P-🔔: A Process Algebra for Modeling Prioritized Stochastic Timed Systems

(Extended Abstract)

Martín Ariel Domínguez *, Gerardo Javier Saiz **, and Pedro R. D'Argenio

FaMAF, Universidad Nacional de Córdoba,
Ciudad Universitaria, 5000 Córdoba, Argentina
{mdoming, gerardo, dargenio}@famaf.unc.edu.ar

Abstract. We present P-🔔, a Stochastic Process Algebra (SPA) that allows for the modeling of timed systems with priorities and urgency. We define the semantics of P-🔔 in terms of Prioritized Stochastic Automata (PSA), an extension of automata with clock events, priorities and probabilistic symbolic transitions. PSAs are symbolic objects that have a concrete semantics on Probabilistic Timed Transition Systems (PTTS). Therefore, P-🔔 has semantics in two steps in terms of PTTS. We also define several operators directly on PTTS. They include parallel composition and a prioritizing operator. We show that this operators applied to PTTS commute (modulo probabilistic bisimulation) with their relatives in P-🔔.

1 Introduction

The design and analysis of systems, like embedded systems, communication and network protocols or multi-media systems, requires insight both in functional aspects and in the real-time and performance aspects. It is known that a separate focus on this aspects has a negative impact in the design process [16, 17]. Usually a system is fully design and functionally tested before making any attempt to determine its performance characteristics [17]. This is negative for the design process since an undesired performance measurement may require to redesign the system. Stochastic Process Algebra (SPA) [18, 14, 15, 3, 9, 4, etc.] provides a framework to integrate both aspects of analysis.

SPA are process algebras [19, 23, 2, 13] extended with operations that allows to represent the occurrence time of events. Moreover, this occurrence time is governed by a probability distribution function. A first group of SPA, the so called called Markovian SPAs, only considers negative exponential distributions (e.g. [18, 15, 3]). These distributions have a nice property —the memoryless property— that allows for good axiomatizations. However, they impose a clear limitation. To overcome this, a second kind of SPA allows for generally distributed timed events (e.g. [14, 20, 9, 4]. In particular, results of [9, 4] overcame the earlier axiomatization problems and moreover, their techniques allow for reasonable symbolic semantics. A profound technical study of SPA with generally distributed timed events is given in [6].

This work extends the second kind of SPAs with prioritized events. Priorities are a key ingredient when designing real-time systems. They allow that urgent events are realized as soon as possible and, moreover, to be executed before less important events. In this paper, we introduce P-🔔, an extension of 🔔 [10, 9] that allows for modeling urgency and a priority scheme on actions.

P-🔔 divides actions in two types: urgent and delayable. A delayable action is an action that, once enabled, can wait indefinitely before actually occurring. Therefore, they are prepared to interact with the environment, waiting as long as necessary to synchronize with another component. Urgent actions, on the contrary, cannot wait: as soon as they become enable, they (or any other action enabled at that time) have to be performed. As a consequence, they cannot interact with

* Junior Research fellow of Agencia Córdoba Ciencia
** Junior Research fellow of CONICET

the environment. Besides, P-🔔 provides a priority order on urgent actions. That is, there are urgent actions that are more important than others. The priority order is simply an order relation on the sets of urgent actions.

Apart from this structure on the set of actions, P-🔔 provides an operation that changes delayable actions into urgent. This operation has a two folded functionality: it closes or encapsulates a system (i.e. it is not open to interact), hence activity becomes urgent, and, at the same time, it imposes a priority order on actions.

In this paper we introduce the syntax and semantics of P-🔔. We give a symbolic semantics in terms of the so-called Prioritized Stochastic Automata (PSA) [6] which, in turn, has a concrete semantics in terms of Probabilistic Timed Transition Systems (PTTS). Hence every P-🔔 term P has also meaning in terms of PTTS. Let us denote it by $[\![P]\!]$. Besides, we define some operations, like parallel composition and the prioritizing operation, directly on PTTSs and state that they commute with respect to P-🔔 operations, namely, that if P and Q are P-🔔 terms then $[\![P \otimes Q]\!]$ and $[\![P]\!] \otimes [\![Q]\!]$ are equivalent modulo probabilistic bisimulation [21, 6], where $\oplus$ is the operator of interest (e.g. parallel composition).

Related Work. Priorities in process algebras were defined in many different styles (e.g. [1, 8, 7]) and they also appear in timed and probabilistic process algebras (e.g [25, 22, 27]). To our knowledge priorities only appear in the Markovian version of SPAs [3, 5]. In this two works, every occurrence of an action is associated to a natural number that represents its priority value. This is not the approach that we follow. Our case is closely related to [1], one of the first works on priorities, in which the set of actions is ordered according to a priority order relation. However, in that work, activity is pruned according to a prioritizing operator. In our case, an operation is used to set priorities to non-prioritized actions (namely, to transform delayable actions into urgent actions). The activity is only pruned in the concrete semantics.

Urgency is present in several SPAs [14, 15, 4] as maximal progress, that is, time is not allowed to progress if a τ step is enabled. We follow exactly the same principle, but instead it is applied to a family of actions rather than to a single one.

Organization of this paper. The paper is organized as follows. Section 2 introduces PSA and its semantics is defined in terms of PTTS in Section 3. Section 4 introduces P-🔔. Its symbolic semantic is defined in Section 5. In Section 6 we define operations on PTTS and show that they commute with the original P-🔔 operations. An illustrating example is shown in section 7. The paper conclusions are reported in Section 8. Proofs are omitted due to space limitations. They can be found in [11].

