

Tuning Temporal Features within the Stochastic π -Calculus

Loïc Paulevé, Morgan Magnin, and Olivier Roux

Abstract—The stochastic π -calculus is a formalism that have been shown of interest for modelling systems where the stochasticity and the delay of transitions are important features, such as the biochemical reactions. Commonly, duration of transitions within stochastic π -calculus models follow an exponential random variable. Underlying dynamics of such distributed models are expressed in terms of continuous-time Markov chains and can then be efficiently simulated and model-checked. However, the exponential law comes with a huge variance making difficult the modelling of systems with accurate temporal constraints. In this report, a technique for tuning temporal features within the stochastic π -calculus is presented. This method relies on the introduction of a stochasticity absorption factor by replacing the exponential distribution by the Erlang distribution, that is the sum of exponential random variables. A construction of this stochasticity absorption factor in the classical exponentially distributed stochastic π -calculus is provided. This report also offers tools for manipulating the stochasticity absorption factor and its link with timed intervals for firing transitions. Finally, the model-checking of such designed models is tackled through the support for the stochasticity absorption factor in the stochastic π -calculus translation to the probabilistic model checker PRISM.

Index Terms—Temporal parameters, π -Calculus, Model checking, Markov processes, Stochastic processes.



1 INTRODUCTION

BY the introduction of temporal and stochastic aspects within models, the designer aims at reproducing real systems more closely. This complexation of models generally raises a compromise between a precise specification of temporal parameters and efficient analyses techniques.

The π -calculus [1] is a concurrent process algebra suitable for modelling independently defined entities communicating through mobile channels. This formalism is widely used for analysing, for instance, communication protocols. Amongst the different extensions of this calculus, the stochastic π -calculus [2] introduces stochastic and temporal features within π -calculus models. The stochastic π -calculus is notably used to model biological systems [3], [4], [5], [6]. Temporal and stochastic features of biochemical reactions are prominent for analysing and predicting the behaviours of such complex systems.

In the stochastic π -calculus framework, time and stochasticity are injected by making the duration of transitions to follow an exponential distribution. The parameter of this distribution is called the *use rate*, and informally, defines the number of times the transition can be used within a time unit. Underlying semantics of stochastic π -calculus expressions can be expressed with continuous-time Markov chains (CTMCs). By exploiting the memoryless property of the exponential law, very efficient simulation algorithms have been designed — such as the *Gillespie's algorithm* [7], *BioSpi* [3] and *SPiM* [8]. The model checking of stochastic π -calculus has been

recently proposed by [9] which offers a translation of the exponentially distributed π -Calculus to PRISM, a probabilistic symbolic model checker.

However, despite use rates bring temporal features within models, the high variance of the exponential distribution seems to prevent a precise modelling of temporal constraints. As an example, one may wonder how to model in stochastic π -calculus a transition taking place only within a given time interval, as usually done with Timed Automata [10].

In this report, we present a technique to tune temporal features within the stochastic π -calculus and we provide a method to analyse such models with PRISM. This tuning permits to model systems with both stochastic and temporal aspects in a more independent manner than in the classical stochastic π -calculus. Such a modelling accuracy is obtained by attaching to actions, in addition of the use rate, a so-called *stochasticity absorption factor*. The higher this factor is, the more the temporal variance is reduced around the mean imposed by the use rate.

The stochasticity absorption of transitions is obtained by replacing the exponential distribution by the Erlang distribution, which is the distribution of the sum of exponential random variables. It is worth to notice that the semantics of the stochastic π -calculus with general distributions has already been studied [11]. The originality of the presented approach is to provide a correct construction of the Erlang distributed stochastic π -calculus into the exponentially distributed stochastic π -calculus. Therefore, such models still have CTMCs semantics and can be simulated and verified with the various existing tools based on the memoryless property of the exponential distribution.

As a first result of this study, the construction of

• IRCCyN, UMR CNRS 6597, École Centrale de Nantes, France.
E-mail: {pauleve,magnin,roux}@irccyn.ec-nantes.fr

the stochasticity absorption factor with the exponentially distributed π -calculus allows to simulate such parametrised transitions with standard simulation algorithms.

As a second result of using this stochasticity absorption factor, a transition can be parametrised by specifying a confidence time interval of firing, referred as the *firing interval*. In other terms, at a given confidence coefficient, to any firing interval corresponds a unique rate and a unique stochasticity absorption factor which ensure the transition is fired within this time interval.

Finally, as a third result, we provide an adaptation of the translation of the exponentially distributed π -calculus into PRISM given by Norman et al. [12] to support the stochasticity absorption factor. In that way, model checking of Erlang distributed stochastic π -calculus can be efficiently performed thanks to PRISM.

As related works, we may cite the modelling of probabilistic timed automata as Markov decision process suitable for model checking with PRISM [13]. The proposed PRISM construction, presented in the last section of this report, follows the same idea of having dedicated transitions for incrementing a clock.

Analysis of complex dynamical systems have been widely done through model-checking algorithms designed for states graphs underlying CTMCs. For this purpose, the branching temporal logic CTL was extended as a stochastic logic called CSL which was initially proposed in [14] and furthermore extended in [15]. Later again, this approach was pursued by the work of [16]. It consisted in defining a new stochastic temporal logic (named CSL^{TA}) intended at limiting the explosion of complexity in analysis. The idea of this work is to use timed automata both to express the property that has to be checked and to specify the system on which it is checked. Thus, the model-checking algorithm works on products of timed automata which prevents from computing the wholly developed reachability graph of the product. Such an avoidance of building a graph the number of states of which may makes it non tractable is an idea that we have used too in a slightly different way in our work.

This report is structured as follows. Section 2 formally presents the stochastic π -calculus. Section 3 introduces the stochasticity absorption factor through the Erlang distribution. A construction of the Erlang distributed to the exponentially distributed stochastic π -calculus is provided. Section 4 establishes the parallel between the firing interval and the rate and stochasticity absorption factor of a transition. Section 5 adapts the translation from the standard stochastic π -calculus into PRISM to support the stochasticity absorption factor. Finally, Section 6 concludes this report and discusses future works.

2 THE STOCHASTIC π -CALCULUS

The π -calculus is a concurrent process algebra where two processes communicate using a shared channel [1]. For

establishing a communication, a sender process outputs on a channel and a receiver process inputs on the same channel. The sender process may outputs *names* on the channel to transmit data to the receiver. A name is either a value or a channel. In that way, π -calculus allows to model the mobility of the communication channels between concurrent processes.

The stochastic extension of the π -calculus affects to each *action* (channel input/output or internal action) a probabilistic distribution for determining the delay until it is effective [2], [11]. Usually, the exponential distribution is preferred. The parameter of this distribution is referred as the *use rate* for the action, and informally, gives the number of time such an action is to be fired within a time unit.

The syntax for the stochastic π -calculus used in this report is presented by Definition 1. It is close to the commonly used definition of the stochastic π -calculus (e.g. [8], [12]).

Definition 1 (Stochastic π -Calculus $S\pi$). *Using P, P_i, Q to range over terms, A to range over definitions of terms, π to range over actions and x, y, z_1, \dots, z_n to range over names:*

$$\begin{aligned} \pi &::= \tau \mid ay \mid \bar{a}y \\ P &::= \mathbf{0} \mid \nu xP \mid P|Q \mid [cond]P \mid \\ &\quad \pi.P \mid A(z_1, \dots, z_n) \\ A(z_1, \dots, z_n) &::= \sum_{i \in I} P_i \end{aligned}$$

Where I is an index set, t is the identifier of the internal action and *cond* is a boolean expression on names.

Hereafter, $P.\mathbf{0}$ is abbreviated as P , $A()$ is abbreviate ad A and z_1, \dots, z_n is abbreviated as \tilde{z} .

The actions of a process are either internal action (τ), input of y on channel a (ay) or free output of y on channel a ($\bar{a}y$). After performing an action π , the process evolves as P , noted as $\pi.P$. The term $\mathbf{0}$ denotes the null process, $P|Q$ the parallel compositions of processes P and Q , $[cond]P$ the process P enabled only if *cond* is satisfied (generally, *cond* is a conjunction of tests upon names), and νxP is the restriction of a new name to the process P . $\sum_{i \in I} P_i$ is a race conditions between processes P_i : only the first fired P_i is considered. Each race condition has a definition of the form $A(\tilde{z}) = \sum_{i \in I} P_i$, where \tilde{z} are the parameters of the process A .

A name y is *bound* to a process P if there exists a term of the form νy or ay within the expression of P . Otherwise, the name y is free. The set of bound names in a process P is noted as $bn(P)$ and the set of free names as $fn(P)$. $n(P) = fn(P) \cup bn(P)$ is the set of names present in the term P .

Figure 1 depicts the operational semantics of the stochastic π -calculus. Transitions are labeled by the performed action, τ_t denotes an internal transition identified by t , \hat{a} denotes a communication on channel a .

