Customized Testing for Probabilistic Systems*

Luis F. Llana-Díaz, Manuel Núñez, and Ismael Rodríguez

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

Abstract. In order to test the correctness of an IUT (*implementation under test*) with respect to a specification, testing its whole behavior is desirable but unfeasible. In some situations, testing the behavior of the IUT assuming that it is stimulated by a given *usage model* is more appropriate. Though considering this approach to test functional behaviors consists simply in testing a subset of the IUT, to study the *probabilistic* behavior of systems by using this *customized testing* approach leads to some new possibilities. If usage models specify the probabilistic behavior of stimuli and specifications define the probabilistic behavior of reactions to these stimuli, then, by composing them, the probabilistic behavior of *any* behavior is completely specified. So, after a finite set of behaviors of the IUT is checked, we can compute an *upper bound* of the probability that a user following the usage model finds an error in the IUT. This can be done by considering the *worst case* scenario, that is, that any unchecked behavior is *wrong*.

1 Introduction

Even though testing the whole behavior of a system is desirable, this implies, in general, applying an infinite number of tests. So, formal testing methodologies usually focus on critical parts or aspects of the system. In particular, it is specially important to check that systems provide some minimal functionalities, even if other less relevant functionalities fail. That is, we check that some critical *usage modes* are correct and remain available as expected. More generally, we can group and abstract a set of usage modes in terms of a (probably abstract) user that produces them, that is, in terms of a user model that represents a subset of manners to use the system. Once we are provided with a suitable user model, this model can be used to particularize the goals of the testing procedure. In other words, we can test the correctness of an implementation under test (IUT) with respect to a specification *under the assumption* that the system is stimulated according to the user model. Let us note that if our testing methodology focuses on checking the *functional* behavior of the IUT (i.e., what it does and what it does not), then testing the IUT with respect to a user model may be

* Research partially supported by the Spanish MCYT project TIC2003-07848-C02-01, the Junta de Castilla-La Mancha project PAC-03-001, and the Marie Curie project MRTN-CT-2003-505121/TAROT.

M.Ü. Uyar, A.Y. Duale, and M.A. Fecko (Eds.): TestCom 2006, LNCS 3964, pp. 87–102, 2006.

easy. In particular, it is enough that, among all the stimulation sequences that can be proposed to the system (i.e., among all stimulation sequences considered in the specification), we consider only those that can be produced by the user model as well. That is, testing the IUT with respect to a specification and a user model actually consists in testing the IUT with respect to a *subset* of the specification.

However, the application of this *user customized* approach to test other types of properties is less trivial and more interesting. This is the case if we consider the *probabilistic* behavior of entities. Let us suppose that the specification provides the desirable probabilistic behavior of the IUT and that, in addition, the user model explicitly defines the probabilistic propensity of each action stimulating the IUT. In this case, we have an environment where the ideal probabilistic behavior of the system, consisting of the IUT and the user model, is *completely specified*. That is, in any interaction between a user and a correct IUT, the probabilistic weight of each choice can be quantified. This extra information allows the testing methodology to go further than other methodologies by providing a relevant diagnosis result: After a finite set of tests is applied, we can compute, for a given *feasibility* degree, an *upper bound* of the probability that a user behaving according to the considered user model finds a wrong behavior in the IUT. This measure will be computed by considering that all the IUT traces that have not been produced yet behave *incorrectly*. Let us note that, after applying a finite set of tests, the number of traces that have not been studied yet is, in general, infinite. However, the cumulated probability of these traces (that is, the probability that any of them is produced) is, like any probability, finite, and we can compute it. Actually, this probability is the complementary of the probability that any *already* analyzed trace is performed. Let us remark that the (ideal) probability of each of these traces is given by the probability defined in the specification for that trace. Unfortunately, we cannot know with certainty whether the traces of the IUT that have been already analyzed actually follow the probabilities defined in the specification. However, by testing each nondeterministic choice of the IUT a high number of times and by applying a suitable contrast hypothesis, we can determine, for a given *feasibility* or *credibility* degree, whether the probabilistic behavior of the IUT corresponds to the one of the specification. For instance, if the specification indicates that, at a given point, the probabilities of performing a and b are the same, then, if the implementation has performed a 507 times and b 493 times then that credibility will be high. However, the credibility will be lower if they were recorded 614 and 386 times, respectively. Since our knowledge of the probabilistic behavior of the IUT will depend on a feasibility degree, the upper bound of the probability that a user finds an error in an IUT will be defined in probabilistic terms as well, that is, for a given feasibility degree.

