A Hierarchy of Equivalences for Probabilistic Processes*

Manuel Núñez and Luis Llana

Dept. Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid, 28040 Madrid, Spain
{mn,llana}@sip.ucm.es

Abstract. We study several process equivalences on a probabilistic process algebra. First, we define an operational semantics. Afterwards we introduce the notion of *passing* a test with a probability. We consider three families of tests according to the intended behavior of an external observer: *Reactive* (sequential tests), *generative* (branching tests), and *limited generative* (equitable branching tests). For each of these families we define three predicates over processes and tests (*may-pass, must-pass,* $pass_p$) which induce three equivalences. Finally, we relate these nine equivalences and provide either alternative characterizations or fully abstract denotational semantics. These semantic frameworks cover from simple traces to probabilistic acceptance trees.

1 Introduction

Process algebras [16,24,1,25,2] are an adequate mechanism to formally specify and analyze networked and distributed systems. The process algebra literature includes numerous semantic models. These semantic frameworks are used to describe the behavior of processes as well as to define relations on them. Testing semantics [7,15] represents one of these semantic frameworks. Intuitively, two processes are *testing equivalent* if they have the same *responses* for all the *tests* belonging to a certain set. Depending on how these responses are analyzed, several testing semantics can be defined: May, must, refusal, fair, etc. We consider that this semantic framework is very suitable because it is easy to understand and allows us to give different equivalences just by modifying the idea of what a test is or when a test is successfully passed.

Once the *basic* frameworks were studied, research in process algebras has tried to close the gap between formal models and real systems. Thus, features which were initially abstracted have been introduced later. This is the case of probabilistic information. In particular, if we concentrate only on probabilistic testing, regardless whether the underlying model is a process algebra or another formalism, several semantics have been defined. This area of research was very active

* This research was partially supported by the Spanish MEC project WEST/FAST TIN2006-15578-C02-01 and the Marie Curie project MRTN-CT-2003-505121/ TAROT.

K. Suzuki et al. (Eds.): FORTE 2008, LNCS 5048, pp. 267–282, 2008.
© IFIP International Federation for Information Processing 2008

in the previous decade (we can mention [5,35,18,30,29,33,26,12,21,6,31]). Moreover, there also exists recent work on the topic (see, for example [19,3,28,34,4,8]). Despite the myriad of papers on the subject, there is a lack of papers comparing several testing approaches under the same umbrella. In fact, most work in this direction is limited to compare a may and a must variants in a given probabilistic setting. That is, we miss a classification in the probabilistic setting similar to the ones provided in [9,10] for nonprobabilistic processes. This paper represents a first step towards such a complete study.

In this paper we study different testing semantics for the probabilistic process algebra PPA defined in [30,28]. This process algebra has two choice operators (external and internal), which are extended with a probability. In addition, it allows the definition of recursive behaviors. The study of the different semantics is performed in a testing framework. We propose three families of tests according to the capabilities of an external observer:

- In the *reactive* model [22] the environment can offer only one action at a given time. Intuitively, an entity interacting with a system *can press only one button at a given time.*
- In the *generative* model [11] the environment can offer more than an action and with different probabilities. So, *more than one button can be pressed at a given time and they can be pressed with different strengths.*
- In the *limited generative* model [29] the environment can offer several actions at a given time, but the probabilities associated with these actions are the same. So, *more than one button can be pressed at a given time, but they have to be pressed with the same strength.*

For each of the above families of tests we consider three different definitions of successfully passing a test:

- P *may-pass* T if the probability with which P *passes* T is greater than zero.
- P *muss-pass* T if the probability with which P *passes* T is equal to one.
- $P\ pass_p\ T$ if the probability with which P *passes* T is equal to p.

The combination of these two ideas (the different families of tests and the different interpretations of successfully passing a test) induce nine equivalence relations. In order to provide real usefulness to these equivalences, we relate them and provide either alternative characterizations or fully abstract denotational semantics.

We will show that the *may* interpretation coincides in the three families of tests. We define a fully abstract denotational semantics, based on traces, for the induced equivalence. For the *must-reactive* interpretation, we show that a fully abstract denotational semantics cannot be defined by using the usual least fixpoint techniques. Thus, we give an alternative characterization based on *must* traces. The *must* interpretation coincides for the *generative* and *limited generative* models, and we give an alternative characterization based on acceptance sets [15]. We show that the reactive testing equivalence is not a congruence and we define an alternative characterization based on *probabilistic traces.* For the

$$\begin{array}{ccccc}
\approx^{\mathcal{R}}_{\text{must}} & \sqsubset & \approx^{\mathcal{LG}}_{\text{must}} & = & \approx^{\mathcal{G}}_{\text{must}} \\
\sqcap & & \sqcap & & \sqcap \\
\approx^{\mathcal{R}} & \sqsubset & \approx^{\mathcal{LG}} & \sqsubset & \approx^{\mathcal{G}} \\
\sqcup & & \sqcup & & \sqcup \\
\approx^{\mathcal{R}}_{\text{may}} & = & \approx^{\mathcal{LG}}_{\text{may}} & = & \approx^{\mathcal{G}}_{\text{may}}
\end{array}$$

Fig. 1. Hierarchy of testing equivalences for PPA

generative model we have a fully abstract denotational semantics based on probabilistic acceptance trees [28]. Finally, we provide an alternative characterization, based on a set of *essential* tests, for the limited generative model. In Figure 1 we show how the different equivalences studied in this paper are related.

As a derived result, our testing equivalences appropriately deal with *unfair divergences* caused by unguarded recursive definitions in the context of an internal choice. In fact, our results about fairness can be compared with those in [27,32], while they are not so similar to those in [14] where fairness is only considered in the context of parallel compositions. A study on the relation between probabilistic testing and fairness can be found in [31].

In terms of related work, our reactive and generative models follow the same intuitive ideas as those in [11], but we do not need to give different operational semantics for each of the models. We have just an operational semantics and the differences between the models come from the considered families of tests. The model described in [35], and other testing frameworks based on it, can be compared to our generative point of view. They define a probabilistic process algebra with two choice operators but, unlike PPA, the external choice does not have an associated probability. This fact simplifies the operational semantics but complicates the definition of a testing semantics: A process passes a test with a set of probabilities. They adapt the notions of *may* and *must* equivalences by computing the infimum and supremum of sets of probabilities. For these equivalences, compositional characterizations are defined in [18] and characterized as simulations in [19]. We think that these characterizations are so complicated because of the absence of probabilities in external choices. In contrast, our operational semantics is slightly harder, because we have to deal with probabilities in the scope of the external choice, but testing equivalences are simpler and more intuitive. Thus, the alternative characterizations for the *may* and *must* interpretations are clearer and more similar to the nonprobabilistic model.

