N}_+ \wedge w \in \mathbb{R}_+\}$, the set of rates, by $AReward = \mathbb{R}$ the set of rewards, and by $Act_r = \{<a, \tilde{\lambda}, r> \in AType \times ARate \times AReward \mid \tilde{\lambda} \in Inf \cup \{*\} \Longrightarrow r = 0\}$ the set of actions. We use $a, b, c, \ldots$ as metavariables for $AType$, $\tilde{\lambda}, \tilde{\mu}, \tilde{\gamma}, \ldots$ for $ARate$, $\lambda, \mu, \gamma, \ldots$ for $\mathbb{R}_+$, and $r, r', r'', \ldots$ for $AReward$. Finally, we denote by $PLevel = \{-1\} \cup \mathbb{N}$ the set of priority levels, and we assume that $* < \lambda < \infty_{l,w}$ for all $\lambda \in \mathbb{R}_+$ and $\infty_{l,w} \in Inf$.

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

Definition 1. The set $\mathcal{L}_r$ of *process terms* of EMPA$_r$ is generated by the following syntax

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

where $L, S \subseteq AType - \{\tau\}$. The set $\mathcal{L}_r$ will be ranged over by $E, F, G, \ldots$. We denote by $\mathcal{G}_r$ the set of guarded and closed terms of $\mathcal{L}_r$. ■

We recall from [1, 2] that the alternative composition operator is parametric in the nature of the choice: the choice is solved according to durations in the case of exponentially timed actions (race policy) and according to priorities and weights in the case of immediate actions (preselection policy), while it is purely nondeterministic in the case of passive actions. We also remind that, concerning the parallel composition operator, a synchronization can occur if and only if the involved actions have the same type belonging to the synchronization set, and at most one of the involved actions is not passive.

The integrated semantics of EMPA$_r$ terms can be defined by exploiting again the idea of potential move: the multiset [1] of the potential moves of a given term is inductively computed, then those potential moves having the highest priority level are selected and appropriately merged. The formal definition is based on the transition relation $\longrightarrow$, which is the least subset of $\mathcal{G}_r \times Act_r \times \mathcal{G}_r$ satisfying the inference rule reported in the first part of Table 1. This rule selects the potential moves having the highest priority level, and then merges together those having the same action type, the same priority level and the same

[1] We use "{|" and "|}" as brackets for multisets, "$\_ \oplus \_$" to denote multiset union, $\mathcal{M}u_{fin}(S)$ ($\mathcal{P}_{fin}(S)$) to denote the collection of finite multisets (sets) over set S, $M(s)$ to denote the multiplicity of element s in multiset M, and $\pi_i(M)$ to denote the multiset obtained by projecting the tuples in multiset M on their i-th component. Thus, e.g., $(\pi_1(PM_2))(<a, *, 0>)$ in the fifth part of Table 1 denotes the multiplicity of tuples of PM_2 whose first component is $<a, *, 0>$.

$$\frac{(<a,\tilde{\lambda},r>,E') \in Melt_r(Select_r(PM_r(E)))}{E \xrightarrow{a,\tilde{\lambda},r} E'}$$

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

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

$PM_r(E/L) = \{\!|\, (<a,\tilde{\lambda},r>,E'/L) \mid (<a,\tilde{\lambda},r>,E') \in PM_r(E) \wedge a \notin L \,|\!\} \oplus$
$\{\!|\, (<\tau,\tilde{\lambda},r>,E'/L) \mid (<a,\tilde{\lambda},r>,E') \in PM_r(E) \wedge a \in L \,|\!\}$

