Technical Report CoSBi 11/2008

Simulation of non-Markovian Processes in BlenX

Davide Prandi
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
prandi@cosbi.eu

Corrado Priami
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
and
Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, Italy
priami@cosbi.eu

Alessandro Romanel
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
and
Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, Italy
romanel@cosbi.eu

Simulation of non-Markovian Processes in BlenX

D. Prandi[1], C. Priami[1,2], and A. Romanel[1,2]

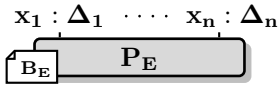
[1] The Microsoft Research - University of Trento Centre for Computational and Systems Biology
[2] Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, Italy
{prandi,priami,romanel}@cosbi.eu

Abstract. BlenX is a programming language designed for modeling entities that can change their behavior in response to external stimuli. The actual framework assumes interactions being exponentially distributed, i.e., an underlying Markov process is associated with BlenX programs. In this paper we relax the Markov assumption by providing formal tools for managing non-Markovian processes within BlenX and we show experimental evidences of the effectiveness of the approach.

1 Introduction

The strength of the interaction between two entities is usually thought as a two value logic, i.e., the interaction is possible or not. For instance, two CCS [1] processes interact iff they can perform complementary actions (input and output) on the same channel. The paradigm *communication by compatibility* [2], recently proposed with the process calculus Beta-binders [3], introduces a "fuzzy" vision, and lets interactions depend on a notion of compatibility of the involved parties. For instance, web-services use XML to describe provided services, and interactions are disciplined by a notion of compatibility between XML descriptions [4]. Also, biological interactions depend on structural and chemical complementarity of molecules, called affinity [5].

BlenX [6] is inspired to Beta-binders and it is designed for modeling entities that can change their behavior in response to external stimuli. A general entity E is depicted as a box $\mathbf{B_E}$:



The program $\mathbf{P_E}$ is written in a process calculi style language, and allows to control the behavior of $\mathbf{B_E}$. In particular, $\mathbf{P_E}$ activates proper replies to external signals caught by interaction sites $\mathbf{x_i} : \mathbf{\Delta_i}$. Type $\mathbf{\Delta_i}$ discriminates among allowed and disallowed interactions, mimicking interaction mechanisms based on compatibility.

The BetaWB framework [7] is a computational tool that supports textual and visual programming with BlenX. The BetaWB can be seen as an *in-silico* laboratory, where (in-silico) experiments can be designed (i.e., a BlenX program is written), simulated and analyzed. The quantitative component of the experiments is guaranteed by the stochastic capability of BlenX, on the line of [8], where a continuous-time Markov process is taken as foundational quantitative model. The goal of this paper is to provide the

formal tools for managing non-Markovian processes within BlenX. Our motivations are *flexibility* and *abstraction*. Assuming, as in Markov processes, that a random variable follows the negative exponential distribution with parameter λ, fixes expected value to λ^{-1} and variance to λ^{-2}, thus limiting the flexibility of the choice about variability in the stochastic model [9]. It is also the case that not all the quantitative data about the basic steps of a Markov process are available, and many steps are abstracted as a single step. Since the composition of negative exponential distributions is not exponentially distributed, general distributions are required to have better abstractions.

We start in Sect. 2 by providing a *proved* reduction semantics for a core subset of BlenX, following the work in [10]. Proved reduction semantics is a rephrase of enhanced operational semantics [11], a conceptual tool for describing the behavior of concurrent systems. In particular, the transitions of the system have rich labels that permit to recover information about the *causal* relation between transitions. A seminal work about causality and Beta-binders can be found in [12]. Here we introduce the notion of *dependency* in Sect. 2.1 to adapt the idea of causality to BlenX. Dependency is then employed in Sect. 2.2 to support *enabling memory discipline* [13], that is, the stochastic distribution of the execution of a transition θ must be influenced by all the transitions fired from the states where θ was firstly enabled. We can therefore compute the right stochastic distribution of a BlenX transition. Sect. 3 proposes some BetaWB simulations of non-Markovian processes. Sect. 4 concludes the paper with some final remarks.

2 Formal Treatment

In this section we provide a proved operational semantics in the style of [10] for a subset of BlenX. In particular, for the sake of clarity, we do not consider events [7].

A *binder* has either the form $\beta(x,\ \Gamma)$, or $\beta^h(x,\ \Gamma)$, where the name x is the *subject* of the binder, and $\Gamma \in \mathcal{T}$ is the *type* of x. The domain $\mathcal{T}$ can be arbitrarily instanced under the proviso that a *symmetric compatibility relation* is also defined, and that the predicate $\alpha(\_,\_) : \mathcal{T} \times \mathcal{T} \to \mathbb{R}^+$, which returns a value greater that 0 iff its argument types are compatible, is decidable. Example of domain $\mathcal{T}$ can be found in [2]. Intuitively, a binder $\beta(x,\ \Gamma)$ represents an active (potentially interacting) site of a box. If a binder has been hidden to prevent interactions, it is represented as $\beta^h(x,\ \Gamma)$. Metavariable β^+ ranges over $\{\beta, \beta^h\}$, and $\Delta, \Delta_1, \ldots, \Gamma, \Gamma_1, \ldots$ range over site types. *Interfaces* are generated by the following grammar:

$$\boldsymbol{I} ::= \beta^+(x,\ \Gamma) \ \Big|\ \beta^+(x,\ \Gamma)\,\boldsymbol{I}.$$

An interface is well-formed when the subjects and the types of its binders are all distinct. We will work only with well-formed interfaces. Auxiliary functions $\mathsf{sub}(\boldsymbol{I})$ and $\mathsf{typ}(\boldsymbol{I})$ give the set of subjects and types of an interface $\boldsymbol{I}$, respectively.