In [20,17,23] several equivalence relations for probabilistic processes are given by adapting nonprobabilistic relations to the probabilistic setting. In spite of being interesting, they relate probabilistic equivalence relations which are not in a common framework. We address our work in a rather different way: We begin

by settling a testing framework, then we define natural semantic equivalences in this framework, and, finally, we provide alternative characterizations for these equivalences and relate them.

The rest of the paper is organized as follows. Section 2 presents the syntax, operational and testing semantics for PPA. In Sections 3, 4, and 5 we study testing semantics and alternative characterizations for the reactive, generative, and limited generative models, respectively. Finally, in Section 6 we present our conclusions.

2 An Overwiew of PPA

In this section, we briefly review the basic concepts of our probabilistic process algebra: Syntax, operational semantics and testing semantics. This process algebra was used in [28], so more details can be found there (in particular, intuitive explanations of the operational semantics rules and the intended meaning of the rules defining the interaction between a process and a test). In addition, in this section we also define the interaction between a process and a test and adapt the *must* and *may* equivalences to our probabilistic framework.

2.1 Syntax and Operational Semantics of PPA

Definition 1. Given a finite set of actions Act and a set of identifiers Id, the set of PPA processes is defined by the following BNF expression:

$$P ::= \ Nil \mid \Omega \mid X \mid a; P \mid P \oplus_p P \mid P +_p P \mid recX.P$$

where $p \in (0,1)$, $a \in Act$, and $X \in Id$. □

From now on, except as noted, we only consider closed processes, that is processes without free occurrences of variables. In this process algebra, *Nil* is a deadlocked process, Ω is a divergent process, $a; P$ denotes the action a prefixing the process P, $P \oplus_p Q$ denotes an internal choice between P and Q with associated probability p, $P +_p Q$ is an external choice between P and Q with associated probability p, and $recX.P$ is used to define recursive processes. We can extend the external choice operator to an n-ary one.

Definition 2. Let $P_1, \ldots, P_n$ be processes and $p_1, \ldots, p_n > 0$ such that $\sum p_i = 1$. We define the *generalized external choice* as:

1. $\sum_{i=1}^{1} [1]\ P = P$.
2. $\sum_{i=1}^{n} [p_i]\ P_i = P_1 +_{p_1} (\sum_{i=1}^{n-1} [\frac{p_{i+1}}{1-p_1}]\ P_{i+1})$. □

Next we give a syntactic definition for the *stability* of a process. It expresses that a process does not have unguarded internal choices, or equivalently that a process will not be able to (immediately) perform an internal transition. We also define a function *live* computing whether a stable process is operationally equivalent to *Nil*.

$$(PRE)\frac{}{a;P \xrightarrow{a}_1 P} \qquad (INT1)\frac{}{P \oplus_p Q >\!\!\longrightarrow_p P} \qquad (INT2)\frac{}{P \oplus_p Q >\!\!\longrightarrow_{1-p} Q}$$

$$(EXT1)\frac{P >\!\!\longrightarrow_q P' \ \wedge\ \mathtt{stable}(Q)}{P +_p Q >\!\!\longrightarrow_q P' +_p Q} \qquad (EXT2)\frac{Q >\!\!\longrightarrow_q Q' \ \wedge\ \mathtt{stable}(P)}{P +_p Q >\!\!\longrightarrow_q P +_p Q'}$$

$$(EXT3)\frac{P >\!\!\longrightarrow_{q_1} P' \ \wedge\ Q >\!\!\longrightarrow_{q_2} Q'}{P +_p Q >\!\!\longrightarrow_{q_1 \cdot q_2} P' +_p Q'}$$

$$(EXT4)\frac{P \xrightarrow{a}_q P' \ \wedge\ \mathtt{stable}(Q)}{P +_p Q \xrightarrow{a}_{p \cdot \hat{q}} P'} \qquad (EXT5)\frac{Q \xrightarrow{a}_q Q' \ \wedge\ \mathtt{stable}(P)}{P +_p Q \xrightarrow{a}_{(1-p) \cdot \hat{q}} Q'}$$

$$(REC)\frac{}{rec\ X.P >\!\!\longrightarrow_1 P\{rec\ X.P/X\}} \qquad (DIV)\frac{}{\Omega >\!\!\longrightarrow_1 \Omega}$$

where $\hat{q} = \frac{q}{p \cdot \mathtt{live}(P) + (1-p) \cdot \mathtt{live}(Q)}$.

Fig. 2. Operational Semantics of PPA

Definition 3. We define the predicate $\mathtt{stable}(P)$ over PPA processes as:

- $\mathtt{stable}(Nil) = \mathtt{stable}(a; P) = \mathtt{true}$
- $\mathtt{stable}(\Omega) = \mathtt{stable}(X) = \mathtt{stable}(P_1 \oplus_p P_2) = \mathtt{stable}(recX.P) = \mathtt{false}$
- $\mathtt{stable}(P_1 +_p P_2) = \mathtt{stable}(P_1) \ \wedge\ \mathtt{stable}(P_2)$

We define the function $\mathtt{live}(P)$ over PPA processes as:

- $\mathtt{live}(Nil) = 0$
- $\mathtt{live}(a; P) = 1$
- $\mathtt{live}(P_1 +_p P_2) = \max(\mathtt{live}(P_1), \mathtt{live}(P_2))$ □

Even though the function $\mathtt{live}(\_)$ is not defined for unstable processes, this fact does not represent a problem since we will apply it only to stable processes. The set of rules that define the operational semantics is given in Figure 2. There are two types of transitions. The intuitive meaning of a transition $P \xrightarrow{a}_p Q$ (external transitions) is that if the environment offers all the actions in Act then the probability with which P performs a and then behaves as Q is equal to p; the meaning of $P >\!\!\longrightarrow_p Q$ (internal transitions) is that the process P evolves to Q with probability p without interaction with the environment.

