Enriching EMPA with Value Passing: A Symbolic Approach based on Lookahead

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 extend the stochastically timed process algebra EMPA by allowing for value passing. The proposal of Hennessy and Lin relying on symbolic labeled transition systems and symbolic bisimulations is adapted to EMPA by providing suitable semantic rules based on lookahead in order to benefit as much as possible from the inherent parametricity of value passing. This turns out to be quite useful from the analysis point of view, especially to take advantage from symmetries as the example at the end of the paper demonstrates.

1 Introduction

The stochastically timed process algebra EMPA [4, 5, 3] provides a framework where concurrent systems can be compositionally described in such a way that it is possible both to verify their functional properties and to compute their performance measures.

Despite of the considerable expressive power of EMPA, there is a large class of concurrent systems that cannot be dealt with. Such a class is composed of those systems where *data* play a fundamental role, so that it is not possible to abstract from them when describing those systems. An example of such systems is any process that can receive messages and undertake different activities depending on the contents of messages. The problem with EMPA is that data are not considered at all.

The purpose of this paper is to suitably extend EMPA in order to cope with the situation mentioned before. Following [9], we modify the syntax firstly by introducing actions which can read or write values: as a consequence, beside *unstructured actions* of the form $<a, \tilde{\lambda}>$, we will have *input actions* of the form $<a?x, \tilde{\lambda}>$ where x is a variable, as well as *output actions* of the form $<a!e, \tilde{\lambda}>$ where e is an expression. Secondly, a new operator is necessary which reflects the influence of data on the execution: this is a *conditional operator* of the form *if* β *then* E_1 *else* E_2 where β is a boolean expression. Thirdly, we should be able to keep track of the data we are interested in: this can be achieved by means of *parametrized constant definitions* of the form $A(\underline{x}) \stackrel{\Delta}{=} E$ where $\underline{x}$ is a vector of variables.

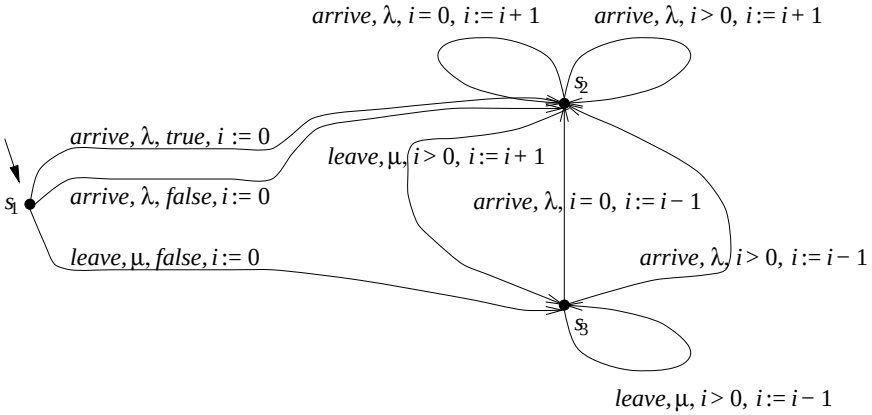
We have listed above all the syntactic ingredients we need in order to cope with value passing. Incidentally, it is worth noting that by means of them it is even possible to give a *finite algebraic representation* of systems that do not admit it in the pure calculus because of behavioral (as opposed to structural) reasons. For example, an unbounded buffer where interarrival times are exponentially distributed with rate λ and interdeparture times are exponentially distributed with rate μ can be described as follows:

$$\begin{aligned} \mathit{Buffer}(i) \quad \stackrel{\Delta}{=} \quad & \mathit{if}\ (i = 0)\ \mathit{then} <\mathit{arrive}, \lambda>.\mathit{Buffer}(i+1) \\ & \mathit{else} <\mathit{arrive}, \lambda>.\mathit{Buffer}(i+1) + <\mathit{leave}, \mu>.\mathit{Buffer}(i-1) \end{aligned}$$

Now the question becomes how to define the semantics for such a value passing calculus. The proposal of [9] of expanding a term of the form $<a?x, \tilde{\lambda}>.E$ according to all the possible values v that variable x can take on, which results in term $\sum_v <a_v, \tilde{\lambda}>.E\{v/x\}$, should be avoided for two reasons: the first (practical)

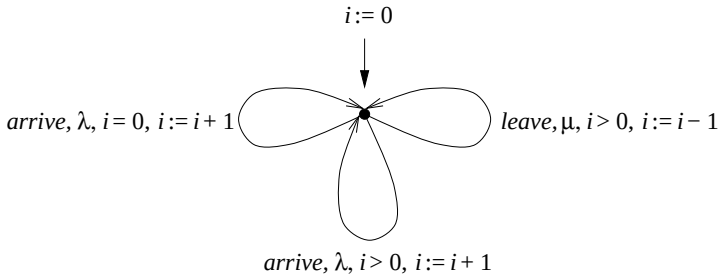
one is that the expansion causes *infinite branching* to arise whenever the value set is infinite, the second (theoretical) one is that the expansion may unexpectedly *increase the speed* at which activities are carried out. To solve this problem, we follow the proposal of [6]: the semantic model of the calculus is given by a *symbolic labeled transition system*, i.e. a labeled transition system where actions are kept symbolic. To be more precise, every state has an associated finite set of free variables and a transition of the form $s \xrightarrow{\alpha,\tilde{\lambda},\beta,\sigma} s'$ means that it is possible to evolve from state s to state s' by executing an action with type α and rate $\tilde{\lambda}$ provided that boolean expression β evaluates to *true* at state s, and after that the free variables of s' are given the values specified by assignment σ evaluated at state s [8].

As an example, the semantic model for $Buffer(0)$ obtained by applying semantic rules similar to those shown in [8] is the following *finite symbolic labeled transition system*:



The transitions leaving initial state s_1, corresponding to $Buffer(0)$, are computed by applying the semantic rules to the defining equation for $Buffer(i)$ under the assignment $i := 0$. As a consequence, only one of those three transitions can be actually executed as one can see by observing the boolean guards labeling the transitions themselves, and the value of free variable i in the target state is 0. The transitions leaving states s_2, corresponding to $Buffer(i+1)$, and s_3, corresponding to $Buffer(i-1)$, are computed in the same manner, the difference being that free variable i must be increased or decreased, respectively.

From the example above it can be noted that the semantic rules proposed in [8] do not exploit the whole parametricity inherent to value passing. To accomplish this, we should *keep states as symbolic as possible* by suitably changing the semantic rule for constants. The idea is not to substitute (by means of an assignment) actual parameters for formal parameters of a constant definition *upon* constant invokation as done in [8]. Instead, we *look ahead* in the target term of each transition so as to set the assignment above *just before* constant invokation. In the case of $Buffer(0)$, we should obtain the following *smaller finite symbolic labeled transition system*:



where we have only one state corresponding to $Buffer(i)$, we keep track of the initial value of formal parameter i so that e.g. the semantic model for $Buffer(0)$ is different from the semantic model for $Buffer(1)$, and the assignments labeling the transitions indicate how formal parameter i must be updated after executing

them. Even from this simple example, it is clear that using lookahead produces more compact symbolic labeled transition systems and proves to be advantageous to detect symmetries a priori, which means that the semantic model should not necessarily undergo to a reduction algorithm after it has been built.

The paper is organized as follows. In Section 2 we present the syntax and the semantics for EMPA extended with value passing. In Section 3 we adapt to our stochastic framework the notion of symbolic bisimulation developed in [6, 8]. In Section 4 we present an example illustrating the advantage gained by means of value passing to exploit symmetries. In Section 5 we report some concluding remarks.

2 Syntax and semantics for EMPA$_{vp}$

In this section we extend the syntax and the semantics for EMPA in order to cope with value passing: the resulting stochastically timed process algebra is called EMPA$_{vp}$.

Before introducing the syntax of EMPA$_{vp}$, we define some syntactic categories that we will use in the sequel. Let *Val* be a set of *values* (ranged over by v), *Var* be a set of *variables* (ranged over by x, y, z), and $Eval = \{\rho : Var \longrightarrow Val\}$ be a set of *evaluations*. Moreover, let *Exp* be a set of *expressions* (ranged over by e) which includes *Val*, *Var* and a set *BExp* of *boolean expressions* (ranged over by β), and let $Sub = \{\sigma : Var \longrightarrow Exp\}$ be a set of *substitutions* which will be sometimes denoted by $\underline{x} := \underline{e}$ (where $\underline{x}$ is a vector of variables and $\underline{e}$ is a vector of expressions of the same length) to specify *assignments*. We shall also write $\rho[x \mapsto v]$ to denote the evaluation which differs from ρ only in that it maps x to v, and $\sigma[x \mapsto e]$ to denote the substitution which differs from σ only in that it maps x to e.

