Testing Semantics for a Probabilistic-Timed Process Algebra *

Carlos Gregorio-Rodríguez, Luis Llana-Díaz,
Manuel Núñez and Pedro Palao-Gostanza

Dept. de Informática y Automática
Universidad Complutense de Madrid. E-28040 Madrid. Spain.
e-mail: {cgregor,llana,manuelnu,ecceso}@dia.ucm.es

Abstract. In this paper we present a probabilistic-timed process algebra, which try to unify the best solutions of previous probabilistic and timed algebras. We provide an operational semantics for the new language (PTPA), and from this operational semantics we define a testing semantics based on the probability with which processes pass tests. Afterwards the induced testing equivalence is operationally characterized by probabilistic timed traces.

1 Introduction

Everyday the systems consisting of a number of components which need to communicate and cooperate each other (control systems, networks, communications, etc) are more numerous and complex. These systems are called Concurrent and Distributed Systems, (CDS), and many of them are also Real-Time systems. In the eighties, process algebras have settled an important theoretical background for the study of CDS. These works were very significant, mainly to shed light on concepts and to open researching methodologies, but, due to the abstraction of the complicated features, models defined by these algebras were still far from real systems and, therefore, some of provided solutions were not specific enough, for instance, those related to real time systems. In the current decade, researches in process algebras have tried to diminish the distance between formal models described by process algebras and real systems. In particular, features which were abstracted before have been introduced in the models. The most significant of these features are time and probabilities.

The interest of embedding probabilistic notions into process algebras is obvious in order to perform a more qualitative analysis of concurrent distributed systems which manage with statistical and probabilistic phenomena, such as random algorithms or failure rates in a communication channel. Several models introduce probabilities into process algebras, in [vGSS95] models are classified with respect to the interpretation of probabilities, by using (strong) bisimulation. Inside the testing framework, there are also several proposals for probabilistic

* Research supported in part by the CICYT project TIC 94-0851-C02-02

process algebras (e.g. [Chr90, CSZ92, YL92, NdFL95, NdF95]). In the present paper the underlying semantic model, from a probabilistic point of view, is along the lines of those in [CSZ92], because, in spite of having some weakness, is a general model: tests have no restrictions and the equivalence defined is not so coarse as the probabilistic extensions of the classical *may* or *must* equivalences (e.g. [JHSY94, JY95]).

We think that the inclusion of probabilities into process algebras is not an easy task. To capture observational equivalence and, consequently to abstract from the τ action in the probabilistic framework is difficult, so in general probabilistic testing models, e.g. [CSZ92, YCDS94, NdF95], $\tau\,;a \not\approx a$, dislike it happened to the nonprobabilistic setting. The same problem arises in models considering bisimulation, in fact, *probabilistic weak bisimulation* remains as an open problem.

Time is another important aspect in concurrent systems that cannot be adequately represented in classical process algebras. For some systems, especially real-time systems, is not only important to know that an action can be executed, but also when it is executed. There have been many proposals to introduce time in a process algebra [RR86, NJ91, Yi91, BB93, Sch95], but most of them are not based on testing. Examples of timed testing semantics appear in [HR95, LdFN96].

There are still not too many proposals including time and probabilities. In [Han91, Tof94] models based on CCS with strong bisimulation are given. In [Low95] a denotational semantics, based on TCSP, is presented. Finally, in [CLLS96], ideas about an integration, using testing methodology, are given, but no characterization is provided for the induced testing equivalence. As far as we know, [CLLS96] is the only proposal trying to integrate probabilities and time into a testing framework, but the syntax is to limited (they do not have recursive processes, and time is modeled by means of an special action) and the semantic framework is under development. We will use testing methodology because we consider it very suitable to, beginning with an intuitive and simple interpretation of the observational equivalence, be able to accomplish semantics for processes.

Discussions above draw the lines which lead our work, that is, to define a process algebra consisting of: a general enough syntax including features of time and probabilities to properly specify real time CDS; an equivalence between processes, based on testing semantics, easy to understand and close to the idea of observational equivalence; finally, characterizations for the equivalence to both gain knowledge about the equivalence classes, and make the equivalence tractable from a computable point of view.

Next we comment many of the considerations taken into account to introduce time and probabilities in our Probabilistic Timed Process Algebra, PTPA .

In this paper we will consider a discrete time domain. This is not a restrictive decision since all computers are ruled by an internal discrete clock, we will consider δ as the duration of a cycle of that clock. Moreover, all instructions of any computer has a non-zero duration, and it is a multiple of the cycle of the clock. Thus we consider as our time domain $\mathcal{T} = \{0, \delta, 2 \cdot \delta ....\}$; and given that

an action specifies an instruction (or set of instructions) the execution of any action takes a non zero multiple of δ period of time (as in [OMdF90]), that will be formally described by the function $d()$ in section 2.1.

Time considerations take special relevance in operators which have to do with the sequentiality of the events. We present a delay prefixing operator $\text{wait}(t); P$ expressing that P is delayed time t, and an action prefix operator $a\{t\}\ ; P$ expressing that the action a is available at time t.

The delay prefixing operator is a derivated one since $\text{wait}(t_1); a\{t_2\}\ ; P$ has the same behavior as $a\{t_1 + t_2\}\ ; P$. So, this operator could be easily removed, but we will keep it in our language because it makes definitions easier.

Probabilistic decisions have to be expressed in operators which relate several processes. Our choice operator is inherited from CCS and is extended with a probability which assigns *weights* to both components of the composition. There exists a general agreement about probabilistic choice, although there are probabilistic models with both probabilistic and nonprobabilistic choices, e.g. [YL92], but that is not the case for the parallel operator. Some proposals consist in adding one probability parameter (e.g. [NdF95, CLLS96]), which is used to give more weight to one of the components when executing interleaving actions; other proposals add two probability parameters (e.g. [BBS95]): one of them plays the same role that the previous one, while the other one gives weight to interleaving actions with respect to synchronization actions. Taking into account that we are mainly interested in unifying time and probabilistic features into a unique process algebra, and considering that the parameters appearing in the parallel operators only influence the probability with which a probabilistic transition is executed, we have preferred to consider a parallel operator, inherited from CSP, without any probabilistic parameter.

More difficult than defining probabilistic operators is assigning semantics. In [vGSS95] the *reactive*, *generative*, and *stratified* models are described and studied in the framework of probabilistic (strong) bisimulation. In the generative model there exists a unique probability distribution relating all actions, in contrast with the reactive model where there exists a probability distribution for each action. The stratified model is similar to the generative, but capturing the branching structure of the pure probabilistic choices made by a process. The first two models have been considered in many papers, while there have not been too much work over the stratified model (maybe because of its inherent complexity). In this paper we use a *generative* interpretation of probabilities because is more *expressive* as well as *simpler* than the reactive interpretation. Although these two concepts may seem contradictories, in this case we will show that they are not.