We denote by $S\pi_e$ the stochastic π -calculus having the duration of each action a following an exponential

$\text{PRE}_\tau \frac{\tau_t.P \xrightarrow{\tau_t} P}{\tau_t.P \xrightarrow{\tau_t} P}$	$\text{PRE}_{\text{IN}} \frac{ay.P \xrightarrow{ay} P}{ay.P \xrightarrow{ay} P}$	$\text{PRE}_{\text{OUT}} \frac{\bar{a}y.P \xrightarrow{\bar{a}y} P}{\bar{a}y.P \xrightarrow{\bar{a}y} P}$	$\text{SUM} \frac{P_j \xrightarrow{\pi} P'_j}{(\sum_{i \in I} P_i) \xrightarrow{\pi} P'_j} j \in I$
$\text{PAR} \frac{P \xrightarrow{\pi} P'}{P Q \xrightarrow{\pi} P' Q} bn(\pi) \cap fn(Q) = \emptyset$			$\text{COM} \frac{P \xrightarrow{ay} P' Q \xrightarrow{\bar{a}z} Q'}{P Q \xrightarrow{\dot{a}} P'\{z/y\} Q'}$
$\text{RES} \frac{P \xrightarrow{\pi} P'}{\nu x P \xrightarrow{\pi} P'} x \notin n(\pi)$			$\text{MATCH} \frac{P \xrightarrow{\pi} P'}{[cond]P \xrightarrow{\pi} P'} cond$

Fig. 1. Operational semantics for the stochastic π -calculus.

random variable of use rate $r_a \in \mathbb{R}^+$. The probability that an action with use rate r is fired within a time t is $1 - \exp(-r.t)$. Its average duration is r^{-1} time units with a variance of r^{-2} . Given x actions having respectively use rates r_1, \dots, r_x , the probability that the y^{th} reaction is fired is $\frac{r_y}{r_1 + \dots + r_x}$. We also consider actions having an infinite rate $r_a = \infty$, i.e. which are instantaneous. Such action are always played first. If two instantaneous actions are possible, the choice of the one to fire is non-deterministic.

3 STOCHASTICITY ABSORPTION

To cope with the strong bind between the average duration of an action and its probability of being fired before a given time brought by the exponential law, we introduce the stochasticity absorption factor.

Instead of having the duration of an action following one exponential random variable at rate r , we propose to have the duration of an action following the sum of sa exponential random variables at rate $r.sa$. This results in an unchanged average duration but a variance divided by sa . In this way, sa stands for the stochasticity absorption factor. The obtained probabilistic distribution is known as the *Erlang distribution*. Therefore, the tuning of the temporal features within the stochasticity π -calculus is achieved by attaching to each action an Erlang distribution of a fixed rate and stochasticity absorption factor. We denote by $S\pi_{Er}$ such a distributed stochastic π -calculus.

This sections starts by presenting the Erlang distribution and functions to compute some standard probabilities. The construction of $S\pi_{Er}$ into $S\pi_e$ is then presented, demonstrating $S\pi_{Er}$ processes can be simulated using standard algorithms based on the exponential law for firing actions. Finally, this section is illustrated by a toy example.

3.1 The Erlang distribution

The Erlang distribution is usually defined by two parameters: the shape $k \in \mathbb{N}^*$ and the rate $\lambda \in \mathbb{R}_+^*$. The Erlang distribution is then the distribution of the sum of k exponential random variables at use rate λ . The Erlang distribution is a particular case of the *gamma distribution* where the shape parameter may be any positive real.

For the sake of consistency, we refer to the Erlang distribution as the distribution of the sum of sa exponential random variables of use rate $r.sa$, where sa is the stochasticity absorption factor and r the use rate of the non-absorbed exponential variable (i.e. when $sa = 1$). Equivalence between these two definitions is given by the relations $k = sa$ and $\lambda = r.sa$.

We recall the probability density function (PDF) (1) and cumulative distribution function (CDF) (2) of an Erlang distribution of rate r and stochasticity absorption factor sa . PDF and CDF with different stochasticity absorption factors but a constant rate are plotted in Figure 2.

$$f_{r,sa}(t) = \frac{(r.sa)^{sa} t^{sa-1} \exp(-r.sa.t)}{(sa-1)!} \quad (1)$$

$$F_{r,sa}(t) = 1 - \exp(-r.sa.t) \sum_{n=0}^{sa-1} \frac{(r.sa.t)^n}{n!} \quad (2)$$

Let be an action having its duration following an Erlang distribution of use rate r and stochasticity absorption factor sa . The average duration of this action is r^{-1} with a variance of $r^{-2}sa^{-1}$. $F_{r,sa}(t)$ (2) gives the probability of firing the action within a time t . Given x actions having respectively use rates r_1, \dots, r_x and stochasticity absorption factors sa_1, \dots, sa_x , the probability that the y^{th} reaction will be fired is given by (3) [11].

$$\int_0^\infty f_{r_y,sa_y}(t) \prod_{w \neq y} (1 - F_{r_w,sa_w}(t)) dt \quad (3)$$

3.2 Stochastic π -Calculus Construction

The paper [11] shows how to simulate stochastic π -calculus models having actions following general probabilistic distributions. However, to our knowledge, no such suitable simulator have been implemented yet. Moreover, the majority of tools around the stochastic π -calculus assume the distribution of actions being exponential [3], [8], [12]. In this section, we propose a construction of $S\pi_{Er}$ processes into $S\pi_e$. This construction allows the simulation of $S\pi_{Er}$ processes with standard algorithms, as previously cited.

The construction is done through a map operator $[[\cdot]]_e$ where \cdot stands for a $S\pi_{Er}$ term. If P is a $S\pi_{Er}$ term, $[[P]]_e$ is a $S\pi_e$ term showing the same behaviour than P (this

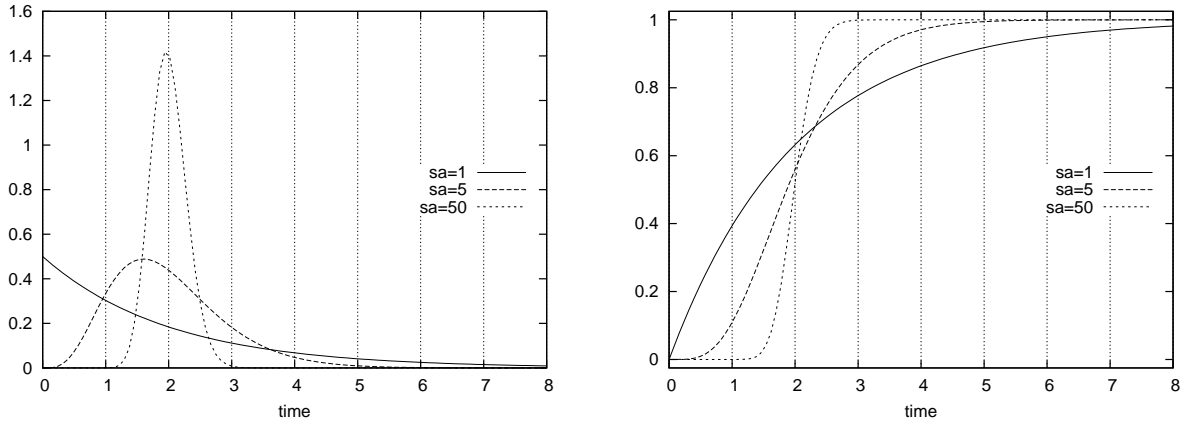


Fig. 2. Probability density function (left) and cumulative distribution function (right) of the Erlang distribution for different stochasticity absorption factors (sa) when rate is $\frac{1}{2}$.

is to be more formally detailed in this section). The full mapping of $S\pi_{Er}$ terms into $S\pi_e$ is given by Definition 2. It is worth to notice that this construction can be applied to any $S\pi_{Er}$ processes.

We sketch this transformation. Any action of rate r has to be replaced by sa actions of rate $r.sa$. Beforehand, to each couple of parameters r_a, sa_a of the $S\pi_{Er}$ process, a parameter $r'_a = r_a.sa_a$ is set for the resulting $S\pi_e$ process. Then, the main idea is to count the number of times an actions has been effective: until this number is equal to the stochasticity absorption factor, the current process is restarted. Let $A(\tilde{z})$ be a race condition, i.e. $A(\tilde{z}) ::= \sum_{i \in I} P_i$. By definition, each term of the sum starts by an action (possibly prefixed by restriction or enabling conditions). Basically, for each of these actions, indexed by i , a counter c_i is attached. This counter is given as an argument for A . Each time the i^{th} action is fired, the counter c_i is incremented by one, and until it reaches the given stochasticity absorption factor, the process A is restarted. This principle is straightforwardly applied to the internal actions (10). Handling channel input and output is more tricky. Consider the following $S\pi_{Er}$ process A_1 :

$$A_1 ::= \bar{a}.0 \quad A_2 ::= a.0 \quad A ::= A_1|A_2|A_2. \quad (4)$$

When counting the number of firing of the channel output \bar{a} , there must be a differentiation between responding instances of A_2 . To achieve this differentiation, the $S\pi_e$ process A_1 initiates a private communications with the responding process by sharing a new name a' . After sa_a enabling of the private communication a' , A_1 outputs on another private channel a'' to inform its correspondent that the correct number of communication has been done. Technically, this behaviour is achieved by storing into c_i the set of current active private communications and the counter of firing for each (11). For the channel input action, no counting is necessary: the process first inputs the two private channels a' and a'' and input on them by adding race conditions. An input on a' means the stochasticity absorption is not complete,

so the current process has to be restarted. An input on a'' means the communication is terminated, the next process is called. As for the channel output, the set of pair of private channels has to be stored in c_i (12).