As usual, the building blocks of EMPA$_{vp}$ are *actions*. Each action is a pair $<\alpha, \tilde{\lambda}>$ consisting of the *type* of the action and the *rate* of the action. Concerning action types, let $AName = AName_U \cup AName_{IO}$ be the set of *action names* (ranged over by a), where $AName_U$ is the set of *unstructured action names* (including τ) and $AName_{IO}$ is the set of *input/output action names*, such that $AName_U \cap AName_{IO} = \emptyset$. Then the set of action types is $AType_{vp} = AType_U \cup AType_I \cup AType_O$ (ranged over by α) where $AType_U = AName_U$ is the set of *unstructured action types*, $AType_I = \{a?x \mid a \in AName_{IO} \wedge x \in Var\}$ is the set of *input action types*, and $AType_O = \{a!e \mid a \in AName_{IO} \wedge e \in Exp\}$ is the set of *output action types*. We also define function $name : AType_{vp} \longrightarrow AName$ by $name(a) = name(a?x) = name(a!e) = a$. The set of action rates is given by $ARate = \mathbf{R}_+ \cup Inf \cup \{*\}$ (ranged over by $\tilde{\lambda}, \tilde{\mu}, \tilde{\gamma}$) where $Inf = \{\infty_{l,w} \mid l \in \mathbf{N}_+ \wedge w \in \mathbf{R}_+\}$, which means that an action can be exponentially timed, immediate with a given priority level and a given weight, or passive. Finally, we let $Act_{vp} = AType_{vp} \times ARate$, and we denote by $APLev = \{-1\} \cup \mathbf{N}$ the set of *action priority levels*.

Let *Const* be a set of *constants* (ranged over by A), and let $ARFun_{vp} = \{\varphi : AName \longrightarrow AName \mid \varphi(\tau) = \tau \wedge \varphi(AName_U - \{\tau\}) \subseteq AName_U - \{\tau\} \wedge \varphi(AName_{IO}) \subseteq AName_{IO}\}$ be a set of *action relabeling functions*. For the sake of convenience, we lift action relabeling functions from action names to action types in the expected way.

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

$$\begin{array}{rcl} E & ::= & F \mid E/L \mid E[\varphi] \mid E \parallel_S E \\ F & ::= & \sum_{i \in I} <\alpha_i, \tilde{\lambda}_i>.F_i \mid \textit{if } (\beta, p) \textit{ then } F \textit{ else } F \mid A(\underline{e}) \end{array}$$

where $L, S \subseteq AName - \{\tau\}$, I is a finite set of indexes such that $\sum_{i \in \emptyset} <\alpha_i, \tilde{\lambda}_i>.F_i$ is denoted by $\underline{0}$, $p \in \mathbf{R}_{]0,1[}$, and $A(\underline{x}) \triangleq F$ with F not a constant whenever such a constant defining equation exists. We denote by $\mathcal{G}_{vp}$ the set of guarded and closed terms of $\mathcal{L}_{vp}$. ■

The new operator with respect to EMPA is the *conditional operator*. Term *if* (β, p) *then* F_1 *else* F_2 behaves like F_1 or F_2 depending on whether boolean expression β is satisfied or not. It is worth noting that condition (β, p) is composed (like actions) of a functional part given by boolean expression β as well as a performance part given by probability p. The latter part expresses (a guess about) the probability that β is satisfied, and is introduced to allow for the construction of the performance model of any term containing occurrences of the conditional operator, which is in its classical form a source of pure nondeterminism. Since we want that conditional choices are resolved independently of the environment and that their resolution takes

precedence over any other action (passive ones included), we introduce a new priority level ω and we thus define $APLev_{vp} = APLev \cup \{\omega\}$, $Inf_{vp} = Inf \cup \{\infty_{\omega,w} \mid w \in \mathbb{R}_+\}$, $ARate_{vp} = \mathbb{R}_+ \cup Inf_{vp} \cup \{*\}$, and $Act'_{vp} = AType_{vp} \times ARate_{vp}$. As a consequence, from the performance standpoint term *if* (β, p) *then* F_1 *else* F_2 can be viewed as $<\tau, \infty_{\omega,p}>.F_1 + <\tau, \infty_{\omega,1-p}>.F_2$.

Another difference with respect to EMPA is that now the null term, the prefix operator and the alternative composition operator are combined together so as to result in a single guarded alternative composition operator. Like in EMPA, 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. The rationale behind the adoption of such a guarded alternative composition operator, beside its greater adequacy from the modeling standpoint, is that we want to avoid terms like $<\alpha, \tilde{\lambda}>.F +$ *if* (β, p) *then* F_1 *else* F_2 which could not be easily handled according to our view of the conditional operator, as well as terms like $A(\underline{e}_1) + A(\underline{e}_2)$ which would complicate the treatment of parameters.

Concerning the parallel composition operator, like in EMPA a synchronization can occur if and only if the involved actions have the same name belonging to the synchronization set, and at most one of the involved actions is not passive. However, in EMPA$_{vp}$ the synchronization is binary instead of multiway in the case of input/output actions. Since variables with the same name occurring in different scopes with respect to parallel composition operators should be kept distinct, we extend the set of variables by means of localities so that $\swarrow x \in Var$ and $\searrow x \in Var$ whenever $x \in Var$, and we accordingly extend *Eval*, *Exp*, *BExp* and *Sub*.

The last difference with respect to EMPA is the two-level syntax definition. The motivation for this, beside the fact that it rules out terms whose state space is not finite because of structural reasons, is that we want to avoid terms like $A(\underline{x}) \triangleq A'(\underline{x})/L$, $A(\underline{x}) \triangleq A'(\underline{x})[\varphi]$, $A(\underline{x}) \triangleq A'(\underline{x}) \parallel_S A''(\underline{x})$ and $A(\underline{x}) \triangleq A'(\underline{x})$ which would complicate the treatment of parameters. To achieve this we require $A(\underline{x}) \triangleq F$ with F not a constant, so the body of every constant definition must be a guarded alternative composition or a conditional.

As anticipated in Section 1, the integrated semantic model for EMPA$_{vp}$ is a smooth adaptation of that proposed in [8].

Definition 2.2 A *symbolic labeled transition system with initial assignment (SLTSIA)* is a sestuple $(S, Label \times BExp \times Sub, \longrightarrow, s_0, \sigma_0)$ where $(S, Label \times BExp \times Sub, \longrightarrow, s_0)$ is a rooted labeled transition system where every state $s \in S$ has an associated set of free variables $fv(s)$, and $\sigma_0 \in Sub$ is the initial assignment for the free variables. Transition $s \xrightarrow{\ell,\beta,\sigma} s'$ means that it is possible to evolve from state s to state s' by executing an action labeled with ℓ provided that boolean expression β evaluates to *true* at state s, and after that $fv(s')$ are assigned the values specified by σ evaluated at state s. ■

As illustrated in Section 1, in order to produce compact semantic models which exploit all the parametricity inherent to value passing, the semantic rules must be set up in such a way that *terms are kept as symbolic as possible* since they are in correspondence with states. We accomplish this by replacing each term of the form $A(\underline{e})$ occurring in an empty or static context of the derivative term of a transition with F whenever $A(\underline{x}) \triangleq F$, and by keeping track of the substitution via assignment $\underline{x} := \underline{e}$. Notice that this implements exactly the idea of looking ahead in the target term of each transition so as to set formal parameters just before constant invokation. Formally, we introduce function $sym : \mathcal{G}_{vp} \longrightarrow \mathcal{G}_{vp}$ defined by structural induction as follows:

$$\begin{array}{rcl}
sym(\sum_{i\in I} <\alpha_i, \tilde{\lambda}_i>.F_i) & = & \sum_{i\in I} <\alpha_i, \tilde{\lambda}_i>.F_i \\
sym(\textit{if } (\beta,p) \textit{ then } F_1 \textit{ else } F_2) & = & \textit{if } (\beta,p) \textit{ then } F_1 \textit{ else } F_2 \\
sym(E/L) & = & sym(E)/L \\
sym(E[\varphi]) & = & sym(E)[\varphi] \\
sym(E_1 \parallel_S E_2) & = & sym(E_1) \parallel_S sym(E_2) \\
sym(A(\underline{e})) & = & F \quad \text{if } A(\underline{x}) \triangleq F
\end{array}$$