2 *PSA*

PSA is introduced in [6] and considers the most commons ingredients presents in the model of stochastic timed systems: clock start, clock termination, execution of actions and probabilistic jumps. In addition, this model also considers priorities and allow to represent maximal progress and urgency. All this ingredients are included in only one symbolic transition. This is a model for represent timed systems with general distributions and is showed that many of the approaches in literature can be mapped into PSA [6].

With PDF we denote the set of all probability distributions functions and with $Prob(\Omega)$ we denote the set of all probability spaces with sample space in a subset of Ω. We let $Prob_d(\Omega)$ denote the subset of $Prob(\Omega)$ containing only discrete probability spaces. We use $\rho, \rho', \rho_i, \ldots$ to denote discrete probability spaces and $\pi, \pi', \pi_i, \ldots$ to denote probability spaces in general. In both cases, we overload the notation and use this letters to represent the probability measure as well.

Definition 1. *A* prioritized stochastic automata *(PSA) is a structure* $(St, Ck, Distr, Act, \longrightarrow, s_0)$ *where*

- St *is a set of* control states *with* $s_0 \in St$ *being the* initial control state.
- Ck *is the set of* clock names. *For the sake of clarity in technical manipulation, we assume that* Ck *is totally ordered, and that if* $C \subseteq Ck$, $\vec{C}$ *is the vector induced by this order.*

- *Distr* : $Ck \to PDF$ *assigns a probability distribution function to each clock. If* $f \in PDF$*, we usually name a clock* $c_f \in Ck$ *to indicate that* $Distr(c_f) = f$*.*
- Act *is set of* actions *partitioned in the following two sets:*
  - Act_d*, the set of* delayable *actions and*
  - Act_u*, the set of* urgent *actions*

  where Act_u *is ordered according to a* priority relation $\prec$ *which is a strict partial order.*
- $\longrightarrow \subseteq St \times 2^{Ck} \times Act \times Prob_d(2^{Ck} \times St)$ *is the* control transition *relation.*

We write $s \xrightarrow{C,a} \rho$ if $(s, C, a, \rho) \in \longrightarrow$ and $s \xrightarrow{C,a}$ if there is a distribution ρ such that $s \xrightarrow{C,a} \rho$. To trigger a control transition like this, all clocks in set C must terminate (the clock c is *active* as long as it does not reach this termination time. Otherwise we say it is *terminated*). Transitions are labeled with actions. They can be delayable, meaning that they need to interact with the environment, or they can be urgent and they would not. Urgent actions impose maximal progress and must be executed as soon as they are enabled. If a conflict between two enabled urgent transitions occurs, it may be solved according to the priority relation on the actions. In addition, when a transition is executed, a probabilistic branching will take place according to the probability space ρ. $\rho(C', s')$ is the probability that all clocks in C' are started and the system reaches the control state s'.

3 *PTTS*

In this section we give a brief description of $PTTS$ as presented in [6], we also look at the so called "residual life time semantics" of PSA in terms of $PTTS$ and define bisimulations in both models.

3.1 The *PTTS* model

The $PTTS$ model [6] is an extension of Segala's simple probabilistic automata [24] with continuous probability spaces and time labeled transitions.

Definition 2. *A* probabilistic timed transition system *(PTTS) is a structure* $(\Sigma, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow, \sigma_0)$ *where*

- Σ *is a set of* states*;*
- Act *is a set of* actions *like in PSA;*
- $\longrightarrow \subseteq \Sigma \times (Act \cup \mathbb{R}_{\geq 0}) \times Prob(\Sigma)$ *is the* transition relation*; and*
- $\sigma_0 \in \Sigma$ *is the initial state.*

If $(\sigma, a, \pi) \in \longrightarrow$*, we write* $\sigma \xrightarrow{a} \pi$*. We also write* $\sigma \xrightarrow{a}$ *if there is a probability space* π *such that* $\sigma \xrightarrow{a} \pi$*,* $\sigma \not\xrightarrow{a}$ *if there is no* π *such that* $\sigma \xrightarrow{a} \pi$*, and* $\sigma \xrightarrow{a} \sigma'$ *if* $\sigma \xrightarrow{a} \pi$ *and* π *is a trivial probability space where the atom* $\{\sigma'\}$ *has measure 1.*

In addition, the following requirements must hold:

1. *maximal progress:* $\forall \sigma \in \Sigma, a \in Act_u.\ \sigma \xrightarrow{a} \implies \nexists t \in \mathbb{R}_{\geq 0}.\ \sigma \xrightarrow{t}$
2. *priority:* $\forall \sigma \in \Sigma, \{a, b\} \subseteq Act_u.\ a \prec b \wedge \sigma \xrightarrow{a} \implies \sigma \not\xrightarrow{b}$

3.2 Concrete semantics of *PSA*

The semantics of PSA is defined in terms of probabilistic timed transition systems. As we consider a continuous time domain the concrete model allows us to represent the behavior of the symbolic probabilistic automata when it is "executed". In this concrete semantics, when a clock c is started, its termination time is sampled according to $Distr(c)$. In this context, a control transition $s \xrightarrow{C,a} \rho$ becomes enabled as soon as all clocks in C are terminated. To carry the time value and the termination value of a clock, we use valuations. A *valuation* is a function v from Ck in the set $\mathbb{R}_{\geq 0}$ of non-negative real numbers. Let Val be the set of all valuations on Ck.