Definition 2 ($\llbracket \cdot \rrbracket_e$). Let $\llbracket \cdot \rrbracket_e$ be the map from $S\pi_{Er}$ processes and definitions into $S\pi_e$ processes and definitions given by rules below.

Notations: \tilde{c} stands for c_1, \dots, c_n . c_i is either a positive integer (internal action), a set of triples $\langle c_{in}^{sa}, c_{in}^{a'}, c_{in}^{a''} \rangle$ (output) – n is the index of the element in the set, c_{in}^{sa} a positive integer, and $c_{in}^{a'}, c_{in}^{a''}$ a couple of channels – or a set of couples of channels $\langle c_{in}^{a'}, c_{in}^{a''} \rangle$ (input). $\tilde{c}\{c_i + = 1\}$ stands for \tilde{c} having c_i incremented by one (c_i has to be a positive integer). $\#c_i$ is the size of the set c_i . $\bar{a}\langle a', a'' \rangle$ (resp. $a\langle a', a'' \rangle$) stands for the outputting (resp. inputting) of the couple of names a', a'' .

$$\llbracket 0 \rrbracket_e = 0 \quad (5)$$

$$\llbracket \nu x P \rrbracket_e = \nu x \llbracket P \rrbracket_e \quad (6)$$

$$\llbracket P|Q \rrbracket_e = \llbracket P \rrbracket_e | \llbracket Q \rrbracket_e \quad (7)$$

$$\llbracket [cond]P \rrbracket_e = [cond]\llbracket P \rrbracket_e \quad (8)$$

Process definition:

$$\llbracket A(\tilde{z}) \rrbracket_e ::= \sum_{i \in I} P_i \rrbracket_e = A(\tilde{z}, \tilde{c}) ::= \sum_{i \in I} \llbracket P_i \rrbracket_e \quad (9)$$

with $\tilde{c} \cap bn(\sum_{i \in I} \llbracket P_i \rrbracket_e) = \emptyset$.

Internal action:

$$\begin{aligned} \llbracket \tau_t.P \rrbracket_e = & [c_i < sa_t] \tau_t.A(\tilde{z}, \tilde{c}\{c_i + = 1\}) \\ & + [c_i = sa_t] \tau_t.\llbracket P \rrbracket_e \end{aligned} \quad (10)$$

Channel output:

$$\begin{aligned} \llbracket \bar{a}y.P \rrbracket_e = & \nu a' \nu a'' \bar{a}\langle a', a'' \rangle.A(\tilde{z}, \tilde{c}') \\ & + \sum_{n=0}^{\#c_i} [c_{in}^{sa} < sa_a] \overline{c_{in}^{a'}}.A(\tilde{z}, \tilde{c}\{c_{in}^{sa} + = 1\}) + \\ & [c_{in}^{sa} = sa_a] \overline{c_{in}^{a''}}y.\llbracket P \rrbracket_e \end{aligned} \quad (11)$$

with $r_{a'} = r_a$ and $r_{a''} = \infty$, $\{a', a''\} \cap (fn(A) \cup fn(P)) = \emptyset$, and \tilde{c}' is \tilde{c} where c_i has a new element $\langle c_{in}^{sa}, c_{in}^{a'}, c_{in}^{a''} \rangle = \langle 1, a', a'' \rangle$, $n = \#c_i + 1$.

Channel input:

$$\begin{aligned} \llbracket ay.P \rrbracket_e &= a\langle a', a'' \rangle.A(\tilde{z}, \tilde{c}') \\ &+ \sum_{n=0}^{\#c_j} c_{jn}^{a'} \cdot A(\tilde{z}, \tilde{c}) + c_{jn}^{a''} y \cdot \llbracket P \rrbracket_e \end{aligned} \quad (12)$$

with $\{a', a''\} \cap (fn(A) \cup fn(P)) = \emptyset$, and \tilde{c}' is \tilde{c} where c_i has a new element $\langle c_{in}^{a'}, c_{in}^{a''} \rangle = \langle a', a'' \rangle$, $n = \#c_i + 1$.

Process call:

$$\begin{aligned} \llbracket A(\tilde{z}) \rrbracket_e &= A(\tilde{z}, \tilde{c}) \\ \text{with } \forall i \in I, c_i &= \begin{cases} 1 & \text{if the first action of } P_i \\ & \text{is an internal action,} \\ \emptyset & \text{else.} \end{cases} \end{aligned} \quad (13)$$

3.3 Correctness of the construction

To ensure that this construction is correct several points have to be proved. First, the qualitative behaviour has to be preserved: there must be an equivalence between transitions from any $S\pi_{Er}$ process $A(\tilde{z})$ to a process $B(\tilde{w})$ by the action π (noted as $A(\tilde{z}) \xrightarrow{\pi} B(\tilde{w})$) and a series of transitions starting from the $S\pi_e$ process $A_e(\tilde{z}, c_{A0})$ to the process $B_e(\tilde{w}, c_{B0})$ (denoted as $A_e(\tilde{z}, c_{A0}) \xrightarrow{e} \dots B_e(\tilde{w}, c_{B0})$), where A_e (resp. B_e) is the constructed $S\pi_e$ process from the $S\pi_{Er}$ process A (resp. B) using $\llbracket \cdot \rrbracket_e$ construction and c_{A0} and c_{B0} are computed by following (13). Finally, it must be ensured a transition in $S\pi_{Er}$ at rate r and stochasticity absorption factor sa results in sa transitions at rate $r \cdot sa$ in $S\pi_e$.

Figure 3 identifies the transition rules for a $S\pi_e$ process resulting from the construction defined in Definition 2.

At first, the inclusion of behaviours of $S\pi_{Er}$ process in the behaviours of the constructed $S\pi_e$ process is proven. Let $A(\tilde{z}) \xrightarrow{\pi} B(\tilde{w})$ be a transition, where π is either an internal action t ($\pi = \tau_t$) or a communication on channel a ($\pi = \dot{a}$). The index of the action π in the sum defined by A is denoted by i .

Internal action ($\pi = \tau_t$)

At the initial call of $A_e(\tilde{z}, \tilde{c})$, $c_i = 1$ (Equation (13)). As long as $c_i < sa_t$, only the rule TAU_{wait} can be applied. After each of these transitions, c_i is incremented by 1. When c_i reaches sa_t , TAU_{eff} is the only rule applicable, resulting in a transition towards $B_e(\tilde{w}, \tilde{d})$.

Thus, for each $A(\tilde{z}) \xrightarrow{\tau_t} B(\tilde{w})$, there exists a chain:

$$A_e(\tilde{z}, \tilde{c}) \xrightarrow{\tau_t} \dots A_e(\tilde{z}, \tilde{c}_k) \xrightarrow{\tau_t} B_e(\tilde{w}, \tilde{d})$$

where \tilde{c}_k is \tilde{c} having $c_i = k$. From the conditions for applying TAU_{eff} , it is obvious that $k = sa_t$. The duration of the chain follows an Erlang distribution of rate r'_t and stochasticity absorption sa_t .

Communication ($\pi = \dot{a}$)

We look for an equivalence of the COM transition $A(\tilde{z})|C(\tilde{z}') \xrightarrow{\dot{a}} B(\tilde{w})|D(\tilde{w}')$ in the $S\pi_e$ translated processes A_e, C_e, B_e, D_e . We assume $A(\tilde{z}) \xrightarrow{\dot{a}} B(\tilde{w})$ and $C(\tilde{z}') \xrightarrow{\dot{a}} D(\tilde{w}')$.