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

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

$PM_r(E_1 \parallel_S E_2) = \{\!|\, (<a,\tilde{\lambda},r>,E_1' \parallel_S E_2) \mid a \notin S \wedge (<a,\tilde{\lambda},r>,E_1') \in PM_r(E_1) \,|\!\} \oplus$
$\{\!|\, (<a,\tilde{\lambda},r>,E_1 \parallel_S E_2') \mid a \notin S \wedge (<a,\tilde{\lambda},r>,E_2') \in PM_r(E_2) \,|\!\} \oplus$
$\{\!|\, (<a,\tilde{\gamma},r>,E_1' \parallel_S E_2') \mid a \in S \wedge$
$(<a,\tilde{\lambda}_1,r_1>,E_1') \in PM_r(E_1) \wedge$
$(<a,\tilde{\lambda}_2,r_2>,E_2') \in PM_r(E_2) \wedge$
$\tilde{\gamma} = Norm_{r,rate}(a,\tilde{\lambda}_1,\tilde{\lambda}_2,PM_r(E_1),PM_r(E_2)) \wedge$
$r = Norm_{r,reward}(a,r_1,r_2,PM_r(E_1),PM_r(E_2)) \,|\!\}$

$PM_r(A) = PM_r(E) \quad$ if $A \stackrel{\Delta}{=} E$

$Select_r(PM) = \{\!|\, (<a,\tilde{\lambda},r>,E) \in PM \mid PL_r(<a,\tilde{\lambda},r>) = -1 \vee$
$\forall (<b,\tilde{\mu},r'>,E') \in PM.\, PL_r(<a,\tilde{\lambda},r>) \geq PL_r(<b,\tilde{\mu},r'>) \,|\!\}$

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

$Melt_r(PM) = \{(<a,\tilde{\lambda},r>,E) \mid (<a,\tilde{\mu},r'>,E) \in PM \wedge$
$\tilde{\lambda} = Min\{\!|\, \tilde{\gamma} \mid (<a,\tilde{\gamma},r''>,E) \in PM \wedge PL_r(<a,\tilde{\gamma},r''>) = PL_r(<a,\tilde{\mu},r'>) \,|\!\} \wedge$
$r = \sum\{\!|\, r'' \mid (<a,\tilde{\gamma},r''>,E) \in PM \wedge PL_r(<a,\tilde{\gamma},r''>) = PL_r(<a,\tilde{\mu},r'>) \,|\!\}\}$

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

$$Norm_{r,rate}(a,\tilde{\lambda}_1,\tilde{\lambda}_2,PM_1,PM_2) = \begin{cases} Split(\tilde{\lambda}_1, 1/(\pi_1(PM_2))(<a,*,0>)) \text{ if } \tilde{\lambda}_2 = * \\ Split(\tilde{\lambda}_2, 1/(\pi_1(PM_1))(<a,*,0>)) \text{ if } \tilde{\lambda}_1 = * \end{cases}$$

$$Norm_{r,reward}(a,r_1,r_2,PM_1,PM_2) = \begin{cases} r_1/(\pi_1(PM_2))(<a,*,0>) \text{ if } r_2 = 0 \\ r_2/(\pi_1(PM_1))(<a,*,0>) \text{ if } r_1 = 0 \end{cases}$$

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

Table 1. Inductive rules for EMPA$_r$ integrated interleaving semantics

derivative term. The first operation is carried out through functions $Select_r : \mathcal{M}u_{fin}(Act_r \times \mathcal{G}_r) \longrightarrow \mathcal{M}u_{fin}(Act_r \times \mathcal{G}_r)$ and $PL_r : Act_r \longrightarrow PLevel$, which are defined in the third part of Table 1. The second operation is carried out through function $Melt_r : \mathcal{M}u_{fin}(Act_r \times \mathcal{G}_r) \longrightarrow \mathcal{P}_{fin}(Act_r \times \mathcal{G}_r)$ and partial function $Min : (ARate \times ARate) \multimap ARate$, which are defined in the fourth part of Table 1. Observe that function $Melt_r$ sums the rewards of the potential moves to merge: this is consistent with the additivity assumption about rewards.

The multiset $PM_r(E) \in \mathcal{M}u_{fin}(Act_r \times \mathcal{G}_r)$ of potential moves of $E \in \mathcal{G}_r$ is defined by structural induction in the second part of Table 1. The normalization of rates and rewards of potential moves resulting from the synchronization of an action with several independent or alternative passive actions is carried out through partial functions $Norm_{r,rate} : (AType \times ARate \times ARate \times \mathcal{M}u_{fin}(Act_r \times \mathcal{G}_r) \times \mathcal{M}u_{fin}(Act_r \times \mathcal{G}_r)) \multimap ARate$ and $Norm_{r,reward} : (AType \times AReward \times AReward \times \mathcal{M}u_{fin}(Act_r \times \mathcal{G}_r) \times \mathcal{M}u_{fin}(Act_r \times \mathcal{G}_r)) \multimap AReward$, and function $Split : (ARate \times \mathbb{R}_{]0,1]}) \longrightarrow ARate$, which are defined in the fifth part of Table 1. Observe that the normalization of rewards is consistent with the additivity assumption about rewards.