For the sake of simplicity, we use multisets of transitions in order to get sets of transitions. So, if a transition can be derived in several ways, each derivation generates a different instance of this transition. For example, let us consider the process $P = a +_{\frac{1}{2}} a$, where trailing occurrences of Nil have been omitted. If we were not careful, we would have the transition $P \xrightarrow{a}_{\frac{1}{2}} Nil$ only once, while we should have this transition twice. This problem is similar for the $\oplus_p$ operator. So, if a transition can be derived in several ways, we consider that each derivation generates a different instance. In particular, when we define the testing semantics we will consider multisets of computations as well. We will use the delimiters $\{\!|$ and $|\!\}$ to denote multisets.

As shown in [28], this operational semantics separates between internal and external transitions. So, a process can perform an external transition only if this process is stable (that is, it cannot perform internal transitions).

$$\frac{P \succ\!\!\longrightarrow_p P' \wedge T_\oplus = 0}{P \mid T \longmapsto_p P' \mid T} \qquad \frac{T \succ\!\!\longrightarrow_p T' \wedge P_\oplus = 0}{P \mid T \longmapsto_p P \mid T'} \qquad \frac{P \succ\!\!\longrightarrow_p P' \wedge T \succ\!\!\longrightarrow_q T'}{P \mid T \longmapsto_{p \cdot q} P' \mid T'}$$

$$\frac{P \xrightarrow{a}_p P' \wedge T \xrightarrow{a}_q T'}{P \mid T \longmapsto_{r_1} P' \mid T'} \qquad \frac{T \xrightarrow{\omega}_p T' \wedge P_\oplus = 0}{P \mid T \overset{\omega}{\longmapsto}_{r_2} Nil}$$

where $r_1 = \frac{f_1^{P,T}(p) \cdot f_2^{P,T}(q)}{\mu(P,T)}$ and $r_2 = \frac{f_2^{P,T}(p)}{\mu(P,T)}$.

$$f_1^{P,T}(p) = \frac{p}{\sum_a \{\!|\ r \mid \exists P', T', p' :\ P \xrightarrow{a}_r P' \wedge T \xrightarrow{a}_{p'} T'\ |\!\}}$$

$$f_2^{P,T}(q) = \frac{q}{\sum_a \{\!|\ r \mid \exists P', T', p' :\ T \xrightarrow{a}_r T' \wedge P \xrightarrow{a}_{p'} P'\ |\!\} + \sum \{\!|\ r \mid \exists T' :\ T \xrightarrow{\omega}_r T'\ |\!\}}$$

$$\mu(P,T) = \begin{array}{l} \sum_a \{\!|\ f_1^{P,T}(p) \cdot f_2^{P,T}(q) \mid \exists P', T' :\ P \xrightarrow{a}_p P' \wedge T \xrightarrow{a}_q T'\ |\!\} \\ + \sum \{\!|\ f_2^{P,T}(p) \mid \exists T' :\ T \xrightarrow{\omega}_p T'\ |\!\} \end{array}$$

Fig. 3. Rules for the parallel composition

Lemma 1. Let P be a process. If there exist p and P' such that $P \succ\!\!\longrightarrow_p P'$ then there do not exist q, a, P'' such that $P \xrightarrow{a}_q P''$. Equivalently, if there exist p, a, P' such that $P \xrightarrow{a}_p P'$ then there do not exist q and P'' such that $P \succ\!\!\longrightarrow_q P''$.

2.2 Testing Semantics

As usual, tests are processes where the alphabet Act is extended with a new action ω indicating successful termination. The operational semantics of tests is the same as that of processes (considering ω as an ordinary action). The rules defining the interaction between a process and a test (modelled by their parallel composition) are given in Figure 3. In these rules we use a *normalization factor* $\mu(P,T)$ similar to that in [6]. In addition, we also use two *prenormalization factors* ($f_1^{P,T}$ and $f_2^{P,T}$) in order to *distribute* the probability associated with those actions which cannot be performed by both sides of the parallel composition among the actions which can be performed by both sides. This represents a small change with respect to the definition given in [28], but it does not change the induced testing equivalence (even though it changes the probability with which processes pass tests).

Example 1. Let $P = (a; Nil) +_{\frac{1}{4}} (b; Nil)$ and $T = (a; Nil) +_{\frac{1}{2}} \omega$. If we would not use prenormalization factors we would have P passes the test T with a probability $\frac{1}{5} = \frac{\frac{1}{4} \cdot \frac{1}{2}}{\frac{1}{4} \cdot \frac{1}{2} + \frac{1}{2}}$ while using prenormalization factors we obtain a probability of $\frac{1}{2} = \frac{\frac{1}{2}}{\frac{1}{2} + \frac{1}{2}}$. This is a more intuitive result because the action b should not *subtract* probability from a.

Definition 4. Let P be a process and T be a test. A *computation* is a *maximal* sequence of transitions of the form

$$C = P \mid T \longmapsto_{p_1} P_1 \mid T_1 \longmapsto_{p_2} \cdots P_{n-1} \mid T_{n-1} \stackrel{?}{\longmapsto}_{p_n} R$$

where ? denotes either an empty label or the special action ω. A sequence is said to be *maximal* when there do not exist $p > 0$ and R' such that $R \stackrel{?}{\longmapsto}_p R'$.

When the last transition is of the form $P_{n-1} \mid T_{n-1} \stackrel{\omega}{\longmapsto}_{p_n} Nil$ we say that the computation is *successful*. We denote by $\tilde{C}$ the *set* of *successful computations* from C. The probability of a successful computation S, denoted by $Pr(S)$, is inductively defined as

$$Pr(Nil) = 1 \text{ (i.e. } T \text{ has succeeded)}$$
$$Pr(P \mid T \stackrel{*}{\longmapsto}_p C) = p \cdot Pr(C)$$

We write P *may-pass* T if $\sum_{S \in \tilde{C}} Pr(S) > 0$. We write P *muss-pass* T if $\sum_{S \in \tilde{C}} Pr(S) = 1$. We write $P\ pass_p\ T$ if $\sum_{S \in \tilde{C}} Pr(S) = p$. □

In the previous definition, by *maximal* we mean that it cannot be extended, that is, from the reached point on, the composition of process and test cannot perform any action. Given a family of tests we can define the corresponding notions of testing, *may*, and *must* equivalences with respect to this family.