We assume two disjoint countable infinite sets: $\mathcal{N}$ of *names* ranged over by $x, y, z, \ldots$ and $\mathcal{S}$ of *delays* ranged over by $\tau_1, \tau_2, \tau_3, \ldots$. *Processes* are defined by the following:

$$\begin{array}{lll} P & ::= \mathsf{nil} \ \Big|\ M \ \Big|\ P \mid P \ \Big|\ \mathsf{rep}\,\pi.\,P & \qquad M ::= \pi.\,P \ \Big|\ M + M \\ \pi & ::= x!v \ \Big|\ y?w \ \Big|\ \pi_\beta & \\ \pi_\beta & ::= \tau_i \ \Big|\ \mathsf{hide}(x) \ \Big|\ \mathsf{unhide}(x) \ \Big|\ \mathsf{expose}(x,\ \Gamma) \ \Big|\ \mathsf{ch}(x,\ \Gamma) & \end{array}$$

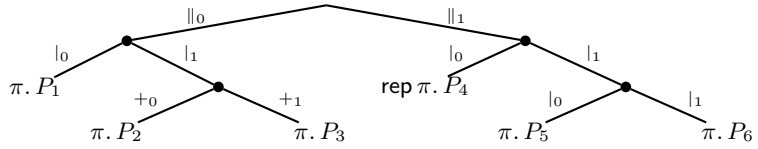


Fig. 1. The tree of the sequential processes within the boxes in the system (1).

Process nil, prefixes output $x!v$, input $y?w$, and delay τ_i, and operators of parallel composition $|$ and choice $+$ work as in π-calculus. Guarded replication $\mathsf{rep}\,\pi.\, P$ was introduced in [14] and spawns a single copy of P if prefix π is consumed. The prefixes $\mathsf{hide}(x)$, $\mathsf{unhide}(x)$, $\mathsf{expose}(x,\, \Gamma)$, and $\mathsf{ch}(x,\, \Gamma)$ manipulate the interface of a box and will be further commented on when the semantics will be introduced. Finally, *systems* are defined by the following:

$$B ::= \mathsf{Nil} \;\Big|\; \boldsymbol{I}\big[\, P \,\big] \;\Big|\; B \parallel B \,.$$

The actual syntax of BlenX does not univocally identify which actions are active in a given box. Consider, for instance, the following

$$\boldsymbol{I}_0\big[\, \pi.\, P_1 \mid (\pi.\, P_2 + \pi.\, P_3) \,\big] \parallel \boldsymbol{I}_1\big[\, \mathsf{rep}\,\pi.\, P_4 \mid (\pi.\, P_5 \mid \pi.\, P_6) \,\big] \tag{1}$$

A notion of address is needed to distinguish among the different instances of the prefix π in (1). An address identifies a sequential component of a box B, namely, a process with a prefix as a top-level operator. In particular, a *b-address* $\vartheta^b \in \{\|_0, \|_1\}^*$, λ^b is the empty one, identifies a box within a system, while a *p-address* $\vartheta^p \in \{|_0, |_1, +_0, +_1\}^*$, λ^p is the empty one, points to a specific sequential component of a process. Consider the abstract syntax tree of (1) in Fig. 1, built assuming parallel composition and choice as main operators. The leaves of the tree are the active processes. The label of the path from the root to a leaf is the address, e.g., process $\pi.\, P_3$ has address $\|_0\, |_1\, +_1$. Once a tree of a system is fixed, an address uniquely identifies an active action.

Systems are decorated with addresses by a labeling function $\mathcal{T}(\_)$. An auxiliary operator $\triangleright$ that distributes addresses among the sequential components is defined:

- $\vartheta^p \triangleright \mathsf{nil} = \mathsf{nil}$
- $\vartheta_1^p \triangleright (\vartheta_2^p\, \pi.\, P) = \vartheta_1^p\, \vartheta_2^p\, \pi.\, (\vartheta_1^p \triangleright P)$
- $\vartheta^p \triangleright (M_1 + M_2) = \vartheta^p \triangleright M_1 + \vartheta^p \triangleright M_2$
- $\vartheta^p \triangleright (P_1 \mid P_2) = \vartheta^p \triangleright P_1 \mid \vartheta^p \triangleright P_2$
- $\vartheta_1^p \triangleright \mathsf{rep}\, \vartheta_2^p\, \pi.\, P = \mathsf{rep}\, \vartheta_1^p\, \vartheta_2^p\, \pi.\, P$
- $\vartheta^b \triangleright \mathsf{Nil} = \mathsf{Nil}$
- $\vartheta^b \triangleright \boldsymbol{I}\big[\, P \,\big] = \vartheta^b\, \boldsymbol{I}\big[\, P \,\big]$
- $\vartheta^b \triangleright (B_1 \parallel B_2) = \vartheta^b \triangleright B_1 \parallel \vartheta^b \triangleright B_2$

The operator behaves as expected (see [10]), except for guarded replication $\mathsf{rep}\,\pi.\, P$. As it will become clear later, the address is not distributed over P in $\mathsf{rep}\,\pi.\, P$, but the task is delayed until the application of structural congruence. Function $\mathcal{T}(\_)$ inspects a system and when a box $\|$, a process parallel $|$, or a choice $+$ is found, function $\triangleright$ is invoked to push the proper address inside the syntactic structure. In the other cases, $\mathcal{T}(\_)$ behaves as the identity. The definition follows.

$P_1 \equiv P_2$ if P_1 and P_2 are α-equivalent
$P \mid \mathsf{nil} \equiv P, \quad P_1 \mid P_2 \equiv P_2 \mid P_1, \quad P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$
$\mathsf{rep}\, \vartheta^p\, \pi.\, P \equiv \vartheta^p\, \pi.\, (\vartheta^p\, {}_{|0} \triangleright \mathcal{T}(P) \mid \mathsf{rep}\, \vartheta^p\, {}_{|1}\, \pi.\, P)$

$\vartheta_1^b\, \boldsymbol{I}\big[\, P_1 \,\big] \equiv \vartheta_2^b\, \boldsymbol{I}\big[\, P_2 \,\big]$ provided $P_1 \equiv P_2$
$\vartheta_1^b\, \boldsymbol{I}_1 \boldsymbol{I}_2 \big[\, P \,\big] \equiv \vartheta_2^b\, \boldsymbol{I}_2 \boldsymbol{I}_1 \big[\, P \,\big]$
$B \equiv B'$ if $(B = \vartheta_1^b\, \boldsymbol{I}^* \beta^+ (x : \Delta) \big[\, P \,\big] \parallel B_3$ and $B' = \vartheta_2^b\, \boldsymbol{I}^* \beta^+ (y : \Delta) \big[\, P\{y/x\} \,\big] \parallel B_3)$ or
$\quad (B' = \vartheta_1^b\, \boldsymbol{I}^* \beta^+ (x : \Delta) \big[\, P \,\big] \parallel B_3$ and $B = \vartheta_1^b\, \boldsymbol{I}^* \beta^+ (y : \Delta) \big[\, P\{y/x\} \,\big] \parallel B_3)$
$\quad$ with y fresh in P and in $\mathsf{sub}(\boldsymbol{I}^*)$
$B \parallel \mathsf{Nil} \equiv B, \quad B_1 \parallel B_2 \equiv B_2 \parallel B_1, \quad B_1 \parallel (B_2 \parallel B_3) \equiv (B_1 \parallel B_2) \parallel B_3$

Table 1. Structural congruence over both processes and boxes.