Regarding simplicity, from the operational point of view the (probabilistic) choice operators immediately induce a unique distribution of probability, just multiplying the probabilities associated with the actions of each component by the corresponding probability parameter of the choice. This is not so direct in the reactive model, because we must manage several probability distributions (see [vGSS95]).

Regarding expressiveness, any process specified using the reactive model can

be translated to the generative model. For example, let us suppose that we want to specify a system such that if a_i is pressed then there exists a (local for each a_i) probability distribution leading to several processes. This purely reactive process can be easily specified in the generative model by the process (using general summations as in [vGSS95])

$$[\frac{1}{n}]\sum_{i\in I_1}[p_i]a_1 \,;\, P_i + [\frac{1}{n}]\sum_{i\in I_2}[p_i]a_2 \,;\, P_i + \cdots + [\frac{1}{n}]\sum_{i\in I_n}[p_i]a_n \,;\, P_i$$

where we only need to add the (*artificial*) factor $[\frac{1}{n}]$. On the contrary, a process like $a +_p b$ which relates the probabilities associated with a and b cannot be specified using a reactive model (i.e. in the reactive model, $a +_p b$ and $a +_q b$ are equivalent). Moreover, from the testing point of view the reactive model can be seen as a particular case of the generative model. Following the intuitive and well known black box analogy [Mil89] where processes are described as black boxes with buttons, we can consider that in the reactive model the environment cannot offer more than one action simultaneously, i.e. only one button can be pressed, which leads tests to be just traces. In the generative model, several buttons can be pressed simultaneously, and with different strengths, leading tests to be general processes (see [NdF95] for a more extended explanation). There exists another reason for choosing the generative model in a testing framework, and it has to do with internal actions. For example, in the generative model the meaning of a process like $P = a +_p \tau \,;\, P'$ is that if the environment offers a then P executes it with a probability p, while with a probability $1-p$, P evolves by an internal transition. But it is not so clear the intuitive meaning of this process in the reactive model because, as the model says, there does not exist a probabilistic relation between different actions. So, what is the probability with which this process executes a? If we would answer "p" then we are recognizing a relation between different actions. If we answer "1 but maybe a is not executed", which is possibly the most accurate answer, then the answer is even more obscure.

Concluding, our model presents a compromise between generality and simplicity, that is, we try to get a model as general as possible to properly specify real systems, but, once fixed a level of generality, we make decisions as simple as possible to obtain a tractable mathematical model. For example, we consider that actions are available at a single instant of time, $a\{t\}$, because the model described is as powerful as if we would have defined that actions are available during a finite interval of time. The case also applies to the definition of $d()$ (which could not only depend on the action but also on the process or on the environment), or to the parallel operator, or even to the set itself of operators which could be enlarged with derived ones to improve the expressiveness of the language.

The rest of the paper is structured as follows. In Section 2 we present our language, namely PTPA . For this language we define an operational semantics, and the induced testing semantics where we give a notion of a process passing a test with a certain probability. Our operational semantics has not transitions labeled with a probability, but the probabilistic information is recorded using

a *transition probability function*. Section 3 is the most technical part of the paper and presents an alternative characterization of the testing semantics, which allows to decide if two processes are (testing) equivalent looking at their operational structures. This alternative characterization will be based on a special kind of probabilistic-timed traces. In Section 4 we present our conclusions and some lines for future work.

2 The PTPA Language

In this section we concrete all considerations above in the syntax of our language and in the definition of the operational semantics. Later, we will define how a process passes a test and the induced testing equivalence.

2.1 Syntax and Operational Semantics of PTPA

We consider the set of actions Act, an special action $\tau \notin \text{Act}$ representing an internal action, the discrete time domain $\mathcal{T}$ previously defined, and a set of identifiers Id; e will usually denote a generic action in $\text{Act} \cup \{\tau\}$ while a will usually denote a generic action in Act.

Definition 1. The set of PTPA *processes* is defined as the set of expressions given by the following BNF-expression:

$$P ::= \text{stop} \mid X \mid \text{wait}(t); P \mid e\{t\}\,; P \mid P +_p P \mid P \parallel_A P \mid P\backslash A \mid \text{rec } X.\ P$$

where $p \in (0,1)$, $e \in \text{Act} \cup \{\tau\}$, $t \in \mathcal{T}$, $A \subseteq \text{Act}$, and $X \in Id$.

From now on we will concentrate only in *closed* expressions (i.e. without free variables) and with *guarded recursion* (i.e. any occurrence of a variable is prefixed by any, visible or not, action). For the sake of clarity we will omit trailing occurrences of the process stop. As we said in the introduction, we consider that the execution of an action takes time. So, we have a function $d:\ \text{Act} \cup \{\tau\} \longrightarrow \mathcal{T}$, such that $d(e) = t$ means that the action e takes time t for its execution; without loosing any generality, we suppose that the duration of the internal action τ is one clock cycle, i.e., $d(\tau) = \delta$.

In order to assign meaning to the syntactic terms we will define an adequate operational semantics. We consider two kinds of transitions. The meaning of an *action* transition $P \xrightarrow{e} P'$ is that the process P executes e, becoming P'. In the following, we use $P \not\longrightarrow$ as a shorthand for $\nexists e, P':\ P \xrightarrow{e} P'$ and $P \xrightarrow{e}\!\!\!\!\!\!/\ \ $ as a shorthand for $\nexists P':\ P \xrightarrow{e} P'$. The meaning of a *time* transition $P \overset{t}{\rightsquigarrow} P'$ is that P behaves as P' after an amount of time equal to t, which is always greater than zero. Moreover, $P \overset{t}{\rightsquigarrow}\!\!\!\!\!\!/\ \ $ stands for $\nexists P':\ P \overset{t}{\rightsquigarrow} P'$, and $\rightsquigarrow^*$ denotes the reflexive and transitive closure of $\rightsquigarrow$, that is $P \overset{t}{\rightsquigarrow}^* P'$ iff there exist $P_1, \ldots, P_n$ and $t_1, \ldots, t_n, t_{n+1}$ such that $P \overset{t_1}{\rightsquigarrow} P_1 \cdots \overset{t_n}{\rightsquigarrow} P_n \overset{t_{n+1}}{\rightsquigarrow} P'$, and $t = \sum t_i$. If $t = 0$ then we have $P \overset{0}{\rightsquigarrow}^* P$.