Definition 2. The *integrated interleaving semantics* of $E \in \mathcal{G}_r$ is the labeled transition system $\mathcal{I}_r[\![E]\!] = (\uparrow E, Act_r, \longrightarrow_E, E)$ where $\uparrow E$ is the set of states reachable from E, and $\longrightarrow_E$ is $\longrightarrow$ restricted to $\uparrow E \times Act_r \times \uparrow E$. ■

As in [1, 2], from the integrated semantic model it is possible to obtain a functional semantic model (by dropping action rates and rewards) as well as a performance semantic model (basically by dropping action types and by lifting rewards from transitions to states according to the additivity assumption). Due to lack of space, we do not show the related definitions here.

4 A Notion of Equivalence for EMPA$_r$

In [1, 2] we developed a notion of equivalence for EMPA called strong extended Markovian bisimulation equivalence and denoted $\sim_{EMB}$. Such an equivalence was defined according to the idea of probabilistic bisimulation [8] on the integrated semantic model, and we proved that it is necessary to define it on the integrated semantic model in order for the congruence property to hold. For the sake of convenience, we can extend $\sim_{EMB}$ to EMPA$_r$ since it disregards rewards, provided that like in [1, 2] we introduce a priority operator "$\Theta(\_)$" and we consider the language $\mathcal{L}_{r,\Theta}$ generated by the following syntax

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

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

$$\frac{(<a, \tilde{\lambda}, r>, E') \in Melt_r(PM_r(E))}{E \xrightarrow{a,\tilde{\lambda},r} E'}$$

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

$$PM_r(\Theta(E)) = Select_r(PM_r(E))$$

It is easily seen that EMPA$_r$ coincides with the set of terms $\{\Theta(E) \mid E \in \mathcal{L}_r\}$. We denote by $\mathcal{G}_{r,\Theta}$ the set of guarded and closed terms of $\mathcal{L}_{r,\Theta}$.

One of the advantages of the algebra-based method, besides its ease of use, is the possibility of defining a notion of equivalence for EMPA$_r$ which relates terms having the same reward, thus allowing for simplification without altering the value of the performance index we are interested in. Exploiting the lesson learnt with $\sim_{EMB}$, we define this new equivalence on the integrated semantic model. For simplicity, one may be tempted to relate strongly extended-Markovian bisimilar terms having the same total reward, intended as the sum of the rewards attached to the actions it can execute. However, in this way one would fail both to capture an equivalence preserving the performance measure at hand and to obtain a congruence.

Example 1. Consider terms

$$A \stackrel{\Delta}{=} <a, \lambda, r>.<b, \mu, r_1>.A$$
$$B \stackrel{\Delta}{=} <a, \lambda, r>.<b, \mu, r_2>.B$$

where $r_1 \neq r_2$. Then $A \sim_{EMB} B$ and A and B have the same total reward r, but if we solve the two underlying performance models we obtain two different values of the performance measure we are interested in: $r \cdot \mu/(\lambda + \mu) + r_1 \cdot \lambda/(\lambda + \mu)$ and $r \cdot \mu/(\lambda + \mu) + r_2 \cdot \lambda/(\lambda + \mu)$. ■

Example 2. Consider terms

$$E_1 \equiv <a, \lambda, r_1>.\underline{0} + <b, \mu, r_2>.\underline{0}$$
$$E_2 \equiv <a, \lambda, r_2>.\underline{0} + <b, \mu, r_1>.\underline{0}$$

where $r_1 \neq r_2$. Then $E_1 \sim_{EMB} E_2$ and E_1 and E_2 have the same total reward $r_1 + r_2$, but e.g. $E_1 \parallel_{\{b\}} \underline{0}$ has total reward r_1 while $E_2 \parallel_{\{b\}} \underline{0}$ has total reward r_2. ■

The examples above show that if we want to preserve the performance measure and to obtain a congruence, we cannot treat rewards separately from the rest of the actions: rewards must be checked in the bisimilarity clause in order to guarantee that, given two equivalent terms, they have the same total reward and any pair of equivalent terms reachable from them have the same total reward.

