Chapter 1

FAIR TESTING THROUGH PROBABILISTIC TESTING

Manuel Núñez and David Rupérez
Dept. de Sistemas Informáticos y Programación
Universidad Complutense de Madrid, Spain
{manuelnu,ruperezd}@eucmax.sim.ucm.es.

Abstract In this paper we define a probabilistic testing semantics which can be used to alternatively characterize *fair* testing. The key idea is to define a probabilistic semantics in such a way that two non-probabilistic processes are fair equivalent iff any *probabilistic version* of both processes are equivalent in our probabilistic testing semantics. In order to get this result we define a simple probabilistic *must* semantics by saying that a probabilistic process *must* pass a test iff the probability with which the process passes the test equals 1. Finally, we present an algorithm for deciding whether the probability with which a finite-state process passes a finite-state test equals 1. Alternatively, this algorithm can be used for computing whether a finite-state process *fairly* passes a finite-state test.

Keywords: Testing semantics, fair testing, probabilistic processes.

1. INTRODUCTION

Formal models of concurrency have been proved to be very useful to properly specify concurrent and distributed systems. In order to verify the properties held by these specifications some kind of semantics must be given to these syntactic processes. Several semantic models have been proposed in the literature, for example, failures model, bisimulations, testing semantics, axiomatic semantics, etc.

In our opinion, among the previously cited semantic models, the closest to *reality* is testing. Testing semantics was defined and thoroughly studied in [de Nicola and Hennessy, 1984, Hennessy, 1988], and this will be the approach considered here. In their model tests are just processes

but they may contain a special action indicating the *successful* termination of the testing procedure. Then, two testing interpretations are defined: a process *may* pass a test if the composition of process and test has a successful computation; a process *must* pass a test if every computation of the composition is successful. From these interpretations three testing semantics can be defined: a process is greater in the may (resp. must) sense than another one if every test that the latter process may (resp. must) pass is also may (resp. must) passed by the former. Besides, a may–must interpretation can be defined, but given the fact that for non-divergent processes must equivalence and may–must equivalence coincide, this interpretation does not add too much information. While may semantics identifies too much processes, because it takes into account only the possible traces of processes, must semantics distinguishes enough processes to be considered a useful relation: it is stronger than may semantics, weaker than weak bisimulation, and equivalent to failures semantics (actually, failures semantics and bisimulation have been characterized in terms of tests [Milner, 1981, Brookes et al., 1984]). Although must semantics is very suitable for giving meaning to processes, further research has shown some weakness in the original definition. For example, there are problems dealing with *conformance* and *fairness* (see [Tretmans, 1996, Francez, 1986] for good overviews on the subjects).

Intuitively, under a conformance assumption, a process is considered *better* than another (or implements another) if the former can execute more sequences of actions than the latter, while presenting less nondeterminism. For example, using a CCS notation, $a + b$ should be better than a, since the former process can execute both a and b deterministically, while the latter cannot execute b. But this does not hold in the must testing semantics: these processes are not related at all. The extension of [Hennessy, 1988] to deal with conformance has been treated in [de Frutos-Escrig et al., 1997, de Frutos-Escrig et al., 1998].

The lack of *fairness* in the *classical testing theory* is the main purpose of this paper. A usual justification for fairness is that *unfair* behavior is unlikely. We will illustrate this problem by showing two simple examples. Consider the processes P_0 and P_1 depicted in Figure 1.1. In the must testing semantics these processes are not equivalent: P_0 may execute an infinite sequence of internal actions τ, so it is considered to be divergent (i.e. somehow *catastrophic*). On the contrary, we can observe that the action a is offered infinitely often (actually, it is offered continuously), so it is reasonable to think that this action should be finally executed, or in other words, that P_0 *should* pass T_0 (this insight was presented in [Koomen, 1985] and treated formally in [Baeten et al., 1987]). Thus, P_0 and P_1 should be considered equivalent under a *fairness* assumption.

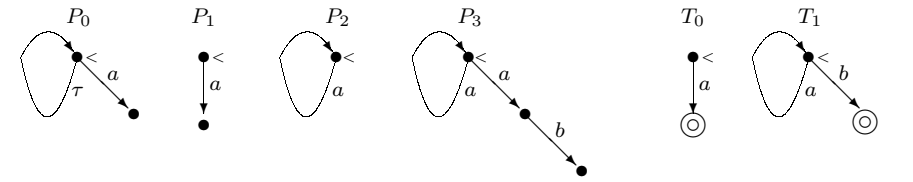

Figure 1.1 Examples of processes and tests.

Consider now the processes P_2 and P_3. Even though P_3 could execute the sequence $a\,;b$, this sequence might not be executed. But considering a fair interpretation, P_3 presents the capability of executing the sequence $a\,;b$, and so, if this sequence is infinitely often offered, P_3 should be able to execute it, while P_2 is not. For instance, P_3 *should* pass T_1 while P_2 should not.