A state in the semantics is a triple (s, v, e) where s is a control state in PSA, v is the *time valuation*, and e is the *enabling valuation*. Therefore, given a clock c, $v(c)$ registers the time that has passed since c was started, and $e(c)$ registers its termination time which was sampled when it was started. Notice that c is active if $v(c) \leq e(c)$. As a consequence, a control transition $s \xrightarrow{C,a} \rho$ is enabled in state (s, v, e) whenever $v(c) \geq e(c)$ for all clocks in C. This is denoted by $enabled(s \xrightarrow{C,a} \rho, v, e)$,

Let $\mathcal{R}(f_1, \ldots, f_k)$ be the probability space in the k-dimensional real space with the unique probability measure induced by the probability distribution functions $f_1, \ldots, f_k$.
If $C = \{c^1_{f_1}, \ldots, c^k_{f_k}\} \subseteq Ck$. $\mathcal{R}(Distr(\vec{C}))$ denotes the probability space $\mathcal{R}(f_1, \ldots, f_k)$.

Besides, we use the following notation. Given a probability space π and $p \in [0, 1]$, $p \cdot \pi$ is the measurable spaced obtained by multiplying p to the probability measure of π. Given a denumerable set of probability spaces π_i, $i \in I$, $\sum_{i \in I} \pi_i$ is the measurable space obtained by appropriately summing the measures of the different probability spaces.

Definition 3. *Let $PSA = (St, Ck, Distr, Act, \longrightarrow, s_0)$. Its concrete semantics is defined by the PTTS $[\![PSA]\!] = (\Sigma, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow, \sigma_0)$ where:*

- $\Sigma \stackrel{\text{def}}{=} St \times Val \times Val$
- $\longrightarrow$ *is defined by the following rules:*

$$\frac{enabled(s \xrightarrow{C,a} \rho, v, e) \quad a \in Act_u \implies \nexists b \in Act_u.\ a \prec b \wedge enabled(s \xrightarrow{C'',b} \rho'', v, e)}{(s, v, e) \xrightarrow{a} \sum_{\substack{s' \in St \\ C' \subseteq Ck}} \rho(C', s') \cdot sample^{s',C'}_{v,e}(\mathcal{R}(Distr(\vec{C}')))} \quad (1)$$

$$\frac{\forall t'.\ 0 \leq t' < t,\ \forall a \in Act_u.\ \neg enabled(s \xrightarrow{C,a} \rho, v + t', e)}{(s, v, e) \xrightarrow{t} (s, v + t, e)} \quad (2)$$

- $\sigma_0 \stackrel{\text{def}}{=} (s_0, v_0, v_0)$*; with* $v_0 \stackrel{\text{def}}{=} 0$ *for all* $c \in Ck$
- $enabled(s \xrightarrow{C,a} \rho, v, e) \stackrel{\text{def}}{\iff} \forall c \in C.\ v(c) \geq e(c)$*;*
- $sample^{s,C}_{v,e}(\vec{t}) \stackrel{\text{def}}{=} (s, v[\vec{C}/0], e[\vec{C}/\vec{t}])$*; and*
- *for all* $c \in Ck$, $(v + t)(c) \stackrel{\text{def}}{=} v(c) + t$, *and* $v[\vec{C}/\vec{t}](c) \stackrel{\text{def}}{=} \vec{t}(i)$ *whenever there is an index* i *such that* $c = \vec{C}(i)$*, otherwise* $v[\vec{C}/\vec{t}](c) \stackrel{\text{def}}{=} v(c)$.

Rule (1) defines the execution of a control transition $s \xrightarrow{C,a} \rho$ requiring, therefore, that it is enabled. Notice that if a is an urgent action, it is also necessary to check that no control transition with higher priority is also enabled. In addition, the postcondition of the concrete transition is a random selection of a control state together with the set of clocks to be started and a sample terminating value for this clocks. Function $sample^{s',C'}_{v,e}$ takes care of appropriately constructing the next state.

Rule (2) controls the passage of time. It states that the system is allowed to stay in the control state s as long as no urgent action becomes enabled. As a consequence, maximal progress on urgent action is ensured.

3.3 Bisimulations

In this section we define bisimulation relations both on the symbolic model and the concrete model. We use concrete bisimulation as our correctness criteria. The definition of the symbolic bisimulation introduced in [6] is a straightforward modification of probabilistic bisimulation [21] in order to fit clocks.

Definition 4. *Given a PSA, a relation $R \subseteq St \times St$ is a* (symbolic) bisimulation on *PSA if the following statements hold:*

1. *R is an equivalence relation,*
2. *whenever $\langle s_1, s_2 \rangle \in R$ and $s_1 \xrightarrow{C',a} \rho_1$, there is a probability space ρ_2 such that*
   (a) *$s_2 \xrightarrow{C',a} \rho_2$ and*
   (b) *$\rho_1(S) = \rho_2(S)$ for every equivalence class $S \in (Ck \times St)/R_{Ck}$ induced by the relation $R_{Ck} \stackrel{\text{def}}{=} \{\langle (C, s_1), (C, s_2) \rangle \mid C \subseteq Ck, \langle s_1, s_2 \rangle \in R\}$.*

Two control states s_1 and s_2 are bisimilar, *notation $s_1 \sim s_2$, if there is a bisimulation R such that $\langle s_1, s_2 \rangle \in R$. Two PSA, PSA_1 and PSA_2 are* bisimilar, *notation $PSA_1 \sim PSA_2$, if the initial states are bisimilar in the disjoint union of PSA_1 and PSA_2.*

The definition of the concrete bisimulation is more involved since it must deal with continuous probability spaces (see e.g. [26, 9, 4]). In particular, we follow the definition given in [4]. We first give some necessary definitions.