Below we show that it is really easy to extend the definition of $\sim_{EMB}$ in such a way that both objectives are achieved. Proofs of results are omitted whenever they are smooth adaptations of the corresponding proofs in [2].

Definition 3. We define partial functions *Rate*, *Reward*, *RR* with domain $\mathcal{G}_{r,\Theta} \times$ *AType* $\times$ *PLevel* $\times \mathcal{P}(\mathcal{G}_{r,\Theta})$ and ranges *ARate*, *AReward*, *ARate* $\times$ *AReward* respectively, by

$$Rate(E, a, l, C) = Min\{\!|\, \tilde{\lambda} \mid E \xrightarrow{a,\tilde{\lambda},r} E' \wedge PL_r(<a, \tilde{\lambda}, r>) = l \wedge E' \in C \,|\!\}$$
$$Reward(E, a, l, C) = \sum\{\!|\, r \mid E \xrightarrow{a,\tilde{\lambda},r} E' \wedge PL_r(<a, \tilde{\lambda}, r>) = l \wedge E' \in C \,|\!\}$$
$$RR(E, a, l, C) = (Rate(E, a, l, C), Reward(E, a, l, C))$$

■

Definition 4. An equivalence relation $\mathcal{B} \subseteq \mathcal{G}_{r,\Theta} \times \mathcal{G}_{r,\Theta}$ is a *strong extended Markovian reward bisimulation (strong EMRB)* iff, whenever $(E_1, E_2) \in \mathcal{B}$, then for all $a \in AType$, $l \in PLevel$ and $C \in \mathcal{G}_{r,\Theta}/\mathcal{B}$

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

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

Proposition 5. *Let $\sim_{EMRB}$ be the union of all the strong EMRBs. Then $\sim_{EMRB}$ is the largest strong EMRB.* ■

Definition 6. We call $\sim_{EMRB}$ the *strong extended Markovian reward bisimulation equivalence (strong EMRBE)*, and we say that $E_1, E_2 \in \mathcal{G}_{r,\Theta}$ are *strongly extended-Markovian reward bisimulation equivalent (strongly EMRBE)* if and only if $E_1 \sim_{EMRB} E_2$. ■

Proposition 7. $\sim_{EMRB} \subset \sim_{EMB}$.

Proof. It follows immediately from the fact that every strong EMRB is a strong EMB too. ■

The following example shows that the inclusion is strict. We would like to point out that this is not inconsistent with $\sim_{EMB}$. The purpose of $\sim_{EMB}$ is to relate terms describing concurrent systems having the same functional and performance properties: if $E_1 \sim_{EMB} E_2$ but $E_1 \not\sim_{EMRB} E_2$, this simply means that we are measuring two different performance indices for E_1 and E_2.

Example 3. Consider terms

$$A \stackrel{\Delta}{=} <a, \lambda, 1>.<b, \mu, 0>.A$$
$$B \stackrel{\Delta}{=} <a, \lambda, 0>.<b, \mu, 1>.B$$

Then $A \sim_{EMB} B$ but $A \not\sim_{EMRB} B$. If we regard a and b as the transmission over two different channels, then by means of A we can compute the utilization of the former channel, whereas by means of B we can compute the utilization of the latter channel. ■

Theorem 8. *Let $E_1, E_2 \in \mathcal{G}_{r,\Theta}$. If $E_1 \sim_{EMRB} E_2$ then:*