We also introduce function $setpar : \mathcal{G}_{vp} \longrightarrow Sub$ defined by structural induction as follows:

$$\frac{(<\alpha,\tilde{\lambda}>,\beta,\sigma,E') \in Melt_{vp}(Select_{vp}(PM_{vp}(E)))}{E \xrightarrow{\alpha,\tilde{\lambda},\beta,\sigma} E'}$$

$$PM_{vp}(\textstyle\sum_{i\in I} <\alpha_i,\tilde{\lambda}_i>.F_i) = \bigoplus_{i\in I} \{\!|\, (<\alpha_i,\tilde{\lambda}_i>, true, setpar(F_i), sym(F_i)) \,|\!\}$$

$$PM_{vp}(if\ (\beta,p)\ then\ F_1\ else\ F_2) = \{\!|\, (<\tau,\infty_{\omega,p}>,\beta,setpar(F_1),sym(F_1)), (<\tau,\infty_{\omega,1-p}>,\neg\beta,setpar(F_2),sym(F_2)) \,|\!\}$$

$$\begin{aligned} PM_{vp}(E/L) = {} & \{\!|\, (<\alpha,\tilde{\lambda}>,\beta,\sigma,E'/L) \mid (<\alpha,\tilde{\lambda}>,\beta,\sigma,E') \in PM_{vp}(E) \wedge name(\alpha) \notin L \,|\!\} \oplus \\ & \{\!|\, (<\tau,\tilde{\lambda}>,\beta,\sigma,E'/L) \mid (<\alpha,\tilde{\lambda}>,\beta,\sigma,E') \in PM_{vp}(E) \wedge name(\alpha) \in L \,|\!\} \end{aligned}$$

$$PM_{vp}(E[\varphi]) = \{\!|\, (<\varphi(\alpha),\tilde{\lambda}>,\beta,\sigma,E'[\varphi]) \mid (<\alpha,\tilde{\lambda}>,\beta,\sigma,E') \in PM_{vp}(E) \,|\!\}$$

$$\begin{aligned} PM_{vp}(E_1 \parallel_S E_2) = {} & \{\!|\, (<\alpha,\tilde{\lambda}>,\swarrow\beta,\swarrow\sigma,E_1' \parallel_S E_2) \mid name(\alpha) \notin S \wedge (<\alpha,\tilde{\lambda}>,\beta,\sigma,E_1') \in PM_{vp}(E_1) \,|\!\} \oplus \\ & \{\!|\, (<\alpha,\tilde{\lambda}>,\searrow\beta,\searrow\sigma,E_1 \parallel_S E_2') \mid name(\alpha) \notin S \wedge (<\alpha,\tilde{\lambda}>,\beta,\sigma,E_2') \in PM_{vp}(E_2) \,|\!\} \oplus \\ & \{\!|\, (<a,\tilde{\gamma}>,\swarrow\beta_1 \wedge \searrow\beta_2,\sigma,E_1' \parallel_S E_2') \mid (<\alpha_1,\tilde{\lambda}_1>,\beta_1,\sigma_1,E_1') \in PM_{vp}(E_1) \wedge \\ & \qquad (<\alpha_2,\tilde{\lambda}_2>,\beta_2,\sigma_2,E_2') \in PM_{vp}(E_2) \wedge \\ & \qquad name(\alpha_1) = name(\alpha_2) = a \in S \wedge \\ & \qquad \tilde{\gamma} = Norm_{vp}(\alpha_1,\alpha_2,\tilde{\lambda}_1,\tilde{\lambda}_2,PM_{vp}(E_1),PM_{vp}(E_2)) \wedge \\ & \qquad \alpha_1,\alpha_2 \in AType_U \Longrightarrow \sigma = \swarrow\sigma_1 \cup \searrow\sigma_2 \wedge \\ & \qquad \alpha_1 = a?x \wedge \alpha_2 = a!e \Longrightarrow \sigma = \swarrow\sigma_1 \cup \searrow\sigma_2 \cup \{\swarrow x := \searrow e\} \wedge \\ & \qquad \alpha_1 = a!e \wedge \alpha_2 = a?x \Longrightarrow \sigma = \swarrow\sigma_1 \cup \searrow\sigma_2 \cup \{\searrow x := \swarrow e\} \,|\!\} \end{aligned}$$

$$PM_{vp}(A(\underline{e})) = PM_{vp}(F) \quad \text{if } A(\underline{x}) \triangleq F$$

$$\begin{aligned} Select_{vp}(PM) = {} & \{\!|\, (<\alpha,\tilde{\lambda}>,\beta,\sigma,E) \in PM \mid (PL(<\alpha,\tilde{\lambda}>) = -1 \wedge \forall (<\alpha',\tilde{\lambda}'>,\beta',\sigma',E') \in PM.\, PL(<\alpha',\tilde{\lambda}'>) < \omega) \vee \\ & \qquad \forall (<\alpha',\tilde{\lambda}'>,\beta',\sigma',E') \in PM.\, PL(<\alpha,\tilde{\lambda}>) \geq PL(<\alpha',\tilde{\lambda}'>) \,|\!\} \end{aligned}$$

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

$$\begin{aligned} Melt_{vp}(PM) = \{(<\alpha,\tilde{\lambda}>,\beta,\sigma,E) \mid {} & (<\alpha,\tilde{\mu}>,\beta,\sigma,E) \in PM \wedge \\ & \tilde{\lambda} = Min\{\!|\, \tilde{\gamma} \mid (<\alpha,\tilde{\gamma}>,\beta,\sigma,E) \in PM \wedge PL(<\alpha,\tilde{\gamma}>) = PL(<\alpha,\tilde{\mu}>) \,|\!\}\} \end{aligned}$$

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

$$Norm_{vp}(\alpha_1,\alpha_2,\tilde{\lambda}_1,\tilde{\lambda}_2,PM_1,PM_2) = \begin{cases} Split(\tilde{\lambda}_1, 1/(\pi_1(PM_2))(<\alpha,*>)) & \text{if } \alpha_1 = \alpha_2 = \alpha \in AType_U \wedge \tilde{\lambda}_2 = * \\ Split(\tilde{\lambda}_2, 1/(\pi_1(PM_1))(<\alpha,*>)) & \text{if } \alpha_1 = \alpha_2 = \alpha \in AType_U \wedge \tilde{\lambda}_1 = * \\ Split(\tilde{\lambda}_1, 1/\sum_{x\in Var}(\pi_1(PM_2))(<a?x,*>)) & \text{if } \alpha_1 \in AType_O \wedge \alpha_2 \in AType_I \wedge \\ & \quad name(\alpha_2) = a \wedge \tilde{\lambda}_2 = * \\ Split(\tilde{\lambda}_2, 1/\sum_{x\in Var}(\pi_1(PM_1))(<a?x,*>)) & \text{if } \alpha_1 \in AType_I \wedge \alpha_2 \in AType_O \wedge \\ & \quad name(\alpha_1) = a \wedge \tilde{\lambda}_1 = * \\ Split(\tilde{\lambda}_1, 1/\sum_{e\in Exp}(\pi_1(PM_2))(<a!e,*>)) & \text{if } \alpha_1 \in AType_I \wedge \alpha_2 \in AType_O \wedge \\ & \quad name(\alpha_2) = a \wedge \tilde{\lambda}_2 = * \\ Split(\tilde{\lambda}_2, 1/\sum_{e\in Exp}(\pi_1(PM_1))(<a!e,*>)) & \text{if } \alpha_1 \in AType_O \wedge \alpha_2 \in AType_I \wedge \\ & \quad name(\alpha_1) = a \wedge \tilde{\lambda}_1 = * \end{cases}$$

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

Table 1: Inductive rules for EMPA$_{vp}$ integrated interleaving semantics

$$
\begin{array}{rcl}
setpar(\sum_{i\in I} <\alpha_i, \tilde{\lambda}_i>.F_i) & = & \emptyset \\
setpar(\mathit{if}\ (\beta, p)\ \mathit{then}\ F_1\ \mathit{else}\ F_2) & = & \emptyset \\
setpar(E/L) & = & setpar(E) \\
setpar(E[\varphi]) & = & setpar(E) \\
setpar(E_1 \parallel_S E_2) & = & \swarrow setpar(E_1) \cup \searrow setpar(E_2) \\
setpar(A(\underline{e})) & = & \underline{x} := \underline{e} \quad \text{if } A(\underline{x}) \stackrel{\Delta}{=} F
\end{array}
$$

The integrated semantics of EMPA$_{vp}$ terms can be defined by exploiting again the idea of potential move, which is now a quadruple $(<\alpha, \tilde{\lambda}>, \beta, \sigma, E)$: 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. We denote by $PMove_{vp} = Act''_{vp} \times BExp \times Sub \times \mathcal{G}_{vp}$ the set of all the potential moves, where $Act''_{vp} = AType'_{vp} \times ARate_{vp}$ with $AType'_{vp} = AType_{vp} \cup AName_{IO}$.