In the previous decade, there were several proposals for including different kinds of fairness into process algebras (e.g. [Hennessy, 1987, Bergstra et al., 1987, Costa and Stirling, 1987, Brinksma, 1988]). Almost simultaneously, two groups of authors [Natarajan and Cleaveland, 1995, Brinksma et al., 1995] have come up with what is called *fair testing*. They proposed two equivalent testing semantics with the property of abstracting from certain divergences, like weak bisimulation, in contrast to the classical must testing. The idea is to modify the classical definition in order to consider as successful those computations such that, under a fairness assumption, success can always be reached. Both papers present alternative characterizations of the new fair testing semantics. In a later work [Brinksma et al., 1996], the framework described in [Brinksma et al., 1995] is extended with a set of sound axioms and more examples showing the usefulness of this fair testing semantics.

In this paper we probabilistically characterize fair testing by showing the connection between probabilities and fairness. This point has been motivated by Koomen's Fair Abstraction Rule (KFAR) [Koomen, 1985], that has been intuitively explained with probabilistic arguments in the sense of "No matter how small the probability of success, if you try often enough you will eventually succeed". Taking into account that fair testing is fully abstract with respect to this rule, it must be possible to justify fair testing in probabilistic terms. This is exactly what we will do in the present paper. As the probabilistic intuitions motivating KFAR have never been formalized before, this paper consolidates these intuitions.

Let us consider the previous examples. When composing the process P_3 and the test T_1 we would like that the process must pass the test

because the sequence $a\,;b$ is available an infinite number of times. Let us add a probability for choosing between both actions a in the process P_3. So, let π be the probability of executing the left hand side a transition (leading to the initial state), and let $1-\pi$ the probability associated with the other a transition (leading to a state where only b is available). It is clear that the probability of executing the trace $\langle a, b\rangle$, by the composition of process and test, equals $1-\pi$; the probability of executing $\langle a, a, b\rangle$ equals $\pi \cdot (1-\pi)$, etcetera; in general, the probability of executing the trace $\langle a^n, b\rangle$ equals $\pi^{n-1} \cdot (1-\pi)$. These traces lead to a successful state of the test, and the sum of these probabilities is given by:

$$\sum_{i=1}^{\infty}(\pi^{i-1} \cdot (1-\pi)) = (1-\pi) \cdot \sum_{i=1}^{\infty}\pi^{i-1} = (1-\pi) \cdot \frac{1}{1-\pi} = 1$$

This example illustrates our technique. We consider non-probabilistic processes and non-probabilistic tests. Then, we assign arbitrary (randomly) probabilities within every choice of the process and the test, and if the probability with which the process passes the test equals 1, then we say that the test is passed, otherwise we say that it is not passed. This final answer is returned for the composition of the (non-probabilistic) process and test. If we consider finite-state processes and tests, as in the previous example, the probability of passing the test can be computed by solving a linear equations system, which is built from the states of the interaction system. Even though there could be more efficient algorithms for checking fair testing (by directly checking the induced graph) this algorithm can be used in more general probabilistic testing semantics (e.g. [Christoff, 1990, Cleaveland et al., 1992, Núñez and de Frutos, 1995]).

It could be thought that we add too much complexity to afford the definition of this semantics but we claim that this is not the case. The definitions of processes, tests, interaction systems, and probabilistic testing are standard. Moreover, tests could be described without any probabilistic information because in our setting probabilities do not add any discriminatory power to tests. Nevertheless, in order to keep compatibility with the classical framework (both in the probabilistic and non-probabilistic cases) we consider that tests are *like* processes (i.e. probabilistic). Note that in general probabilistic frameworks, probabilities add distinguishing power to tests (see, for example, [Christoff, 1990, Núñez and de Frutos, 1995] where different probabilistic semantics are defined according to different sets of tests).

There are also previous papers relating probabilistic and classical testing. For instance, [Cleaveland et al., 1992] compares in some settings satisfaction with probability 1 to ordinary must testing. In [Núñez and

de Frutos, 1995], failing a test with probability 0 is related to the classical must testing. In [Kumar et al., 1998] some notions of non-probabilistic and probabilistic testing are related. Regarding fair testing, our proposal extends the last one given the fact that we deal with infinite and divergent processes and our characterization of fair testing works in both ways, that is, from our probabilistic must testing semantics to fair testing and back.

The rest of the paper is structured as follows. In Section 2 we present the probabilistic model: probabilistic processes and tests, interaction systems and the definition of the probabilistic must testing semantics. In this semantics we consider that a process *must* pass a test if the test is passed with a probability 1. In Section 3 we probabilistically characterize fair testing. We compare fair satisfaction to satisfaction with probability 1. We recover the *fair* testing passing of tests (and so the fair testing preorder) from our probabilistic must semantics. In Section 4, we give an algorithm which effectively computes the probability with which a finite-state process passes a finite-state test (or equivalently, it decides whether a process *fairly* passes a test). Finally, in Section 5 we present our conclusions and some lines for future work.