Let us note that to provide an upper bound of the probability of error is not only useful for a (probabilistic) diagnostic of the IUT correctness. In fact, it may guide the testing process itself. The task of selecting, among an infinite set of tests, a finite set of tests to be applied during the (finite) time assigned to testing is not simple. Thus, we will be interested in tests with high discrimination power.

In other words, we should choose tests such that, when successfully passed, induce a high certainty about the correctness of the IUT. Actually, the upper bound of error probability implicitly provides a guide for selecting tests. We will prefer those sets of tests such that, when correctly passed by the IUT, provide a *lower* upper bound of error, that is, provide a higher certainty of the IUT correctness. That is, we will prefer those sets providing upper bounds with higher feasibility degrees.

In this paper we develop these ideas and we construct a testing methodology for testing probabilistic systems that interact with user models. We define two implementation relations. The first one directly compares the probabilities of the traces in the IUT and in the specification. The second one indirectly compares these probabilities by applying a contrast hypothesis to a sample collected by interacting with the IUT. Besides, we show how the measure that we commented before is computed from an IUT sample. In terms of related work, there is significant work on testing preorders and equivalences for probabilistic processes [2, 10, 12, 3, 1, 13, 7, 5]. Most of these proposals follow the *de Nicola and Hennessy's style* [6, 4], that is, two processes are equivalent if the application of any test belonging to a given set returns the same result. Instead, we are interested in checking whether an implementation conforms to a specification. In particular, our relations are more similar to the ones introduced in [14, 8]. Regarding probabilistic user models, it is worth to point out that these previous works do not explicitly consider this notion. User models have been used in specific software testing scenarios (e.g., to test C++ templates [11]). Other work deals with user models in the testing context [16, 15], but they do not consider formal conformance testing techniques.

The rest of the paper is structured as follows. In the next section we present some basic notions to denote specifications, IUTs, and user models. In Section 3 we show how these notions are related and we define tests. In Section 4 we present our (probabilistic) conformance relations. Then, in Section 5 we give the upper bound of the probability that a user finds an error in an IUT. Finally, in Section 6 we present our conclusions.

2 Basic Notions

In this section we present some basic notions used in the paper. First, we introduce some statistics notions. An *event* is any reaction we can detect from a system or environment; a random variable is a function associating each event with its probability.

Definition 1. Let $\mathcal{A}$ be a set of events and $\xi : \mathcal{A} \to [0, 1]$ be a function such that $\sum_{\alpha \in \mathcal{A}} \xi(\alpha) = 1$. We say that ξ is a *random variable* for the set of events $\mathcal{A}$.

If we observe that the event $\alpha \in \mathcal{A}$ is produced by a random source whose probabilistic behavior is given by ξ then we say that α *has been generated by* ξ. We extend this notion to sequences of events as expected: If we observe that the sequence of events $H = \langle \alpha_1, \ldots, \alpha_n \rangle$ is consecutively produced by a random

source whose probabilistic behavior is given by ξ then we say that H *has been generated by* ξ or that H is a *sample* of ξ.

Given the random variable ξ and a sequence of events H, we denote the *confidence* that H is *generated by* ξ by $\gamma(\xi, H)$. □

This definition introduces a simple version of discrete random variable where all the events are independent. The actual definition of a *random variable* is more complex but it is pointless to use its generality in our setting. In the previous definition, the application of a suitable *hypothesis contrast* is abstracted by the function γ. We have that $\gamma(\xi, H)$ takes a value in $[0, 1]$. Intuitively, a sample will be *rejected* if the probability of observing that sample from a given random variable is low. At the end of this section we present a working definition of the function γ. It is worth to point out that the results of this paper do not depend on the formulation of γ, being possible to *abstract* the actual definition.

Next we present the formalism we will use to define *specifications* and *implementations*. A *probabilistic finite state machine* is a finite state machine where each transition is equipped with a probability denoting its probabilistic propensity. Thus, a transition $s \xrightarrow{i/o}_p s'$ denotes that, when the machine is in state s and the input i is received, then, with probability p, it moves to the state s' and produces the output o. We will assume that the environment stimulates the machine with a single input at any time. Given an input, the machine probabilistically chooses the transition it takes from its current state. Hence, the probability of a transition allows to compare its propensity with the one of any other transition departing from the same state and receiving the *same* input. That is, given s and i, the addition of all values p such that there exist o, s' with $s \xrightarrow{i/o}_p s'$ must be equal to 1. In contrast, there is no requirement binding the probabilities departing from the same state and receiving different inputs because each one describes (part of) a different probabilistic choice of the machine.