Let $(\Omega, \mathcal{F}, \mu)$ and $(\Omega', \mathcal{F}', \mu')$ be two probability spaces. We say that they are *equivalent* (notation $(\Omega, \mathcal{F}, \mu) \approx (\Omega', \mathcal{F}', \mu')$) if (a) for all $A \in \mathcal{F}$, $A \cap \Omega' \in \mathcal{F}'$ and $\mu(A) = \mu'(A \cap \Omega')$, and (b) for all $A' \in \mathcal{F}'$, $A' \cap \Omega \in \mathcal{F}$ and $\mu'(A') = \mu(A' \cap \Omega)$.

Given an equivalence relation R on a set Σ and a set $I \subseteq \Sigma$, we define the function $EC_{I,R} : I \to \Sigma/R$ which maps each state $\sigma \in I$ into the corresponding equivalence class $[s]_R$ in Σ.

Definition 5. *Given a PTTS, a relation $R \subseteq \Sigma \times \Sigma$ is a* bisimulation on *PTTS if the following statements hold:*

1. *R is an equivalence relation, and*
2. *whenever $\langle \sigma_1, \sigma_2 \rangle \in R$ and $\sigma_1 \xrightarrow{a} \pi_1$, there is a probability space π_2 such that*
   (a) *$\sigma_2 \xrightarrow{a} \pi_2$ and*
   (b) *$EC_{\Sigma_1,R}(\pi_1) \approx EC_{\Sigma_2,R}(\pi_2)$ where Σ_i is the sample space of π_i.*

Two states σ_1 and σ_2 are bisimilar, *notation $\sigma_1 \sim \sigma_2$, if there is a bisimulation R such that $\langle \sigma_1, \sigma_2 \rangle \in R$. Two PTTS, PT_1 and PT_2 are* bisimilar, *notation $PT_1 \sim PT_2$, if $\sigma_0^1 \sim \sigma_0^2$ in the disjoint union of PT_1 and PT_2.*

4 P-♤

Most of the approaches to formal specifications and analysis of concurrent systems whose activity duration depends on general probabilistic functions agree in a three levels structure: process algebra level, symbolic semantics level and concrete semantic level. PSA fits in the symbolic semantic level, the concretes semantics level is given by means of $PTTS$. However, no adequate (in the sense of taking full advantage of expressiveness) process algebra for PSA was presented. That is the issue we want to address with our process algebra P-♤. P-♤ is based in ♤ [9], and extends a transitional process algebra with operators for handling random clocks, prioritized actions and probabilistic transitions. $C \mapsto P$ with $C \subseteq Ck$ is the clocks triggering operator, $a.\sum_{i=1}^{n}(p_i, \{\!|C_i|\!\}, Q_i)$ with $a \in Act, C_i \subseteq Ck, p_i \in [0,1], \sum_{i=1}^{n} p_i = 1, Q_i \in$ P-♤ is the probabilistic prefix and $\theta_f(P)$ with $f : Act_d \longrightarrow Act_u$ being a partial function, is the priority operator.

4.1 Syntax

Let $Act = Act_d \cup Act_u$ a set of actions, Ck a set of clocks having each one a distribution function associated. The syntax of P-🔔 is defined according to the following grammar:

$$P ::= \mathbf{0} \mid X \mid P + P \mid a.\sum_{i=1}^{n}(p_i, \{\!|C_i|\!\}, P) \mid P \,||_A\, P \mid C \mapsto P \mid \theta_f(P) \mid P[\phi]$$

where $a \in Act$ is an action name, $\sum_{i=1}^{n} p_i = 1$, $C, C_i \in 2^{Ck}$ are sets of random clocks, $A \subseteq Act_d$ is the set of synchronization, $\phi : Act \longrightarrow Act$ is a homomorphic rename function, $f : Act_d \longrightarrow Act_u$ is a partial prioritizing function and X is a process variable belonging to the set V of process variables.

5 Symbolic Semantic of P-🔔

The rules for the semantics of P-🔔 in PSA are given in table 1, they capture the intuitive behavior explained in the following. As usual, process $\mathbf{0}$ represents a process that cannot perform any action but is allowed to idle indefinitely. $a.\sum_{i=1}^{n}(p_i, \{\!|C_i|\!\}, Q_i)$, is the process that, after performing action a, it resets clocks in C_i, and behave like Q_i with probability p_i. That induce a discrete probabilistic space ρ such that $\rho(C_i, Q_i) = p_i$, and $\rho(C, P) = 0$ if $(C, P) \notin \{(C_1, P_1), ..., (C_n, P_n\}$. $C \mapsto P$ can perform any activity P can, but with the restriction that any delayable activity must wait until all clocks in C are terminated. $P + Q$ behave as P or Q but not both. The parallel composition $||_A$ executes P and Q in parallel, and they are synchronized by the set of delayable actions A. We remark that synchronization may happen if both process are ready to do it, so we do not allow urgent actions to synchronize. Synchronization implies one of the involved parts may have to wait, but we do not want to delay an enabled urgent action. This can also be noticed in the rule for $C \mapsto P$. Process $\theta_f(P)$ behaves like P, except that delayable actions from P in the domain of f are renamed to urgent actions and hence prioritized.