The formal definition is based on the transition relation $\longrightarrow$, which is the least subset of $\mathcal{G}_{vp} \times (Act''_{vp} \times BExp \times Sub) \times \mathcal{G}_{vp}$ 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, the same boolean guard, the same assignment, and the same derivative term. The first operation is carried out through functions $Select_{vp} : \mathcal{M}u_{fin}(PMove_{vp}) \longrightarrow \mathcal{M}u_{fin}(PMove_{vp})$ and $PL : Act''_{vp} \longrightarrow APLev_{vp}$, which are defined in the third part of Table 1. It is worth noting that the selection of the highest priority level potential moves does not interfere with boolean guards because nontrivial boolean guards are associated only with potential moves generated by the conditional operator and these have the same priority level and take precedence over any other potential move. The second operation is carried out through function $Melt_{vp} : \mathcal{M}u_{fin}(PMove_{vp}) \longrightarrow \mathcal{P}_{fin}(PMove_{vp})$ and partial function $Min : (ARate_{vp} \times ARate_{vp}) \multimap ARate_{vp}$, which are defined in the fourth part of Table 1.

The multiset $PM_{vp}(E) \in \mathcal{M}u_{fin}(PMove_{vp})$ of potential moves of $E \in \mathcal{G}_{vp}$ is defined by structural induction in the second part of Table 1. The normalization of rates of potential moves resulting from the synchronization of an active action with several independent or alternative passive actions of the same name is carried out through partial function $Norm_{vp} : (AType'_{vp} \times AType'_{vp} \times ARate_{vp} \times ARate_{vp} \times \mathcal{M}u_{fin}(PMove_{vp}) \times \mathcal{M}u_{fin}(PMove_{vp})) \multimap ARate_{vp}$ and function $Split : (ARate_{vp} \times \mathbf{R}_{]0,1]}) \longrightarrow ARate_{vp}$, which are defined in the fifth part of Table 1.

Definition 2.3 The *integrated interleaving semantics* of $E \in \mathcal{G}_{vp}$ is the SLTSIA

$$\mathcal{I}_{vp}[\![E]\!] = (\uparrow E, Act''_{vp} \times BExp \times Sub, \longrightarrow_E, sym(E), setpar(E))$$

where $\uparrow E$ is the set of states reachable from $sym(E)$, and $\longrightarrow_E$ is $\longrightarrow$ restricted to $\uparrow E \times (Act''_{vp} \times BExp \times Sub) \times \uparrow E$. ■

Definition 2.4 $E \in \mathcal{G}_{vp}$ is *performance closed* if and only if $\mathcal{I}_{vp}[\![E]\!]$ does not contain passive transitions. We denote by $\mathcal{E}_{vp}$ the set of performance closed terms of $\mathcal{G}_{vp}$. ■

Definition 2.5 The *functional semantics* of $E \in \mathcal{G}_{vp}$ is the SLTSIA

$$\mathcal{F}_{vp}[\![E]\!] = (\uparrow E, AType'_{vp} \times BExp \times Sub, \longrightarrow_{E,\mathcal{F}_{vp}}, sym(E), setpar(E))$$

where $\longrightarrow_{E,\mathcal{F}_{vp}}$ is $\longrightarrow_E$ restricted to $\uparrow E \times (AType'_{vp} \times BExp \times Sub) \times \uparrow E$. ■

Definition 2.6 The *Markovian semantics* of $E \in \mathcal{E}_{vp}$ is the probabilistically rooted labeled transition system

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

obtained from $\mathcal{I}_{vp}[\![E]\!]$ by applying the algorithm described in [4] after dropping boolean guards and assignments from transitions. ■

[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., $(\pi_1(PM_2))(<\alpha, *>)$ in the fifth part of Table 1 denotes the multiplicity of tuples of PM_2 whose first component is $<\alpha, *>$.

We now report two properties of the semantic models of EMPA$_{vp}$ terms. The latter shows that EMPA$_{vp}$ is a conservative extension of EMPA.

Proposition 2.7 For any $E \in \mathcal{G}_{vp}$, every nondeadlocked state of $\mathcal{I}_{vp}[\![E]\!]$ is such that all of its transitions either have type τ and priority level ω, or have priority level less than ω and boolean guard *true*. ■

Proposition 2.8 If $E \in \mathcal{G}_{vp}$ has no actions whose type belongs to $AType_I \cup AType_O$, no conditional operators and no parametrized constants, then:

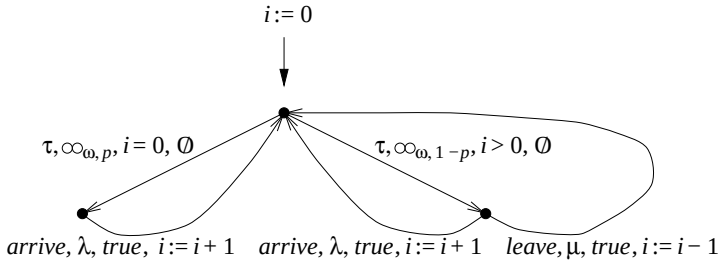
1. $\mathcal{I}_{vp}[\![E]\!]$ is isomorphic to $\mathcal{I}[\![E]\!]$ up to boolean guards and assignments.
2. $\mathcal{F}_{vp}[\![E]\!]$ is isomorphic to $\mathcal{F}[\![E]\!]$ up to boolean guards and assignments.
3. $\mathcal{M}_{vp}[\![E]\!]$ is p-isomorphic to $\mathcal{M}[\![E]\!]$. ■

We conclude the section by illustrating a couple of examples.

Example 2.9 Let us consider again the unbounded buffer introduced in Section 1. According to the syntax of EMPA$_{vp}$, the buffer is specified as follows:

$$\begin{aligned} Buffer(i) \;\stackrel{\Delta}{=}\; & \textit{if } (i = 0, p) \textit{ then } <arrive, \lambda>.Buffer(i+1) \\ & \textit{else } <arrive, \lambda>.Buffer(i+1) + <leave, \mu>.Buffer(i-1) \end{aligned}$$

where $p = 1 - \lambda/\mu$ since the buffer basically represents an $M/M/1$ queueing system with arrival rate λ and service rate μ [7] hence p must reflect the probability that the system is empty. Below we show $\mathcal{I}_{vp}[\![Buffer(0)]\!]$:



This SLTSIA has three states: the initial one corresponds to term $Buffer(i)$, while the others correspond to terms $<arrive, \lambda>.Buffer(i+1)$ and $<arrive, \lambda>.Buffer(i+1)+<leave, \mu>.Buffer(i-1)$, respectively. Notice that this SLTSIA differs from the one with only one state and three transitions presented in Section 1. This stems from our treatment of the conditional operator, which requires choices to be solved before executing actions. Though less compact, the model above might turn out to be more convenient than the one in Section 1 because each state explicitly denotes its average sojourn time. Moreover, this model is more compact than the one obtainable by applying the semantic rules in [8] since the latter has nine transitions instead of five. ■

Example 2.10 In our approach the substitution of actual parameters for formal parameters occurs just before constant invokation instead of upon constant invokation. The following term shows that this cannot occur at anytime before constant invokation:

$$A(i) \stackrel{\Delta}{=} <a!i, \lambda>.<a!i, \lambda>.Buffer(0)$$

If we consider $A(1)$ and we set $i := 0$ after the execution of the first a action because we encounter $Buffer(0)$ while looking ahead, then the second a action would output 0 in place of 1. ■

3 Symbolic Markovian bisimulations

Bisimulation equivalence [9] is a useful semantic theory for process algebras, and it is quite convenient to redefine it at the more abstract level of SLTSIA since they make it possible to work with finite semantic representations of value passing terms. The purpose of this section is to develop bisimulation equivalences for EMPA$_{vp}$ following the guidelines of [6, 8].