1. *For every $<a, \tilde{\lambda}, r> \in Act_r$, $<a, \tilde{\lambda}, r>.E_1 \sim_{EMRB} <a, \tilde{\lambda}, r>.E_2$.*
2. *For every $L \subseteq AType - \{\tau\}$, $E_1/L \sim_{EMRB} E_2/L$.*
3. *For every $\varphi \in RFun$, $E_1[\varphi] \sim_{EMRB} E_2[\varphi]$.*
4. *$\Theta(E_1) \sim_{EMRB} \Theta(E_2)$.*
5. *For every $F \in \mathcal{G}_{r,\Theta}$, $E_1 + F \sim_{EMRB} E_2 + F$ and $F + E_1 \sim_{EMRB} F + E_2$.*
6. *For every $F \in \mathcal{G}_{r,\Theta}$ and $S \subseteq AType - \{\tau\}$, $E_1 \parallel_S F \sim_{EMRB} E_2 \parallel_S F$ and $F \parallel_S E_1 \sim_{EMRB} F \parallel_S E_2$.* ■

Theorem 9. *$\sim_{EMRB}$ is preserved by recursive definitions.* ■

Theorem 10. *Let $\mathcal{A}_r$ be the set of axioms in Table 2. The deductive system $Ded(\mathcal{A}_r)$ is sound and complete with respect to $\sim_{EMRB}$ for the set of nonrecursive terms of $\mathcal{G}_{r,\Theta}$.* ■

$(\mathcal{A}_{r,1})$ $(E_1 + E_2) + E_3 = E_1 + (E_2 + E_3)$

$(\mathcal{A}_{r,2})$ $E_1 + E_2 = E_2 + E_1$

$(\mathcal{A}_{r,3})$ $E + \underline{0} = E$

$(\mathcal{A}_{r,4})$ $<a, \tilde{\lambda}_1, r_1>.E + <a, \tilde{\lambda}_2, r_2>.E = <a, \tilde{\lambda}_1 \; Min \; \tilde{\lambda}_2, r_1 + r_2>.E$
if $PL_r(<a, \tilde{\lambda}_1, r_1>) = PL_r(<a, \tilde{\lambda}_2, r_2>)$

$(\mathcal{A}_{r,5})$ $\underline{0}/L = \underline{0}$

$(\mathcal{A}_{r,6})$ $(<a, \tilde{\lambda}, r>.E)/L = \begin{cases} <a, \tilde{\lambda}, r>.(E/L) \text{ if } a \notin L \\ <\tau, \tilde{\lambda}, r>.(E/L) \text{ if } a \in L \end{cases}$

$(\mathcal{A}_{r,7})$ $(E_1 + E_2)/L = E_1/L + E_2/L$

$(\mathcal{A}_{r,8})$ $\underline{0}[\varphi] = \underline{0}$

$(\mathcal{A}_{r,9})$ $(<a, \tilde{\lambda}, r>.E)[\varphi] = <\varphi(a), \tilde{\lambda}, r>.(E[\varphi])$

$(\mathcal{A}_{r,10})$ $(E_1 + E_2)[\varphi] = E_1[\varphi] + E_2[\varphi]$

$(\mathcal{A}_{r,11})$ $\Theta(\underline{0}) = \underline{0}$

$(\mathcal{A}_{r,12})$ $\Theta(\sum_{i \in I} <a_i, \tilde{\lambda}_i, r_i>.E_i) = \sum_{j \in J} <a_j, \tilde{\lambda}_j, r_j>.\Theta(E_j)$
where $J = \{i \in I \mid \tilde{\lambda}_i = * \vee \forall h \in I.\, PL_r(<a_i, \tilde{\lambda}_i, r_i>) \geq PL_r(<a_h, \tilde{\lambda}_h, r_h>)\}$

$(\mathcal{A}_{r,13})$ $\underline{0} \parallel_S \underline{0} = \underline{0}$

$(\mathcal{A}_{r,14})$ $(\sum_{i \in I} <a_i, \tilde{\lambda}_i, r_i>.E_i) \parallel_S \underline{0} = \sum_{j \in J} <a_j, \tilde{\lambda}_j, r_j>.(E_j \parallel_S \underline{0})$ where $J = \{i \in I \mid a_i \notin S\}$