Initially, only rules IN_{init} and OUT_{init} can be applied:

$$A_e(\tilde{z}, \tilde{c})|C_e(\tilde{z}', \tilde{c}') \xrightarrow{\dot{a}} A_e(\tilde{z}, \tilde{c}_1)|C_e(\tilde{z}', \tilde{c}'_1)$$

where \tilde{c}_1 and \tilde{c}'_1 are computed according to (12) and (11). During this communication, B_e sends to A_e two fresh names a' and a'' . These names are shared only between $A_e(\tilde{z}, \tilde{c}_1)$ and $B_e(\tilde{z}', \tilde{c}'_1)$. After this first communication, $sa_a - 1$ transitions OUT_{wait} are necessary for applying the final OUT_{eff} :

$$\begin{aligned} A_e(\tilde{z}, \tilde{c}_1)|C_e(\tilde{z}', \tilde{c}'_1) &\xrightarrow{\dot{a}'} \dots A_e(\tilde{z}, \tilde{c}_{sa_a})|C_e(\tilde{z}', \tilde{c}'_{sa_a}) \\ &\xrightarrow{\dot{a}''} B_e(\tilde{w}, \tilde{d})|D(\tilde{w}', \tilde{d}') \end{aligned}$$

As a'' is an instantaneous transition and the rate of a' is the same as the rate of a , the duration of the chain follows an Erlang distribution of rate r'_a and stochasticity absorption sa_a .

Finally, from rules depicted in Figure 3, it can be deduced that $A_e(\tilde{z}, \tilde{c}) \xrightarrow{\pi} B_e(\tilde{w}, \tilde{d})$ implies $A(\tilde{z}) \xrightarrow{\pi} B(\tilde{w})$ — and there is a finite number of transitions between $A_e(\tilde{z}, \tilde{c})$ and $B_e(\tilde{w}, \tilde{d})$. From the previous points, it has been shown that the correct number of transition has to be done between the initial call of A_e and its final transition to B_e . This ends the correction of the proposed construction of the stochasticity absorption factor.

3.4 Toy Example

We apply the results obtained in this section to a toy example: an infinite looping sequence of two processes (14).

$$A_0 ::= \tau_t.A_1 \quad A_1 ::= \tau_t.A_0 \quad (14)$$

Basically, starting from A_0 , we expect to observe an infinite sequence of A_0 becoming A_1 becoming A_0 , etc. Between each transition, the interval action τ_t is performed.

Figure 4 plots the presence of A_0 and A_1 during simulations of $\llbracket A_0 \rrbracket_e$ under SPIM [8], [17] with an identical rate but different stochasticity absorption factors.

When no stochasticity absorption is applied, we observe a strong variance of the duration of the internal actions, as imposed by the exponential distribution. By increasing the stochasticity absorption factor, this variance reduces. Regular oscillations are observed with a high stochasticity absorption factor.

4 FIRING INTERVALS

The aim of this section is to point up a link between the couple use rate and stochasticity absorption factor of an action and the interval of time within it is fired

$\text{TAU}_{\text{wait}} \frac{A(\tilde{z}) \xrightarrow{r_t} {}_E B(\tilde{w})}{A_e(\tilde{z}, \tilde{c}) \xrightarrow{r'_t} {}_e A_e(\tilde{z}, \tilde{c}\{c_i += 1\})} c_i < sa_t$	$\text{TAU}_{\text{eff}} \frac{A(\tilde{z}) \xrightarrow{r_t} {}_E B(\tilde{w})}{A_e(\tilde{z}, \tilde{c}) \xrightarrow{r'_t} {}_e B_e(\tilde{w}, \tilde{c}_{B0})} c_i = sa_t$
$\text{IN}_{\text{init}} \frac{A(\tilde{z}) \xrightarrow{a_y} {}_E B(\tilde{w})}{A_e(\tilde{z}, \tilde{c}) \xrightarrow{a < a', a'' >} {}_e A_e(\tilde{z}, \tilde{c}')} $	$\text{OUT}_{\text{init}} \frac{A(\tilde{z}) \xrightarrow{\bar{a}_y} {}_E B(\tilde{w})}{A_e(\tilde{z}, \tilde{c}) \xrightarrow{\bar{a} < a', a'' >} {}_e A_e(\tilde{z}, \tilde{c}'')} $
$\text{IN}_{\text{wait}} \frac{A(\tilde{z}) \xrightarrow{a_y} {}_E B(\tilde{w})}{A_e(\tilde{z}, \tilde{c}) \xrightarrow{a'} {}_e A_e(\tilde{z}, \tilde{c})} \exists n, c_{in}^{a'} = a'$	$\text{OUT}_{\text{wait}} \frac{A(\tilde{z}) \xrightarrow{\bar{a}_y} {}_E B(\tilde{w})}{A_e(\tilde{z}, \tilde{c}) \xrightarrow{\bar{a}'} {}_e A_e(\tilde{z}, \tilde{c}\{c_{in}^{sa} += 1\})} \exists n, c_{in}^{a'} = a', c_{in}^{sa} < sa_a$
$\text{IN}_{\text{eff}} \frac{A(\tilde{z}) \xrightarrow{a_y} {}_E B(\tilde{w})}{A_e(\tilde{z}, \tilde{c}) \xrightarrow{a''_y} {}_e B_e(\tilde{w}, \tilde{c}_{B0})} \exists n, c_{in}^{a''} = a''$	$\text{OUT}_{\text{eff}} \frac{A(\tilde{z}) \xrightarrow{\bar{a}_y} {}_E B(\tilde{w})}{A_e(\tilde{z}, \tilde{c}) \xrightarrow{\bar{a}''_y} {}_e B_e(\tilde{w}, \tilde{c}_{B0})} \exists n, c_{in}^{a''} = a'', c_{in}^{sa} = sa_a$

Fig. 3. Operational semantics derived from Definition 2. i is the index of the action π in the sum defined by the $S\pi_{Er}$ process A . \tilde{c}' and \tilde{c}'' are computed respectively as in (12) and (11). \tilde{c}_{B0} is computed as in (13).

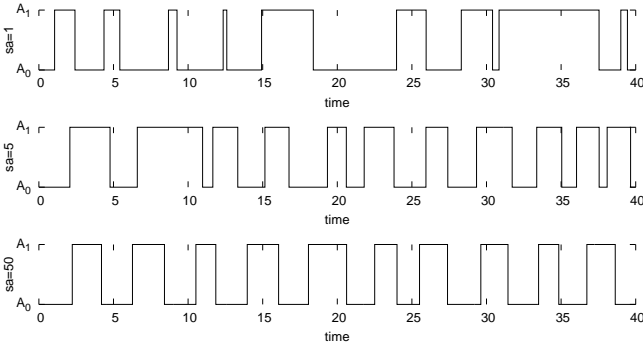


Fig. 4. Simulation of the $S\pi_{Er}$ process defined by (14) with rate $\frac{1}{2}$ and different absorption factors sa .

at a given confidence. That timed interval is referred as the *firing interval* (Definition 3). This section results in a set of statistical tools that may help the modelling and the checking of temporal and stochastic systems where durations of transitions follow an Erlang distribution.

Definition 3 (Firing Interval). *Given a use rate r and a stochasticity absorption factor sa , the Firing Interval at confidence coefficient $1 - \alpha$ is noted $FI_\alpha(r, sa) = [d; D]$ where $F_{r,sa}(d) = \frac{\alpha}{2}$ and $F_{r,sa}(D) = 1 - \frac{\alpha}{2}$.*

At the modelling time, and more especially when parametrising a model, it may be more natural to reason on firing intervals than on couples rate and stochasticity absorption factor. Here, we provide estimators and approximating functions to translate from and back a firing interval to rate and stochasticity absorption parameters.

These bring to the stochastic absorption factor different usages, depending on the knowledge and on the nature of the modelled system. On the one hand, the stochasticity absorption factor expresses a certainty on the precise action duration: the higher the confidence in the action duration is, the more the stochasticity absorption factor can be raised. On the other hand, the stochasticity absorption supplies the ability to reproduce actions with intrinsic stochasticity where time bounds

are known.

As supplementary modelling helpers, we point out there exists methods for inferring the shape [18] and the scale [19] parameters of a gamma distribution from a set of time measurement data. The conversion from gamma parameters to Erlang parameters is discussed below.

In the following of this section, the computation of the time interval from rate and stochasticity absorption parameters is first introduced. Finally, the computation of the rate and stochasticity absorption corresponding to a given time interval is tackled. To our knowledge, no prior study has been done in that way.

4.1 From Parameters to Firing Interval

Given the rate and stochasticity absorption parameters of an action, we are interested in computing the confidence interval for the time at which the action is fired. Let $1 - \alpha$ be the confidence coefficient for computing the firing interval. We search d and D such that $F_{r,sa}(d) = \frac{\alpha}{2}$ and $F_{r,sa}(D) = 1 - \frac{\alpha}{2}$, where $F_{r,sa}$ is the CDF of the Erlang distribution of use rate r and stochasticity absorption factor sa (2). The function which associates to $0 \leq x \leq 1$ the time t such that $F_{r,sa}(t) = x$ is known as the *quantile function*, and is noted F^{-1} .