Definition 5. Given a family of probabilistic tests $\mathcal{T}$ and two processes P and Q we say:

- P and Q are *probabilistic testing equivalent* with respect to $\mathcal{T}$, denoted by $P \approx^{\mathcal{T}} Q$, iff for all $T \in \mathcal{T}$ we have $P\ pass_p\ T \Longleftrightarrow Q\ pass_p\ T$.
- P and Q are *probabilistic may-testing equivalent* with respect to $\mathcal{T}$, denoted by $P \approx^{\mathcal{T}}_{\text{may}} Q$, iff for all $T \in \mathcal{T}$ we have P *may-pass* $T \Longleftrightarrow Q$ *may-pass* T.
- P and Q are *probabilistic must-testing equivalent* with respect to $\mathcal{T}$, denoted by $P \approx^{\mathcal{T}}_{\text{must}} Q$, iff for all $T \in \mathcal{T}$ we have P *muss-pass* $T \Longleftrightarrow Q$ *muss-pass* T.

□

As shown in [28], the set of tests can be reduced by considering tests without internal choices and without nondeterminism caused by prefixing by the same action in an external choice.

Lemma 2. Let P be a process, T and T' be tests, and $a \in Act$. We have:

1. $P\ pass_q\ (T \oplus_p T')$ iff $P\ pass_{q_1}\ T \wedge P\ pass_{q_2}\ T' \wedge p \cdot q_1 + (1-p) \cdot q_2 = q$
2. $P\ pass_q\ (a;T) +_p (a;T')$ iff $P\ pass_q\ a;(T \oplus_p T')$

3 The Reactive Model

In the reactive model the environment can offer only one action each time, that is, *only one button can be pressed at a given time*. In a testing framework, this interpretation gives rise to tests being just traces finishing with the acceptance action ω.

Definition 6. (Reactive Tests) The set of *reactive tests*, denoted by $\mathcal{R}$, is defined by the BNF expression $T = \omega \mid a;T$. □

3.1 Alternative Characterization for $\approx^{\mathcal{R}}$

The next example shows that $\approx^{\mathcal{R}}$, that is, the test equivalence induced by considering $\mathcal{R}$ as set of tests, is not a congruence. So, we cannot define a fully abstract (compositional) denotational semantics for this equivalence.

Example 2. Consider $P = (a;c) +_{\frac{1}{2}} b$ and $P' = (a;c) +_{\frac{1}{3}} b$. Obviously, $P \approx^{\mathcal{R}} P'$, but if we consider $Q = a;b$, we have $P +_{\frac{1}{2}} Q \not\approx^{\mathcal{R}} P' +_{\frac{1}{2}} Q$. For example, if $T = a;b;\omega$ then we have $P +_{\frac{1}{2}} Q\, pass_{\frac{2}{3}}\, T$ while $P' +_{\frac{1}{2}} Q\, pass_{\frac{3}{4}}\, T$.

However, the rest of the operators are congruent for $\approx^{\mathcal{R}}$.

Proposition 1. Let P and P' be processes such that $P \approx^{\mathcal{R}} P'$. Then,

- For all action $a \in Act$ we have $a;P \approx^{\mathcal{R}} a;P'$.
- For all process Q and probability $p \in (0,1)$ we have $P \oplus_p Q \approx^{\mathcal{R}} P' \oplus_p Q$ and $Q \oplus_p P \approx^{\mathcal{R}} Q \oplus_p P'$.

Let $P(X)$ and $P'(X)$ be two terms with the free occurrence of the identifier X (that is, processes but with a free variable). If for all process Q we have $P(Q) \approx^{\mathcal{R}} P'(Q)$ then $recX.P \approx^{\mathcal{R}} recX.P'$.

We will define an alternative characterization of this equivalence based on the operational behavior of processes, considering *probabilistic traces*. First we extend $\longrightarrow$ to sequences of actions and $\rightarrowtail$ to sequences of internal transitions.

Definition 7. Let P and P' be processes. We write $P \rightarrowtail^*_p P'$ if this transition can be derived from the following rules:

$$P \rightarrowtail^*_1 P \text{ if } \mathtt{stable}(P)$$
$$P \rightarrowtail^*_p P' \text{ if } \exists q, q', Q :\ P \rightarrowtail_q Q \rightarrowtail^*_{q'} P' \wedge p = q \cdot q'$$

We write $P \xrightarrow{s}_p P'$ if this transition can be derived from the following rules:

$$P \xrightarrow{\epsilon}_p P' \text{ if } P \rightarrowtail^*_p P'$$
$$P \xrightarrow{\langle a \rangle \circ s'}_p P' \text{ if } \exists P_1, P_2, p_1, p_2, p_3 :\ P \rightarrowtail^*_{p_1} P_1 \wedge P_1 \xrightarrow{a}_{p_2} P_2 \wedge P_2 \xrightarrow{s'}_{p_3} P'$$

where $p = p_1 \cdot \frac{p_2}{\sum_{P'} \{\!|\ q \mid P_1 \xrightarrow{a}_q P' \ |\!\}} \cdot p_3$ and $s \circ s'$ denotes the concatenation of s and s'. We write $P \xrightarrow{s}_0 P'$ if $P \xrightarrow{s}_p P'$ cannot be derived from the previous rules. □

Let us note that if $P \rightarrowtail^*_p P'$ then P' must be stable. So, $\rightarrowtail^*$ is not *exactly* the reflexive and transitive closure of $\rightarrowtail$. Let us also note that the value p appearing in $P \xrightarrow{s}_p P'$ indicates the product of the probabilities associated with

the nondeterministic choices involved in the execution of the sequence s. This nondeterminism can be produced by internal transitions but also by external transitions labelled with the same action. As in the case of $\succ\!\!\longrightarrow$ and $\longrightarrow$, we must consider the possible repetitions of *generalized* transitions of the form $P \stackrel{s}{\longrightarrow}_p P'$ and $P \succ\!\!\longrightarrow_p^* P'$.

Definition 8. Let P be a process. We define the *set of probabilistic traces* of P as $ptraces(P) = \{(s,p) \mid p = \sum_{P'} \{\!\{ q \mid P \stackrel{s}{\longrightarrow}_q P' \}\!\} \wedge p > 0\}$. Given a trace s, we define the function $prob(P,s)$ as p if $(s,p) \in ptraces(P)$ and as 0, otherwise. □

Lemma 3. Let P be a process. We have $prob(P,s) = p$ iff $P\ pass_p\ \tilde{s}$, where $\tilde{s}$ denotes the reactive test conformed by the actions of the sequence s finishing with ω.