The results of this paper are an extension of those appearing in the second author's Master Thesis [Rupérez, 1997], where only finite-state processes were considered.

2. THE PROBABILISTIC MODEL

In this section, we present the probabilistic model. Probabilistic processes will be modeled by using *probabilistic labeled transitions systems*, which are an extension of usual labeled transition systems by labeling the arcs by an action and a probability. Later, we will define the notion of *probabilistic test*. Then, we will give a notion of a process passing a test with a certain probability, as well as the definition of the *probabilistic must testing semantics*.

We suppose that we are working with a fixed and finite set of actions Act, and we assume the existence of a special action $\tau \notin \text{Act}$, which represents an internal action. We denote by Act_τ the set $\text{Act} \cup \{\tau\}$; a will denote a generic action in Act, while e will denote a generic action in Act_τ.

Definition 1 A *probabilistic process* P is a triple (S, μ, s_0) where S is a (possibly infinite) *set of states*, ranged over $s_0, s_1, \ldots$; $s_0 \in S$ is the *initial state*, and $\mu : S \times \text{Act}_\tau \times S \longrightarrow [0, 1]$ is the *transition probability distribution function*, which satisfies that for all $s \in S$, $\sum_{e,s'} \mu(s, e, s')$ equals 1, or 0 (blocked state). □

We will use the following conventions:

- $s \xrightarrow{e}_p s'$ stands for $\mu(s, e, s') = p$.
- $s \xrightarrow{e} s'$ stands for $\mu(s, e, s') > 0$.
- $s \longrightarrow$ if $\exists e \in \mathrm{Act}_\tau, s' \in S : \ s \xrightarrow{e} s'$.
- $s \not\longrightarrow$ if $\nexists e \in \mathrm{Act}_\tau, s' \in S : \ s \xrightarrow{e} s'$.
- $\mu(s, e) = \sum_{s'} \mu(s, e, s')$.

In the following definition, we present some process algebraic operators. These operators will be used in the last section of this paper. Specifically, a prefix operator $a\,;P$, a CCS-like probabilistic choice operator $P +_\pi Q$ and a stop operator.

Definition 2 Let $P = (S, \mu, s_0)$ be a probabilistic process and $e_0 \in \mathrm{Act}_\tau$. We define the *prefixing* of P by e_0, denoted by $e_0\,;P$, as the probabilistic process $P' = (S \cup \{s'_0\}, \mu', s'_0)$ where $s'_0 \notin S$, and the function μ' is defined as

$$\mu'(s, e, s') = \begin{cases} \mu(s, e, s') & \text{if } s, s' \in S \\ 1 & \text{if } s = s'_0 \ \wedge \ s' = s_0 \ \wedge \ e = e_0 \\ 0 & \text{otherwise} \end{cases}$$

Given n probabilistic processes $P_i = (S_i, \mu_i, s_i)$ such that if $i \neq j$ then $S_i \cap S_j = \emptyset$, and $p_1, p_2, \ldots, p_n > 0$ such that $\sum p_i = 1$, we define the *probabilistic choice* of P_i with associated probabilities p_i, denoted by $\sum_{i=1}^{n}[p_i]P_i$, as the process $P = (\bigcup S_i \cup \{s_0\}, \mu, s_0)$ where $s_0 \notin \bigcup S_i$, and the function μ is defined as:

$$\mu(s, e, s') = \begin{cases} \mu_i(s, e, s') & \text{if } s, s' \in S_i \\ \frac{p_i \cdot \mu_i(s_i, e, s')}{R} & \text{if } s = s_0 \ \wedge \ s' \in S_i \ \wedge \ R \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

where $R = \sum_{i=1}^{n} p_i \cdot \delta_i$, δ_i being equal to 1 if $s_i \longrightarrow$, and 0 otherwise.

We consider stop as the probabilistic process without transitions, that is, the probabilistic process $(\{s\}, \mu, s)$ such that $\mu(s, e, s) = 0$ for any $e \in \mathrm{Act}_\tau$. □

In the definition of $\sum[p_i]P_i$ we use the factor R in order to avoid *sub-stochastic* processes in the case that some of the P_i's are deadlocked (i.e. equivalent to stop). Also note that we keep the transitions from the initial states of the original processes for keeping the possible *loops* of the former processes.

A probabilistic test is just a probabilistic process (*plts*) extended with a set of successful states. As in [Cleaveland et al., 1992, Yuen et al., 1994] we slightly differ from the standard formulation of tests given in [Hennessy, 1988] because successful states, instead of a distinguished acceptance event, are used to denote the (successful) passing of tests. Anyway, these successful states can be easily simulated by adding a new action *ok* to the set of actions.

Definition 3 A *probabilistic test* is a 4-tuple (S, μ, s_0, Suc) such that (S, μ, s_0) is a probabilistic process and $Suc \subseteq S$ is the set of *successful states*.