Definition 2. A *Probabilistic Finite State Machine*, in short `PFSM`, is a tuple $M = (S, I, O, \delta, s_0)$ where

- S is the *set of states* and $s_0 \in S$ is the *initial state*.
- I and O, with $I \cap O = \varnothing$, denote the sets of *input* and *output* actions, respectively.
- $\delta \subseteq S \times I \times O \times (0, 1] \times S$ is the *set of transitions*. We will write $s \xrightarrow{i/o}_p s'$ to denote $(s, i, o, p, s') \in \delta$.

Transitions and states fulfill the following additional conditions:

- For all $s \in S$ and $i \in I$, the probabilities associated with outgoing transitions add up to 1, that is, $\sum\{p \mid \exists\, o \in O,\, s' \in S :\ s \xrightarrow{i/o}_p s'\} = 1$.
- `PFSMs` are *free of non-observable non-determinism*, that is, if whenever we have the transitions $s \xrightarrow{i/o}_{p_1} s_1$ and $s \xrightarrow{i/o}_{p_2} s_2$ then $p_1 = p_2$ and $s_1 = s_2$.
- In addition, we will assume that implementations are *input-enabled*, that is, for all state s and input i there exist o, p, s' such that $s \xrightarrow{i/o}_p s'$. □

Although PFSMs will be used to define specifications, a different formalism will be used to define *user models*. Specifically, we will use *probabilistic labeled transition systems*. A user model represents the external environment of a system. User models actively produce inputs that stimulate the system, while passively receive outputs produced by the system as a response. The states of a user model are split into two categories: *Input states* and *output states*. In input states, all outgoing transitions denote a different input action. Since inputs are probabilistically chosen by user models, any input transition is endowed with a probability. In particular, $s \xrightarrow{i}_p s'$ denotes that, with probability p, in the input state s, the input i is produced and the state is moved to s'. Given an input state s, the addition of all probabilities p such that there exists i, s' with $s \xrightarrow{i}_p s'$ must be *lower* than or equal to 1. If it is lower then we consider that the remainder up to 1 implicitly denotes the probability that the interaction with the system *finishes* at the current state. Regarding output states, all transitions departing from an output state are labeled by a different output action. However, output transitions do not have any probability value (let us remind that outputs are chosen by the system). Input and output states will strictly alternate, that is, for any input state s, with $s \xrightarrow{i}_p s'$, s' is an output state, and for any output state s, with $s \xrightarrow{o} s'$, s' is an input state.

Definition 3. A *probabilistic labeled transition system*, in short PLTS, is a tuple $U = (S_I, S_O, I, O, \delta, s_0)$ where

- S_I and S_O, with $S_I \cap S_O = \varnothing$, are the sets of *input* and *output* states, respectively. $s_0 \in S_I$ is the *initial state*.
- I and O, with $I \cap O = \varnothing$, are the sets of *input* and *output* actions, respectively.
- $\delta \subseteq (S_I \times I \times (0,1] \times S_O) \cup (S_O \times O \times S_I)$ is the *transition relation*. We will write $s \xrightarrow{i}_p s'$ to denote $(s, i, p, s') \in S_I \times I \times (0,1] \times S_O$ and $s \xrightarrow{o} s'$ to denote $(s, o, s') \in S_O \times O \times S_I$.

Transitions and states fulfill the following additional conditions:

- For all input states $s \in S_I$ and input actions $i \in I$ there exists at most one outgoing transition from s: $|\{s \xrightarrow{i}_p s' \mid \exists\, p \in (0,1], s' \in S_O\}| \leq 1$.
- For all output states $s \in S_O$ and output actions $o \in O$ there exists exactly one outgoing transition labeled with o: $|\{s \xrightarrow{o} s' \mid \exists\, s' \in S_I\}| = 1$.
- For all input state $s \in S_I$ the addition of the probabilities associated with the outgoing transitions is lower than or equal to 1, that is, $\mathtt{cont}(s) = \sum\{p \mid \exists\, s' \in S_O : s \xrightarrow{i}_p s'\} \leq 1$. So, the probability of stopping at that state s is $\mathtt{stop}(s) = 1 - \mathtt{cont}(s)$. □

By iteratively executing transitions, both PFSMs and PLTSs can produce sequences of inputs and outputs. The probabilities of these sequences are given by the probabilities of the transitions. Next we introduce some *trace* notions. A *probability trace* is a sequence of probabilities, a *trace* is a sequence of inputs and outputs, and a *probabilistic trace* is a tuple containing both.