Definition 6. *Let $\longrightarrow$ be the least relation satisfying the rules in Table 1. For $P \in$ P-🔔 define $PSA(P) = (St_P, Ck, Distr, Act, \longrightarrow, P)$ where St_P as in is the set of all* P-🔔 *processes reachable from P trough $\longrightarrow$, Ck and $Distr$ are just like in* P-🔔*, and P is the initial state.*

Now, we can obtain the concrete semantic of a process by first obtaining its symbolic semantic in terms of PSA as in definition 6 and then deriving the concrete semantic of the resulting prioritized stochastic automaton as in definition 3. So, let $P \in$ P-🔔 we write $[\![P]\!]$ as the result of this two steps semantic: $[\![P]\!] \stackrel{\text{def}}{=} PTTS[\![PSA(P)]\!]$.

6 Compositionality of P-🔔 operators

We showed in the previous section how the concrete semantic of a P-🔔 process can be obtained. In this section we define the behavior of several P-🔔 operators over arbitrary $PTTS$'s and prove that these operators are compositional respect of the concrete semantics. We do that for the parallel composition, the nondeterministic sum, the priority operator and the renaming operator. Since in an arbitrary $PTTS$ clocks do not exists, we cannot do the same for the clock triggering operator or the probabilistic prefix.

So, let $PT = (\Sigma, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow, \sigma_0)$, $PT_1 = (\Sigma_1, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow_1, \sigma_0^1)$, $PT_2 = (\Sigma_2, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow_2, \sigma_0^2)$, $P \in$ P-🔔, and for $\sigma \in \Sigma$ we define ρ_σ as $\rho_\sigma \in PROB(\Sigma)$ such that $\rho_\sigma(\sigma) = 1$. We redefine $||_A$, $+$, θ_f and $[\phi]$ to run on arbitrary PTTSs. They can be found in

$$a.\sum_{\mathrm{I}}(p_i, \{\!|C_i|\!\}, Q_i) \xrightarrow{\emptyset,a} \rho' \qquad \frac{P \xrightarrow{C_1,a} \rho \; a \in Act_d}{C \mapsto P \xrightarrow{C \cup C_1,a} \rho} \qquad \frac{P \xrightarrow{C_1,a} \rho \; a \in Act_u}{C \mapsto P \xrightarrow{C_1,a} C \mapsto (\rho)}$$

$$\frac{P \xrightarrow{C_1,a} \rho_1 \quad Q \xrightarrow{C_2,a} \rho_2 \quad a \in A}{P \,||_A\, Q \xrightarrow{C_1 \cup C_2,a} M_A(\rho_1, \rho_2)} \qquad \frac{P \xrightarrow{C,a} \rho \quad a \notin A}{\begin{array}{l} P \,||_A\, Q \xrightarrow{C,a} M_{A,Q}(\rho) \\ Q \,||_A\, P \xrightarrow{C,a} M_{Q,A}(\rho) \end{array}} \qquad \frac{P \xrightarrow{C,a} \rho}{\begin{array}{l} P + Q \xrightarrow{C,a} \rho \\ Q + P \xrightarrow{C,a} \rho \end{array}}$$

$$\frac{P \xrightarrow{C,a} \rho \quad a \notin Dom(f)}{\theta_f(P) \xrightarrow{C,a} \theta_f(\rho)} \qquad \frac{P \xrightarrow{C,a} \rho \quad a \in Dom(f)}{\theta_f(P) \xrightarrow{C,f(a)} \theta_f(\rho)} \qquad \frac{P \xrightarrow{C,a} \rho}{P[\phi] \xrightarrow{C,\phi(a)} \rho[\phi]}$$

- $\rho'(\Sigma, \mathcal{F}, P)$, $\Sigma = St \times 2^{Ck}$, $P(Q, C) = \begin{cases} p_i \text{ if } Q = Q_i \wedge C = C_i \\ 0 \qquad \text{otherwise} \end{cases}$
- $M_{A,Q}(\rho)$ is a space such that:
  $P_{M_{A,Q}(\rho)}(< P \,||_A\, Q, C >) = P_\rho(< P, C >)$
  $P_{M_{A,Q}(\rho)}(< R, C >) = 0$ if $\nexists P.R \equiv P \,||_A\, Q$
- $M_A(\rho_1, \rho_2)$ is a space such that:
  $P_{M_A(\rho_1,\rho_2)}(< P_1 \,||_A\, P_2, C_1 \cup C_2 >) = P_{\rho_1}(< P_1, C_1 >).P_{\rho_2}(< P_2, C_2 >)$
  $P_{M_A(\rho_1,\rho_2)}(< R, C >) = 0$ if $\nexists P_1, P_2.R \equiv P_1 \,||_A\, P_2$
- $\theta_f(\rho)$ is a space such that
  $P_{\theta_f(\rho)}(\theta_f(P), C) = P_\rho(< P, C >)$
  and $P_{\theta_f(\rho)}(< R, C >) = 0$ if $\nexists P.R \equiv \theta_f(P)$
- $C \mapsto (\rho)$ is a space such that
  $P_{C \mapsto (\rho)}(< C \mapsto P, C >) = P_\rho(< P, C >)$
  and $P_{C \mapsto (\rho)}(< R, C' >) = 0$ if $\nexists P.R \equiv C \mapsto P$
- $\phi : Act \longrightarrow Act$ is an homomorphism with respect to the priority order $\prec$

Table 1. Rules for P-🔔

Table 2. We explain the intuitive behavior for the rules in the following. For $PT_1 \,||_A\, PT_2$ the states of the composition are pairs of states of each $PTTS$. Actions not in A are executed independently, while actions in A are required to synchronize. Urgent actions can be executed only if no other action with higher priority is enabled. If both $PTTS$ can idle for t time units, then the composition can also idle t time units. Finally, actions in A impose synchronization. The non deterministic sum behaves as one of the $PTTS$'s but not both, the only considerations are that it can let time pass before deciding which branch it takes (if both $PTTS$'s can) and if the first action of each option is urgent, the higher prioritized path must be chosen if it exists. The definitions of operators θ_f and ϕ are straightforward, only considering that the prioritizing operator can enable new urgent transitions, that is, the common restriction for urgent actions must be checked again.