Because of its relation with the incomplete gamma function, the quantile function of the gamma distribution, hence of the Erlang distribution, has no easy analytical expression and can not be used directly [20]. However, efficient approximation algorithms for F^{-1} exist [21], [22]. The widely used statistical tool R [23] proposes an implementation of such an algorithm. As R is distributed with a C programming language library, one can easily access to this implementation from independent programs. To compute the quantiles for the Erlang distribution using R , the `qgamma` function can be used. Here is an instance of a R session for computing the firing interval of an action of use rate r and stochasticity absorption factor sa :

```
d ← qgamma(α/2, shape = sa, rate = r * sa)
D ← qgamma(1 - α/2, shape = sa, rate = r * sa)
```

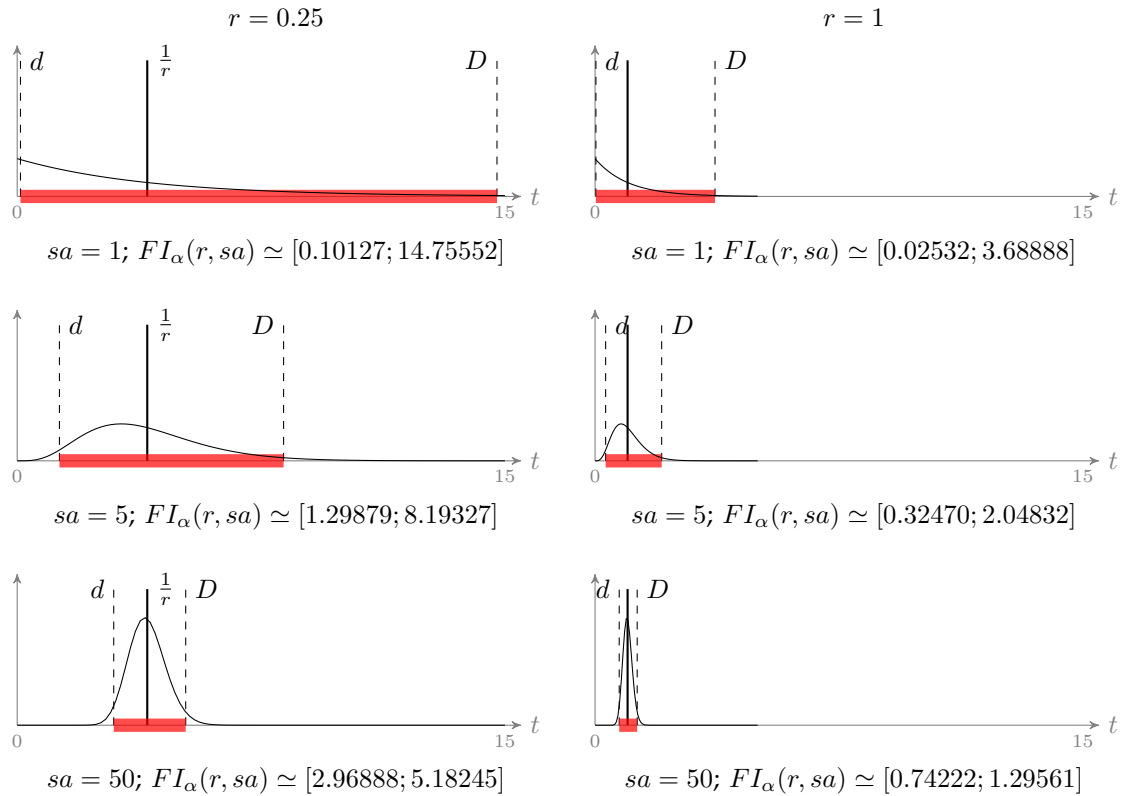


Fig. 5. Evolution of the firing interval when the stochasticity absorption factor sa increases. The thick vertical line is the average duration for each parameters. Confidence coefficient has been fixed to 95% ($\alpha = 0.05$).

Figure 5 shows the influence of the use rate and the stochasticity absorption factor on the firing interval.

4.2 From Firing Interval to Parameters

Given a firing interval $[d; D]$ at a confidence coefficient $1 - \alpha$, we look for a rate r and stochastic absorption factor sa such that $FI_\alpha(r, sa) = [d, D]$. To achieve this goal, an estimator of r and sa (respectively \hat{r}_α and \hat{sa}_α) is built in function of the lower and upper bound of the firing interval d and D .

In a first phase, the integer constraint on the stochasticity absorption factor is released to consider the gamma distribution of use rate $r \in \mathbb{R}_+^*$ and stochasticity absorption factor $sa \in \mathbb{R}_+^*$. As there is no analytical expression of the quantile function of the gamma distribution, expressing r and sa in function of the confidence interval can not be done analytically either. Therefore, to be able to build an estimator, firing intervals have been computed for a huge amount of use rates r and stochasticity absorption factors. Estimators are then obtained by a regression on the generated data. For this report, d and D have been computed at a confidence coefficient of 95% for all rounded $\log sa$ between 0 and 4.5 (step of 0.1) and all $\log r$ between -8 and 2 (step of 0.1). Figure 6 shows the heat map for r and sa parameters in function of the bounds d and D of firing intervals for such a generation.

From this obtained 3-D plots, regressions fitting the data have been manually determined. Parameters of

these regressions have then been abstracted as they may depend on the confidence coefficient value. The estimator obtained for the use rate, noted \hat{r}_α , is given by (15), and the one for the stochasticity absorption factor, noted \hat{sa}_α is given by (16).

$$\hat{r}_\alpha = (w + x \exp(-y \cdot d))(d + D)^{-1} \quad (15)$$

$$\hat{sa}_\alpha = \exp\left(u \left(\frac{D}{d}\right)^v\right), \quad (16)$$

where u, v, w, x, y are parameters depending on the confidence coefficient $1 - \alpha$.

Finally, R has been used to valuate the parameters of these estimators from a set of generated data points. Table 1 sums up the estimators found for r and sa at different confidence coefficients. The lack of usable analytical expressions around the gamma distribution makes it difficult to evaluate. Figure 7 shows an attempt to evaluate the quality of these estimators by comparing, for generated data, the expected value and the estimated value.

It is worth to notice that the estimators are bijective functions, i.e. to each firing interval corresponds one and only one use rate and stochasticity absorption factor. The resting point is the integer constraints on the stochasticity absorption factor. Because of the bijectivity of the estimators, if the estimated stochasticity absorption is not an integer, there is no Erlang distribution fitting with the given firing interval. From this observation, the search for approximating Erlang distribution parameters

is at the confidence of the modeller. However, one can notice that rounding the estimated stochasticity absorption factor to the upward (resp. downward) integer results in a firing interval included by (resp. including the) originally given firing interval. In that way, one can easily estimate an over or under approximation of the couple of parameters matching an arbitrary firing interval.

4.3 Sequence of Actions

To conclude this section, we briefly discuss about the distribution of a sequence of actions and its relation with firing intervals.

Given k actions at respectively rates r_1, \dots, r_k and stochasticity absorption factors sa_1, \dots, sa_k , and considering they are fired successively, what is the firing interval of the sum of the actions?

It can be easily checked that the firing interval $[d; D]$ of a sequence of Erlang distributed actions is not the sum of the firing intervals of the individual actions — i.e. $d \neq d_1 + \dots + d_k$ where d_i is the lower bound of the firing interval of the i^{th} action ; D is computed similarly. However, the sum of Erlang distributed random variables with different parameters has been studied in [24], [25] and [26] gives an easily computable expression of the CDF for such a distribution. In that way, the firing interval of the sum of Erlang distributed random variables can be computed using standard approximation techniques of the quantile function (like bisections). The dual operation consisting of inferring the parameters of the Erlang sum from the firing interval raises several difficulties, such as the lost of bijectivity. We consider such an issue as out of the scope of this report.

5 MODEL CHECKING USING PRISM

The probabilist model checker PRISM [9] offers efficient model checking for CTMCs. In PRISM, actions are defined within PRISM modules. Each module has a finite set of local variables. The union of the local variables of all modules gives the global state of the model, denoted by V .

$$[act] \text{ guard} \rightarrow r : (x'_1 = u_1) \ \& \ \dots \ \& \ (x'_k = u_k)$$

where act is an optional action label, $guard$ a predicate over V , x_i is a local variable and u_i a function over V . $r \in \mathbb{R}_+^*$ is the use rate of the action and is assumed to 1 when omitted. To be applicable, a labelled action has to be synchronised with an action of the same label in another module. The rate of such a synchronised action is the product of the rate of both actions.

An efficient translation of the classical exponentially distributed stochastic π -calculus ($S\pi_e$) into PRISM has been proposed by Norman et al. [12]. Their translation requires the overall process structure to be rearrangeable to the form $P = \nu x_1 \dots \nu x_k (P_1 | \dots | P_n)$ where each P_i contains no ν operator nor recursive use of the $|$ operator, especially to ensure a finite number of states. In that

way, given a $S\pi_{Er}$ process P respecting these constraints, the constructed $S\pi_e$ process $\llbracket P \rrbracket_e$ does not respect this limitation. Indeed, our proposed construction of the output action into $S\pi_e$, presented in (11), involves a recursive generation of fresh names a', a'' . A solution is then to directly translate the $S\pi_{Er}$ process to PRISM.