Definition 4. A *probability trace* π is a finite sequence of probabilities, that is, a possibly empty sequence $\langle p_1, p_2, \ldots, p_n \rangle \in (0,1]^*$. The symbol ϵ denotes the empty probability trace. Let $\pi = \langle p_1, p_2, \ldots, p_n \rangle$ be a probability trace. We define its *sef-product*, denoted by $\prod \pi$, as $\prod_{1 \leq i \leq n} p_i$. Since $\prod_{a \in \varnothing} = 1$, we have $\prod \epsilon = 1$. Let $\pi = \langle p_1, p_2, \ldots, p_n \rangle$ and $\pi' = \langle p'_1, p'_2, \ldots, p_m \rangle$ be probability traces. Then, $\pi \cdot \pi'$ denotes their concatenation that is, $\langle p_1, p_2, \ldots, p_n, p'_1, p'_2, \ldots, p_m \rangle$, while $\pi * \pi'$ denotes its pairwise product, that is, $\langle p_1 * p'_1, p_2 * p'_2, \ldots, p_r * p'_r \rangle$, where $r = \min(n, m)$.

A *trace* ρ is a finite sequence of input/output actions $(i_1/o_1, i_2/o_2, \ldots, i_n/o_n)$. The symbol ϵ denotes the empty trace. Let ρ and ρ' be traces. Then, $\rho \cdot \rho'$ denotes their concatenation. A *probabilistic trace* is a pair (ρ, π) where ρ is a trace $(i_1/o_1, i_2/o_2, \ldots, i_n/o_n)$ and $\pi = \langle p_1, p_2, \ldots, p_n \rangle$ is a probability trace. If ρ and π are both empty then we have the *empty probabilistic trace*, written as (ϵ, ϵ). Let (ρ, π) and (ρ', π') be probabilistic traces. Then, $(\rho, \pi) \cdot (\rho', \pi')$ denotes their concatenation, that is, $(\rho \cdot \rho', \pi \cdot \pi')$. □

Next we define how to extract traces from `PFSMs` and `PLTSs`. First, we consider the reflexive and transitive closure of the transition relation, and we call it *generalized* transition. Then, probabilistic traces are constructed from generalized transitions by considering their sequences of actions and probabilities.

Definition 5. Let M be a `PFSM`. We inductively define the *generalized* transitions of M as follows:

- If $s \in S$ then $s \stackrel{\epsilon}{\Longrightarrow}_{\epsilon} s$ is a generalized transition of M.
- If $s \in S$, $s \stackrel{\rho}{\Longrightarrow}_{\pi} s'$, and $s' \xrightarrow{i/o}_{p} s_1$ then $s \stackrel{\rho \cdot i/o}{\Longrightarrow}_{\pi \cdot \langle p \rangle} s_1$ is a generalized transition of M.

We say that (ρ, π) is a *probabilistic trace* of M if there exists $s \in S$ such that $s_0 \stackrel{\rho}{\Longrightarrow}_{\pi} s$. In addition, we say that ρ is a *trace* of M. The sets $\mathtt{pTr}(M)$ and $\mathtt{tr}(M)$ denote the sets of *probabilistic traces* and *traces* of M, respectively. □

The previous notions can be defined for `PLTSs`. In order to obtain sequences of paired inputs and outputs, traces begin and end at input states; generalized transitions are constructed by taking *pairs* of consecutive `PLTS` transitions.

Definition 6. Let M be a `PLTS`. We inductively define the *generalized* transitions of U as follows:

- If $s \in S_I$ then $s \stackrel{\epsilon}{\Longrightarrow}_{\epsilon} s$ is a generalized transition of U.
- If $s \in S_I$, $s \stackrel{\rho}{\Longrightarrow}_{\pi} s'$, and $s' \xrightarrow{i}_{p} s'' \xrightarrow{o} s_1$ then $s \stackrel{\rho \cdot i/o}{\Longrightarrow}_{\pi \cdot \langle p \rangle} s_1$ is a generalized transition of U.