The following theorem proves compositionality for each P-🔔 operator in Table 2 respect of the concrete semantic. We use bisimulation as our correctness criteria.

Theorem 1. *Let $P, P_1, P_2 \in$ P-🔔 with associated clocks sets Ck, Ck_1 and Ck_2 respectively:*

- $[\![P_1]\!] \,||_A\, [\![P_2]\!] \sim [\![P_1 \,||_A\, P_2]\!]$ *provided* $Ck_1 \cap Ck_2 = \emptyset$
- $[\![P_1]\!] + [\![P_2]\!] \sim [\![P_1 + P_2]\!]$
- $\theta_f([\![P]\!]) \sim [\![\theta_f(P)]\!]$
- $[\![P]\!][\phi] \sim [\![P[\phi]]\!]$

$PT_1 \|_{_A} PT_2 \stackrel{\text{def}}{=} (\Sigma_1 \times \Sigma_2, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow_{12}, (\sigma_0^1, \sigma_0^2))$

$$\frac{\sigma_1 \xrightarrow{a}_1 \rho_1 \ \wedge (a \in Act_u \implies \sigma_2 \not\xrightarrow{b}_2 \forall b : b \succ a)}{(\sigma_1, \sigma_2) \xrightarrow{a}_{12} \rho_1 \times \rho_{\sigma_2}} \qquad \frac{\sigma_2 \xrightarrow{a}_2 \rho_2 \ \wedge (a \in Act_u \implies \sigma_1 \not\xrightarrow{b}_1 \forall b : b \succ a)}{(\sigma_1, \sigma_2) \xrightarrow{a}_{12} \rho_{\sigma_1} \times \rho_2}$$

$$\frac{\sigma_1 \xrightarrow{t}_1 \sigma_1',\ \sigma_2 \xrightarrow{t}_2 \sigma_2'}{(\sigma_1, \sigma_2) \xrightarrow{t}_{12} (\sigma_1', \sigma_2')} \qquad \frac{\sigma_1 \xrightarrow{a}_1 \rho_1,\ \sigma_2 \xrightarrow{a}_2 \rho_2,\ \ a \in A}{(\sigma_1, \sigma_2) \xrightarrow{a}_{12} \rho_1 \times \rho_2}$$

$PT_1 + PT_2 \stackrel{\text{def}}{=} (\Sigma_1 \cup \Sigma_2 \cup \Sigma_1 \times \Sigma_2, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow_+, (\sigma_0^1, \sigma_0^2))$

$$\frac{\sigma_i \xrightarrow{a}_i \rho \qquad a \in Act \wedge (a \in Act_u \implies \nexists b \succ a : \sigma_{i'} \xrightarrow{b}_{i'} \rho', i, i' \in \{1,2\}, i \neq i')}{(\sigma_i, \sigma_{i'}) \xrightarrow{a}_+ \rho}$$

$$\frac{\sigma_1 \xrightarrow{t}_1 \sigma_1',\ \sigma_2 \xrightarrow{t}_2 \sigma_2'}{\begin{array}{c}(\sigma_1, \sigma_2) \xrightarrow{t}_+ (\sigma_1', \sigma_2') \\ (\sigma_2, \sigma_1) \xrightarrow{t}_+ (\sigma_2', \sigma_1')\end{array}} \qquad \frac{\sigma \xrightarrow{a}_i \rho}{\sigma \xrightarrow{a}_+ \rho}\ i \in \{1,2\}$$

$\theta_f(PT) \stackrel{\text{def}}{=} (\Sigma_f, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow_f, \theta_f(\sigma_0)),\ \Sigma_f = \theta_f(\Sigma)$

$$\frac{\sigma \xrightarrow{a} \rho \qquad a \notin Dom(f) \wedge (a \in Act_u \implies \nexists b \in Dom(f) : f(b) \succ a : \sigma \xrightarrow{f(b)})}{\theta_f(\sigma) \xrightarrow{a}_f \theta_f(\rho)}$$

$$\frac{\sigma \xrightarrow{t} \sigma' \ \wedge\ \forall a \in Dom(f), t' < t : \nexists \sigma''.\ \sigma \xrightarrow{t'} \sigma'' \xrightarrow{a}}{\theta_f(\sigma) \xrightarrow{t}_f \theta_f(\sigma')}$$

$$\frac{\sigma \xrightarrow{a} \rho,\ a \in Dom(f) \wedge\ \nexists b \succ f(a) : \sigma \xrightarrow{b} \wedge\ \ \nexists b \in Dom(f) : f(b) \succ f(a) \wedge\ \sigma \xrightarrow{b}}{\theta_f(\sigma) \xrightarrow{f(a)}_f \theta_f(\rho)}$$

$PT[\phi] \stackrel{\text{def}}{=} (\Sigma_\phi, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow_\phi, \sigma_0[\phi]),\ \ \Sigma_\phi = \{\sigma[\phi] : \sigma \in \Sigma\}$

$$\frac{\sigma \xrightarrow{a} \rho}{\sigma[\phi] \xrightarrow{\phi(a)}_\phi \rho[\phi]} \qquad \frac{\sigma \xrightarrow{t} \rho}{\sigma[\phi] \xrightarrow{t}_\phi \rho[\phi]}$$