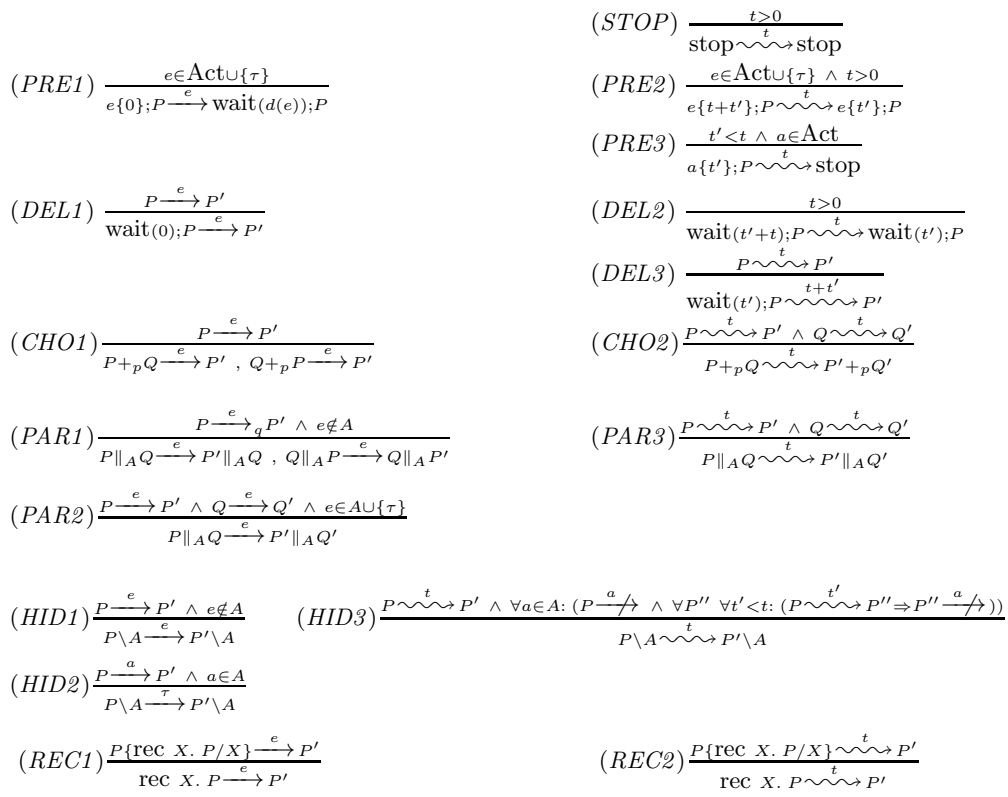

Fig. 1. Operational Semantics of PTPA .

The operational semantics is given in Figure 1. The rules appearing in the left column deal with action transitions, and they are similar to those of a classical (without time or probability) process algebra. Differences appear in rules (*PRE1*), (*DEL1*), and (*PAR2*). (*PRE1*) indicates that an action e can be executed only if its time parameter is 0, and that this execution takes time $d(e)$, and so, the continuation of the process must be delayed this amount of time. (*DEL1*) just indicates that a delayed process can execute actions only if the delay has been consumed. Finally, in (*PAR2*) we allow the synchronization not only in the actions belonging to the synchronization set A, but also in τ, as in [CSZ92, YCDS94]. Note that τ's can also be executed without synchronization (rule (*PAR1*)).

Example 1. Let $P = (a\{0\}\,;P_1 +_p \tau\{0\}\,;P_2) \parallel_{\{a\}} (a\{0\}\,;P_3 +_q \tau\{0\}\,;P_4)$. Then, P has the following transitions:

- $P \xrightarrow{a} (\text{wait}(d(a)); P_1) \parallel_{\{a\}} (\text{wait}(d(a)); P_3)$
- $P \xrightarrow{\tau} (\text{wait}(\delta); P_2) \parallel_{\{a\}} (a\{0\}\,;P_3 +_q \tau\{0\}\,;P_4)$
- $P \xrightarrow{\tau} (a\{0\}\,;P_1 +_q \tau\{0\}\,;P_1) \parallel_{\{a\}} (\text{wait}(\delta); P_4)$
- $P \xrightarrow{\tau} (\text{wait}(\delta); P_2) \parallel_{\{a\}} (\text{wait}(\delta); P_4)$

Now, we will briefly explain the rules appearing in the right column (i.e. rules dealing with time transitions). (*STOP*) indicates that the process stop can idle forever. (*PRE2*) indicates that a prefix operator can consume any amount of time less than or equal to its associated parameter, while (*PRE3*) indicates that if there is a passage of time greater than the parameter associated with an (visible) action then the process becomes stop. Let us note that this rule cannot be applied if the action is τ, and so we get that internal actions are urgent, according with the usual meaning in timed process algebras. (*CHO2*) and (*PAR3*) indicate that, in order to get a time transition from the composition, both components must evolve simultaneously by a time transition. (*HID3*) is usually included in timed process algebras for assuring the urgency of hidden actions. Intuitively, this rule means that a process $P\backslash A$ cannot evolve by a time transition labeled by t if there exist $t' < t$ and $a \in A$ such that a can be executed at time t'.

Since the rule (*HID3*) has a negative premise we have to provide a way to guarantee that the generated transition system is consistent. This is achieved by defining a *stratification*, as detailed in [Gro93]. We consider the following function $f(P \overset{t}{\rightsquigarrow} Q) = t$, that is indeed a stratification. Now, we present some interesting results showing that the operational semantics maintains some *good* time properties. The proof of the following proposition is easy from the operational semantics.

Proposition 2. For any PTPA process P, $t, t' \in \mathcal{T}$, the following hold:
Time determinism: $(P \overset{t}{\rightsquigarrow} P' \wedge P \overset{t}{\rightsquigarrow} P'') \Rightarrow P'$ is syntactically identical to P''.
Time additivity: $(P \overset{t}{\rightsquigarrow} P' \wedge P' \overset{t'}{\rightsquigarrow} P'') \Rightarrow P \overset{t+t'}{\rightsquigarrow} P''$.
Time closure: $0 < t < t' \;\Rightarrow\; (P \overset{t'}{\rightsquigarrow} P' \Rightarrow \exists P'' :\; P \overset{t}{\rightsquigarrow} P'' \overset{t'-t}{\rightsquigarrow} P')$

Finally, given that our actions take time to be executed, and that we are working within a discrete time domain, there do not exist *Zeno processes* in our language, that is, processes which can execute an infinite number of actions within a finite time.

Let us remark that we have not included any probabilistic information in the operational transitions and, obviously, we need to get this information in some way, for example, in order to differentiate the processes $a\{0\}+_p b\{0\}$ and $a\{0\}+_q b\{0\}$, $(p \neq q)$. To record this information we will use a *transition probability function* μ, as it is done in [SS96, CLLS96][2]. This approach has (at least) two

[2] The way of defining a transition probability function is not exactly the same in these papers. Our definition follows [CLLS96], but considering recursive processes like in [SS96].

advantages. First, probabilistic information is separated from purely operational behavior of processes. Second, and more important, it elegantly solves the well known problem[3] of deriving the same (probabilistic) transition several times.

Example 2. Let $P = a\{0\} +_{\frac{1}{2}} a\{0\}$. If no care is taken, and considering set of transitions, we would only derive $P \xrightarrow{a}_{\frac{1}{2}}$ stop, instead of the desired result $P \xrightarrow{a}_{1}$ stop.