We consider *ok* as the probabilistic test without transitions, and with a unique successful state, that is, the probabilistic test $(\{t\}, \mu, t, \{t\})$ such that $\mu(t, e, t) = 0$ for any $e \in \mathrm{Act}_\tau$. □

As we commented in the introduction of this paper, in order to define our probabilistic must semantics the probabilities appearing in the tests are completely irrelevant, so we could be working with non-probabilistic tests. We have preferred to keep probabilities in tests in order to follow the classical testing theory where processes and tests are (almost) the same. In the rest of this section, we usually omit the adjective "probabilistic" when referring to probabilistic processes (or probabilistic tests) and no confusion arises.

Now we will follow the process used, for example, in [Cleaveland et al., 1996]. First, we define the total probability of the successful computations of a test. Afterwards, we define the composition of a process and a test, resulting into a test. Then, in order to compute the probability with which a process passes a test we just study the corresponding composition as a test.

Definition 4 Let $T = (S, \mu, s_0, Suc)$ be a test, and let $s_1 \in S$. We call *computations from* s_1 the (possibly infinite) sequences of transitions:

$$C = s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} \cdots s_n \xrightarrow{e_n} \cdots$$

We say that the computation C is *successful* if there exists n such that $s_n \in Suc$ and such that $\forall\ 1 \leq i < n :\ s_i \notin Suc$. We will denote by $\tilde{C}_{s_1}$ the *set* of successful computations from s_1.

We say that the state $s_i \in S$ is *non-successful* in C, denoted by $ns(s_i, C)$, if there does not exist any successful computation extending the partial computation $C_1 = s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} \cdots s_i$. In this case, we say that C_1 is a *Non-Successful Partial* computation, in short NSP, if we have that $ns(s_i, C)$ holds, $ns(s_j, C)$ does not hold for any $j < i$, and $s_1, s_2, \ldots, s_i \notin Suc$.

Successful and NSP computations are defined as the *maximal* computations. For finite computations we say that $length(C) = n - 1$. Given a finite computation C, we define the *probability of executing* C as:

$$Pr(C) = \prod_{i=1}^{length(C)} \mu(s_i, e_i, s_{i+1})$$

Finally, we define the probability of *reaching* a successful state from s_1, denoted by $Pr(T, s_1)$, as $\sum_{C \in \tilde{C}_{s_1}} Pr(C)$. □

Note that what we call *maximal* computations are always finite. Let us remark that finite computations have positive probability, while infinite computations may have zero probability. We will see later that the composition of processes and tests will produce tests such that no further action can be executed from a successful state. Thus, the condition imposing that no previous state in the computation belongs to *Suc* in the definition of successful computations would be redundant.

Next we define the way a process and a test interact by using *interaction systems*. Intuitively, an interaction system will have a transition labeled by a visible action from a state iff the corresponding states of process and test have a transition labeled by this visible action.

Definition 5 Let $P = (S, \mu_P, s_0)$ be a process and $T = (S', \mu_T, s'_0, Suc)$ be a test. The *interaction system* for P and T, denoted $P \mid T$, is a test $I = (S \times S', \mu_I, (s_0, s'_0), S \times Suc)$. In order to define μ_I, we need a *normalization function* ν_I. This function computes the sum of the probabilities associated with the actions that the interaction system can execute, and is formally defined as:

$$\nu_I(s, s') = \sum_{a \in \mathrm{Act}} (\mu_P(s, a) \cdot \mu_T(s', a)) \; + \mu_P(s, \tau) \; + \; \mu_T(s', \tau)$$

The function μ_I is defined as:

$$\mu_I((s_1, s'_1), e, (s_2, s'_2)) = \begin{cases} \frac{\mu_P(s_1, e, s_2)}{\nu_I(s_1, s'_1)} & \text{if} \quad s'_1 = s'_2 \; \wedge \; e = \tau \\ & \quad \wedge \; s'_1 \notin Suc \\ \frac{\mu_T(s'_1, e, s'_2)}{\nu_I(s_1, s'_1)} & \text{if} \quad s_1 = s_2 \; \wedge \; e = \tau \\ & \quad \wedge \; s'_1 \notin Suc \\ \frac{\mu_P(s_1, e, s_2) \cdot \mu_T(s'_1, e, s'_2)}{\nu_I(s_1, s'_1)} & \text{if} \quad e \neq \tau \; \wedge \; s'_1 \notin Suc \\ 0 & \text{otherwise} \end{cases}$$

In the first three cases of the definition of μ_I we assume $\nu_I(s_1, s'_1) > 0$ (otherwise the result is 0). We say that P *passes* T with a probability p

if $Pr(I, (s_0, s'_0)) = p$, that is, if the probability of reaching a successful state from the initial state (s_0, s'_0) equals p. □

A normalization function ν_I is 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. [Christoff, 1990, Cleaveland et al., 1992, Núñez and de Frutos, 1995, D'Argenio et al., 1999, Núñez, 1999]). Given that we do not allow the synchronization in τ, our normalization factor is similar to that in [Núñez and de Frutos, 1995]. Let us comment that our definition of interaction systems is slightly different from that in classical testing. Following [Natarajan and Cleaveland, 1995] we avoid useless computations after the interaction system has reached a successful state (that is why the clause $s_1 \notin Suc$ appears in the definition of μ_I). Thus, once the test reaches a successful state, no more transitions are possible. Note that this decision does not change the rest of the semantic treatment. Finally, we define the desired relation between processes.