The standard definitions of bisimulation equivalence are given for operational semantics defined on closed terms of the calculus. Moreover, the standard definitions of operational semantics for value passing calculi are based on syntactic substitutions: for example, we would have rules of the form $<a?x, \tilde{\lambda}>.E \xrightarrow{a?v,\tilde{\lambda}} E\{v/x\}$. Since the states of a SLTSIA have an associated set of free variables and they cannot keep track of substitutions like the previous one, the proper analogue of an open term is a pair composed of a term and a substitution:

$$Term = \{\langle E, \sigma\rangle \in \mathcal{G}_{vp} \times Sub\}$$

Because of the remarks above which tell us that we cannot use directly the semantics defined in the previous section, before introducing our notion of bisimulation equivalence we thus have to redefine the semantics for EMPA$_{vp}$. To accomplish this, there are two different approaches named *early* and *late*, respectively, where the difference relies on whether the behavior of processes with respect to input actions is related or not to the actual input values. We shall take the late view since the early view can lead to unexpected rate increase as argued in Section 1. Actually, we shall define two different semantics (and consequently two different bisimulation equivalences) called *concrete late* and *symbolic late*, respectively, and we shall investigate the relationships between them.

In the concrete late case, boolean guards and output expressions are evaluated according to a given evaluation ρ supplying values for free variables, so that a concrete instance of the symbolic model $\mathcal{I}_{vp}[\![E]\!]$ is obtained where boolean guards and output expressions no longer label transitions. We define $AType'_O = \{a!v \mid a \in AName_{IO} \wedge v \in Val\}$, $AType''_{vp} = AType_U \cup AName_{IO} \cup AType_I \cup AType'_O$, and $Act'''_{vp} = AType''_{vp} \times ARate_{vp}$.

Definition 3.1 The *concrete late integrated interleaving semantics* of $E \in \mathcal{G}_{vp}$ with respect to $\rho \in Eval$ is the SLTS

$$\mathcal{I}^{CL,\rho}_{vp}[\![E]\!] = (S_{E,CL,\rho}, Act'''_{vp}, \longrightarrow_{E,CL,\rho}, \langle sym(E), setpar(E)\rangle)$$

where $S_{E,CL,\rho}$ is the set of states reachable from $\langle sym(E), setpar(E)\rangle$, and $\longrightarrow_{E,CL,\rho}$ is the restriction of $\longrightarrow_{CL,\rho}$ defined in Table 2 to $S_{E,CL,\rho} \times Act'''_{vp} \times S_{E,CL,\rho}$. ■

Definition 3.2 For any $\rho \in Eval$ we define partial function $Rate_{CL,\rho} : (Term \times AType''_{vp} \times APLev_{vp} \times Term) \longrightarrow ARate_{vp}$ by

$$Rate_{CL,\rho}(t, \alpha, l, C) = Min\{\!|\, \tilde{\lambda} \mid t \xrightarrow{\alpha,\tilde{\lambda}}_{CL,\rho} t' \wedge PL(<\alpha, \tilde{\lambda}>) = l \wedge t' \in C \,|\!\}$$ ■

Definition 3.3 Let $\mathcal{R} = \{\mathcal{B}^\rho \subseteq Term \times Term \mid \mathcal{B}^\rho$ equivalence $\wedge \rho \in Eval\}$, and let $CLB(\mathcal{R}) = \{CLB(\mathcal{R})^\rho \subseteq Term \times Term \mid CLB(\mathcal{R})^\rho$ equivalence $\wedge \rho \in Eval\}$ be defined by $(t_1, t_2) \in CLB(\mathcal{R})^\rho$ if and only if:

(*i*) $\forall \alpha \in AType_U \cup AName_{IO}. \forall l \in APLev. \forall C \in T/\mathcal{B}^\rho$

$$Rate_{CL,\rho}(t_1, \alpha, l, C) = Rate_{CL,\rho}(t_2, \alpha, l, C)$$

(*ii*) $\forall \alpha = a?x \in AType_I. \forall v \in Val. \forall l \in APLev. \forall C \in T/\mathcal{B}^{\rho[x \mapsto v]}$

$$Rate_{CL,\rho}(t_1, \alpha, l, C) = Rate_{CL,\rho}(t_2, \alpha, l, C)$$

$$\frac{E \xrightarrow{\alpha,\tilde{\lambda},\beta,\sigma'} E'}{\langle E,\sigma\rangle \xrightarrow{\alpha,\tilde{\lambda}}_{CL,\rho} \langle E',\sigma'\sigma\rangle} \quad \alpha \in AType_U \cup AName_{IO} \wedge \rho(\beta\sigma) = true$$

$$\frac{E \xrightarrow{a?x,\tilde{\lambda},true,\sigma'} E'}{\langle E,\sigma\rangle \xrightarrow{a?z,\tilde{\lambda}}_{CL,\rho} \langle E',\sigma'\sigma[x \mapsto z]\rangle} \quad z \in Var \text{ fresh}$$

$$\frac{E \xrightarrow{a!e,\tilde{\lambda},true,\sigma'} E'}{\langle E,\sigma\rangle \xrightarrow{a!\rho(e\sigma),\tilde{\lambda}}_{CL,\rho} \langle E',\sigma'\sigma\rangle}$$

Table 2: Rule for EMPA$_{vp}$ concrete late semantics

(iii) $\forall \alpha \in AType'_O. \forall l \in APLev. \forall C \in T/\mathcal{B}^\rho$
$$Rate_{CL,\rho}(t_1,\alpha,l,C) = Rate_{CL,\rho}(t_2,\alpha,l,C)$$

We say that $\mathcal{R}$ is a *strong concrete late extended Markovian bisimulation (strong CLEMB)* if and only if $\mathcal{R} \subseteq CLB(\mathcal{R})$. ■

Definition 3.4 We define the *strong concrete late extended Markovian bisimulation equivalence (strong CLEMBE)* with respect to $\rho \in Eval$, denoted $\sim^\rho_{CLEMB}$, as the ρ-th component of the greatest fixed point of functional CLB. ■

Since functional CLB is componentwise monotonic, the definition above is well-founded. Furthermore, $\sim^\rho_{CLEMB}$ turns out to be an equivalence relation.

In the symbolic late case, boolean guards and output expressions are carried in transitions instead of getting evaluated, so the resulting semantic model is more abstract because it does not refer to evaluations.

Definition 3.5 The *symbolic late integrated interleaving semantics* of $E \in \mathcal{G}_{vp}$ is the SLTS
$$\mathcal{I}^{SL}_{vp}[\![E]\!] = (S_{E,SL}, Act''_{vp} \times BExp, \longrightarrow_{E,SL}, \langle sym(E), setpar(E)\rangle)$$
where $S_{E,SL}$ is the set of states reachable from $\langle sym(E), setpar(E)\rangle$, and $\longrightarrow_{E,SL}$ is the restriction of $\longrightarrow_{SL}$ defined in Table 3 to $S_{E,SL,} \times (Act''_{vp} \times BExp) \times S_{E,SL}$. ■

Definition 3.6 We define partial function $Rate_{SL} : (Term \times AType'_{vp} \times APLev_{vp} \times Term) \longrightarrow ARate_{vp}$ by
$$Rate_{SL}(t,\alpha,l,C) = Min\{\!|\, \tilde{\lambda} \mid t \xrightarrow{\alpha,\tilde{\lambda},\beta}_{SL} t' \wedge PL(<\alpha,\tilde{\lambda}>) = l \wedge t' \in C \,|\!\}$$
and partial function $Bool_{SL} : (Term \times AType'_{vp} \times APLev_{vp} \times Term) \longrightarrow \mathcal{P}(BExp)$ by
$$Bool_{SL}(t,\alpha,l,C) = \{\beta \mid t \xrightarrow{\alpha,\tilde{\lambda},\beta}_{SL} t' \wedge PL(<\alpha,\tilde{\lambda}>) = l \wedge t' \in C\}$$ ■