- $\mathcal{T}(\mathsf{nil}) = \mathsf{nil}$
- $\mathcal{T}(\pi.\, P) = \pi.\, \mathcal{T}(P)$
- $\mathcal{T}(\mathsf{rep}\, \pi.\, P) = \mathsf{rep}\, \pi.\, P$
- $\mathcal{T}(P_0 \mid P_1) = ({}_{|0} \triangleright \mathcal{T}(P_0) \mid ({}_{|1} \triangleright \mathcal{T}(P_1))$
- $\mathcal{T}(M_0 + M_1) = ({}_{+0} \triangleright \mathcal{T}(M_0)) + ({}_{+1} \triangleright \mathcal{T}(M_1))$
- $\mathcal{T}(\mathsf{Nil}) = \mathsf{Nil}$
- $\mathcal{T}(\boldsymbol{I}\big[\, P \,\big]) = \boldsymbol{I}\big[\, \mathcal{T}(P) \,\big]$
- $\mathcal{T}(B_0 \parallel B_1) = {}_{\|0} \triangleright \mathcal{T}(B_0) \parallel {}_{\|1} \triangleright \mathcal{T}(B_1)$

It is straightforward proving that $\mathcal{T}()$ is a bijection between processes and boxes and their labeled version, its inverse being the function that discards addresses. For this reason, in the following we will omit adjective labeled, and we will refer to processes and boxes leaving the context to discriminate.

The proved reduction semantics of BlenX requires the use of the structural congruence over both processes and boxes of Tab. 1. We overload the symbol $\equiv$ to denote both congruences and let the context disambiguate the intended relation. The laws of structural congruence over processes are the typical π-calculus axioms except for the rule of replication. In fact, the structural congruence rule for replication adds a parallel component after the prefix π. Suppose to have a process $\mathsf{rep}\, \tau_1\, .\, \tau_2$. The rule computes the addresses of τ_1 and τ_2 for each application of the structural congruence:
$\mathsf{rep}\, \tau_1\, .\, \tau_2 \equiv \tau_1\, .\, ({}_{|0}\, \tau_2 \mid \mathsf{rep}\, {}_{|1}\, \tau_1\, .\, \tau_2) \equiv \tau_1\, .\, ({}_{|0}\, \tau_2 \mid {}_{|1}\, \tau_1\, .\, ({}_{|1}\, {}_{|0}\, \tau_2 \mid \mathsf{rep}\, {}_{|1}\, {}_{|1}\, \tau_1\, .\, \tau_2))$
The meaning of the laws for boxes follows. First, the structural congruence of internal processes is lifted at the level of boxes. B-addresses are ignored. Second, the actual ordering of binders within an interface is irrelevant. Third, the subject of a binder can be refreshed under the proviso that name clashes in the internal process are avoided and that well-formedness of the interface is preserved. Finally, the monoidal axioms for the parallel composition of boxes are assumed.

Tab.2 shows our proved reduction semantics for BlenX. Arrows carry labels holding the information needed to compute dependency relations. Labels, with metavariable θ, have the form:

- $\vartheta^b\, \vartheta^p\, \pi_\beta$: a prefix π_β with p-address ϑ^p is consumed within box ϑ^b;
- $\vartheta^b\, \vartheta^p \langle {}_{|i}\, \vartheta_i^p\, x?w, {}_{|1-i}\, \vartheta_{1-i}^p\, x!z \rangle$: a communication within box ϑ^b is taking place; the communicating processes have a common context specified by ϑ^p, and specific p-addresses ${}_{|i}\, \vartheta_i^p$ and ${}_{|1-i}\, \vartheta_{1-i}^p$;

(intra) $$\frac{P \equiv \vartheta^p \,|_i\, \vartheta_i^p\; x?w\,.\,P_1 + M_1 \mid \vartheta^p \,|_{1-i}\, \vartheta_{1-i}^p\; x!z\,.\,P_2 + M_2 \mid P_3}{\vartheta^b\, \boldsymbol{I}\big[\, P \,\big] \xrightarrow{\vartheta^b\, \vartheta^p \langle |_i\, \vartheta_i^p\; x?w, |_{1-i}\, \vartheta_{1-i}^p\; x!z \rangle} \vartheta^b\, \boldsymbol{I}\big[\, P_1\{z/w\} \mid P_2 \mid P_3 \,\big]}$$

(tau) $$\vartheta^b\, \boldsymbol{I}\big[\, \vartheta^p\; \tau_i\,.\,P_1 + M_1 \mid P_2 \,\big] \xrightarrow{\vartheta^b\, \vartheta^p\; \tau_i} \vartheta^b\, \boldsymbol{I}\big[\, P_1 \mid P_2 \,\big]$$

(hide) $$\frac{P \equiv \vartheta^p\; \mathsf{hide}(x)\,.\,P_1 + M_1 \mid P_2}{\vartheta^b\, \boldsymbol{I}^*\, \beta(x,\, \Gamma)\big[\, P \,\big] \xrightarrow{\vartheta^b\, \vartheta^p\; \mathsf{hide}(x)} \vartheta^b\, \boldsymbol{I}^*\, \beta^h(x,\, \Gamma)\big[\, P_1 \mid P_2 \,\big]}$$

(unhide) $$\frac{P \equiv \vartheta^p\; \mathsf{unhide}(x)\,.\,P_1 + M_1 \mid P_2}{\vartheta^b\, \boldsymbol{I}^*\, \beta^h(x,\, \Gamma)\big[\, P \,\big] \xrightarrow{\vartheta^b\, \vartheta^p\; \mathsf{unhide}(x)} \vartheta^b\, \boldsymbol{I}^*\, \beta(x,\, \Gamma)\big[\, P_1 \mid P_2 \,\big]}$$