Table 2. Rules for $\|_{_A}$, $+$, θ_f and ϕ operators

7 Modeling using P-🔔

We will use the shorthand notations $a.(\{\!|C_1|\!\}, S_1)$ for $a.\sum_{i=1}^{1}(p_i, \{\!|C_i|\!\}, S_i)$ if $p_1 = 1$, and if additionally $C_1 = \emptyset$ we use $a.S_1$. When n is fixed we explicit the summation using $\oplus$ symbol. For example we use $a.((p_1, \{\!|C_1|\!\}, P_1) \oplus (p_2, \{\!|C_2|\!\}, P_2))$ for $a.\sum_{i=1}^{2}(p_i, \{\!|C_i|\!\}, P_i)$

In the following example we try to show how the operators we introduce can be used for modeling, we use the probabilistic jump and the prioritizing operators. The last one have three main functionalities. First, it allows us to make urgent a delayable action that not longer needs to communicate, hence hiding it from the others processes (in the sense of communication). This change in the action status also implies that, as soon as the action is enabled, it cannot let time pass any more, this is the second functionality. The third is the ordering of delayable actions that

become urgent, allowing us to define an order relation among them. In the next example, priorities are not required. For an integral example we refer to [11].

The alternate bit protocol [2,12] is an algorithm for the correct transmission of a data stream form one place to another via so called *faulty channel*. Here a *faulty channel* is a medium that can garble, duplicate or lose data in some way. At the macroscopic level the problem can be described as a black box such that in the left side there is an *in* of stream data and in the right side there is an *out*. The intention is that the stream ejected be equal to the stream injected. Internally (inside of the box), we have a sender S, sending elements a to the receiver R via the *faulty channel* K. After having received a certain data element, R will send an acknowledge to S via the same channel K. If an *ack* is not received by S a timeout occurs and the data must be retransmitted. In this example we model with P-🔔 (in Table 3) a variant of the alternate bit protocol where channel K has two ways to fail, lose the message (with a probability of 0.05) or garble it (with a probability of 0.05). The delayable actions are $a(i), b(i), c(i), d(i), b(\bot), c(\bot), in, out$ and the urgent ones are $\underline{a(i)}, \underline{b(i)}, \underline{c(i)}, \underline{d(i)}, \underline{b(\bot)}, \underline{c(\bot)}, \underline{\tau}$ for $i \in \{0,1\}$.

The Sender alternates between process $S(0)$ and $S(1)$ as follows: when data arrive from outside via delayable action in, $S(i)$ sends a message to the channel appending alternating bit i (this is a control bit, used to detect duplicate messages) with the action $a(i)$ and resets the clock y, which has a trivial discrete distribution function. There are three possibilities:

- a timeout occurs; then the message is sent again,
- the Sender receives an *ack* with alternating bit i, the message was successfully transmitted, the sender waits for a new *in* message in process $S(1-i)$,
- $S(i)$ receives an *ack* with alternating bit $1-i$ or an incorrect *ack* $b(\bot)$ (we suppose that if the message is corrupted by the channel we can detect it, we use the symbol $\bot$ for such corrupted message), the message is discarded and $S(i)$ keeps waiting for a correct *ack*.

The channel, for each received message, has a 5% chance of losing it, and 5% of garble it. In any case the process takes d time units with an error of $\pm\epsilon$ uniformly distributed (i.e. according to a uniform distribution $U(d-\epsilon, d+\epsilon)$) to send a message.

The Receiver alternates between process $R(1)$ and $R(0)$: when $R(i)$ receives a message $c(j)$, with $j \in \{i, 1-i, \bot\}$, if $j = i$, $R(i)$ just sends and *ack* with alternating bit i, if $j = 1-i$ sends an *ack* back, outputs the message, and continues with its normal execution switching to process $R(1-i)$. In the other case, it discards the message and waits for a new one.

$$
\begin{array}{l}
S_i = in.a(i).(\{\!|y|\!\}, S_i') \\
S_i' = \left(\{y\} \mapsto a(i).(\{\!|y|\!\}, S_i')\right) \quad + \quad \left(b(i).S_{1-i} + b(\bot).S_i' + b(1-i).S_i'\right) \\
\\
K = \sum_{i=0}^{1} D_i + L_i \\
D_i = a(i).\left(\{\!|x|\!\}, \left(\{x\} \mapsto \underline{\tau}.((0.05, \emptyset, K) \oplus (0.05, \emptyset, c(\bot).K) \oplus (0.9, \emptyset, c(i).K))\right)\right) \\
L_i = d(i).\left(\{\!|x|\!\}, \left(\{x\} \mapsto \underline{\tau}.((0.05, \emptyset, K) \oplus (0.05, \emptyset, b(\bot).K) \oplus (0.9, \emptyset, b(i).K))\right)\right) \\
\\
R_i = c(i).R_i' + c(\bot).R_i' + c(1-i).R_i'' \\
R_i' = d(i).R_i \\
R_i'' = d(1-i).out.R_{1-i}
\end{array}
$$

Table 3. P-🔔 specification for the alternating bit protocol

The process modeling the communication protocol is

$$\theta_f(S_0 \,||_{SK}\, K \,||_{RK}\, R_1)$$

where $SK = \{a(0), a(1), b(0), b(1), b(\bot)\}$, $RK = \{c(0), c(1), c(\bot), d(0), d(1)\}$ and $f : SK \cup RK \to \underline{SK} \cup \underline{RK}$ with $f(x) = \underline{x}$ for all $x \in SK \cup RK$. It is important to remark that operator θ_f makes that the environment can only communicate with this process just through actions *in*, *out*. This is the reason the entire process behave like a black box as explained above.