Definition 3. The *transition probability function* μ : PTPA $\times$ Act $\cup \{\tau\} \times$ PTPA $\longrightarrow [0,1]$ is defined at the least fixed point of the recursive equation $\mu = \mathcal{P}(\mu)$, where $\mathcal{P}$ is defined as:

$$\mathcal{P}(\mu)(\text{stop}, e, R) = 0$$

$$\mathcal{P}(\mu)(\text{wait}(t); P, e, R) = \begin{cases} \mu(P, e, R) & \text{if } t = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{P}(\mu)(a\{t\}\,; P, e, R) = \begin{cases} 1 \text{ if } t = 0 \ \wedge \ a = e \ \wedge \ R \equiv P \\ 0 \text{ otherwise} \end{cases}$$

$$\mathcal{P}(\mu)(P +_p Q, e, R) = \frac{p}{\nu_+(P,Q,p)} \cdot \mu(P, e, R) + \frac{1-p}{\nu_+(P,Q,p)} \cdot \mu(Q, e, R)$$

$$\mathcal{P}(\mu)(P \parallel_A Q, e, R) = \begin{cases} \frac{\mu(P,e,P') \cdot \mu(Q,e,Q')}{\nu_\parallel(P,Q,A)} & \text{if } \ e \in A \cup \{\tau\} \ \wedge \ R \equiv P' \parallel_A Q' \\ \frac{\mu(P,e,P') + \mu(Q,e,Q')}{\nu_\parallel(P,Q,A)} & \text{if } \ e \notin A \cup \{\tau\} \ \wedge \ R \equiv P' \parallel_A Q' \\ & \quad \wedge \ (P \equiv P' \text{ xor } Q \equiv Q') \\ \frac{\mu(P,\tau,P') \cdot (1-\mu(Q,\tau)) + \mu(Q,\tau,Q') \cdot (1-\mu(P,\tau))}{\nu_\parallel(P,Q,A)} & \text{if } \ e = \tau \ \wedge \ R \equiv P' \parallel_A Q' \\ & \quad \wedge \ (P \equiv P' \text{ xor } Q \equiv Q') \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{P}(\mu)(P\backslash A, e, R) = \begin{cases} \mu(P, e, R') & \text{if } e \notin A \cup \{\tau\} \ \wedge \ R \equiv R'\backslash A \\ \sum_{e' \in A \cup \{\tau\}} \mu(P, e', R') & \text{if } e = \tau \ \wedge \ R \equiv R'\backslash A \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{P}(\mu)(\text{rec } X.\ P, e, R) = \mu(P\{\text{rec } X.\ P/X\}, e, R)$$

where $\equiv$ denotes syntactic identity of processes, φ xor ψ means that either φ holds or ψ holds but not both of them, and $\mu(R, e) = \sum_{R'} \mu(R, e, R')$.

Intuitively, $\mu(P, e, Q)$ computes the probability with which the process P can execute the action e becoming Q. In the definition of μ appear two auxiliary functions: ν_+ and $\nu_\parallel$. The first function computes the probability of executing actions at time 0 by both components of a choice (multiplying these probabilities

[3] Other solutions to this problem are to index transitions (e.g. [GJS90]), to increment the number of operational rules (e.g. [LS91]), or to consider multisets of transitions (e.g. [NdFL95, NdF95]).

by p and $1-p$ respectively). This function is used to avoid processes being *substochastic*. In our model, the sum of the probabilities associated with the actions that the process can execute is either 0, if the process can execute no action, or 1. Formally,

$$\nu_+(P,Q,p) = p \cdot \sum_{R',e} \mu(P,e,R') + (1-p) \cdot \sum_{R',e} \mu(Q,e,R')$$

The following example illustrates the kind of problems we avoid with this function.

Example 3. Let us suppose that our definition of μ for the choice operator were

$$\mathcal{P}(\mu')(P +_p Q, e, R) = p \cdot \mu'(P,e,R) + (1-p) \cdot \mu'(Q,e,R)$$

Let $P_1 = e\{0\} +_{\frac{1}{2}} \text{stop}$. Then, $\mu'(P_1, e, \text{stop}) = \frac{1}{2}$, while we would like to obtain 1. In particular, the unit law $P +_p \text{stop} \approx P$ would not hold. Moreover, let $P_2 = e_1\{0\} +_{\frac{1}{2}} e_2\{1\}$. Again, $\mu'(P_2, e_1, \text{stop}) = \frac{1}{2}$, while this probability should be equal to 1, because the only action that P_2 can execute at time 0 is b_1.

The function $\nu_\parallel$ is the *normalization* function used in most probabilistic models having a notion of parallel composition (i.e. a parallel operator, or the parallel composition of a process and a test; e.g. [Chr90, CSZ92, YCDS94, NdFL95, NdF95]). This function computes the sum of the probabilities associated with the actions that the composition can execute, that is, in our case it computes the sum of the probabilities appearing in the first three branches of the definition of μ for the parallel composition. Given that we allow the synchronization in τ, our normalization factor will be similar to that in [CSZ92, YCDS94], but considering that the synchronization set is not forced to be the whole set of actions Act. Formally,

$$\begin{aligned}\nu_\parallel(P,Q,A) = & \sum_{R,(e \notin A)} (\mu(P,e,R) + \mu(Q,e,R)) \quad + \sum_{P',Q',(e \in A)} (\mu(P,e,P') \cdot \mu(Q,e,Q')) \\ & - \sum_{P'} \mu(P,\tau,P') \cdot \sum_{Q'} \mu(Q,\tau,Q')\end{aligned}$$

While the intuition behind the definition of μ is clear for most of the cases, we will explain the definition for the parallel operator. The first branch computes the probability of executing actions belonging to the synchronization set or being τ, and it considers the product of the corresponding probabilities associated with P and Q. The second branch captures the probability of P or Q executing an action not belonging to $A \cup \{\tau\}$. Finally, the third branch deals with the possibility of executing τ without synchronization (remember that we allow to synchronize in τ as well as to execute τ in an interleaving manner). In this case, the probability will be the product of the one associated with the executed τ and the probability with which the other process does not execute τ. Note that if a process only can execute τ, then the other one is not allowed to executed its τ's in an interleaving manner. Let us remark that in the last two branches only one of the processes *evolves* while the other one does not change (because of the clause $P \equiv P'$ xor $Q \equiv Q'$).

Once μ has been completely defined we must show that it has been *well defined*, that is, that the least fixed point actually exists. We follow a similar reasoning to [SS96]. First, $[0,1]$ is a cpo under the usual ordering, and this ordering induces a pointwise ordering on the set of all functions μ taking triples (P,e,R) to $[0,1]$, so that this set also is a cpo. Now, it is easy to prove that $\mathcal{P}$ is a continuous mapping from this cpo to itself, so that the least fixed point exists, and can be characterized using finite approximations. That is, let μ_0 be the identically zero function, and for any $i \geq 0$ let $\mu_{i+1} = \mathcal{P}(\mu_i)$, and then, $\mu = \sup_{0 \leq i} \mu_i$. Moreover, the function μ is indeed a probability distribution function, as the following result states.