Theorem 1. Let P and P' be processes. We have $P \approx^{\mathcal{R}} P'$ iff $ptraces(P) = ptraces(P')$.

3.2 The *must* Reactive Equivalence

The next example shows that we cannot define a fully abstract denotational semantics using the usual least fixpoint technique. Instead, we will define an alternative characterization from the probabilistic traces of a process.

Example 3. Let $Q = recX.P(X)$, where $P(X) = (a; Nil) \oplus_p X$. We have that $Q \approx^{\mathcal{R}}_{\text{must}} a; Nil$. However, Ω is fixpoint of the equation $X = P(X)$ because $\Omega \approx^{\mathcal{R}}_{\text{must}} (a; Nil) \oplus_p \Omega$. Obviously, Ω is the least fixpoint. So, if we define the semantics of a recursive process with the usual technique we do not obtain the desired results.

Definition 9. Let P be a process. We define the *set of must probabilistic traces* of P as $must\text{-}traces(P) = \{s \mid (s,1) \in ptraces(P)\}$. □

That is, given a process, we consider its probabilistic traces which have 1 as associated probability. The induced equivalence coincides with $\approx^{\mathcal{R}}_{\text{must}}$ as it is stated in the next result.

Theorem 2. Let P, P' be processes. We have $P \approx^{\mathcal{R}}_{\text{must}} P'$ iff $must\text{-}traces(P) = must\text{-}traces(P')$.

3.3 The *may* Reactive Equivalence

Following a similar reasoning to that used for the *must* reactive equivalence, we can define an alternative characterization for $\approx^{\mathcal{R}}_{\text{may}}$, considering the traces of a process and *forgetting* the probabilistic information.

Definition 10. Let P be a process. We define the *set of may probabilistic traces* of P as $may\text{-}traces(P) = \{s \mid (s,p) \in ptraces(P)\}$. □

Theorem 3. Let P and P' be processes. We have $P \approx^{\mathcal{R}}_{\text{may}} P'$ iff *may-traces*(P) = *may-traces*(P').

But in this equivalence we can go further because a fully abstract denotational semantics for $\approx^{\mathcal{R}}_{\text{may}}$ can be given in terms of traces. The semantic domain, denoted by $\mathbf{TRA}_{Act}$, will be the sets of traces. We denote by $R, R_1, \ldots$ the elements of $\mathbf{TRA}_{Act}$ and by $[\![P]\!]^{\mathcal{R}}_{\text{may}}$ the semantics of process P.

Definition 11. Let $R_1, R_2 \in \mathbf{TRA}_{Act}$. We write $R_1 \sqsubseteq_{\mathbf{TRA}} R_2$ if for all $s \in Act^*$ we have $s \in R_1$ implies $s \in R_2$. We write $R_1 =_{\mathbf{TRA}} R_2$ if for all $s \in Act^*$ we have $s \in R_1$ iff $s \in R_2$. □

Lemma 4. $(\mathbf{TRA}_{Act}, \sqsubseteq_{\mathbf{TRA}})$ is a cpo.

Nil only has the empty trace. So, $[\![Nil]\!]^{\mathcal{R}}_{\text{may}} = \{\epsilon\}$. Ω can execute no trace because the only transition which Ω can perform is $\Omega \rightarrowtail_1 \Omega$. So, this process cannot be stable. Thus, $[\![\Omega]\!]^{\mathcal{R}}_{\text{may}} = \emptyset$.

For all $a \in Act$ we define the semantic function $a;\_ :: \mathbf{TRA}_{Act} \longrightarrow \mathbf{TRA}_{Act}$. The element $a; R \in \mathbf{TRA}_{Act}$ has the same traces as R but prefixed by the action a. So, $a; R = \{(a \circ s) \,|\, s \in R\} \cup \{\epsilon\}$. We add the empty trace because the syntactic process *associated* with this semantic process is stable.

For all $p \in (0,1)$ the functions $\oplus_p :: \mathbf{TRA}_{Act} \times \mathbf{TRA}_{Act} \longrightarrow \mathbf{TRA}_{Act}$ return the union of the corresponding sets of traces. That is, $R_1 \oplus_p R_2 = R_1 \cup R_2$.

In the case of the external choice we must consider if any of the processes is *equivalent* to Ω. For all $p \in (0,1)$ the function $+_p :: \mathbf{TRA}_{Act} \times \mathbf{TRA}_{Act} \longrightarrow \mathbf{TRA}_{Act}$ is defined as

$$R_1 +_p R_2 = \begin{cases} \emptyset & \text{if } R_1 =_{\mathbf{TRA}} \emptyset \ \vee \ R_2 =_{\mathbf{TRA}} \emptyset \\ R_1 \cup R_2 & \text{otherwise} \end{cases}$$

Proposition 2. For all $a \in Act$ the semantic function $a;\_$ is continuous. For all $p \in (0,1)$ the semantic functions $\oplus_p$ and $+_p$ are continuous.

As usual when defining a denotational semantics, the meaning of recursive expressions $recX.P(X)$ is given by the limit of its finite approximations

$$P_0 = \Omega, P_1 = P(\Omega), \ldots, P_n = P^n(\Omega)$$

Since all the operators of the language are continuous, this limit is the least fixpoint of the equation $X = P(X)$. That is, $[\![recX.\ P(X)]\!]^{\mathcal{R}}_{\text{may}} = \bigsqcup_{n=0}^{\infty} [\![P_n]\!]^{\mathcal{R}}_{\text{may}}$.

Lemma 5. Let P be a process. We have $s \in [\![P]\!]^{\mathcal{R}}_{\text{may}}$ iff P *may-pass* $\tilde{s}$, where $\tilde{s}$ denotes the reactive test conformed by the actions of the sequence s finishing with ω.

Theorem 4. Let P, P' be processes. We have $[\![P]\!]^{\mathcal{R}}_{\text{may}} =_{\mathbf{TRA}} [\![P']\!]^{\mathcal{R}}_{\text{may}}$ iff $P \approx^{\mathcal{R}}_{\text{may}} P'$.

Let us remark that a fully abstract denotational semantics can be easily defined, from this one, for $\approx^{\mathcal{R}}$ (reactive testing equivalence) if we consider the subset of PPA without external choices. It is enough to modify the semantic functions adding a probability equal to 1 to the empty trace, taking into account the probability associated with the traces of R, when defining $a; R$, and considering the probability associated with internal choices (see [28]).