$(\mathcal{A}_{r,15})$ $\underline{0} \parallel_S (\sum_{i \in I} <a_i, \tilde{\lambda}_i, r_i>.E_i) = \sum_{j \in J} <a_j, \tilde{\lambda}_j, r_j>.(\underline{0} \parallel_S E_j)$ where $J = \{i \in I \mid a_i \notin S\}$

$(\mathcal{A}_{r,16})$ $(\sum_{i \in I_1} <a_i, \tilde{\lambda}_i, r_i>.E_i) \parallel_S (\sum_{i \in I_2} <a_i, \tilde{\lambda}_i, r_i>.E_i) =$

$$\sum_{j \in J_1} <a_j, \tilde{\lambda}_j, r_j>.(E_j \parallel_S \sum_{i \in I_2} <a_i, \tilde{\lambda}_i, r_i>.E_i) +$$
$$\sum_{j \in J_2} <a_j, \tilde{\lambda}_j, r_j>.(\sum_{i \in I_1} <a_i, \tilde{\lambda}_i, r_i>.E_i \parallel_S E_j) +$$
$$\sum_{k \in K_1 \wedge h \in H_k} <a_k, Split(\tilde{\lambda}_k, 1/n_k), r_k/n_k>.(E_k \parallel_S E_h) +$$
$$\sum_{k \in K_2 \wedge h \in H_k} <a_k, Split(\tilde{\lambda}_k, 1/n_k), r_k/n_k>.(E_h \parallel_S E_k)$$

where $J_1 = \{i \in I_1 \mid a_i \notin S\}$
$J_2 = \{i \in I_2 \mid a_i \notin S\}$
$K_1 = \{i_1 \in I_1 \mid \exists i_2 \in I_2.\, a_{i_1} = a_{i_2} \in S \wedge \tilde{\lambda}_{i_2} = *\}$
$K_2 = \{i_2 \in I_2 \mid \exists i_1 \in I_1.\, a_{i_1} = a_{i_2} \in S \wedge \tilde{\lambda}_{i_1} = *\}$
$H_k = \{h \in I_2 \mid a_k = a_h \wedge \tilde{\lambda}_h = *\}$ with $k \in K_1$
$H_k = \{h \in I_1 \mid a_k = a_h \wedge \tilde{\lambda}_h = *\}$ with $k \in K_2$
$n_k = |H_k|$

Table 2. Axioms for $\sim_{EMRB}$

5 Conclusion

In this paper we have introduced an algebra-based method to attach rewards with EMPA terms in order to derive performance measures. As observed in Sect. 2, though less powerful in general than the logic-based method proposed in [3], the algebra-based method may be convenient due to its ease of use, its low computational cost and the possibility of defining a notion of equivalence accounting for rewards. Furthermore, it has been a really easy task to extend the theory developed for EMPA in order to take into account rewards according to the algebra-based method.

Concerning future work, we could allow the designer to associate rewards with immediate actions as well, because in this way we could derive performance measures also when we restrict ourselves to the probabilistic kernel [2] of EMPA. Finally, the algebra-based method will be implemented in a software tool (we are currently developing) based on EMPA for the modeling and analysis of functional and performance properties of concurrent systems.

Acknowledgements

This research has been partially funded by MURST and CNR.

References

1. M. Bernardo, R. Gorrieri, *"Extended Markovian Process Algebra"*, in Proc. of the *7th Int. Conf. on Concurrency Theory (CONCUR '96)*, LNCS 1119:315-330, Pisa (Italy), 1996
2. M. Bernardo, R. Gorrieri, *"A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time"*, Technical Report UBLCS-96-17, University of Bologna (Italy), 1996
3. 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
4. M. Hennessy, R. Milner, *"Algebraic Laws for Nondeterminism and Concurrency"*, in Journal of the ACM 32:137-161, 1985
5. J. Hillston, *"A Compositional Approach to Performance Modelling"*, Cambridge University Press, 1996
6. R.A. Howard, *"Dynamic Probabilistic Systems"*, John Wiley & Sons, 1971
7. L. Kleinrock, *"Queueing Systems"*, John Wiley & Sons, 1975
8. K.G. Larsen, A. Skou, *"Bisimulation through Probabilistic Testing"*, in Information and Computation 94(1):1-28, 1991
9. 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