In this section, the translation of $S\pi_e$ process to PRISM proposed by Norman et al. is adapted to the translation of $S\pi_{Er}$ process respecting previously cited structure. Therefore, this allows an efficient model checking of $S\pi_{Er}$ processes, which is a new result.

To end, the overall approach of this report is illustrated by a toy example.

5.1 Construction of the Stochasticity Absorption

Let P be a stochastic π -calculus expression of the form $P = \nu x_1 \dots \nu x_k (P_1 | \dots | P_n)$ where each P_i contains neither the ν or $|$ operator. In that way, each process P_i can be described by a transition graph where nodes are race conditions annotated by Q_i, R_i, \dots [12]. Hence, each Q_i, R_i, \dots stands for a state of the process P_i . Using the translation detailed in [12], the PRISM model corresponding to P can be computed. It results in n PRISM modules, one per P_i , each having a variable s_i representing the current state of the process P_i . Variables representing names to be sent or to be received are also attached to modules. Obtained actions are of three different main forms:

$$\llbracket (s_i = Q_i) \ \& \ M \rightarrow r_t : (s'_i = R_i) \rrbracket \quad (17)$$

$$[a_P_i_P_j_y] (s_i = Q_i) \ \& \ M \rightarrow r_a : (s'_i = R_i) \quad (18)$$

$$[a_P_j_P_i_y] (s_i = Q_i) \ \& \ M \rightarrow (s'_i = R_i) \ \& \ (z' = y) \quad (19)$$

Each of these forms represents the process P_i at state Q_i applying a certain action under the condition M and changing to state R_i . These actions are respectively the internal action (17), the output (either bound or free) of y on a channel a to P_j (18) and the input of y as z on a channel a from P_j (19). Depending on the boundness of the sent name y , supplementary conditions are added to M . In the following of this section, we assume that any identical action label inside a same module have disjoint guards, i.e. both are never part of a same race condition.

As for the construction of the stochasticity absorption factor in $S\pi_e$, a counter is attached to each PRISM action. For identifying internal actions, a label is attached to them: P_i_t is the label of the internal action τ_t of P_i . The set of labels of internal and output actions of P_i is denoted by $\mathcal{L}_{P_i \diamond} = \{a_P_i_P_j_y, \dots, P_i_t, \dots\}$, and the set of labels of input actions of P_i is denoted by $\mathcal{L}_{\diamond P_i} = \{a_P_j_P_i_y, \dots\}$. Basically, there is one counter for each action having a label in $l \in \mathcal{L}_{P_i \diamond}$. This counter is defined as a local variable c_l of the PRISM module for P_i . Each time an output or an internal action is performed, the corresponding counter is incremented by one. The update of the action is proceed when this

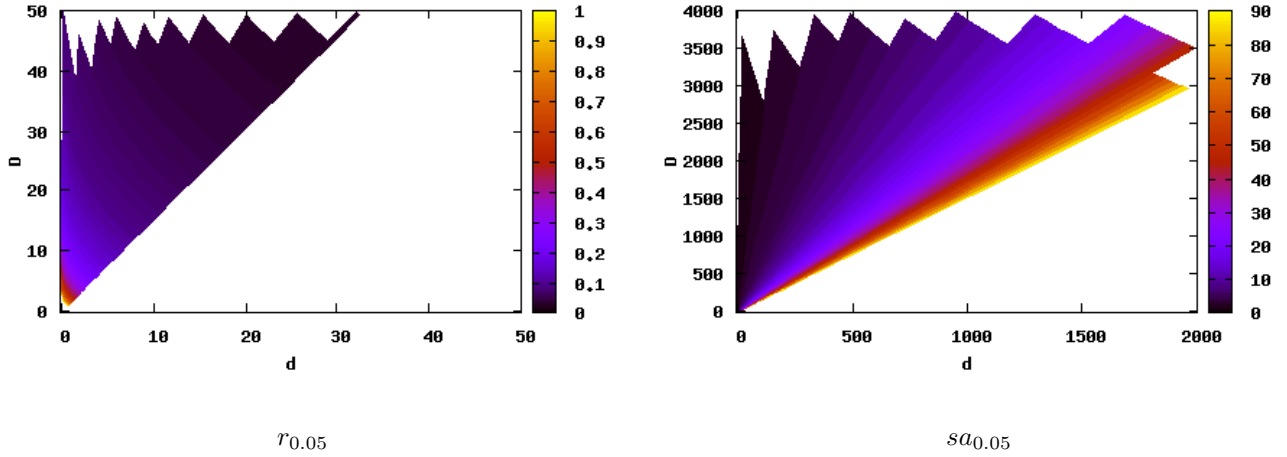
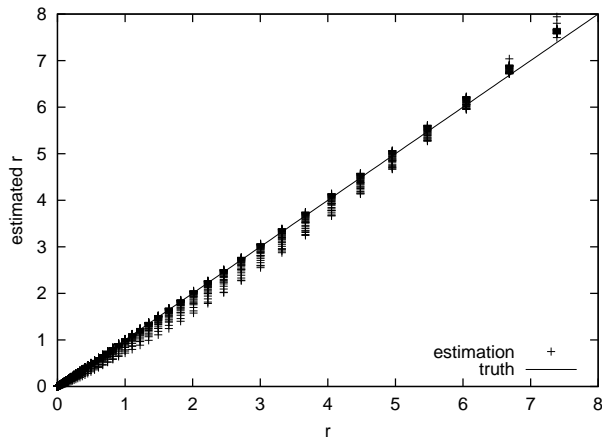


Fig. 6. Heat map of the use rate (left) and stochasticity absorption factor (right) in function of the bounds d and D of the Firing Interval $FI_{0.05} = [d, D]$ for some generated data.

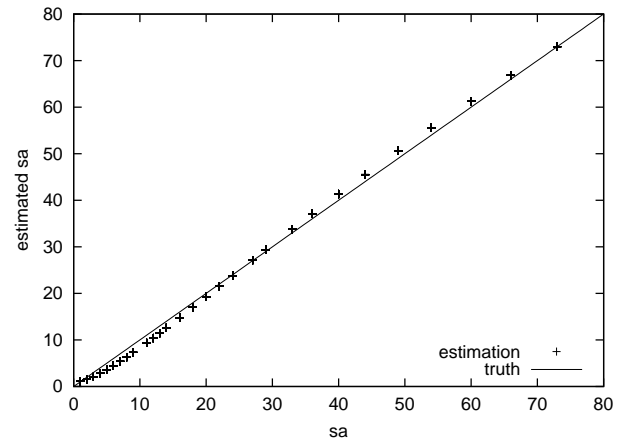
TABLE 1

Obtained estimators of parameters r and sa for the Firing Interval $[d; D]$ at different confidence coefficients.

	$1 - \alpha = 0.90$	$1 - \alpha = 0.95$	$1 - \alpha = 0.99$
\hat{r}_α	$(2.13 + 2.89 \exp(-44.46d))(d + D)^{-1}$	$(2.06 + 1.93 \exp(-36.49d))(d + D)^{-1}$	$(2.03 + 1.39 \exp(-33.33d))(d + D)^{-1}$
\hat{sa}_α	$\exp\left(6.39 \left(\frac{d}{D}\right)^{-0.66}\right)$	$\exp\left(6.41 \left(\frac{d}{D}\right)^{-0.87}\right)$	$\exp\left(6.41 \left(\frac{d}{D}\right)^{-1.04}\right)$



evaluation of $\hat{r}_{0.05}$



evaluation of $\hat{sa}_{0.05}$

Fig. 7. Evaluation of obtained estimators for use rate r and stochasticity absorption factor sa with confidence coefficient of 95%. Points are the estimated parameter value. A perfect estimator would put all points on the truth line.

counter reaches the expected stochasticity absorption factor.

To ensure a correct count of absorption, all counters related P_i have to be reset when P_i changes its state. This raises a concern concerning the reset of counters not owned by P_i , as this is the case for all actions labelled by $\mathcal{L}_{\diamond P_i}$. To cope with such an issue, a boolean variable d_l is defined in the module P_i for each $l = a_{P_j} P_i y \in \mathcal{L}_{\diamond P_i}$, and is set to true when the reset of c_l is required.

The following subsection precisely describes transformations to apply to each of the PRISM modules P_i to

add support for the stochasticity absorption factor.