As a concluding remark, we have that *may* and *must* reactive equivalences do not imply reactive equivalence (obviously, reactive equivalence implies *may* and *must* equivalences). For example, let us consider $P = a \oplus_{\frac{1}{3}} b$ and $Q = a \oplus_{\frac{1}{2}} b$, where trailing occurrences of *Nil* have been removed. We have $P \approx^{\mathcal{R}}_{\text{may}} Q$ and $P \approx^{\mathcal{R}}_{\text{must}} Q$ but $P \not\approx^{\mathcal{R}} Q$, because, for example, $P\, pass_{\frac{1}{3}}\, a; \omega$ while $Q\, pass_{\frac{1}{2}}\, a; \omega$.

Finally, let us mention that reactive equivalences are in general too weak. For example, they cannot distinguish between $a +_p b$ and $a +_q b$. Also, these equivalences identify processes which will be divergent in the next step. For example, reactive equivalences identify the processes $a; \Omega$ and $b; \Omega$ (and both are equivalent to *Nil*).

4 The Generative Model

In the generative model the environment can offer several actions each time, and with different probabilities. That is, *several buttons can be pressed at the same time and with different strengths*. So, in this model the family of tests, denoted by $\mathcal{G}$, is the whole set of probabilistic tests. Nevertheless, the set of tests can be reduced by applying Lemma 2 and by considering that recursive tests do not increase the distinguishing power of tests.

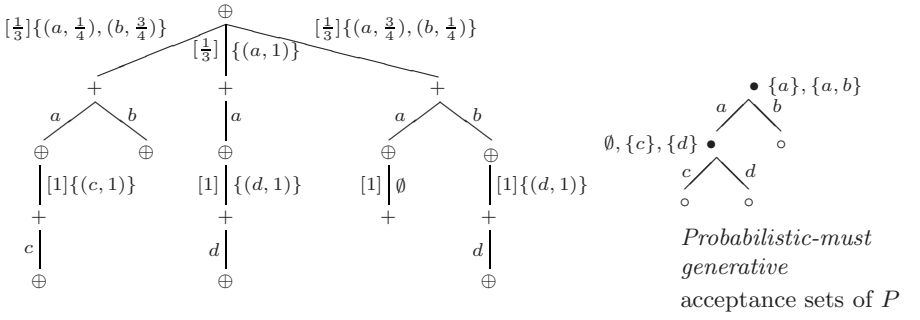
For the generative testing equivalence ($\approx^{\mathcal{G}}$) a fully abstract denotational semantics was defined in [28] and extensively studied. This denotational semantics is based on the notion of *probabilistic acceptance trees* which are a natural extension of acceptance trees [13]. These trees have two kinds of nodes: *Internal* and *external*. The root is an internal node. Arcs outgoing from internal nodes are labelled with different *states* (sets of pairs ⟨action, probability⟩ where the actions are different, and if the state is not empty then the probabilities sum up to one), with an associated probability. The sum of these probabilities is less than or equal to 1, and the difference between one and this sum denotes the probability of divergence at this point. These arcs go to external nodes. The arcs outgoing from external nodes are labelled with the actions belonging to the state labelling the ingoing arc. For any action in that state, there is a (unique) arc labelled with this action. These arcs go to internal nodes. An example is shown in Figure 4.

There are two important differences with respect to nonprobabilistic acceptance trees (of course, out of probabilities). First, there can be more than one state with the same actions. For example, a process may have the states: $\{(a, \frac{1}{3}), (b, \frac{2}{3})\}$ and $\{(a, \frac{1}{4}), (b, \frac{3}{4})\}$. Second, the *continuations* after the same

action are not joined. So the process $(a;P) \oplus_p ((a;P') +_q (b;Q))$ is not necessarily equivalent to the process $(a;(P \oplus_{q'} P')) \oplus_p ((a;(P \oplus_{q''} P')) +_q (b;Q))$.

Due to lack of space, we do not present the semantic functions corresponding to the syntactic operators (they can be found in [28]) and we just repeat the final result.

Theorem 5. Let P, P' be processes. We have $P \approx^{\mathcal{G}} P'$ iff $[\![P]\!]^{\mathcal{G}} = [\![P']\!]^{\mathcal{G}}$.



Generative probabilistic acceptance tree of P

Probabilistic-must generative acceptance sets of P

Fig. 4. Semantics of $P = (a;d;\Omega) \oplus_{\frac{1}{3}} (((a;c;\Omega) +_{\frac{1}{4}} b;\Omega) \oplus_{\frac{1}{2}} ((a;Nil) +_{\frac{3}{4}} b;d;\Omega))$

Even though we have increased the set of tests, this fact does not influence the *may* interpretation, that is, as in the nonprobabilistic case, it is enough to consider *sequential* tests to characterize a *may* equivalence. Thus, if two processes are *may* reactive equivalent, they will be *may* generative equivalent. This is so because if one process passes a generative test with a probability greater than zero then each successful computation generates a sequential test conformed by the actions taking part in the computation. Because they are *may* reactive equivalent, this computation is also successful for the other process, and so it passes the generative test with a probability greater than 0.

Theorem 6. Let P and P' be processes. We have $P \approx^{\mathcal{R}}_{\text{may}} P'$ iff $P \approx^{\mathcal{G}}_{\text{may}} P'$.

4.1 The *must* Generative Equivalence

Using a similar argument to that of Example 3, we cannot define a fully abstract denotational semantics using the usual least fixpoint techniques for this equivalence. Instead, we will define an alternative characterization, in the same way that for the nonprobabilistic case, based on *acceptance sets* [7,15].