Definition 3.7 Let $\mathcal{R} = \{\mathcal{B}^\beta \subseteq Term \times Term \mid \mathcal{B}^\beta \text{ equivalence } \wedge\, \beta \in BExp\}$, and let $SLB(\mathcal{R}) = \{SLB(\mathcal{R})^\beta \subseteq Term \times Term \mid SLB(\mathcal{R})^\beta \text{ equivalence } \wedge\, \beta \in BExp\}$ be defined by $(t_1,t_2) \in SLB(\mathcal{R})^\beta$ if and only if there exists $B \subseteq BExp$ such that for all $\beta' \in B$:

$$\frac{E \xrightarrow{\alpha,\tilde{\lambda},\beta,\sigma'} E'}{\langle E,\sigma\rangle \xrightarrow{\alpha,\tilde{\lambda},\beta\sigma}_{SL} \langle E',\sigma'\sigma\rangle} \quad \alpha \in AType_U \cup AName_{IO}$$

$$\frac{E \xrightarrow{a?x,\tilde{\lambda},true,\sigma'} E'}{\langle E,\sigma\rangle \xrightarrow{a?z,\tilde{\lambda},true}_{SL} \langle E',\sigma'\sigma[x \mapsto z]\rangle} \quad z \in Var \text{ fresh}$$

$$\frac{E \xrightarrow{a!e,\tilde{\lambda},true,\sigma'} E'}{\langle E,\sigma\rangle \xrightarrow{a!e\sigma,\tilde{\lambda},true}_{SL} \langle E',\sigma'\sigma\rangle}$$

Table 3: Rule for EMPA$_{vp}$ symbolic late semantics

(i) $\forall \alpha \in AType_U \cup AName_{IO}. \forall l \in APLev. \forall C \in T/\mathcal{B}^{\beta'}$
$$Rate_{SL}(t_1,\alpha,l,C) = Rate_{SL}(t_2,\alpha,l,C)$$
$$\beta \wedge \bigwedge Bool_{SL}(t_1,\alpha,l,C) \wedge \bigwedge Bool_{SL}(t_2,\alpha,l,C) \Longrightarrow \bigvee B$$
$$\beta' \Longrightarrow \bigwedge Bool_{SL}(t_1,\alpha,l,C) \wedge \bigwedge Bool_{SL}(t_2,\alpha,l,C)$$

(ii) $\forall \alpha \in AType_I. \forall l \in APLev. \forall C \in T/\mathcal{B}^{\beta'}$
$$Rate_{SL}(t_1,\alpha,l,C) = Rate_{SL}(t_2,\alpha,l,C)$$
$$\beta \Longrightarrow \bigvee B$$

(iii) $\forall \alpha \in AName_{IO}. \forall e_1,e_2 \in Exp. \forall l \in APLev. \forall C \in T/\mathcal{B}^{\beta'}$
$$Rate_{SL}(t_1,a!e_1,l,C) = Rate_{SL}(t_2,a!e_2,l,C)$$
$$\beta \Longrightarrow \bigvee B$$
$$\beta' \Longrightarrow e_1 = e_2$$

We say that $\mathcal{R}$ is a *strong symbolic late extended Markovian bisimulation (strong SLEMB)* if and only if $\mathcal{R} \subseteq SLB(\mathcal{R})$. ■

Definition 3.8 We define the *strong symbolic late extended Markovian bisimulation equivalence (strong SLEMBE)* with respect to $\beta \in BExp$, denoted $\sim^{\beta}_{SLEMB}$, as the β-th component of the greatest fixed point of functional SLB. ■

Since functional SLB is componentwise monotonic, the definition above is well-founded. Furthermore, $\sim^{\beta}_{SLEMB}$ turns out to be an equivalence relation.

We conclude by showing that the strong SLEMBE captures the strong CLEMBE. As recognized in [6, 8], this has the advantage that the checking of strong CLEMBE can be reduced to the checking of strong SLEMBE on the more abstract (hence frequently finite) semantic model.

Lemma 3.9 We have that:

(i) $\langle E,\sigma\rangle \xrightarrow{\alpha,\tilde{\lambda}}_{CL,\rho} \langle E',\sigma'\rangle$ if and only if $\langle E,\sigma\rangle \xrightarrow{\alpha,\tilde{\lambda},\beta}_{SL} \langle E',\sigma'\rangle$ and $\rho(\beta) = true$, where $\alpha \in AType_U \cup AName_{IO}$.

(*ii*) $\langle E,\sigma\rangle \xrightarrow{\alpha,\tilde{\lambda}}_{CL,\rho} \langle E',\sigma'\rangle$ if and only if $\langle E,\sigma\rangle \xrightarrow{\alpha,\tilde{\lambda},true}_{SL} \langle E',\sigma'\rangle$, where $\alpha \in AType_I$.

(*iii*) $\langle E,\sigma\rangle \xrightarrow{a!v,\tilde{\lambda}}_{CL,\rho} \langle E',\sigma'\rangle$ if and only if $\langle E,\sigma\rangle \xrightarrow{a!e,\tilde{\lambda},true}_{SL} \langle E',\sigma'\rangle$ and $\rho(e) = v$. ■

Proposition 3.10 Let $\mathcal{R} = \{\mathcal{B}^\beta \subseteq Term \times Term \mid \mathcal{B}^\beta$ equivalence $\wedge\, \beta \in BExp\}$ be a strong SLEMB. If $\mathcal{R}' = \{\mathcal{B}^\rho \subseteq Term \times Term \mid \mathcal{B}^\rho$ equivalence $\wedge\, \rho \in Eval\}$ is such that $\mathcal{B}^\rho = \{(t_1,t_2) \in Term \times Term \mid \exists \beta \in BExp.\, \rho(\beta) = true \wedge (t_1,t_2) \in \mathcal{B}^\beta\}$, then $\mathcal{R}'$ is a strong CLEMB.

Proof We must prove that $\mathcal{R}' \subseteq CLB(\mathcal{R}')$. Given $(t_1,t_2) \in \mathcal{B}^\rho$, i.e. assumed that there exists $\beta \in BExp$ such that $\rho(\beta) = true \wedge (t_1,t_2) \in \mathcal{B}^\beta$, let us demonstrate that $(t_1,t_2) \in CLB(\mathcal{R}')^\rho$.

- Let $\alpha \in AType_U \cup AName_{IO}$. Given $l \in APLev$ and $C \in Term/\mathcal{B}^\rho$, we must prove that
$$Rate_{CL,\rho}(t_1,\alpha,l,C) = Rate_{CL,\rho}(t_2,\alpha,l,C)$$
Since $(t_1,t_2) \in \mathcal{B}^\beta$ and $\mathcal{R}$ is a strong SLEMB, there must exist $B \subseteq BExp$ such that for every $\beta' \in B$ and $C' \in Term/\mathcal{B}^{\beta'}$
$$\begin{array}{l} Rate_{SL}(t_1,\alpha,l,C') = Rate_{SL}(t_2,\alpha,l,C') \\ \beta \wedge \bigwedge Bool_{SL}(t_1,\alpha,l,C') \wedge \bigwedge Bool_{SL}(t_2,\alpha,l,C') \Longrightarrow \bigvee B \\ \beta' \Longrightarrow \bigwedge Bool_{SL}(t_1,\alpha,l,C') \wedge \bigwedge Bool_{SL}(t_2,\alpha,l,C') \end{array}$$
Since $\rho(\beta) = true$ by initial hypothesis, $\rho(\bigwedge Bool_{SL}(t_1,\alpha,l,C') \wedge \bigwedge Bool_{SL}(t_2,\alpha,l,C')) = true$ by Lemma 3.9, and $\beta \wedge \bigwedge Bool_{SL}(t_1,\alpha,l,C') \wedge \bigwedge Bool_{SL}(t_2,\alpha,l,C') \Longrightarrow \bigvee B$, we have $\rho(\bigvee B) = true$ thus there must exist $\beta' \in B$ such that $\rho(\beta') = true$. Therefore, the equivalence classes with respect to $\mathcal{B}^\rho$ are unions of equivalence classes with respect to $\mathcal{B}^{\beta'}$ (if $(u_1,u_2) \in \mathcal{B}^{\beta'}$, from $\rho(\beta') = true$ and the definition of $\mathcal{B}^\rho$ it follows that $(u_1,u_2) \in \mathcal{B}^\rho$). The result then follows by virtue of Lemma 3.9.