Lemma 4. Let P be a PTPA process. Then, $\sum_{R,e} \mu(P,e,R) \in \{0,1\}$.
Proof: Similar to that in [SS96], considering that we do not have *sub-stochastic* processes, and so, our sum is either 0 or 1, while their sum belongs to the interval $[0,1]$. □

We finish this section with a result, whose proof is easy by structural induction, showing that all the operational rules not dealing with time and with a probability greater than zero of execution are inferable from the definition of μ.

Lemma 5. Let P be a PTPA process. If $\mu(P,a,P') > 0$ then $P \xrightarrow{a} P'$.

2.2 Testing Semantics

Observational equivalence is intuitively described as follows: two processes are equivalent if an external observer cannot distinguish them by registering their responses to the same stimuli. In the testing framework the agents which play a role in the observational equivalence are formally described: the possible behaviors of the observer, that is, the different stimuli the observer offers, are modeled by the set of tests; the experiment itself is modeled by the interaction system between the process and the observer (tests); this interaction system gives a response which we have to judge to be successful or not; finally, two processes are defined to be testing equivalent if for every test the interpretation of the responses of both interaction systems (processes and test) are the same.

In this section we present our model of testing, namely, the set of tests, the interaction systems, a measure of success with respect to a given test and, finally, the testing equivalence.

A test is just a process where the alphabet has been extended with a new action $ok \notin \mathrm{Act}$, which indicates *successful* termination of the test. As in [LdFN96], we consider that the successful termination event ok is urgent (like the action τ). So, the operational semantics given in Figure 1 must be extended with the following rules:

$$(SUC1)\ \frac{}{ok\{0\}\,;T \xrightarrow{ok} \mathrm{wait}(d(ok))T} \qquad (SUC2)\ \frac{t > 0}{ok\{t+t'\}\,;T \overset{t}{\rightsquigarrow} ok\{t'\}\,;T}$$

Now, we will define the way a process and a test interact. This definition will be based on the classical testing semantics by Hennessy [Hen88], where we have

to take into account time and probabilistic informations. Testing is performed by studying the computations of the process that is obtained by composing in parallel a test with the tested process, taking as synchronization set the full set of actions (i.e. *ok* is not included). Test application will be represented by $P \mid T$, and is formally defined as follows:

$$P \mid T = (P \parallel_{\mathrm{Act}} T) \backslash \mathrm{Act}$$

Now, we must define when a test has been *successfully passed* as well as the probability with which a process *passes* a test.

Definition 6. Let P be a process and T be a test. We call *computations* to the (possibly infinite) sequences of transitions from $S_0 = P \mid T$ of the form

$$C = S_0 \overset{t_1}{\rightsquigarrow}{}^{*} S'_1 \xrightarrow{e_1} S_1 \overset{t_2}{\rightsquigarrow}{}^{*} S'_2 \xrightarrow{e_2} S_2 \cdots \overset{t_n}{\rightsquigarrow}{}^{*} S'_n \xrightarrow{e_n} S_n \cdots$$

We say that a computation is *successful*, if there exists n such that $S'_n \xrightarrow{ok}$, and such that for all $1 \leq i < n$, $S'_i \not\xrightarrow{ok}$. In this case, we say that $\mathit{length}(C) = n$. Note that if C is a successful computation, then all the e_i's are τ's, but the last one.

Given a successful computation C, such that $\mathit{length}(C) = n$, we define the probability of executing C as $Pr(C) = \prod_{i=1}^{n-1} \mu(S'_i, \tau, S_i)$ (we consider $\prod_{i=1}^{0} \mu(S'_i, \tau, S_i) = 1$).

We will denote by $\tilde{C}_{P \mid T}$ the *set* of *successful computations* starting from $P \mid T$. Finally, we say that P *passes* T with a *probability* equal to p, denoted by $P\ \mathit{pass}_p\ T$, if

$$\sum_{C \in \tilde{C}_{P \mid T}} Pr(C) = p$$

Let us remark that $P \mathit{pass}_p T$ iff $p = \lim_{n \to \infty} \sum_{C \in \tilde{C}_{P \mid T}} \{Pr(C) \mid \mathit{length}(C) < n\}$. The previous definition induces the corresponding notion of testing equivalence.

Definition 7. (**Testing Equivalence**)
Let P, Q be *PTPA* processes. We write $P \approx_{\mathcal{T}} Q$ iff $\forall\ T :\ P\ \mathit{pass}_p\ T \iff Q\ \mathit{pass}_p\ T$.

There have been quite a number of equivalence definitions in process algebra literatura, but they all try to capture *observational equivalence*, quoting from Milner [Mil89], "the behavior of a system is exactly what is observable, and to observe a system is exactly to communicate with it". Provided that the intended meaning of the τ action is that of an internal action, this action cannot be seen by the environment, so the processes $\tau\,;a\,;$stop and $a\,;$stop have to be equivalent according to the definition of observational equivalence. That is the case, for example, in CCS w.r.t. weak bisimulation. As we mentioned in Section 1, some testing probabilistic models do not verify $\tau\ ;\ a\ ;$ stop $\approx a\ ;$ stop. We think that this problem, and others which have to do with equivalences, can be solved if we mix time and probabilities into a process algebra. The reason is that if

we enlarge the syntax of the language we can specify with more precision the behavior of processes and, consequently, the equivalence classes can be more exactly delimited. In our model $a\{0\} \not\approx_{\mathcal{T}} \tau\{0\};a\{0\}$ because the test $a\{0\};ok\{0\}$ is passed with probability 1 by the first process but it is not passed for the second one. Now we have an intuitive reason for this result: τ takes time δ to execute. Nevertheless, now we also have equivalences like $\tau\{0\}\,;a\{0\} \approx_{\mathcal{T}} a\{\delta\}$.

We conclude this section by showing that the set of tests can be reduced, but conserving the testing equivalence, resulting that infinite tests are not necessary.

Lemma 8. Let P, Q be processes. If there exists a recursive test T such that P *pass*$_p$ T, Q *pass*$_q$ T, and $p \neq q$, then there exists a finite test T' (i.e. without occurrences of the recursion operator) such that P *pass*$_{p'}$ T', Q *pass*$_{q'}$ T', and $p' \neq q'$.