(expose) $$\frac{P \equiv \vartheta^p\; \mathsf{expose}(x,\, \Gamma)\,.\,P_1 + M_1 \mid P_2}{\vartheta^b\, \boldsymbol{I}\big[\, P \,\big] \xrightarrow{\vartheta^b\, \vartheta^p\; \mathsf{expose}(x,\, \Gamma)} \vartheta^b\, \boldsymbol{I}\, \beta(x,\, \Gamma)\big[\, P_1 \mid P_2 \,\big]}, \quad x \notin \mathsf{sub}(\boldsymbol{I}) \text{ and } \Gamma \notin \mathsf{typ}(\boldsymbol{I})$$

(change) $$\frac{P \equiv \vartheta^p\; \mathsf{ch}(x,\, \Delta)\,.\,P_1 + M_1 \mid P_2}{\vartheta^b\, \boldsymbol{I}^*\, \beta(x,\, \Gamma)\big[\, P \,\big] \xrightarrow{\vartheta^b\, \vartheta^p\; \mathsf{ch}(x,\, \Delta)} \vartheta^b\, \boldsymbol{I}^*\, \beta(x,\, \Delta)\big[\, P_1 \mid P_2 \,\big]}, \quad \Delta \notin \mathsf{typ}(\boldsymbol{I})$$

(inter) $$\frac{P \equiv \vartheta_P^p\; y?w\,.\,P_1 + M_1 \mid P_2 \qquad Q \equiv \vartheta_Q^p\; x!z\,.\,Q_1 + N_1 \mid Q_2}{B_0 \parallel B_1 \xrightarrow{\vartheta^b \langle \|_i\, \vartheta_i^b\, \vartheta_P^p\; y?w, \|_{1-i}\, \vartheta_{1-i}^b\, \vartheta_Q^p\; x!z \rangle} B_0' \parallel B_1'}$$

where:

$-B_0 = \vartheta^b \,\|_i\, \vartheta_i^b\, \beta(y,\, \Gamma)\, \boldsymbol{I}_0^*\big[\, P \,\big] \qquad B_0' = \vartheta^b \,\|_i\, \vartheta_i^b\, \beta(y,\, \Gamma)\, \boldsymbol{I}_0^*\big[\, P_1\{z/w\} \mid P_2 \,\big]$

$-B_1 = \vartheta^b \,\|_{1-i}\, \vartheta_{1-i}^b\, \beta(x,\, \Delta)\, \boldsymbol{I}_1^*\big[\, Q \,\big] \quad B_1' = \vartheta^b \,\|_{1-i}\, \vartheta_{1-i}^b\, \beta(x,\, \Delta)\, \boldsymbol{I}_1^*\big[\, Q_1 \mid Q_2 \,\big]$

$-\alpha(\Gamma, \Delta) > 0$

(redex) $$\frac{B \xrightarrow{\theta} B'}{B \parallel B'' \xrightarrow{\theta} B' \parallel B''} \qquad \text{(struct)}\ \frac{B \equiv B_1 \qquad B_1 \xrightarrow{\theta} B_2 \qquad B_2 \equiv B'}{B \xrightarrow{\theta} B'}$$

Table 2. Proved reduction semantics for BlenX.

– $\vartheta^b \langle \|_i\, \vartheta_i^b\, \vartheta_i^p\; y?w, \|_{1-i}\, \vartheta_{1-i}^b\, \vartheta_{1-i}^p\; x!z \rangle$: a common context ϑ^b allows to reach communicating boxes $\|_i\, \vartheta_i^b$ and $\|_{1-i}\, \vartheta_{1-i}^b$; p-addresses ϑ_i^p and ϑ_{1-i}^p identify the involved input and output prefixes, respectively.

The axiom (intra) defines communications between processes within the same box. The axiom reads as follows. If the internal process P is structurally equivalent to $\vartheta^p \,|_i\, \vartheta_i^p\; x?w\,.\,P_1 + M_1 \mid \vartheta^p \,|_{1-i}\, \vartheta_{1-i}^p\; x!z\,.\,P_2 + M_2 \mid P_3$, then the box can perform a reduction leading to a new box with unchanged interface and internal process $P_1\{z/w\} \mid P_2 \mid P_3$. The axiom (tau) models the consumption of delay τ_i. The axiom (hide) forces a binder to become hidden, and therefore not available for interactions. The dual prefix $\mathsf{unhide}(x)$ makes visible a hidden binder. The axiom (expose) adds a new binder to a box. The name x declared in the prefix $\mathsf{expose}(x,\, \Gamma)$ is a placeholder which can be renamed to avoid clashes with the subjects of the other binders of the containing box. To

guarantee the well-formedness of the interface new type Γ cannot be in the set of types of $\boldsymbol{I}$, i.e., $\Gamma \notin \mathsf{typ}(\boldsymbol{I})$. The axiom (change) modifies the type of a binder, provided well-formedness of the interface is preserved. The axiom (inter) defines the interaction of boxes with complementary internal actions (i.e., input and output) over sites with compatible types. The compatibility predicate $\alpha(\Delta, \Gamma)$ is left unspecified and different typing policies and notions of compatibility may be adopted according to distinct modeling needs. However, independently from the notion of type compatibility assumed, the communication ability is only determined by the types of the involved interfaces and not by their subjects. Information flows from the box containing the process which exhibits the output prefix to the box enclosing the process that performs the input action. Finally, the rule (redex) interprets the reduction of a parallel subcomponent as a reduction of the system, and the rule (struct) infers a reduction after a structural shuffling of the system at hand.

The axioms and rules above give a detailed description of one step of computation, i.e., given a system B, the semantics describes how to obtain $B_1, \ldots, B_k$ such that $B \xrightarrow{\theta_i} B_i$, $1 \leq i \leq k$. *Proved computation* lifts one step of computation to n steps of computation. If $B_0 \xrightarrow{\theta} B_1$ is a transition, then B_0 is the *source* of the transition and B_1 is its *target*. A proved computation of B_0 is a sequence of transitions $B_0 \xrightarrow{\theta_0} B_1 \xrightarrow{\theta_1} \cdots$ such that the target of any transition is the source of the next one.

To simplify the treatment, hereafter we suppose α-equivalence implemented a la De Bruijn [15]. In this way α-equivalence coincides with first-order equality.