Additional local variables

For each label $l \in \mathcal{L}_{P_i \diamond}$, the variable c_l stands for the counter of the stochasticity absorption.

$$c_l : [1..sa_l] \text{ init } 1;$$

For each label $l \in \mathcal{L}_{\diamond P_i}$, the boolean variable d_l is true if P_i has changed its state while the absorption of the action played by l has started. In other terms, d_l is

true if the associated counter c_l has to be reset.

$$d_l : \text{bool init false};$$

Hereafter, the PRISM update for the reset of all stochasticity absorption counter is denoted by $R_{P_i \diamond}$ (20). The update of d_l variables, $l \in \mathcal{L}_{\diamond P_i}$, is denoted by $S_{\diamond P_i}$ (21). Basically, d_l is set to true if and only if the associated counter c_l is different from its default value.

$$R_{P_i \diamond} \stackrel{\text{def}}{=} \bigwedge_{l \in \mathcal{L}_{\diamond P_i}} (c_l' = 1) \quad (20)$$

$$S_{\diamond P_i} \stackrel{\text{def}}{=} \bigwedge_{l \in \mathcal{L}_{\diamond P_i}} (d_l' = c_l > 1) \quad (21)$$

with $\bigwedge_{i \in \{1, \dots, k\}} u_i = u_1 \ \& \ \dots \ \& \ u_k$.

Internal action

Let $l = P_i.t$ be the label of an action resulting from the translation of an internal action τ_t . The use rate and stochasticity absorption factor of this internal action are respectively r_t and sa_t . The PRISM action respects the following form:

$$[] G \rightarrow r_t : U;$$

where G stands for the guard of the action and U for the updates to perform. The stochasticity absorption of the action is achieved by replacing the previous statement by the actions below:

$$\begin{aligned} [] G \ \& \ (c_l < sa_t) \rightarrow r_t * sa_t : (c_l' = c_l + 1); \\ [] G \ \& \ (c_l = sa_t) \rightarrow r_t * sa_t : U \ \& \ R_{P_i \diamond} \ \& \ S_{\diamond P_i}; \end{aligned}$$

Channel output

Let $l = a.P_i.P_j.y$ be the label an action resulting from the translation of an output of name y on channel a . The use rate and stochasticity absorption factor of this channel are respectively r_a and sa_a . The PRISM action respects the following form:

$$[l] G \rightarrow r_a : U;$$

where G stands for the guard of the action and U for the updates to perform. The stochasticity absorption of the action is achieved by replacing the previous statement by the actions below.

$$\begin{aligned} [l_wait] G \ \& \ d_l \rightarrow r_a * sa_a : (c_l' = 2); \\ [l_wait] G \ \& \ !d_l \ \& \ (c_l < sa_a) \rightarrow r_a * sa_a : \\ & \ (c_l' = c_l + 1); \\ [l] G \ \& \ !d_l \ \& \ (c_l = sa_a) \rightarrow r_a * sa_a : \\ & \ U \ \& \ R_{P_i \diamond} \ \& \ S_{\diamond P_i}; \end{aligned}$$

Channel input

Let $l = a.P_j.P_i.y$ be the label of an action resulting from the translation of an input of name y on channel a . The PRISM action respects the following form:

$$[l] G \rightarrow U;$$

where G stands for the guard of the action and U for the updates to perform. The stochasticity absorption of the action is achieved by replacing the previous statement by the actions below:

$$\begin{aligned} [l_wait] G \rightarrow (d_l' = false); \\ [l] G \rightarrow U \ \& \ R_{P_i \diamond} \ \& \ S_{\diamond P_i}; \end{aligned}$$

5.2 Toy Example

As an application of the use of the PRISM model checker for analysing $S\pi_{Er}$ processes, and as an illustration of the overall method presented by this report, we propose the study of an example process P defined in (22).

$$\begin{aligned} A_1 &::= a.A_0 + \bar{b}.A_1 & B_1 &::= b.B_0 + \bar{a}.B_1 \\ A_0 &::= \perp & B_0 &::= \perp \\ P &::= \nu a \nu b \nu \perp (A_1 | B_1) \end{aligned} \quad (22)$$

Intuitively, two stories can happen: either A_1 outputs first on b then B_1 becomes B_0 and the system ends in deadlock ; or B_1 outputs first on a then A_1 becomes A_0 and the system ends in deadlock.

For this toy example, we search for reducing to near zero the probability that A_1 becomes A_0 , i.e. the probability p that B_1 outputs on a . As supplementary constraints, the average duration for using channels a and b is fixed to respectively 4 and 1 time units (i.e. $r_a = 0.25$ and $r_b = 1$). For instance, these constraints may have been imposed by observations of the real system modelled by P . The property to verify is expressed in PRISM as $\mathbf{P}=? [\mathbf{F} (s_a=0)]$ which means the probability that A_0 is reached.

We first study this model as a $S\pi_e$ process. Listing 1 shows the result of the translation of P into PRISM using the method of Norman et al. Verifying the previously given property with PRISM results in a probability for reaching A_0 of 0.2.

Let us consider P as a $S\pi_{Er}$ process. For the sake of simplicity, we will consider that both a and b have the same stochasticity absorption factor.

In terms of firing interval, we look for a stochasticity absorption factor with which the firing interval of b is entirely before the firing interval of a . By computing the firing interval for both of these actions at different absorptions factors — as done in Figure 8 — one can observe the probability of firing a before b will be seriously decreased with a stochasticity absorption factor of 5. By using a stochasticity absorption factor of 50, we expect the probability of firing a first to be near zero.

To confirm these results, we now turn to the model checking of the $S\pi_{Er}$ process P using PRISM. Listing 2 shows the translation of P into PRISM using the construction presented above. With a stochasticity absorption of 5, the probability that A_1 reaches A_0 is divided by 100 (almost 0.02) comparing without stochasticity absorption. As expected, increasing this stochasticity absorption factor to 50 drops down this probability to approximately 10^{-11} .

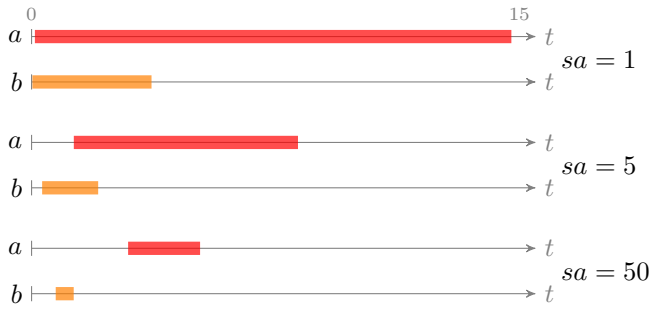


Fig. 8. Firing intervals at confidence coefficient 95% for a and b (22) with different stochasticity absorption factors but constant use rates $r_a = 0.25$ and $r_b = 1$.

```
ctmc
const double r_a = 0.25;
const double r_b = 1;

module proc_A
  s_a: [0..1] init 1;

  [a_B1_A1] (s_a=1) -> (s_a'=0);
  [b_A1_B1] (s_a=1) -> r_b: (s_a'=1);
endmodule

module proc_B
  s_b: [0..1] init 1;

  [b_A1_B1] (s_b=1) -> (s_b'=0);
  [a_B1_A1] (s_b=1) -> r_a: (s_b'=1);
endmodule
```

Listing 1. Translation of the example stochastic π -calculus model (22) as a $S\pi_e$ process into PRISM.

6 CONCLUSION

In this report, we presented and proved a technique for tuning temporal features within the stochastic π -calculus. This tuning is done through a stochasticity absorption factor which reduces the variance around the average duration of transitions. This absorption is achieved by replacing the exponential distribution for firing actions by the Erlang distribution.

We proved a translation of the Erlang distributed stochastic π -calculus into the exponential distributed one. In this way, such tuned models can still be efficiently simulated with standard algorithms. The assignment of the temporal and stochastic parameters has been discussed and we detailed the link with the classical timed interval for firing transitions. We showed that by using estimators it is possible to parameterise stochastic π -calculus models by specifying either the parameters of the Erlang distribution or the firing interval at a given confidence level. Finally, we added support for this stochasticity absorption factor within the translation of the stochastic π -calculus to PRISM, allowing efficient model-checking of these models.

As further works, we are interested in improving the efficiency of the model-checking for Erlang distributed stochastic π -calculus by the study of the sum of Erlang random variables with different parameters.

```
ctmc
const double r_a = 0.25; const int sa_a = 5;
const double r_b = 1; const int sa_b = 5;

module proc_A
  s_a: [0..1] init 1;
  c_b_A1_B1: [1..sa_b] init 1;
  d_a_B1_A1: bool init false;

  [a_B1_A1_wait] (s_a=1) -> (d_a_B1_A1'=false);
  [a_B1_A1] (s_a=1) -> (s_a'=0) & (d_a_B1_A1'=false) &
    (c_b_A1_B1'=1);

  [b_A1_B1_wait] (s_a=1) & d_b_A1_B1 -> r_b:
    (c_b_A1_B1'=2);
  [b_A1_B1_wait] (s_a=1) & !d_b_A1_B1 & (c_b_A1_B1<sa_b)
    -> r_b: (c_b_A1_B1'=c_b_A1_B1+1);
  [b_A1_B1] (s_a=1) & !d_b_A1_B1 & (c_b_A1_B1=sa_b) ->
    r_b: (s_a'=1) & (c_b_A1_B1'=1) &
    (d_a_B1_A1'=c_a_B1_A1+1);
endmodule

module proc_B
  s_b: [0..1] init 1;
  c_a_B1_A1: [1..sa_a] init 1;
  d_b_A1_B1: bool init false;

  [b_A1_B1_wait] (s_b=1) -> (d_b_A1_B1'=false);
  [b_A1_B1] (s_b=1) -> (s_b'=0) & (d_b_A1_B1'=false) &
    (c_a_B1_A1'=1);

  [a_B1_A1_wait] (s_b=1) & d_a_B1_A1 -> r_a:
    (c_a_B1_A1'=2);
  [a_B1_A1_wait] (s_b=1) & !d_a_B1_A1 & (c_a_B1_A1<sa_a)
    -> r_a: (c_a_B1_A1'=c_a_B1_A1+1);
  [a_B1_A1] (s_b=1) & !d_a_B1_A1 & (c_a_B1_A1=sa_a) ->
    r_a: (s_b'=1) & (c_a_B1_A1'=1) &
    (d_b_A1_B1'=c_b_A1_B1+1);
endmodule
```

Listing 2. Translation of the example stochastic π -calculus model (22) as a $S\pi_{Er}$ process into PRISM.