Proof: Let T be a recursive test such that P *pass*$_p$ T, Q *pass*$_q$ T, and $p \neq q$. So,

$$p = \lim_{n\to\infty} \sum_{C\in\tilde{C}_{P\mid T}} \{Pr(C) \mid \mathit{length}(C) < n\} \neq \lim_{n\to\infty} \sum_{C\in\tilde{C}_{Q\mid T}} \{Pr(C) \mid \mathit{length}(C) < n\} = q$$

If these two limits are different, then there exist n_0 such that for all $n \geq n_0$ we have

$$\sum_{C\in\tilde{C}_{P\mid T}} \{Pr(C) \mid \mathit{length}(C) < n\} \neq \sum_{C\in\tilde{C}_{Q\mid T}} \{Pr(C) \mid \mathit{length}(C) < n\}$$

Now, we consider the finite test T' resulting from T by unwinding n_0 times every occurrence of recursion in T, and then replacing any occurrence of recursion by stop. Given that T' simulates the first n_1 transitions of T (for some $n_1 \geq n_0$), and so $P|T'$ (resp. $Q|T'$) simulates, at least, the first n_1 transitions of $P|T$ (resp. $Q\,|\,T$), we get that the probability with which P and Q pass T' are different. □

3 Alternative Characterization

In this section we present an operational characterization of the relation $\approx_{\mathcal{T}}$. This alternative characterization will be based on the sequences of actions that processes can execute, together with some additional information. As in other probabilistic and/or timed models we have to consider not only sequences of actions but also the *states* where these actions are executed. In our model, a *probabilistic-timed state*, which is a generalization of the *classical* notion of state [Hen88], represents a local configuration of a process and it contains the actions that a process can execute in such configuration. As we are considering a probabilistic-timed process algebra, each action has also information about the time when it is executed as well as the probability with which is executed.

Definition 9. A *state* is a set $A \subseteq \mathrm{Act} \times \mathcal{T} \times (0..1]$ such that for any $t \in \mathcal{T}$ the sum of the probabilities at such time $\sum\{\!|\, p \mid (a,t,p) \in A \,|\!\}$ is either 0 or 1.

As we said previously, traces will not be just sequences of actions because we have to *remember* the different states during the execution of actions by a process. Nevertheless, we only need to consider those states whose actions can be executed before the corresponding action in the trace (last condition in the following definition).

Definition 10. A *trace* is a sequence $(A_1, a_1, t_1)(A_2, a_2, t_2) \cdots (A_n, a_n, t_n)$ where,

- $n \geq 0$; if $n = 0$ we have the empty trace denoted by ϵ.
- For all $0 \leq i \leq n$, A_i is a state, $a_i \in \text{Act}$ and $t_i \in \mathcal{T}$.
- For all $0 \leq i \leq n$, there exists p such that $(a_i, t_i, p) \in A_i$.
- For all $0 \leq i \leq n$, and for all $(a', t', p') \in A_i$ we have $t' \leq t_i$.

In order to define the traces of a process, first we define some auxiliary concepts.

Definition 11. Let P be a process. We define the set of *immediate available probabilistic-timed actions* as $S(P) = \left\{(a, 0, p) \mid \mu(P, a) > 0 \ \wedge \ p = \frac{\mu(P,a)}{1-\mu(P,\tau)}\right\}$. Let A be a state, and $t \in \mathcal{T}$. We define $(A + t) = \{(a, t + t', p) \mid (a, t', p) \in A\}$.

$S(P)$ is the set of actions that the process can execute at time 0, but dividing the associates probabilities by $1 - \mu(P, \tau)$, so the sum is equal to 1 (or 0 if there do not exist visible actions). The state $A + t$ represents the state A but adding t units of time.

Definition 12. Let P, Q be processes. We inductively define $P \stackrel{s}{\Longrightarrow}_p P'$ as follows:

- $P \stackrel{\epsilon}{\Longrightarrow}_1 P$.
- If $p = \mu(P, a, P') > 0$ and $P' \stackrel{s}{\Longrightarrow}_q Q$ then we have $P \stackrel{(S(P),a,0)\cdot s}{\Longrightarrow}_{p'} Q$, where $p' = q(1 - \mu(P, \tau))\left(\frac{p}{\mu(P,a)}\right)$.
- If $p = \mu(P, \tau, P') > 0$ and $P' \stackrel{(A,a,t)\cdot s}{\Longrightarrow}_q Q$ then we have the transitions $P \stackrel{s}{\Longrightarrow}_{pq} Q$ and $P \stackrel{((S(P)\cup A),a,t)\cdot s}{\Longrightarrow}_{(1-p)q} P'$.
- If $P \stackrel{t_0}{\rightsquigarrow} P' \stackrel{(A,a,t)\cdot s}{\Longrightarrow}_q Q$ and the transition $P \stackrel{t_0}{\rightsquigarrow} P'$ is maximal[4] with respect to $(A, a, t) \cdot s$, then $P \stackrel{(A',a,(t+t_0))\cdot s}{\Longrightarrow}_q P'$ where $A' = (A + t_0) \cup \bigcup_{t' < t_0} \{S(R) \mid P \stackrel{t'}{\rightsquigarrow} R\}$.

$P \stackrel{s}{\Longrightarrow}_p P'$ is the extension of $\rightarrow$ and $\mu()$ to traces, and computes the probability with which P executes the trace s reaching P'. Now we should define the traces of a process, but as we are working with a probabilistic (timed) process algebra, it is enough to define the probability with which the process executes the trace.

Definition 13. Let P be a process and s be a trace. We define the *probability* with which the process P *executes* s as $\mu_{tr}(P, s) = \sum\{\!\!\{\, p \mid \exists Q : P \stackrel{s}{\Longrightarrow}_p Q \,\}\!\!\}$

To illustrate the above definitions, consider the following example.

Example 4. Let $P = (b\{0\}\,; P_b) +_{\frac{1}{7}} (a\{0\}\,; P_{a_1} +_{\frac{1}{3}} (a\{0\}\,; P_{a_2} +_{\frac{3}{4}} \tau\{0\}\,; c\{0\}\,; P_c))$. This process has the following transitions:

$$P \xrightarrow{b} \text{wait}(d(b)); P_b \quad \mu() = \tfrac{1}{7}, \qquad P \xrightarrow{\tau} \text{wait}(d(\tau)); c\{0\}\,; P_c \ \mu() = \tfrac{1}{7},$$
$$P \xrightarrow{a} \text{wait}(d(a)); P_{a_1} \ \mu() = \tfrac{2}{7}, \qquad P \xrightarrow{a} \text{wait}(d(a)); P_{a_2} \qquad \mu() = \tfrac{3}{7}.$$

[4] It cannot be extended, i.e., $t = 0$ or there exists P'' such that $P' \xrightarrow{\tau} P''$

So we have $\mu(P,a) = \frac{5}{7}$, $\mu(P,b) = \frac{1}{7}$ and $\mu(P,\tau) = \frac{1}{7}$. And then we have the transitions

$$P \xrightarrow{(A,b,0)}_{\frac{6}{7}} \mathrm{wait}(d(b)); P_b, \quad P \xrightarrow{(A,a,0)}_{\frac{6}{7}\cdot\frac{2}{5}} \mathrm{wait}(d(a)); P_{a_1}, \quad P \xrightarrow{(A,a,0)}_{\frac{6}{7}\cdot\frac{3}{5}} \mathrm{wait}(d(a)); P_{a_2},$$

where $A = \{(a, 0, \frac{5}{6}), (b, 0, \frac{1}{6})\}$. Thus we have

$$\mu_{tr}(P, (A, b, 0)) = \frac{6}{7} \quad \text{and} \quad \mu_{tr}(P, (A, a, 0)) = \frac{6}{7}.$$