Definition 6 Let $P = (S, \mu_P, s_0)$ be a process and $T = (S', \mu_T, s'_0, Suc)$ be a test. We say P must T if $Pr(I, (s_0, s'_0)) = 1$, where $I = P \mid T$. Moreover, given two processes P, P' we write $P \sqsubseteq_{\text{must}} P'$ if for any test T we have P must T implies P' must T. □

Next we give two results about the computations of interaction systems. They will be used in order to prove that the previously defined testing semantics characterizes, in a way that we will specify later, fair testing. Whereas the results hold for finite-state systems, they are not correct for general infinite-state systems. In the latter case we have to add two *regularity* conditions which will ensure that the considered infinite-state systems do not have *strange* behaviors.

Definition 7 Let P be a process, T be a test, and consider $P \mid T = (S, \mu, s_0, Suc)$. We say the composition of P and T is *regular* if the two following conditions hold:

- There exists a non-zero probability π_m such that for any $s, s' \in S$, $e \in \text{Act}$ we have $\mu(s, e, s') > 0$ implies $\mu(s, e, s') > \pi_m$.
- Let us consider the digraph induced by S and μ where probabilities are discarded. There exists a value $M > 0$, such that for any $s \in S$, if there exists a path from s to a successful state then there exists $s' \in Suc$ such that a path from s to s' has length less than M.

□

From now on we will assume that the considered compositions are always regular. Let us note that finite-state systems are trivially regular.

Also note that the first condition implies that systems are finite branching (any state has, at most, $\frac{1}{\pi_m}$ transitions). Finally, let us remark that regularity conditions could have been defined on probabilistic transitions systems, instead of using our formulation on interaction systems.

The first condition indicates that the probabilities appearing in interaction systems have a lower bound, or equivalently that probabilities less than one have an upper bound. This condition is necessary in order to avoid *strange* successions of probabilities. By strange we mean an infinite succession of probabilities such that all of its members are different from one, but the product of the components is different from zero. For example, in the succession $a_n = (\frac{1}{2})^{\frac{1}{n!}}$ we have $\prod_{n=0}^{\infty} a_n = (\frac{1}{2})^e$. This is formalized in the following lemma which states that if an infinite computation is executed with a probability greater than zero then there exists a point in the computation such that from this point on all the probabilities associated with the transitions are equal to 1, and so, a prefix of the computation is already a NSP computation. The proof is not difficult and follows from the existence of the previously commented lower bound.

Lemma 1 Let $P = (S, \mu_P, s_0)$ be a process, $T = (S', \mu_T, s'_0)$ be a test, and $I = P \mid T$ be the corresponding interaction system. If I is regular and there exists an infinite computation of I

$$C = (s_0, s'_0) \xrightarrow{e_1} (s_1, s'_1) \xrightarrow{e_2} (s_2, s'_2) \cdots \xrightarrow{e_n} (s_n, s'_n) \cdots \qquad [Pr(C) > 0]$$

then there exists n_0 such that $\mu((s_{n-1}, s'_{n-1}), e_n, (s_n, s'_n)) = 1$ for all $n > n_0$.

The second regularity condition is necessary in order to ensure that we cannot get a set of infinite computations having positive measure, and such that they do not have NSP prefixes. Note that we need this condition because the first regularity condition ensures that there will not be what we called strange computations (previous lemma) but dealing with infinite systems we can define a set of computations behaving like one of this *undesirable* computations that we want to avoid. In particular, this extra condition allows us to formulate the following result. The proof is highly technical and it is based on the idea that from prefixes of non-NSP computations we can always find a successful state whose associated probability is not arbitrarily small.

Lemma 2 Let P be a process and T be a test T such that $P|T$ is regular. We have that the sum of the probabilities of the *maximal* computations (successful and NSP computations) of $P \mid T$ is equal to 1.

3. COMPARING FAIR AND PROBABILISTIC TESTING

In this section we probabilistically characterize the fair testing semantics given in [Natarajan and Cleaveland, 1995] by comparing fair satisfaction to satisfaction with probability 1. First, we formally define the concept *made from*, that is, we obtain a probabilistic process by putting arbitrary nonzero probabilities on the transitions of a non-probabilistic process. Then, we show that the specific probabilities associated with a process do not change its probabilistic must semantics. We will assume again that all the interaction systems appearing in this section are regular.