APPENDIX

AN ALTERNATIVE CONSTRUCTION FOR THE STOCHASTICITY ABSORPTION FACTOR WITHIN THE STOCHASTIC π -CALCULUS

In this appendix, we provide an alternative to that of Definition 2 for the construction of the stochasticity absorption factor within the stochastic π -calculus. The main motivation for this new construction is to prevent the use of an indexing set for the race condition which depend on parameters of the process. That is the case for the proposed construction of the channel output (11) and input (11). The construction hereafter can then be fully implemented in SPIM, for instance.

Basically, instead of having a race condition growing after each communication instantiation, a process handling the stochasticity absorption for each new communication is composed in parallel. To identify the process P_i winning the race condition $\sum_{i \in I} P_i$, a name c_i is bound to each component of the race condition. When the process handling a communication has completed its absorption, it outputs on this channel c_i . In that way, the translation of the call for the race condition (13) is replaced by:

$$[A(\tilde{z})]_e = A(\tilde{z}, \tilde{c}) \quad (23)$$

$$\text{with } \forall i \in I, c_i = \begin{cases} 1 & \text{if the first action of } P_i \\ & \text{is an internal action,} \\ \nu c_i & \text{else, with } r_{c_i} = \infty \end{cases}$$

Channel output: The process defined below takes as parameters the stochasticity absorption factor to reach sa , the counter of firing c , channels a', a'' as defined in (11), and the channel r for outputting when the absorption is completed.

$$A_{send}(sa, c, a', a'', r) ::= \begin{aligned} & [c < sa] \bar{a}'.A_{send}(sa, c + 1, a', a'', r) \\ & + [c = sa] \bar{r}a'' \end{aligned} \quad (24)$$

(11) is then replaced by the following:

$$[\bar{a}y.P]_e = \begin{aligned} & \nu a' \nu a'' \bar{a} < a', a'' > .(A(\tilde{z}, \tilde{c})|A_{send}(sa, 1, a', a'', c_i)) \\ & + c_i a'' . \bar{a}'' y . [P]_e \end{aligned} \quad (25)$$

with $r'_{a'} = r'_{a''} = \infty$ and $\{a', a''\} \cap (fn(A) \cup fn(P))$.

Channel input: Similarly, the process defined below inputs either on the channel a' (absorption in progress) on the channel a'' (communication done), and outputs on r the data received by a'' to the parent race condition.

$$A_{recv}(a', a'', r) ::= \begin{aligned} & a'.A_{recv}(a', a'', r) \\ & + a'' y . \bar{r} y \end{aligned} \quad (26)$$

Finally, (12) is replaced as follows:

$$[ay.P]_e = \begin{aligned} & a < a', a'' > .(A(\tilde{z}, \tilde{c})|A_{recv}(a', a'', c_i)) \\ & + c_i y . [P]_e \end{aligned} \quad (27)$$

with $\{a', a''\} \cap (fn(A) \cup fn(P)) = \emptyset$.

REFERENCES

- [1] R. Milner, *Communicating and mobile systems: the π -calculus*. New York, NY, USA: Cambridge University Press, 1999.
- [2] C. Priami, "Stochastic pi-Calculus," *The Computer Journal*, vol. 38, no. 7, pp. 578–589, 1995.
- [3] C. Priami, A. Regev, E. Shapiro, and W. Silverman, "Application of a stochastic name-passing calculus to representation and simulation of molecular processes," *Inf. Process. Lett.*, vol. 80, no. 1, pp. 25–31, 2001.
- [4] C. Kuttler and J. Niehren, "Gene regulation in the pi calculus: Simulating cooperativity at the lambda switch," *Transactions on Computational Systems Biology*, vol. 4230, no. VII, pp. 24–55, Nov. 2006.
- [5] R. Blossey, L. Cardelli, and A. Phillips, "Compositionality, stochasticity and cooperativity in dynamic models of gene regulation," *HFSP Journal*, vol. 2, no. 1, pp. 17–28, Feb 2008.
- [6] L. Cardelli, E. Caron, P. Gardner, O. Kahramanogullari, and A. Phillips, "A process model of actin polymerisation," *Electronic Notes in Theoretical Computer Science*, vol. 229, no. 1, pp. 127–144, January 2009.
- [7] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.
- [8] A. Phillips and L. Cardelli, "Efficient, correct simulation of biological processes in the stochastic pi-calculus," in *Computational Methods in Systems Biology*, ser. LNCS, vol. 4695. Springer, Sep 2007, pp. 184–199.
- [9] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: A tool for automatic verification of probabilistic systems," in *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, ser. LNCS, H. Hermanns and J. Palsberg, Eds., vol. 3920. Springer, 2006, pp. 441–444.
- [10] R. Alur and D. Dill, *Lecture Notes in Computer Science*, 1992, ch. The theory of timed automata, pp. 45–73.
- [11] C. Priami, "Stochastic π -calculus with general distributions," in *Proc. of the 4th Workshop on Process Algebras and Performance Modelling, CLUT*, 1996, pp. 41–57.
- [12] G. Norman, C. Palamidessi, D. Parker, and P. Wu, "Model checking probabilistic and stochastic extensions of the π -calculus," *IEEE Transactions on Software Engineering*, vol. 35, no. 2, pp. 209–223, 2009.
- [13] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston, "Performance analysis of probabilistic timed automata using digital clocks," *Formal Methods in System Design*, vol. 29, pp. 33–78, 2006.
- [14] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, "Model-checking continuous-time markov chains," *ACM Trans. Comput. Logic*, vol. 1, no. 1, pp. 162–170, 2000.
- [15] C. Baier and H. Hermanns, "Model-checking algorithms for continuous-time markov chains," *IEEE Trans. Softw. Eng.*, vol. 29, no. 6, pp. 524–541, 2003, senior Member-Haverkort, Boudewijn and Member-Katoen, Joost- Pieter.
- [16] S. Donatelli, S. Haddad, and J. Sproston, "Model checking timed and stochastic properties with CSL^{TA} ," *IEEE Transactions on Software Engineering*, vol. 35, no. 2, pp. 224–240, 2009.
- [17] A. Phillips, *SPIM*, <http://research.microsoft.com/~aphillip/spim>.
- [18] A. Zaigraev and A. Podraza-Karakulska, "On estimation of the shape parameter of the gamma distribution," *Statistics & Probability Letters*, vol. 78, no. 3, pp. 286 – 295, 2008.
- [19] J. Mi and A. Naranjo, "Inferences about the scale parameter of the gamma distribution based on data mixed from censoring and grouping," *Statistics & Probability Letters*, vol. 62, no. 3, pp. 229 – 243, 2003.
- [20] D. J. Wilkinson, *Stochastic Modelling for Systems Biology (Mathematical and Computational Biology)*. Chapman & Hall/CRC, April 2006.
- [21] D. Best and D. Roberts, "Algorithm as 91: The percentage points of the chi-squared distribution," *Applied Statistics*, vol. 24, no. 3, pp. 385–390, 1975.
- [22] A. R. DiDonato and A. H. Morris, Jr., "Computation of the incomplete gamma function ratios and their inverse," *ACM Trans. Math. Softw.*, vol. 12, no. 4, pp. 377–393, 1986.
- [23] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2009, ISBN 3-900051-07-0. [Online]. Available: <http://www.R-project.org>
- [24] S. Amari and R. Misra, "Closed-form expressions for distribution of sum of exponential random variables," *Reliability, IEEE Transactions on*, vol. 46, no. 4, pp. 519–522, Dec 1997.
- [25] S. Nadarajah, "A review of results on sums of random variables," *Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications*, vol. 103, no. 2, pp. 131–140, Sep 2008.
- [26] S. Favaro and S. Walker, "On the distribution of sums of independent exponential random variables via wilks' integral representation," *Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications*, 2008.