- Let $\alpha = a?x \in AType_I$. Given $l \in APLev$, $v \in Val$ and $C \in Term/\mathcal{B}^{\rho[x\mapsto v]}$, we must prove that
$$Rate_{CL,\rho}(t_1,\alpha,l,C) = Rate_{CL,\rho}(t_2,\alpha,l,C)$$
Since $(t_1,t_2) \in \mathcal{B}^\beta$ and $\mathcal{R}$ is a strong SLEMB, there must exist $B \subseteq BExp$ such that for every $\beta' \in B$ and $C' \in Term/\mathcal{B}^{\beta'}$
$$\begin{array}{l} Rate_{SL}(t_1,\alpha,l,C') = Rate_{SL}(t_2,\alpha,l,C') \\ \beta \Longrightarrow \bigvee B \end{array}$$
Since $\rho(\beta) = true$ by initial hypothesis and $\beta \Longrightarrow \bigvee B$, we have $\rho(\bigvee B) = true$ thus there must exist $\beta' \in B$ such that $\rho(\beta') = true$. Since x is a fresh variable, we obtain $\rho[x \mapsto v](\beta') = true$. Therefore, the equivalence classes with respect to $\mathcal{B}^{\rho[x\mapsto v]}$ are unions of equivalence classes with respect to $\mathcal{B}^{\beta'}$. The result then follows by virtue of Lemma 3.9.

- Let $\alpha = a!v \in AType'_O$. Given $l \in APLev$ and $C \in Term/\mathcal{B}^\rho$, we must prove that
$$Rate_{CL,\rho}(t_1,\alpha,l,C) = Rate_{CL,\rho}(t_2,\alpha,l,C)$$
Since $(t_1,t_2) \in \mathcal{B}^\beta$ and $\mathcal{R}$ is a strong SLEMB, there must exist $B \subseteq BExp$ such that for every $\beta' \in B$, $e_1,e_2 \in Exp$ and $C' \in Term/\mathcal{B}^{\beta'}$
$$\begin{array}{l} Rate_{SL}(t_1,a!e_1,l,C') = Rate_{SL}(t_2,a!e_2,l,C') \\ \beta \Longrightarrow \bigvee B \\ \beta' \Longrightarrow e_1 = e_2 \end{array}$$
Since $\rho(\beta) = true$ by initial hypothesis and $\beta \Longrightarrow \bigvee B$, we have $\rho(\bigvee B) = true$ thus there must exist $\beta' \in B$ such that $\rho(\beta') = true$. Since $\beta' \Longrightarrow e_1 = e_2$, we obtain $\rho(e_1) = \rho(e_2) = v$. Therefore, the equivalence classes with respect to $\mathcal{B}^\rho$ are unions of equivalence classes with respect to $\mathcal{B}^{\beta'}$. The result then follows by virtue of Lemma 3.9. ■

Proposition 3.11 Let $\mathcal{R} = \{\mathcal{B}^\rho \subseteq Term \times Term \mid \mathcal{B}^\rho$ equivalence $\wedge\, \rho \in Eval\}$ be a strong CLEMB. If $\mathcal{R}' = \{\mathcal{B}^\beta \subseteq Term \times Term \mid \mathcal{B}^\beta$ equivalence $\wedge\, \beta \in BExp\}$ is such that $\mathcal{B}^\beta = \{(t_1,t_2) \in Term \times Term \mid \forall \rho \in Eval.\, \rho(\beta) = true \Longrightarrow (t_1,t_2) \in \mathcal{B}^\rho\}$, then $\mathcal{R}'$ is a strong SLEMB.

Proof We must prove that $\mathcal{R}' \subseteq SLB(\mathcal{R}')$. Given $(t_1,t_2) \in \mathcal{B}^\beta$, i.e. assumed that $(t_1,t_2) \in \mathcal{B}^\rho$ for any $\rho \in Eval$ such that $\rho(\beta) = true$, let us demonstrate that $(t_1,t_2) \in SLB(\mathcal{R}')^\beta$.

- Let $\alpha \in AType_U \cup AName_{IO}$. We must prove that there exists $B \subseteq BExp$ such that for every $\beta' \in B$, $l \in APLev$ and $C \in T/\mathcal{B}^{\beta'}$

$$Rate_{SL}(t_1, \alpha, l, C) = Rate_{SL}(t_2, \alpha, l, C)$$
$$\beta \wedge \bigwedge Bool_{SL}(t_1, \alpha, l, C) \wedge \bigwedge Bool_{SL}(t_2, \alpha, l, C) \Longrightarrow \bigvee B$$
$$\beta' \Longrightarrow \bigwedge Bool_{SL}(t_1, \alpha, l, C) \wedge \bigwedge Bool_{SL}(t_2, \alpha, l, C)$$

  Let $B = \{\beta' = \beta_1 \wedge \beta_2 \in BExp \mid t_1 \stackrel{\alpha,\tilde{\lambda}_1,\beta_1}{\longrightarrow}_{SL} t_1' \wedge t_2 \stackrel{\alpha,\tilde{\lambda}_2,\beta_2}{\longrightarrow}_{SL} t_2' \wedge PL(<\alpha, \tilde{\lambda}_1>) = PL(<\alpha, \tilde{\lambda}_2>) = l \wedge (t_1', t_2') \in \mathcal{B}^\rho\}$. We first check that $\beta \wedge \bigwedge Bool_{SL}(t_1, \alpha, l, C) \wedge \bigwedge Bool_{SL}(t_2, \alpha, l, C) \Longrightarrow \bigvee B$. Assuming $\rho(\beta \wedge \bigwedge Bool_{SL}(t_1, \alpha, l, C) \wedge \bigwedge Bool_{SL}(t_2, \alpha, l, C)) = true$, it follows that $\rho(\beta) = true$ hence $(t_1, t_2) \in \mathcal{B}^\rho$. Since $\mathcal{R}$ is a strong CLEMB, for every $C' \in Term/\mathcal{B}^\rho$

$$Rate_{CL,\rho}(t_1, \alpha, l, C') = Rate_{CL,\rho}(t_2, \alpha, l, C')$$

  Then, from Lemma 3.9 and the definition of B it follows that $\rho(\beta') = true$ for any $\beta' \in B$, hence $\rho(\bigvee B) = true$. Furthemore, the equivalence classes with respect to $\mathcal{B}^{\beta'}$ are unions of equivalence classes with respect to $\mathcal{B}^\rho$ (if $(u_1, u_2) \in \mathcal{B}^\rho$, from $\rho(\beta') = true$ and the definition of $\mathcal{B}^{\beta'}$ it follows that $(u_1, u_2) \in \mathcal{B}^{\beta'}$), so by virtue of Lemma 3.9 we obtain $Rate_{SL}(t_1, \alpha, l, C) = Rate_{SL}(t_2, \alpha, l, C)$. Finally, $\beta' \Longrightarrow \bigwedge Bool_{SL}(t_1, \alpha, l, C) \wedge \bigwedge Bool_{SL}(t_2, \alpha, l, C)$ is straightforward.

- Let $\alpha = a?x \in AType_I$. We must prove that there exists $B \subseteq BExp$ such that for every $\beta' \in B$, $l \in APLev$ and $C \in Term/\mathcal{B}^{\beta'}$

$$Rate_{SL}(t_1, \alpha, l, C) = Rate_{SL}(t_2, \alpha, l, C)$$
$$\beta \Longrightarrow \bigvee B$$

  Let $B = \{\beta' = \beta_1 \wedge \beta_2 \in BExp \mid t_1 \stackrel{\alpha,\tilde{\lambda}_1,\beta_1}{\longrightarrow}_{SL} t_1' \wedge t_2 \stackrel{\alpha,\tilde{\lambda}_2,\beta_2}{\longrightarrow}_{SL} t_2' \wedge PL(<\alpha, \tilde{\lambda}_1>) = PL(<\alpha, \tilde{\lambda}_2>) = l \wedge (t_1', t_2') \in \mathcal{B}^{\rho[x \mapsto v]}\}$ where $v \in Val$. We first check that $\beta \Longrightarrow \bigvee B$. Assuming $\rho(\beta) = true$, we have $(t_1, t_2) \in \mathcal{B}^\rho$. Since $\mathcal{R}$ is a strong CLEMB, for every $C' \in Term/\mathcal{B}^{\rho[x \mapsto v]}$

$$Rate_{CL,\rho}(t_1, \alpha, l, C') = Rate_{CL,\rho}(t_2, \alpha, l, C')$$

  Then, from Lemma 3.9 and the definition of B it follows that $\rho(\beta') = true$ for any $\beta' \in B$, hence $\rho(\bigvee B) = true$. Furthemore, the equivalence classes with respect to $\mathcal{B}^{\beta'}$ are unions of equivalence classes with respect to $\mathcal{B}^{\rho[x \mapsto v]}$, so by virtue of Lemma 3.9 we obtain $Rate_{SL}(t_1, \alpha, l, C) = Rate_{SL}(t_2, \alpha, l, C)$.