Finally we have

$$\mathrm{wait}(d(\tau)); c\{0\}; P_c \overset{d(\tau)}{\rightsquigarrow} c\{0\}; P_c \xrightarrow{c}_{P\ c} \quad \text{so} \quad \mathrm{wait}(d(\tau)); c\{0\}; P_c \xrightarrow{\{(c,d(\tau),1)\}cd(\tau)}_1 P_c,$$

so we have the transitions

$$P \xrightarrow{\left(\{(c,d(\tau),1)\},c,d(\tau)\right)}_{\frac{1}{7}} P_c \quad \text{and} \quad P \xrightarrow{\left(A\cup\{(c,d(\tau),1)\},c,d(\tau)\right)}_{\frac{6}{7}} P_c,$$

so $\mu_{tr}(P, \big(\{(c, d(\tau), 1)\}, c, d(\tau)\big)) = \frac{1}{7}$ and $\mu_{tr}(P, \big(A \cup \{(c, d(\tau), 1)\}, c, d(\tau)\big)) = \frac{6}{7}$

Definition 13 induces an equivalence relation between processes:

Definition 14. Let P and Q be processes, we define

$$P \approx_{\mathcal{S}} Q \quad \Longleftrightarrow \quad \forall s:\ \mu_{tr}(P, s) = \mu_{tr}(Q, s)$$

Next, we will prove the equivalence between $\approx_{\mathcal{S}}$ and $\approx_{\mathcal{T}}$. First we have

Proposition 15. Let P and Q be processes such that $P \approx_{\mathcal{S}} Q$ and let T be a finite test, then $P\ pass_p\ T$ iff $Q\ pass_p\ T$.

Proof: The proof is made by structural induction on the test. □

As a corollary, we have $P \approx_{\mathcal{S}} Q \Rightarrow P \approx_{\mathcal{T}} Q$ since infinite tests are not necessary (see Lemma 8). In order to prove the other implication, we recall a result from [Núñ96] (a similar result can be found in [YCDS94]):

Lemma 16. Let f and f' be two rational functions of $n \geq 0$ variables $x_1, x_2 \ldots x_n$, defined as follows:

$$f = \sum_{i\in I} \frac{c_i}{1 + \sum_{j=1}^{n} d_{j,i} \cdot x_j} \qquad f' = \sum_{i'\in I'} \frac{c'_{i'}}{1 + \sum_{j=1}^{n} d'_{j,i'} \cdot x_j}$$

where I, I' are finite sets of indices; $c_i, c'_{i'} > 0$, and for each distinct $r, s \in I$, the tuples $(d_{1,r}, d_{2,r}, \ldots, d_{r,r})$ and $(d_{1,s}, d_{2,s}, \ldots, d_{r,s})$ are distinct; and for each distinct $r, s, \in I'$, the tuples $(d'_{1,r}, d'_{2,r}, \ldots, d'_{r,r})$ and $(d'_{1,s}, d'_{2,s}, \ldots, d'_{r,s})$ are distinct. If $f = f'$, then there exists a bijection $h : I \longrightarrow I'$ such that $d_{j,i} = d'_{j,h(i)}$ and $c_i = c'_{h(i)}$ for all $i \in I$ and $1 \leq j \leq n$.

We will also need the following result whose proof is easy by transition induction.

Lemma 17. Let P be a process, A be a state, and $t = \max\{t' \mid \exists(a', t', p') \in A\} < \infty$. Then, $\quad (a_1, t, p_1), (a_2, t, p_2) \in A \quad \Longrightarrow \quad \mu_{tr}(P, (A, a_1, t)) = \mu_{tr}(P, (A, a_2, t))$.

For instance, in Example 4, $A = \{(a, 0, \frac{5}{6}), (b, 0, \frac{1}{6})\}$, $t = 0$ and $\mu_{tr}(P, (A, a, 0)) = \mu_{tr}(P, (A, b, 0)) = \frac{6}{7}$.

Proposition 18. Let P, Q be processes and s be a trace such that $\mu_{tr}(P, s) \neq \mu_{tr}(Q, s)$. Then there exists a test T such that P *pass*$_p$ T and Q *pass*$_q$ T, with $p \neq q$.

Proof (sketch): We present the proof only for s having length 1. For a longer trace the argument is very similar, considering tests for any step of the trace. Let S be the set of actions appearing in P and Q, i.e. $S = \{a_1, \ldots, a_n\}$. Let $t \in \mathcal{T}$ the minimum time satisfying that there exists a state A, and an action a such that $\mu_{tr}(P, (A, a, t)) \neq \mu_{tr}(Q, (A, a, t))$. Let us consider the set of all possible[5] states made up of S until time t, $\mathcal{A} = \{A_1, \ldots, A_m\}$. For any A_i, let us take $t_i = \max\{t' \mid \exists (a', t', p') \in A\} < t$. By Lemma 17 we can take

$$p_k = \mu_{tr}(P, (A_k, a', t_k)) \quad \text{and} \quad q_k = \mu_{tr}(P, (A_k, a', t_k))$$

for any $a' \in \text{Act}$ such that there exists p_k satisfying $(a', t_k, p_k) \in A_k$. Then, for any $1 \leq i \leq m$, $1 \leq j \leq n$ and $0 \leq t' \leq t$ we take

$$p_{ij}^{t'} = \begin{cases} p \text{ if } (a_j, t', p) \in A_i \\ 0 \text{ otherwise} \end{cases}.$$

For any $0 \leq t' \leq t$ let us consider a probability distribution $\langle r_1^{t'}, \ldots, r_n^{t'}, r_{n+1}^{t'} \rangle$, and then consider the tests

$$T_{t'} = \sum_{i=1}^{n+1} [r_i^{t'}] e_i \{0\}; T_{t'}^{\delta} \quad \text{where} \quad T_{t'}^{\delta} = \begin{cases} \text{stop} & \text{if } 1 \leq i \leq n \\ T_{t'+\delta} & \text{if } i = n+1 \ \wedge \ 0 \leq t' < t \\ ok & \text{if } i = n+1 \ \wedge \ t' = t \end{cases}$$

where $e_{n+1} = \tau$ and $\forall\, 1 \leq i \leq n : \ e_i = a_i \in S$. Then we have P *pass*$_p$ T and Q *pass*$_q$ T with

$$p = \sum_{k=1}^{m} \frac{p_k \cdot r_{n+1}^0 \cdot pt(\delta, k)}{r_{n+1}^0 + \sum_{i=1}^{n} r_i^{t'} \cdot p_{ki}^{t'}} \quad \text{and} \quad q = \sum_{k=1}^{m} \frac{q_k \cdot r_{n+1}^0 \cdot pt(\delta, k)}{r_{n+1}^0 + \sum_{i=1}^{n} r_i^{t'} \cdot p_{ki}^{t'}}$$

where

$$pt(t', k) = \begin{cases} \dfrac{r_{n+1}^{t'} \cdot pt(t' + \delta, k)}{r_{n+1}^{t'} + \sum_{i=1}^{n} r_i^{t'} \cdot p_{ki}^{t'}} \text{ if } t' < t \\ \\ \dfrac{r_{n+1}^{t'}}{r_{n+1}^{t'} + \sum_{i=1}^{n} r_i^{t'} \cdot p_{ki}^{t'}} \text{ if } t' = t \end{cases}$$