2.1 Dependency Relation

We are ready to introduce the relation of dependency between the transitions of a computation. Intuitively, given a computation $B_0 \xrightarrow{\theta_0} B_1 \xrightarrow{\theta_1} \ldots \xrightarrow{\theta_n} B_{n+1}$, the transition $B_n \xrightarrow{\theta_n} B_{n+1}$ depends on a transition $B_i \xrightarrow{\theta_i} B_{i+1}$, $i < n$, if the θ_n transition cannot appear before the transition θ_i. Consider a simple computation:[3]

$B_0 \triangleq \boldsymbol{I}\big[\ \tau_1 \,.\, \tau_2 \,.\, \tau_3\ \big] \xrightarrow{\tau_1} B_1 \triangleq \boldsymbol{I}\big[\ \tau_2 \,.\, \tau_3\ \big] \xrightarrow{\tau_2} B_2 \triangleq \boldsymbol{I}\big[\ \tau_3\ \big] \xrightarrow{\tau_3} B_3 \triangleq \boldsymbol{I}\big[\ \mathsf{nil}\ \big]$

It is clear that $B_2 \xrightarrow{\tau_3} B_3$ depends upon $B_0 \xrightarrow{\tau_1} B_1$, because prefix τ_3 is "covered" by prefix τ_1. Following this intuition, we define the notion of structural dependency between transitions. Note that below we use label θ to denote a transition $B \xrightarrow{\theta} B'$ as shorthand, if no ambiguity arises. We need an auxiliary definition that flats labels:

- $\mathsf{f}(\vartheta^b\, \vartheta^p\, \pi_\beta) = \{\vartheta^b\, \vartheta^p\, \pi_\beta\}$
- $\mathsf{f}(\vartheta^b\, \vartheta^p \langle |_i\, \vartheta_i^p\, x?w, |_{1-i}\, \vartheta_{1-i}^p\, x!z\rangle) = \{\vartheta^b\, \vartheta^p\, |_i\, \vartheta_i^p\, x?w, \vartheta^b\, \vartheta^p\, |_{1-i}\, \vartheta_{1-i}^p\, x!z\}$
- $\mathsf{f}(\vartheta^b \langle \|_i\, \vartheta_i^b\, \vartheta_i^p\, y?w, \|_{1-i}\, \vartheta_{1-i}^b\, \vartheta_{1-i}^p\, x!z\rangle) = \{\vartheta^b\, \|_i\, \vartheta_i^b\, \vartheta_i^p\, y?w, \vartheta^b\, \|_{1-i}\, \vartheta_{1-i}^b\, \vartheta_{1-i}^p\, x!z\}$

Definition 1. *Given a computation $B_0 \xrightarrow{\theta_0} B_1 \xrightarrow{\theta_1} B_2 \ldots \xrightarrow{\theta_n} B_{n+1}$, θ_n has a* direct structural dependency *on θ_h ($\theta_h \prec^I_{str} \theta_n$) iff $h < n$, $\vartheta^b\, \vartheta^p\, \pi \in \mathsf{f}(\theta_h)$ and $\vartheta^b\, \vartheta^p\, \vartheta^{p\prime}\, \pi' \in \mathsf{f}(\theta_n)$.* Structural dependency *is defined as the reflexive and transitive closure of $\prec^I_{str}$, i.e., $\prec_{str} = (\prec^I_{str})^*$.*

[3] Note, it is a proved computation even if no address is provided, because there is a single box with only a sequential process.

Structural dependency does not catch possible relations between transitions that depend on the notion of binder. For instance, consider the following computation:
$\beta(x,\,\Gamma)\left[\ {}_{|0}\,\mathsf{unhide}(x)\mid {}_{|1}\,\mathsf{hide}(x)\ \right] \xrightarrow{{}_{|1}\,\mathsf{hide}(x)} \beta^h(x,\,\Gamma)\left[\ {}_{|0}\,\mathsf{unhide}(x)\ \right] \xrightarrow{{}_{|0}\,\mathsf{unhide}(x)} \beta(x,\,\Gamma)\left[\ \mathsf{nil}\ \right]$
Here ${}_{|0}\,\mathsf{unhide}(x)$ and ${}_{|1}\,\mathsf{hide}(x)$ are not structurally related, but the former cannot take place before the latter has hidden the binder x. We call *binder dependency* this notion, because it depends on the BlenX notion of binders.

Definition 2. *Given a computation* $B_0 \xrightarrow{\theta_0} B_1 \xrightarrow{\theta_1} \ldots \xrightarrow{\theta_n} B_{n+1}$, θ_n *has a direct binder dependency on* θ_h *(*$\theta_h \prec^I_{bin} \theta_n$*) iff* $h < n$ *and*

1. $\theta_n = \vartheta^b\,\vartheta^p\,\mathsf{unhide}(x)$, $\theta_h = \vartheta^b\,\vartheta^{p\prime}\,\mathsf{hide}(x)$
2. $\theta_n = \vartheta^b\,\vartheta^p\,\mathsf{hide}(x)$, $\theta_n = \vartheta^b\,\vartheta^{p\prime}\,\mathsf{unhide}(x)$
3. $\theta_n = \vartheta^b\langle\|_i\,\vartheta^b_i\,\vartheta^p_i\ y?w, \|_{1-i}\,\vartheta^b_{1-i}\,\vartheta^p_{1-i}\ x!z\rangle$ *and* $\theta_h = \vartheta^{b\prime}\,\vartheta^p\,\mathsf{unhide}(k)$ *and* *((*$\vartheta^{b\prime} = \vartheta^b\,\|_i\,\vartheta^b_i$ *and* $y = k$*) or (*$\vartheta^{b\prime} = \vartheta^b\,\|_{1-i}\,\vartheta^b_{1-i}$ *and* $x = k$*))*
4. $\theta_n = \vartheta^b\langle\|_i\,\vartheta^b_i\,\vartheta^p_i\ y?w, \|_{1-i}\,\vartheta^b_{1-i}\,\vartheta^p_{1-i}\ x!z\rangle$ *and* $\theta_h = \vartheta^{b\prime}\,\vartheta^p\,\mathsf{ch}(k,\,\Delta)$ *and* *((*$\vartheta^{b\prime} = \vartheta^b\,\|_i\,\vartheta^b_i$ *and* $y = k$*) or (*$\vartheta^{b\prime} = \vartheta^b\,\|_{1-i}\,\vartheta^b_{1-i}$ *and* $x = k$*))*
5. $\theta_n = \vartheta^b\langle\|_i\,\vartheta^b_i\,\vartheta^p_i\ y?w, \|_{1-i}\,\vartheta^b_{1-i}\,\vartheta^p_{1-i}\ x!z\rangle$ *and* $\theta_h = \vartheta^{b\prime}\,\vartheta^p\,\mathsf{expose}(k,\,\Delta)$ *and* *((*$\vartheta^{b\prime} = \vartheta^b\,\|_i\,\vartheta^b_i$ *and* $y = k$*) or (*$\vartheta^{b\prime} = \vartheta^b\,\|_{1-i}\,\vartheta^b_{1-i}$ *and* $x = k$*))*
6. $\theta_n = \vartheta^b\,\vartheta^p\,\mathsf{ch}(x,\,\Delta)$ *and* $\theta_h = \vartheta^b\,\vartheta^{p\prime}\,\mathsf{expose}(x,\,\Gamma)$