Definition 12. Given a stable process P, the set of actions that can be (immediately) performed by P is given by $S(P) = \{a \mid \exists P', p : \ P \xrightarrow{a}_p P'\}$. Given a

process P and a sequence of actions $s \in Act^*$, we define the *acceptance sets* of P *after* s as:

$$\mathcal{A}(P,s) = \begin{cases} \emptyset & \text{if } \sum_Q \{\!|\, p \mid P \xrightarrow{s}_p Q \,|\!\} < 1 \\ \{S(P') \mid \exists P', p :\ P \xrightarrow{s}_p P' \ \wedge \ p > 0\} & \text{otherwise} \end{cases}$$

□

The previous definition needs some explanation. In order to compute the acceptance sets of a process after a sequence s, first we compute the processes to which the process may evolve by performing the generalized external transition corresponding to s. If the sum of the probabilities associated with these transitions is less than one, then there exists a *computation* reaching a divergent process. In this case, as in the nonprobabilistic case, we consider that from this point on the process is divergent, and we return $\emptyset$ as acceptance set.[1] For example, let us consider $P = (a;\Omega) +_{\frac{1}{2}} (a;b;Nil)$. We have $\sum \{\!|\, p \mid P \xrightarrow{\langle a \rangle}_p Q \,|\!\} = \frac{1}{2}$ because we only can derive $P \xrightarrow{\langle a \rangle}_{\frac{1}{2}} b;Nil$ (we cannot derive $P \xrightarrow{\langle a \rangle}_{\frac{1}{2}} \Omega$ because Ω is not stable). Thus, $\mathcal{A}(P,\langle a \rangle) = \emptyset$ (as in the nonprobabilistic case, this process is *must equivalent* to $a;\Omega$). If the sum of these probabilities is equal to 1 then we compute the acceptance sets as in the nonprobabilistic setting (an example is shown in Figure 4).

Let us note that, as in the *must-reactive* case, this alternative characterization considers *unfair divergences* caused by unguarded recursive definitions in internal choices. For example, let $P = recX.(a;Nil) \oplus_{\frac{1}{3}} X$. We have

$$P \rightarrow\!\!\!\!\rightarrow^*_{\frac{1}{3}} a;Nil,\ P \rightarrow\!\!\!\!\rightarrow^*_{\frac{2}{9}} a;Nil,\ P \rightarrow\!\!\!\!\rightarrow^*_{\frac{4}{27}} a;Nil\ \ldots$$

and the probabilities associated with these transitions sum up to 1 since we have $\frac{1}{3}\sum_{i=0}^{\infty}(1-\frac{1}{3})^i = 1$. So, $\mathcal{A}(P,\epsilon) = \{\{a\}\}$ and then P is equivalent to $a;Nil$.

Next, we define an equivalence relation which coincides with $\approx^{\mathcal{G}}_{\text{must}}$.

Definition 13. Let $\mathcal{A}$ and $\mathcal{B}$ be acceptance sets. We write $\mathcal{A} \simeq \mathcal{B}$ if for every $A \in \mathcal{A}$ there exists $B \in \mathcal{B}$ such that $B \subseteq A$, and for every $B \in \mathcal{B}$ there exists $A \in \mathcal{A}$ such that $A \subseteq B$.

Let P and P' be processes. We write $P =_{\text{must}} P'$ if for all $s \in Act^*$ we have $\mathcal{A}(P,s) \simeq \mathcal{A}(P',s)$. □

Theorem 7. Let P and P' be processes. We have $P \approx^{\mathcal{G}}_{\text{must}} P'$ iff $P =_{\text{must}} P'$.

5 The Limited Generative Model

In the limited generative model we consider deterministic[2] tests and such that there is an equitable distribution of probabilities among the offered actions at

[1] Let us remark that $\mathcal{A}(P,s) = \emptyset$ is not the same as $\mathcal{A}(P,s) = \{\emptyset\}$. The former corresponds to a divergent process (i.e. equivalent to Ω) while the latter corresponds to a deadlocked process (i.e. equivalent to Nil).

[2] This is not a real constraint because, by Lemma 2, nondeterministic tests do not increase the distinguishing power.

a given time. This means that *several buttons can be simultaneously pressed but all of them with the same strength*.

Definition 14. The set of *limited generative tests*, denoted by $\mathcal{LG}$, is defined by the BNF expression $T = Nil \mid \sum_{i=1}^{n} [\frac{1}{n}] b_i; T_i$, where $b_i \in Act \cup \omega$ are different actions. □

The *may* interpretation in this model coincides with that of the reactive model. In the case of the *must* interpretation, we have that $\approx^{\mathcal{LG}}_{\text{must}}$ coincides with $\approx^{\mathcal{G}}_{\text{must}}$. Intuitively, if a process passes a test with probability 1 it does not matter the possible distribution of probabilities among the offered actions.

Theorem 8. Let P and P' be processes. We have $P \approx^{\mathcal{LG}}_{\text{may}} P'$ iff $P \approx^{\mathcal{G}}_{\text{may}} P'$. We have $P \approx^{\mathcal{LG}}_{\text{must}} P'$ iff $P \approx^{\mathcal{G}}_{\text{must}} P'$.

In [29], an alternative characterization based on limited probabilistic barbs was given for the limited generative testing equivalence ($\approx^{\mathcal{LG}}$), for a probabilistic version of LOTOS. These results can be adapted to our framework. First, we precisely define the new set of tests.

Definition 15. The set of *limited probabilistic barbs*, denoted by $\mathcal{LPB}$, is defined by the BNF expression: $T = \omega \mid \sum_{i=1}^{n} [\frac{1}{n}] b_i; R_i$, where $b_i \in Act \cup \omega$ are different actions, and $R_i = Nil$ for all $1 \leq i < n$ while $R_n = T$.

We write $P \approx^{\mathcal{LGB}} P'$ if for all $T \in \mathcal{LGB}$ we have $P\, pass_p\, T$ iff $P'\, pass_p\, T$. □

That is, a limited probabilistic barb is a limited probabilistic test such that at most only one of the offered actions at each time has a continuation (the rest of actions are prefixing the process *Nil*). This family of tests has the same distinguishing power as the whole family, as it is stated in the next result.

Theorem 9. Let P and P' be processes. We have $P \approx^{\mathcal{LG}} P'$ iff $P \approx^{\mathcal{LGB}} P'$.

But this alternative characterization is not suitable to be extended to a denotational semantics. A fully abstract denotational semantics could be given for this equivalence by modifying the probabilistic acceptance trees model, but the definitions are too involved since several states must be joined into one. For example, we have $(a +_{\frac{1}{4}} b) \oplus_{\frac{1}{2}} (a +_{\frac{3}{4}} b) \approx^{\mathcal{LG}} a +_{\frac{1}{2}} b$ (*Nil*'s have been omitted), while these two processes are not equivalent in the generative model (the test $T = (a; \omega) +_{\frac{1}{3}} (b; Nil)$ distinguishes them).

6 Conclusion

In this paper we have proposed a number of testing equivalences on PPA. Since we use a testing framework, the definitions of these equivalences are very natural and easy to understand. Testing equivalences are intuitive but they are not suitable for a practical use since they are based on the behavior of a process with respect to an infinite set of tests. So, we have given alternative characterizations and fully abstract denotational semantics. The hierarchy we have settled brings up both useful information about different probabilistic testing equivalences and relationships between denotational models.