Then, after some algebraic manipulations, we can apply Lemma 16, obtaining $p \neq q$. □

From the previous result we immediately obtain $P \approx_{\mathcal{T}} Q \ \Rightarrow P \approx_{\mathcal{S}} Q$. So, by Propositions 15 and 18 we get the final result.

Theorem 19. Let P, Q be processes. Then, we have $P \approx_{\mathcal{T}} Q$ iff $P \approx_{\mathcal{S}} Q$.

[5] This set is finite as we are considering only the actions of S and the time domain is discrete.

4 Conclusions and Future Work

The goal of our research has been to reduce the gap between real-world systems and models described by process algebras. To reach this goal we aim expressiveness, simplicity and utility.

Expressiveness: we have defined a syntax for a process algebra which deal with concepts, time and probabilities, which are necessary to properly specify and verify real-time concurrent distributed systems.

Simplicity: To easily understand the behavior of the processes we have defined an operational semantics for the syntactic processes. Action and time transitions are used and, to clearly differentiate the probabilistic information from the operational one, a probability function which records the probabilistic branching is used. Then we have followed the intuitive testing methodology to define an equivalence between processes.

Utility: To make the equivalence tractable we have defined the (probabilistic-timed) traces of a process and we have proved that the induced trace equivalence characterizes the testing equivalence.

The way we have started to approach process algebra to real systems can be continued in two directions: A theoretical direction to get a throughly knowledge about the model, for example, by presenting it from different points of view. In particular, it would be desirable to characterize a congruence relation from the testing equivalence. Later a denotational semantics, as well as a sound and complete set of axioms w.r.t. the induced congruence would be given. The second direction, a practical one, by enlarging the syntax of the language with derived operators from the basic ones and by implementing tools using the theoretical background we are developing.

Acknowledgments. We would like to thank Scott A. Smolka who suggested us to use a transition probability function and provided us with [SS96].

References

[BB93] J. C. M. Baeten and J. A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3:142–188, 1993.

[BBS95] J.C.M. Baeten, J.A. Bergstra, and S.A. Smolka. Axiomatizing probabilistic processes: ACP with generative probabilities. *Information and Computation*, 121(2):234–255, 1995.

[Chr90] I. Christoff. Testing equivalences and fully abstract models for probabilistic processes. In *CONCUR'90, LNCS 458*, pages 126–140, 1990.

[CLLS96] R. Cleaveland, I. Lee, P. Lewis, and S.A. Smolka. A theory of testing for soft real-time processes. In *8th International Conference on Software Engineering and Knowledge Engineering*, 1996.

[CSZ92] R. Cleaveland, S.A. Smolka, and A.E. Zwarico. Testing preorders for probabilistic processes. In *19th ICALP, LNCS 623*, pages 708–719, 1992.

[GJS90] A. Giacalone, C.-C. Jou, and S.A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of Working Conference on Programming Concepts and Methods, IFIP TC 2*, 1990.

[Gro93] Jan Friso Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118:263–299, 1993.

[Han91] Hans A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. PhD thesis, Department of Computer Systems. Uppsala University, 1991.

[Hen88] M. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.

[HR95] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117(2):221–239, 1995.

[JHSY94] B. Jonsson, C. Ho-Stuart, and W. Yi. Testing and refinement for nondeterministic and probabilistic processes. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 863*, pages 418–430, 1994.

[JY95] B. Jonsson and W. Yi. Compositional testing preorders for probabilistic processes. In *10th IEEE Symposium on Logic In Computer Science*, pages 431–443, 1995.

[LdFN96] L. Llana, D. de Frutos, and M. Núñez. Testing semantics for urgent timed algebras. In *3rd AMAST Workshop on Real-Time Systems*, 1996.

[Low95] G. Lowe. Probabilistic and prioritized models of timed CSP. *Theoretical Computer Science*, 138:315–352, 1995.

[LS91] K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[NdF95] M. Núñez and D. de Frutos. Testing semantics for probabilistic LOTOS. In *Formal Description Techniques VIII*, pages 365–380, 1995.

[NdFL95] M. Núñez, D. de Frutos, and L. Llana. Acceptance trees for probabilistic processes. In *CONCUR'95, LNCS 962*, pages 249–263, 1995.

[NJ91] X. Nicollin and J.Sifakis. An overview and synthesis on timed process algebras. In *Computer Aided Verification'91, LNCS 575*, pages 376–398, 1991.

[Núñ96] M. Núñez. *Semánticas de Pruebas para Álgebras de Procesos Probabilísticos*. PhD thesis, Universidad Complutense de Madrid, 1996. In Spanish (also in English).

[OMdF90] Y. Ortega-Mallén and D. de Frutos. Timed observations: a semantic model for real-time concurrency. In *Proceedings of Working Conference on Programming Concepts and Methods, IFIP TC 2*, 1990.

[RR86] G.M. Reed and A.W. Roscoe. A timed model for CSP. In *13th ICALP, LNCS 226*, pages 314–323, 1986.

[Sch95] S. Schneider. An operational semantics for timed CSP. *Information and Computation*, 116:193–213, 1995.

[SS96] E.W. Stark and S.A. Smolka. A complete axiom system for finite-state probabilistic processes, 1996.

[Tof94] C. Tofts. Processes with probabilities, priority and time. *Formal Aspects of Computing*, 6:536–564, 1994.

[vGSS95] R. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative and stratified models of probabilistic processes. *Information and Computation*, 121(1):59–80, 1995.

[YCDS94] S. Yuen, R. Cleaveland, Z. Dayar, and S.A. Smolka. Fully abstract characterizations of testing preorders for probabilistic processes. In *CONCUR'94, LNCS 836*, pages 497–512, 1994.

[Yi91] W. Yi. CCS+ Time = an interleaving model for real time systems. In *18th ICALP, LNCS 510*, pages 217–228, 1991.

[YL92] W. Yi and K.G. Larsen. Testing probabilistic and nondeterministic processes. In *Protocol Specification, Testing and Verification XII*, pages 47–61, 1992.