Binder dependency is the reflexive and transitive closure of $\prec^I_{bin}$*, i.e.,* $\prec_{bin} = (\prec^I_{bin})^*$.

We comment on the various conditions of the definition above. Item 1 states that an unhide in a box ϑ^b depends upon an hide on the same binder within the same box ϑ^b. Item 2 states if the unhide happens before the hide, then the hide depends upon the unhide. Items 3, 4, and 5 work on the same idea: an inter box communication cannot take place if one of the involved binders is hidden, or has the wrong type, or it is not yet available, respectively. Finally, a $\mathsf{ch}(x,\,\Delta)$ depends upon the exposition of a binder named x. The hypothesis about α-equivalence at the end of Sect. 2 makes simpler this definition avoiding complex labels to record information about α-conversion. Moreover, here only b-addresses are used because we only need to know the box where action is taking place. As usual, the *dependency relation* is defined as $\prec = (\prec_{str} \cup \prec_{bin})^*$.

Finally, we define *immediate dependency*, the basic relation for managing non-Markovian processes. The idea is that θ_n has an immediate dependency on θ_i if θ_n depends upon θ_i, and there are not other transitions in between the two on which θ_n depends.

Definition 3. *Given a proved computation* $B_0 \xrightarrow{\theta_0} B_1 \xrightarrow{\theta_1} \cdots \xrightarrow{\theta_n} B_{n+1}$, θ_n *has an immediate dependency on* θ_i, $\theta_i \prec_I \theta_n$, *iff* $\theta_i \prec \theta_n$, *and* $\forall j,\ i < j < n$, $\theta_j \not\prec \theta_n$.

2.2 General Distributions

In this section we define the formal tools to manage general continuous probability distributions within BlenX providing a stochastic extension of proved computations.

Given a set $\mathcal{F}$ of continuous probabilistic distribution functions with positive support, we assign a cost to each label θ via a function $\$(\_)$ such that $\$(\theta) = F_\theta \in \mathcal{F}$.

Relying on cost function $\$(\_)$, we make the qualitative model independent from quantitative considerations, allowing modelers to play with quantities. The density function corresponding to distribution F_θ is f_θ, where $F_\theta(x) = \int_{-\infty}^{x} f_\theta(t)dt$. The following results show how to derive useful probabilities and distributions from a proved transition (see [16, Th. 3.1]). The probability of a transition $B \xrightarrow{\theta_i} B_i$ is

$$p_i = \int_0^\infty f_i(t) \prod_{B \xrightarrow{\theta_j} B_j}^{j \neq i} (1 - \$(\theta_j)(t))dt$$

and the distribution $\tilde{F}_i$ of the random variable T_i which describes the time interval associated with $B \xrightarrow{\theta_i} B_i$ is

$$\tilde{F}_i = P[T_i < t] = \left(\int_0^t f_i(x) \prod_{B \xrightarrow{\theta_j} B_j}^{j \neq i} (1 - \$(\theta_j)(x))dx \right) / p_i$$

The random variable T_i describes the time a transition $B \xrightarrow{\theta_i} B_i$ requires to complete. In a Markovian setting, T_i is exponentially distributed and therefore it is independent from the waited time. Consider, for instance:

$$\boldsymbol{I}\left[\ {}_{|0}\,\tau_0 \mid {}_{|1}\,\tau_1\ \right] \xrightarrow{{}_{|0}\,\tau_0} \boldsymbol{I}\left[\ {}_{|1}\,\tau_1\ \right] \xrightarrow{{}_{|1}\,\tau_1} \boldsymbol{I}\left[\ \mathsf{nil}\ \right] \tag{2}$$

Under Markovian hypothesis the time required for transition ${}_{|1}\,\tau_1$ is independent from the time consumed by transition ${}_{|0}\,\tau_0$. In a general setting, both τ_0 and τ_1 are active in $\boldsymbol{I}\left[\ {}_{|0}\,\tau_0 \mid {}_{|1}\,\tau_1\ \right]$ and the time required to complete ${}_{|0}\,\tau_0$ affects the time to complete ${}_{|1}\,\tau_1$. Therefore, the distribution of $T_{|_1\,\tau_1}$ has to be updated considering that transition ${}_{|0}\,\tau_0$ already happened. Generalizing, given a proved computation $B_0 \xrightarrow{\theta_0} B_1 \xrightarrow{\theta_1} \ldots \xrightarrow{\theta_n} B_{n+1}$, the time distribution T_n of θ_n depends upon the time distributions T_i of transitions θ_i, $0 \leq i < n$. But not all transitions θ_i have to be considered. The following computation, that looks similar to computation (2),

$$\boldsymbol{I}\left[\ \tau_0\ .\ \tau_1\ \right] \xrightarrow{\tau_0} \boldsymbol{I}\left[\ \tau_1\ \right] \xrightarrow{\tau_1} \boldsymbol{I}\left[\ \mathsf{nil}\ \right] \tag{3}$$

has a completely different quantitative behavior. In this case the time consumed by τ_0 does not affect the time to complete τ_1, because τ_1 becomes active only when τ_0 finished. The definition of immediate dependency helps in generalizing this idea. Note that, by Def. 3, any pair of transitions in a given computation is either in a dependency relation or not. Thus, once found the maximum i such that $\theta_i \prec_I \theta_n$, all the transitions occurred after θ_i, must influence the time distribution of transition θ_n. The following definition formalizes this idea.