Definition 8 A *non-probabilistic process* P is a triple $(S, \rightarrow, s_0)$ where S is a *set of states*, s_0 is the *initial state*, and $\rightarrow \subseteq S \times \mathrm{Act}_\tau \times S$ is the *transition relation*. We say that the probabilistic process $P_\pi = (S', \mu, s'_0)$ is *made from* P if there exists a bijection $f : S \longrightarrow S'$ s.t. $f(s_0) = s'_0$ and $\forall s_1, s_2 \in S, e \in \mathrm{Act}_\tau : (s_1, e, s_2) \in \rightarrow$ iff $\mu(f(s_1), e, f(s_2)) > 0$. □

Let us note that P_π is a probabilistic process, so for any $s' \in S'$ we have $\sum_{e,s''} \mu(s', e, s'')$ equals 1, or 0 if no transitions are possible. The key of the proof of the following results relays on the equivalence: $P_1 \mid T$ has a NSP-computation iff $P_2 \mid T$ has one.

Proposition 1 Let P_1, P_2 be two probabilistic processes made from a non-probabilistic process P and let T be a probabilistic test. We have P_1 must T iff P_2 must T.

Proposition 2 Let P be a probabilistic process and T_1, T_2 be two probabilistic tests made from a non-probabilistic test T. We have P must T_1 iff P must T_2.

These results allow us to define a testing semantics for non-probabilistic processes, just by assigning whatever positive probabilities to the transitions of the labeled transition system (with the restriction that the sum of these probabilities equals one for any state having transitions) and getting its probabilistic must semantics, which by Propositions 1 and 2 does not depend on the assigned probabilities. That is, in order to decide if a process passes a test (both non-probabilistic), we assign positive probabilities to process and test, and check whether the associated probabilistic process passes the test. This reasoning leads us to the next definition.

Definition 9 Let P, T be a process and a test, both non-probabilistic. Let P_{π_1} be a probabilistic process made from P, and let T_{π_2} be a probabilistic test made from T. We write P must* T iff P_{π_1} must T_{π_2}. □

Now we will relate our testing equivalence with *fair testing* as described in [Natarajan and Cleaveland, 1995].

Definition 10 Let P, T be a process and a test, both non-probabilistic, and let us consider $I = P \mid T$. We say that a state of I is *potentially successful* if there exists a path leading to a successful state. A computation is *potentially successful* if all the states along the computation are potentially successful. A test is passed by a process, denoted by P must$_{NC}$ T, if all the computations of $P \mid T$ are potentially successful. Given two processes P, P' we write $P \sqsubseteq_{\text{must}_{NC}} P'$ if for all the tests we have P must$_{NC}$ T implies P' must$_{NC}$ T □

Now, we can formally give the result we were looking for: The recovering of fair testing from our *non-probabilistic* must semantics.

Theorem 1 Let P, T be a process and a test, both non-probabilistic. We have P must* T iff P must$_{NC}$ T.
Proof: For the right to left implication, suppose P must* T does not hold. Then, by Lemma 2, there exists a maximal computation C of $P \mid T$ being NSP. If C is NSP then every computation that extends C is not potentially successful, so P must$_{NC}$ T does not hold.

For the left to right implication, suppose P must* T does hold. Then, the successful computations of $P \mid T$ sum up 1. So, again by Lemma 2, there does not exist any NSP computation. Let us consider an arbitrary computation of $P \mid T$, $C = s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} \cdots s_n \xrightarrow{e_n} \cdots$. Suppose that C is not potentially successful and we get a contradiction. If C were not potentially successful then there would exist a state s_i in the computation such that there is no path from s_i leading to a successful state. Then, there exists $j \leq i$ such that $C' = s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} \cdots s_j$ is a NSP computation, which it is a contradiction. □

Corollary 1 Let P, Q be non-probabilistic processes and let P_{π_1}, Q_{π_2} be probabilistic processes made from P and Q, respectively. Then, $P \sqsubseteq_{\text{must}_{NC}} Q$ iff $P_{\pi_1} \sqsubseteq_{\text{must}} Q_{\pi_2}$.

4. COMPUTING MUST TESTING FOR FINITE-STATE PROCESSES

In this section, we restrict our study to finite-state processes. We present an algorithm to decide whether a finite-state process passes a finite-state test in our must semantics, or equivalently if a finite-state process fairly passes a finite-state test. Finally, we show some examples about how this algorithm works.