References

1. Baeten, J.C.M., Weijland, W.P.: Process Algebra. In: Cambridge Tracts in Computer Science 18, Cambridge University Press, Cambridge (1990)
2. Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.): Handbook of Process Algebra. North-Holland, Amsterdam (2001)
3. Cazorla, D., Cuartero, F., Valero, V., Pelayo, F.L., Pardo, J.J.: Algebraic theory of probabilistic and non-deterministic processes. Journal of Logic and Algebraic Programming 55(1–2), 57–103 (2003)
4. Cheung, L., Stoelinga, M., Vaandrager, F.: A testing scenario for probabilistic processes. Journal of the ACM 54(6), Article 29 (2007)
5. Christoff, I.: Testing equivalences and fully abstract models for probabilistic processes. In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR 1990. LNCS, vol. 458, pp. 126–140. Springer, Heidelberg (1990)
6. Cleaveland, R., Dayar, Z., Smolka, S.A., Yuen, S.: Testing preorders for probabilistic processes. Information and Computation 154(2), 93–148 (1999)
7. de Nicola, R., Hennessy, M.C.B.: Testing equivalences for processes. Theoretical Computer Science 34, 83–133 (1984)
8. Deng, Y., van Glabbeek, R., Hennessy, M., Morgan, C., Zhang, C.: Characterising testing preorders for finite probabilistic processes. In: 22nd Annual IEEE Symposium on Logic in Computer Science, LICS 2007, pp. 313–325. IEEE Computer Society Press, Los Alamitos (2007)
9. van Glabbeek, R.: The linear time-branching time spectrum II. In: Best, E. (ed.) CONCUR 1993. LNCS, vol. 715, pp. 66–81. Springer, Heidelberg (1993)
10. van Glabbeek, R.: The linear time-branching time spectrum I. The semantics of concrete, sequential processes. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of process algebra, ch. 1, North-Holland, Amsterdam (2001)
11. van Glabbeek, R., Smolka, S.A., Steffen, B.: Reactive, generative and stratified models of probabilistic processes. Information and Computation 121(1), 59–80 (1995)
12. Gregorio, C., Núñez, M.: Denotational semantics for probabilistic refusal testing. In: PROBMIV 1998. Electronic Notes in Theoretical Computer Science, vol. 22. Elsevier, Amsterdam (1999)
13. Hennessy, M.: Acceptance trees. Journal of the ACM 32(4), 896–928 (1985)
14. Hennessy, M.: An algebraic theory of fair asynchronous communicating processes. Theoretical Computer Science 49, 121–143 (1987)
15. Hennessy, M.: Algebraic Theory of Processes. MIT Press, Cambridge (1988)
16. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs (1985)
17. Huynh, D.T., Tian, L.: On some equivalence relations for probabilistic processes. Fundamenta Informaticae 17, 211–234 (1992)
18. Jonsson, B., Yi, W.: Compositional testing preorders for probabilistic processes. In: 10th IEEE Symposium on Logic In Computer Science, pp. 431–443. IEEE Computer Society Press, Los Alamitos (1995)
19. Jonsson, B., Yi, W.: Testing preorders for probabilistic processes can be characterized by simulations. Theoretical Computer Science 282, 33–51 (2002)
20. Jou, C.-C., Smolka, S.A.: Equivalences, congruences and complete axiomatizations for probabilistic processes. In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR 1990. LNCS, vol. 458, pp. 367–383. Springer, Heidelberg (1990)

21. Kwiatkowska, M., Norman, G.J.: A testing equivalence for reactive probabilistic processes. In: EXPRESS 1998. Electronic Notes in Theoretical Computer Science, vol. 16. Elsevier, Amsterdam (1998)
22. Larsen, K., Skou, A.: Bisimulation through probabilistic testing. Information and Computation 94(1), 1–28 (1991)
23. López, N., Núñez, M.: An overview of probabilistic process algebras and their equivalences. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 89–123. Springer, Heidelberg (2004)
24. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
25. Milner, R.: Communicating and Mobile Systems: the π-Calculus. Cambridge University Press, Cambridge (1999)
26. Narayan Kumar, K., Cleaveland, R., Smolka, S.A.: Infinite probabilistic and nonprobabilistic testing. In: Arvind, V., Ramanujam, R. (eds.) FST TCS 1998. LNCS, vol. 1530, pp. 209–220. Springer, Heidelberg (1998)
27. Natarajan, V., Cleaveland, R.: Divergence and fair testing. In: Fülöp, Z., Gecseg, F. (eds.) ICALP 1995. LNCS, vol. 944, pp. 648–659. Springer, Heidelberg (1995)
28. Núñez, M.: Algebraic theory of probabilistic processes. Journal of Logic and Algebraic Programming 56(1–2), 117–177 (2003)
29. Núñez, M., de Frutos, D.: Testing semantics for probabilistic LOTOS. In: 8th IFIP WG6.1 Int. Conf. on Formal Description Techniques, FORTE 1995, pp. 365–380. Chapman & Hall, Boca Raton (1995)
30. Núñez, M., de Frutos, D., Llana, L.: Acceptance trees for probabilistic processes. In: Lee, I., Smolka, S.A. (eds.) CONCUR 1995. LNCS, vol. 962, pp. 249–263. Springer, Heidelberg (1995)
31. Núñez, M., Rupérez, D.: Fair testing through probabilistic testing. In: Formal Description Techniques for Distributed Systems and Communication Protocols (XII), and Protocol Specification, Testing, and Verification (XIX), pp. 135–150. Kluwer Academic Publishers, Dordrecht (1999)
32. Rensink, A., Vogler, W.: Fair testing. Information and Computation 205(2), 125–198 (2007)
33. Segala, R.: Testing probabilistic automata. In: Sassone, V., Montanari, U. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 299–314. Springer, Heidelberg (1996)
34. Stoelinga, M., Vaandrager, F.: A testing scenario for probabilistic automata. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 464–477. Springer, Heidelberg (2003)
35. Yi, W., Larsen, K.G.: Testing probabilistic and nondeterministic processes. In: 12th IFIP/WG6.1 Int. Symposium on Protocol Specification, Testing and Verification, PSTV 1992, pp. 47–61. North-Holland, Amsterdam (1992)