Definition 4. *If $B_0 \xrightarrow{\theta_0} B_1 \xrightarrow{\theta_1} \cdots \xrightarrow{\theta_n} B_{n+1}$ is a proved computation, then the distribution of the random variable T_n describing the time interval associated with transition $B_n \xrightarrow{\theta_n} B_{n+1}$ is*

$$\tilde{F}_n = P\left[T_n \leq t + \sum_{h=i+1}^{n-1} T_h \mid T_n > \sum_{h=i+1}^{n-1} T_h\right] \text{ with } \theta_i \prec_I \theta_n$$

assuming $\sum_\emptyset T_h = 0$.

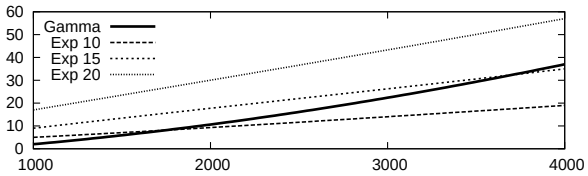


Fig. 2. Simulation time of a chain of exponentially distributed steps vs. the equivalent Erlang step varying the length of the chain.

The expert reader has already noticed that the distribution T_n of θ_n can be computed only after computing the distribution T_i of θ_i, $0 \leq i < n$. This is essential to correctly calculate of $\sum_{h=i+1}^{n-1} T_h$. We conclude this section giving a constructive definition of a stochastic computation.

Definition 5. *Given a proved computation* $\xi = B_0 \xrightarrow{\theta_0} B_1 \xrightarrow{\theta_1} \ldots \xrightarrow{\theta_n} B_{n+1}$*, the corresponding* stochastic computation *is*

$$\xi_{n+1} = B_0 \xrightarrow{\theta_0, \tilde{F}_0} B_1 \xrightarrow{\theta_1, \tilde{F}_1} \ldots \xrightarrow{\theta_n, \tilde{F}_n} B_{n+1}$$

defined, for $i \geq 0$*, as*

$$\xi_i = \texttt{if}\ \ i = 0\ \texttt{then}\ \xi\ \texttt{else}\ \xi_{i-1}\left\{{}^{(B_{i-1} \xrightarrow{\theta_{i-1}} B_i)}\!/_{(B_{i-1} \xrightarrow{\theta_{i-1}, \tilde{F}_{i-1})} B_i}\right\}$$

where $\tilde{F}_i = P\left[T_i \leq t + \sum_{h=j+1}^{i-1} T_h \mid T_n > \sum_{h=j+1}^{i-1} T_h\right]$ *with* $\theta_j \prec_I \theta_i$.

3 Experimental Results

We extended the BlenX language and the BetaWB with a prototypical implementation of the concepts presented in the previous sections. Here we show the effectiveness of the approach by presenting two simple bio-inspired examples that underline the importance of being able to simulate non-Markovian processes.

In the first example we consider the following proved computation:

$$\xi = B_0 \xrightarrow{\theta_0} B_1 \xrightarrow{\theta_1} \ldots \xrightarrow{\theta_{n-1}} B_n$$

where B_0 undergoes an n-step transformation becoming B_n. Each step is described by a negative exponential distribution with parameter λ, i.e., $\$(\theta_i) = 1 - e^{-\lambda t}$. A similar path can be found, for instance, in the lambda phage model described in [17]. If the focus is on simulating the production of B_n , without considering intermediates B_i, $i \in [1, n-1]$, the system can be approximated as

$$\xi_A = B_0 \xrightarrow{\theta} B_n$$

where θ follows an Erlang distribution with scale λ and shape n,

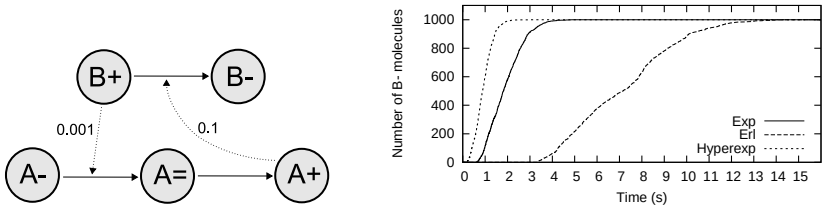
$$\$(\theta) = \sum_{j=0}^{n-1} \frac{e^{-\lambda t}(\lambda t)^j}{n!}$$

Distribution	Parameters	Mean	Variance
Exponential	$\lambda = 0.0078$	128.2051	16436.554
Erlang	$k = 2$, $\lambda = 0.0156$	128.2051	8218.2774
Hyperexponential	$p_1 = 0.3$, $p_2 = 0.7$	128.2051	79755.80924
	$\lambda_1 = 0.0025$, $\lambda_2 = 0.085312$		

Table 3. The three distributions used to model the individual conformational change of protein A from intermediate to active state.

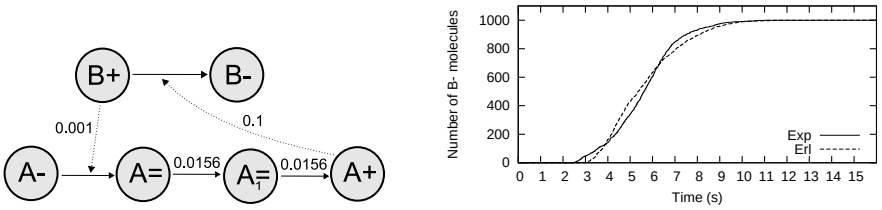
The abstraction is correct because an Erlang distribution with shape n and scale λ is the sum of n exponentially distributed random variables with parameters λ. Fig. 2 shows the simulation time vs. the number of boxes in B_0, where ξ and ξ_A are simulated with BetaWB, for different values of n. Notice that the simulation time of ξ_A is independent form the length of the chain n. Moreover, Fig. 2 points out also an argument regarding the computational efficiency of the simulation, meaning that there are cases in which the use of an Erlang step instead of a chain of exponentially distributed steps is useful not only as a process abstraction, but also for speeding up the simulation time.

In the second example we consider a simple feedback mechanism (Fig. 3(a)) composed of two interacting protein A and B. Protein B can be in an inactive (B^-) or active (B^+) form, while protein A can be in an inactive (A^-), intermediate ($A^=$) or active (A^+) form. B^+ interact with A^-, transforming it into its intermediate form $A^=$, which in turn is subject to an individual conformational change that leads to the active form A^+. Now, consider the individual conformational change. We tried to model this reaction in three different ways by using an exponential, an Erlang and a hyperexponential distribution with same means but different variances (see Tab.3). We ran stochastic simulations of the three models with initially a number 1000 of B^+ and A^- molecules. Fig. 3(b) reports simulation results and in particular the number of B^- molecules over time, showing the speed at which, through the feedback mechanism, the initial amount of B^+ is consumed. It is important to note that a different choice in the probability distribution that drives the protein A conformational change has a fundamental impact on the overall behavior of the system.