In order to compute passing of tests we give an algorithm for computing the probability which a process passes a test with. This algo-

rithm is given by solving a linear equations system. Note that we will have a finite number of equations because in this section we work with finite-state processes. Specifically, given a process P and a test T, let $I = P \,|\, T$ be the associated interaction system $I = (S, \mu, s_0, Suc)$, where $S = \{s_0, s_1, \ldots, s_n\}$. We denote by q_i the probability with which P passes T from s_i (i.e. $Pr(I, s_i)$) and by a_{ij} the sum $\sum_e \mu(s_i, e, s_j)$. The probability with which a process passes a test is calculated by solving the next linear equations system consisting of $n + 1$ equations:

$$\forall i = 0..n : \; q_i = \sum_{j=0}^{n} a_{ij} \cdot q_j$$

We also have the following restrictions:

$$q_i = 1 \text{ if } s_i \in Succ \quad \wedge \quad \forall i = 0..n : \; \sum_{j=0}^{n} a_{ij} \in \{0, 1\}$$

which are imposed by the definition of passing tests (meaning that a successful state has been reached) and by the restriction that the sum of the probabilities associated with the transitions from one state is either 1 or 0 (the last case corresponds to a blocked state). Once the system is solved (for instance, by using Gauss method) we can obtain relations over the variables with an infinite number of solutions, that is, relations of the form

$$q_i = \sum_{k \in K \subseteq \{0..n\}} (a'_k \cdot q_k) \; + \; \pi$$

In this case, we assign 0 to all the values in K, and so $q_i = \pi$, given the fact that such a relation indicates that no successful state can be reached from the states in K.

Finally, in order to decide whether a process passes a test in the probabilistic must semantics, it is enough to check if q_0 is 1. Due to Theorem 1 this method can be applied to decide if a non-probabilistic process passes a test in the *fair* sense.

Let us remark that this algorithm can be trivially adapted for checking properties about ongoing execution behavior. For instance to check if a process will execute an action (for example the action a in the process P_0 in Figure 1.1) or a sequence of actions (for example the process P_3 in Figure 1.1), etcetera, with a probability greater than π, just by controlling the value of q_0.

Example 1 Let $P_0 = (\tau \,;\, P_0) +_\pi \; a$ and $P_1 = a$. Consider the test $T_0 = a \,;\, ok$ and the interaction system $I = P_0 \mid T_0$. The associated

system has got 2 equations (I has 2 states), and it is:

$$\begin{cases} q_0 = \pi \cdot q_0 + (1-\pi) \cdot q_1 \\ q_1 = 1 \end{cases}$$

where $I_{s_1} = \text{stop} \mid ok$, so $s_1 \in Suc$. In order to solve the system we just have the equation $q_0 = \pi \cdot q_0 + (1-\pi)$, that is, $q_0 \cdot (1-\pi) = (1-\pi)$. Given that $0 < \pi < 1$, the only solution is $q_0 = 1$. So, P_0 must T_0.

Consider the interaction system $I = P_1 \mid T_0$. The associated system of 2 equations is:

$$\begin{cases} q_0 = 1 \cdot q_1 \\ q_1 = 1 \end{cases}$$

where $I_{s_1} = \text{stop} \mid ok$, so $s_1 \in Suc$. We have $q_0 = 1$, and so P_1 must T_0.

Example 2 Let $P_2 = a\,;P_2$ and $P_3 = (a\,;P_3) +_\pi (a\,;b)$. Consider the test $T_1 = (a\,;T_1) +_{\pi'} (b\,;ok)$ and the interaction system $I = P_3 \mid T_1$. The associated equations system is:

$$\begin{cases} q_0 = \pi \cdot q_0 + (1-\pi) \cdot q_1 \\ q_1 = 1 \cdot q_2 \\ q_2 = 1 \end{cases}$$

where $I_{s_1} = b \mid T_1$ and $I_{s_2} = \text{stop} \mid ok$, so $s_2 \in Suc$. In order to solve the system we get the same equation of the previous example, $q_0 = \pi \cdot q_0 + (1-\pi)$, and so P_3 must T_1.

Consider the interaction system $I = P_2 \mid T_1$. The associated equations system has got only one equation $q_0 = 1 \cdot q_0$, where $q_0 \notin Suc$, so $q_0 = 0$, that is, P_2 must T_1 does not hold.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have studied *fair* testing by using an intermediate probabilistic semantics. We have defined a probabilistic *must* testing semantics, and we have shown that two probabilistic processes are related by our probabilistic semantics iff the corresponding non-probabilistic processes are equivalent under *fair* testing. Finally, we have given an algorithm that computes whether a finite-state probabilistic process *must* pass a finite-state probabilistic test.

As future work we plan to define a denotational characterization of fair testing based on a probabilistic model (e.g. [Núñez et al., 1995, Gregorio and Núñez, 1999]). We also would like to study the extension of *fair* testing to timed models. Following a probabilistic testing approach, we will use a probabilistic-timed model like those given in [Cleaveland et al., 1996, Gregorio and Núñez, 1996, Gregorio et al., 1997]. Finally, it would

be interesting to study how the framework could be adapted to deal with other notions of fairness.

Acknowledgments. This research has been partially supported by the CICYT project TIC 97-0669-C03-01. We would like to thank David de Frutos-Escrig for fruitful discussions on the topic. We also would like to thank a referee of a previous version of this paper for the careful reading and for pointing out a mistake in the characterization.