- Let $a \in AName_{IO}$. We must prove that there exists $B \subseteq BExp$ such that for every $\beta' \in B$, $e_1, e_2 \in Exp$, $l \in APLev$ and $C \in Term/\mathcal{B}^{\beta'}$

$$Rate_{SL}(t_1, \alpha, l, C) = Rate_{SL}(t_2, \alpha, l, C)$$
$$\beta \Longrightarrow \bigvee B$$
$$\beta' \Longrightarrow e_1 = e_2$$

  Let $B = \{\beta' = \beta_1 \wedge \beta_2 \in BExp \mid t_1 \stackrel{a!e_1,\tilde{\lambda}_1,\beta_1}{\longrightarrow}_{SL} t_1' \wedge t_2 \stackrel{a!e_2,\tilde{\lambda}_2,\beta_2}{\longrightarrow}_{SL} t_2' \wedge PL(<a!e_1, \tilde{\lambda}_1>) = PL(<a!e_2, \tilde{\lambda}_2>) = l \wedge (t_1', t_2') \in \mathcal{B}^\rho \wedge \rho(e_1) = \rho(e_2) = v\}$ where $v \in Val$. We first check that $\beta \Longrightarrow \bigvee B$. Assuming $\rho(\beta) = true$, we have $(t_1, t_2) \in \mathcal{B}^\rho$. Since $\mathcal{R}$ is a strong CLEMB, for every $C' \in Term/\mathcal{B}^\rho$

$$Rate_{CL,\rho}(t_1, a!v, l, C') = Rate_{CL,\rho}(t_2, a!v, l, C')$$

  Then, from Lemma 3.9 and the definition of B it follows that $\rho(\beta') = true$ for any $\beta' \in B$, hence $\rho(\bigvee B) = true$. Furthemore, the equivalence classes with respect to $\mathcal{B}^{\beta'}$ are unions of equivalence classes with respect to $\mathcal{B}^\rho$, so by virtue of Lemma 3.9 we obtain $Rate_{SL}(t_1, a!e_1, l, C) = Rate_{SL}(t_2, a!e_2, l, C)$. Finally, $\beta' \Longrightarrow e_1 = e_2$ is straightforward. ■

Theorem 3.12 $t_1 \sim^\beta_{SLEMB} t_2$ if and only if $t_1 \sim^\rho_{CLEMB} t_2$ for any $\rho \in Eval$ such that $\rho(\beta) = true$. ■

4 Exploiting symmetries: the alternating bit protocol

In this section we report an example which should illustrate the advantage that can be gained by exploiting the inherent parametricity of value passing when analyzing symmetric systems. The example is concerned with the alternating bit protocol. This protocol has been modeled in [4] by using EMPA since it is not

necessary to resort to value passing. However, we shall see now that it is extremely advantageous to resort to value passing in this case.

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

In order to compositionally model such a protocol, like in [4] we figure out to deal with four interacting entities: *Sender*, *Receiver*, $Line_m$, $Line_a$. The interaction between *Sender* and $Line_m$ is described by action type *tm* standing for "transmit message", the interaction between $Line_m$ and *Receiver* is described by action type *dm* standing for "deliver message", the interaction between *Receiver* and $Line_a$ is described by action type *ta* standing for "transmit acknowledgement", and the interaction between $Line_a$ and *Sender* is described by action type *da* standing for "deliver acknowledgement":

$$\begin{array}{rcl} ABP(b) & \triangleq & Sender(b) \parallel_S (Line_m \parallel_\emptyset Line_a) \parallel_R Receiver(b) \\ S & = & \{tm, da\} \\ R & = & \{dm, ta\} \end{array}$$

Since EMPA allows us to express only exponentially distributed durations as well as zero durations, we suppose that the length of a message and the length of an acknowledgement are exponentially distributed, so that the message/acknowledgement transmission/propagation/delivery times are exponentially distributed. However, the three phases given by message/acknowledgement transmission/propagation/delivery are temporally overlapped, i.e. they constitute a pipeline. As a consequence, in order to determine correctly the time taken by a message/acknowledgement to reach *Receiver*/*Sender*, we model actions related to transmission and delivery as immediate and we associate the actual timing (i.e. the duration of the slowest stage of the pipeline) with actions related to propagation: we assume that the message propagation time is exponentially distributed with rate δ, and the acknowledgement propagation time is exponentially distributed with rate γ.

Concerning *Sender*, its local activities are described by action types *gm* standing for "generate message" and *to* standing for "timeout":

$$\begin{array}{rcl} Sender(b) & \triangleq & <gm, \lambda>.<tm!b, \infty_{1,1}>.Sender'(b), \\ Sender'(b) & \triangleq & <da?x, *>.Sender''(b, x) + <to, \theta>.Sender'''(b), \\ Sender''(b, x) & \triangleq & \mathit{if}\ (x = b, p_{pt})\ \mathit{then}\ Sender(\neg b)\ \mathit{else}\ Sender'(b), \\ Sender'''(b) & \triangleq & <tm!b, \infty_{1,1}>.Sender'(b) + <da?x, *>.Sender''(b, x) \end{array}$$

Here the assumption is that the message generation time is exponentially distributed with rate λ, and that the timeout period is exponentially distributed with rate θ: this is not realistic, but EMPA does not enable us to express deterministic durations, and a Markovian analysis would not be possible otherwise. Note that p_{pt} expresses (a guess about) the probability of a premature timeout, i.e. a timeout caused by slow propagation rate rather than actual loss.

The only local activities of $Line_m$ and $Line_a$ are described by action types *pm* standing for "propagate message" and *pa* standing for "propagate acknowledgement" respectively:

$$\begin{array}{rcl} Line_m & \triangleq & <tm?x, *>.<pm, \delta>.(<\tau, \infty_{1,1-p_{loss}}>.<dm!x, \infty_{1,1}>.Line_m + <\tau, \infty_{1,p_{loss}}>.Line_m) \\ Line_a & \triangleq & <ta?x, *>.<pa, \gamma>.(<\tau, \infty_{1,1-p_{loss}}>.<da!x, \infty_{1,1}>.Line_a + <\tau, \infty_{1,p_{loss}}>.Line_a) \end{array}$$

Notice that the loss probability p_{loss} has been easily expressed by means of the weights of immediate actions.

Finally, the only local activity of *Receiver* is described by action type *cm* standing for "consume message":

$$Receiver(b) \stackrel{\Delta}{=} <dm?x, *>.if\ (x = b, p_{pt})\ then <cm, \infty_{1,1}>.<ta!x, \infty_{1,1}>.Receiver(\neg b)$$
$$else <ta!x, \infty_{1,1}>.Receiver(b)$$

Observe that message consumption is an immediate action in that irrelevant from the performance viewpoint.

This description of the protocol is more compact than the one in [4] and, most importantly, the underlying state space is smaller: we have about 150 states instead of about 300. This is due to the fact that by means of our treatment of constant parameters based on lookahead we are able to exploit all the parametricity inherent to value passing so as to lump together at semantic model construction time the ways in which the protocol works according to the two possible values of the alternating bit which labels messages and acknowledgements.

5 Conclusion

In this paper we have extended the theory developed for EMPA in order to cope with value passing. This has been accomplished by adapting to our stochastic framework the proposal of [6, 8] relying on symbolic labeled transition systems and symbolic bisimulations through suitable semantic rules based on lookahead. Though based on EMPA, our work on value passing can be smoothly adapted to other process algebras.

The careful treatment of constant parameters by means of lookahead has proven to be very advantageous since it allows compact models to be produced and system symmetries to be detected during the construction of models themselves. Making such compact models available should also be profitable for simulation purposes. In [2] it has been recognized that the simulative analysis of systems described with EMPA is feasible because the (possibly huge or infinite) underlying semantic models can be built by need at simulation time. In the case of EMPA_{vp}, it might be convenient to generate the whole (frequently compact finite) semantic model a priori so as to reduce the time needed to carry out the simulation runs.

Concerning future work, it would be interesting to allow for variable rates and also variable rewards in the case of EMPA_r [3], as well as to develop algorithms to check strong SLEMBE by following [6, 8].

Acknowledgements

This research has been partially funded by MURST, CNR and Esprit Project n. 8130 LOMAPS.

References

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

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

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

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

[5] M. Bernardo, R. Gorrieri, *"A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time"*, to appear in Theoretical Computer Science, 1997

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

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

[8] H. Lin, *"Symbolic Transition Graph with Assignment"*, in Proc. of the *7th Int. Conf. on Concurrency Theory (CONCUR '96)*, LNCS 1119:50-65, Pisa (Italy), 1996

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