(a) Example model (b) Simulation results

Fig. 3. Example model and simulation results for the three alternatives using different distributions to model the conformational change of protein A from intermediate to active form.



(a) Example model modified (b) Simulation results

Fig. 4. Example model with conformational change expressed as a 2-step transformation and comparison of the simulation results with the one of the model in Fig. 3(a) that uses the Erlang distribution.

Although important itself from a modeling point of view, this fact suggests us that playing with non-Markovian processes is a useful tool to form hypotheses. Consider indeed a scenario in which the experimental data fits with the simulation results obtained using the Erlang distribution. By the first example we know that our Erlang (Tab.3) can be seen also as a chain of two exponential steps of rate $\lambda = 0.0156$, which can lead us to the hypothesis that maybe our model is incomplete and that the conformational change is a 2-step transformation passing through another intermediate protein form $A_1^=$. This hypothesis could be used to refine the model in Fig. 3(a) as the one in Fig. 4(a), for which the simulation results in Fig. 4(b) shows the perfect fit with the simulation results for the model in Fig. 3(a) that uses the Erlang distribution, and eventually to drive wet experiments to confirm the hypothesis.

4 Conclusions

We presented the tools to cope with general distributions within the BetaWB framework. The proved reduction semantics introduced for BlenX allows us to derive a notion of dependency between transitions without changing the BlenX syntax. Moreover, we exploit the notion of dependency only for quantitative purposes, but also qualitative aspects can be retrieved, as, for instance, localities [10]. In the literature there have been many attempts to extend process calculi with general probabilistic distributions (see, e.g., [18–21, 16]), but, to the best of our knowledge, this is the first time an effective simulation tool is available. The examples presented in Sect. 3 outline that reasoning about general distributions could be useful and need further investigations. In particular, the last example proposed non-Markovian processes as a tool to form hypothesis based on experimental observations. Clearly the example is simple and ad-hoc, and a systematic way for constructing hypothesis is needed for validating the approach. Nevertheless, the tool presented here is an important step in this direction, because it allows playing with non-Markovian processes at a reasonable computational cost.

References

1. Milner, R.: Communication and Concurrency. International Series in Computer Science. Prentice hall (1989)
2. Prandi, D., Priami, C., Quaglia, P.: Communicating by compatibility. JLAP **75** (2008) 167
3. Priami, C., Quaglia, P.: Beta binders for biological interactions. In: CMSB 2004. Volume 3082 of LNCS., Springer (2004) 20
4. Meredith, G., Bjorg, S.: Contracts and types. Comm. of the ACM **46**(10) (2003) 41–47
5. Tame, J.: Scoring Functions - the First 100 Years. Journal of Computer-Aided Molecular Design **19** (2005) 445
6. Dematté, L., Priami, C., Romanel, A.: The BlenX Language: A Tutorial. In LNCS, ed.: SFM 2008, Springer-Verlag (2008) 313–365
7. Dematté, L., Priami, C., Romanel, A.: The Beta Workbench: a computational tool to study the dynamics of biological systems. Briefings in Bioinformatics (2008)
8. Degano, P., Prandi, D., Priami, C., Quaglia, P.: Beta-binders for biological quantitative experiments. Electr. Notes Theor. Comput. Sci **164**(3) (2006) 101–117
9. Mura, I.: Exactness and Approximation of the Stochastic Simulation Algorithm. Technical Report 12/2008, The Microsoft Research - University of Trento Centre for Computational and Systems Biology (2008)
10. Curti, M., Degano, P., Priami, C., Baldari, C.: Modelling biochemical pathways through enhanced π-calculus. TCS **325**(1) (2004) 111
11. Degano, P., Priami, C.: Enhanced operational semantics: A tool for describing and analysing concurrent systems. ACM Computing Surveys **33**(2) (2001) 135–176
12. Guerriero, M.: From Intuitive Descriptions of Biochemical Systems to Their Formal Analysis. PhD in Information and Communication Technologies, International Doctorate School in Information and Communication Technologies – University of Trento (2007)
13. Marsan, M., Balbo, G., Bobbio, A., Chiola, G., Conte, G., Cumani, A.: The Effect of Execution Policies on the Semantics and Analysis of Stochastic Petri Nets. IEEE Transactions on Software Engeneering **15**(7) (1989) 832–846
14. Phillips, A., Cardelli, L.: Efficient, Correct Simulation of Biological Processes in the Stochastic Pi-calculus. In: CMSB 2007. Volume 4695 of LNCS., Springer (2007) 184
15. de Bruijn, N.: Lambda calculus notation with nameless dummies. Indagationes Mathematicae **34** (1972) 381–392
16. Priami, C.: Language-based performance prediction for distributed and mobile systems. Information and Computation **175** (2002)
17. Arkin, A., Ross, J., McAdams, H.: Stochastic Kinetic Analysis of Developmental Pathway Bifurcation in Phage λ-Infected *Escherichia coli* Cells. Genetics **149** (1998) 1633
18. Götz, N., Herzog, U., Rettelbach, M.: TIPP - Introduction and Application to Protocol Performance Analysis. In: Formale Methoden für verteilte Systeme, GI/ITG-Fachgespräch, Magdeburg, 10.-11. Juni 1992. (1992) 105–125
19. Marsan, M., Bianco, A., Ciminiera, L., Sisto, R., Valenzano, A.: A LOTOS extension for the performance analysis of distributed systems. IEEE/ACM Transactions Networking **2**(2) (1994) 151–165
20. Brinksma, E., Katoen, J., Langerak, R., Latella, D.: A Stochastic Causality-Based Process Algebra. the Computer Journal **38**(7) (1995) 552–565
21. Bravetti, M., Bernardo, M., Gorrieri, R.: Towards Performance Evaluation with General Distributions in Process Algebras. In: CONCUR '98: Concurrency Theory, 9th International Conference, Nice, France, September 8-11, 1998, Proceedings. (1998) 405–422