References

[Baeten et al., 1987] Baeten, J., Bergstra, J., and Klop, J. (1987). On the consistency of Koomen's fair abstraction rule. *Theoretical Computer Science*, 51(1-2):129–176.

[Bergstra et al., 1987] Bergstra, J., J.W.Klop, and Olderog, E. (1987). Failures without chaos: A new process semantics for fair abstraction. In *Formal Description of Programming Concepts III*, pages 77–103. Elsevier.

[Brinksma, 1988] Brinksma, E. (1988). A theory for the derivation of tests. In *Protocol Specification, Testing and Verification VIII*, pages 63–74. North Holland.

[Brinksma et al., 1995] Brinksma, E., Rensink, A., and Vogler, W. (1995). Fair testing. In *CONCUR'95, LNCS 962*, pages 313–327. Springer.

[Brinksma et al., 1996] Brinksma, E., Rensink, A., and Vogler, W. (1996). Applications of fair testing. In *Formal Description Techniques for Distributed Systems and Communication Protocols (IX), and Protocol Specification, Testing, and Verification (XVI)*, pages 145–160. Chapman & Hall.

[Brookes et al., 1984] Brookes, S., Hoare, C., and Roscoe, A. (1984). A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599.

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

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

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

[Costa and Stirling, 1987] Costa, G. and Stirling, C. (1987). Weak and strong fairness in CCS. *Information and Computation*, 87:207–244.

[D'Argenio et al., 1999] D'Argenio, P. R., Hermanns, H., and Katoen, J.-P. (1999). On generative parallel composition. In *PROBMIV'98, Electronics Notes in Theoretical Computer Science 21*. Elsevier.

[de Frutos-Escrig et al., 1997] de Frutos-Escrig, D., Llana-Díaz, L., and Núñez, M. (1997). Friendly testing as a conformance relation. In *Formal Description Techniques for Distributed Systems and Communication Protocols (X), and Protocol Specification, Testing, and Verification (XVII)*, pages 283–298. Chapman & Hall.

[de Frutos-Escrig et al., 1998] de Frutos-Escrig, D., Llana-Díaz, L., and Núñez, M. (1998). An invitation to friendly testing. *Journal of Computer Science and Technology*, 13(6):531–545.

[de Nicola and Hennessy, 1984] de Nicola, R. and Hennessy, M. (1984). Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133.

[Francez, 1986] Francez, N. (1986). *Fairness*. Springer.

[Gregorio et al., 1997] Gregorio, C., Llana, L., Núñez, M., and Palao, P. (1997). Testing semantics for a probabilistic-timed process algebra. In *4th International AMAST Workshop on Real-Time Systems, Concurrent, and Distributed Software, LNCS 1231*, pages 353–367. Springer.

[Gregorio and Núñez, 1996] Gregorio, C. and Núñez, M. (1996). Specifying and verifying the Alternating Bit Protocol with Probabilistic-Timed LOTOS. In *COST 247 International Workshop on Applied Formal Methods in System Design*, pages 38–50.

[Gregorio and Núñez, 1999] Gregorio, C. and Núñez, M. (1999). Denotational semantics for probabilistic refusal testing. In *PROBMIV'98, Electronics Notes in Theoretical Computer Science 21*. Elsevier.

[Hennessy, 1987] Hennessy, M. (1987). An algebraic theory of fair asynchronous communicating processes. *Theoretical Computer Science*, 49:121–143.

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

[Koomen, 1985] Koomen, C. (1985). Algebraic specification and verification of communications protocols. *Science of Computer Programming*, 5:1–36.

[Kumar et al., 1998] Kumar, K. N., Cleaveland, R., and Smolka, S. (1998). Infinite probabilistic and nonprobabilistic testing. In *18th Conference on Foundations of Software Technology and Theoretical Computer Science, LNCS 1530*, pages 209–220.

[Milner, 1981] Milner, R. (1981). A modal characterization of observable machine-behaviour. In *6th CAAP, LNCS 112*, pages 25–34. Springer.

[Natarajan and Cleaveland, 1995] Natarajan, V. and Cleaveland, R. (1995). Divergence and fair testing. In *22nd ICALP, LNCS 944*, pages 648–659. Springer.

[Núñez, 1999] Núñez, M. (1999). An axiomatization of probabilistic testing. In *5th AMAST Workshop on Real-Time and Probabilistic Systems, LNCS 1601*, pages 130–150. Springer.

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

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

[Rupérez, 1997] Rupérez, D. (1997). Una semántica de pruebas *fair* mediante un modelo probabilístico. Master Thesis. Dept. Sistemas Informáticos y Programación. Universidad Complutense de Madrid.

[Tretmans, 1996] Tretmans, J. (1996). Conformance testing with labelled transition systems: Implementation relations and test generation. *Computer Networks and ISDN Systems*, 29:49–79.

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