8 Conclusion

We introduced P-🔔, a process algebra that extends SPAs with a framework to model priorities and urgency and give semantics in terms of PSA. Since PSA has semantics in terms of PTTS, every P-🔔 term has concrete meaning in a PTTS. Moreover, we defined several operations directly on PTTS, namely, parallel composition, summation, the prioritizing operation, and the renaming operation. We showed that this operations commute modulo bisimulation with their relatives in P-🔔. We finally gave an example illustrating how P-🔔 can be used for system modeling.

Much work is still to be done. For instance, the finding of an axiomatization for P-🔔 or a tool support that allows to analyze systems modeled in this SPA. In [6] several SPA are shown to have semantics in terms of PSA. As a corollary of this, it seems that P-🔔 is more expressive than those algebras. A deeper study on the relation of P-🔔 and those algebras, seems to be also a reasonable research direction.

References

1. J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae*, IX(2):127–168, 1986.
2. J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990.
3. M. Bernardo. *Theory and Application of Extended Markovian Process Algebras*. PhD thesis, Dottorato di Ricerca in Informatica. Università di Bologna, Padova, Venezia, February 1999.
4. M. Bravetti. *Specification and Analysis of Stochastic Real-Time Systems*. PhD thesis, Dottorato di Ricerca in Informatica. Università di Bologna, Padova, Venezia, February 2002.
5. M. Bravetti and M. Bernardo. Compositional asymmetric cooperations for process algebra with probabilities, priorities, and time. Technical Report UBLCS-2000-01, Laboratory for Computer Science, University of Bologna, February 2000.
6. Mario Bravetti and Pedro R. D'Argenio. Tutti le algebre insieme: Concepts, discussions and relations of stochastic process algebras with general distributions. In *Verification of Stochastic Timed Systems, GI/Dagstuhl Research Seminar, December 8-11, 2002, Schloss Dagstuhl*, Lecture Notes in Computer Science. Springer-Verlag, 2003. To appear.
7. J. Camilleri and G. Winskel. CCS with priority choice. In *Proceedings* 6^{th} *Annual Symposium on Logic in Computer Science,* Amsterdam, pages 246–255. IEEE Computer Society Press, 1991.
8. R. Cleaveland and M. Hennessy. Priorities in process algebra. In *Proceedings* 3^{th} *Annual Symposium on Logic in Computer Science,* Edinburgh, pages 193–202. IEEE Computer Society Press, 1988.
9. P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, Department of Computer Science, University of Twente, 1999.
10. P.R. D'Argenio, J.-P. Katoen, and E. Brinksma. An algebraic approach to the specification of stochastic systems (extended abstract). In D. Gries and W.-P. de Roever, editors, *Proceedings of the IFIP Working Conference on Programming Concepts and Methods, PROCOMET'98,* Shelter Island, New York, USA, IFIP Series, pages 126–147. Chapman & Hall, 1998.
11. M. Domínguez, G. Saiz, and P.R. D'Argenio. P-🔔: A process algebra for modelling prioritized stochastic timed systems (full report). Technical report, Fa.M.A.F., UNC, 2003.
12. W.H. J. Feijen and A. J. M. van Gasteren. *On a method of Multi-Programming*. Monographs in Computer Science. Springer-Verlag, 2000.
13. W. Fokkink. *Introduction to Process Algebra*. EATCS. Springer-Verlag, 2000.
14. P.G. Harrison and B. Strulo. SPADES: Stochastic process algebra for discrete event simulation. *Journal of Logic and Computation*, 10(1):3–42, 2000.
15. H. Hermanns. *Interactive Markov Chains*, volume 2428 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

16. U. Herzog. Formal description, time and performance analysis. A framework. In T. Härder, H. Wedekind, and G. Zimmermann, editors, *Entwurf und Betrieb verteilter Systeme*, pages 172–190. Springer-Verlag, 1990.
17. U. Herzog. Formal methods for performance evaluation. In E. Brinksma, H. Hermanns, and J.-P. Katoen, editors, *Lectures on Formal Methods and Performance Analysis. First Euro/EEF-Summerschool on Trends in Computer Science.* Berg en Dal, The Netherlands, July 2000, volume 2090 of *Lecture Notes in Computer Science*, pages 1–37. Springer-Verlag, 2001.
18. J. Hillston. *A Compositional Approach to Performance Modelling.* Distinguished Dissertation in Computer Science. Cambridge University Press, 1996.
19. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.
20. J.-P. Katoen. *Quantitative and Qualitative Extensions of Event Structures.* PhD thesis, Department of Computer Science, University of Twente, April 1996.
21. K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.
22. G. Lowe. Prioritized and probabilistic models of Timed CSP. Technical Report PRG-TR-24-91, Oxford University Computing Laboratory, Programming Research Group, 1994.
23. R. Milner. *Communication and Concurrency.* Prentice-Hall International, 1989.
24. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1995.
25. S.A. Smolka and B. Steffen. Priority as extremal probability. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*, pages 456–466. Springer-Verlag, 1990.
26. B. Strulo. *Process Algebra for Discrete Event Simulation.* PhD thesis, Department of Computing, Imperial College, University of London, 1993.
27. C. Tofts. Processes with probabilities, priority and time. *Formal Aspects of Computing*, 6:536–564, 1994.