We say that (ρ, π) is a *probabilistic trace* of U if there exists $s \in S_I$ such that $s_0 \stackrel{\rho}{\Longrightarrow}_{\pi} s$. In addition, we say that ρ is a *trace* of U. We define the probability of U to stop after ρ, denoted by $\mathtt{stop}_U(\rho)$, as $\mathtt{stop}(s)$. The sets $\mathtt{pTr}(U)$ and $\mathtt{tr}(U)$ denote the set of *probabilistic traces* and *traces* of U, respectively. □

2.1 Definition of a Hypothesis Contrast: Pearson's χ^2

In this paper we consider *Pearson's* χ^2 *contrast* but other contrasts could be used. The mechanism is the following. Once we have collected a sample of size n we perform the following steps:

- We split the sample into k classes covering all the possible range of values. We denote by O_i the *observed frequency* in class i (i.e., the number of elements belonging to the class i).
- We calculate, according to the proposed random variable, the probability p_i of each class i. We denote by E_i the *expected frequency* of class i, that is, $E_i = np_i$.
- We calculate the *discrepancy* between observed and expected frequencies as $X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$. When the model is correct, this discrepancy is approximately distributed as a χ^2 random variable.
- The number of freedom degrees of χ^2 is $k-1$. In general, this number is equal to $k-r-1$, where r is the number of parameters of the model which have been estimated by maximal likelihood over the sample to estimate the values of p_i. In our framework we have $r = 0$ because the model completely specifies the values of p_i before the samples are observed.
- We will *accept* that the sample follows the proposed random variable if the probability to obtain a discrepancy greater than or equal to the detected discrepancy is high enough, that is, if $X^2 < \chi^2_\alpha(k-1)$ for some α high enough. Actually, as such margin to accept the sample decreases as α increases, we can obtain a measure of the validity of the sample as $\max\{\alpha | X^2 \leq \chi^2_\alpha(k-1)\}$.

According to the previous steps, next we present an operative definition of the function γ that was introduced in Definition 1. We will consider two sets of events $\mathcal{A}$ and $\mathcal{A}'$, with $\mathcal{A} \subseteq \mathcal{A}'$. The set $\mathcal{A}$ gives the domain of the random variable ξ, while the events denoted by H belong to $\mathcal{A}'$. If the sample includes any event a that is not considered by the random variable (i.e., $a \notin \mathcal{A}$) then the sample cannot be generated by the random variable and the minimal *feasibility*, that is 0, is returned. Otherwise, we return the maximal feasibility α such that the hypothesis contrast is passed.

Definition 7. Let $\mathcal{A}$ and $\mathcal{A}'$ be sets of events, with $\mathcal{A} \subseteq \mathcal{A}'$, and H be a sample of elements belonging to $\mathcal{A}'$. Let $\xi : \mathcal{A} \to (0, 1]$ be a random variable. We define the confidence of ξ on H, denoted by $\gamma(\xi, H)$, as follows:

$$\gamma(\xi, H) = \begin{cases} 0 & \texttt{if } H \cap \bar{\mathcal{A}} \neq \varnothing \\ \max\{\alpha \mid X^2_\xi \leq \chi^2_\alpha(k-1)\} & \texttt{otherwise} \end{cases}$$

where X^2_ξ denotes the discrepancy level of the sample H on ξ, calculated as explained above by considering that the sampling space is $\mathcal{A}$. □

3 Tests and Composition of Machines

In this section we define our tests as well as the interaction between the notions introduced in the previous section (`PFSMs` and `PLTSs`). As we said before, we will

use PLTSs to define the behavior of the external environment of a system, that is, a user model. Moreover, PLTSs are also appropriate to define the *tests* we will apply to an IUT. Tests are PLTSs fulfilling some additional conditions. Basically, a test defines a finite sequence of inputs that can be interrupted depending on the outputs produced by the IUT as response: If one of the expected outputs is received then the next input is applied, otherwise the interaction with the IUT finishes. Since tests consider a single sequence of inputs, each intermediate input state of the sequence contains a single outgoing transition labeled by the next input and probability 1. Output states offer transitions with different outputs. Only one of the input states reached by these transitions offers a (single) transition; the interaction finishes in the rest of them.

Definition 8. A test $T = (S_I, S_O, I, O, \delta, s_0)$ is a PLTS such that for all $s \in S_I$ there is at most one transition $s \xrightarrow{i}_p s'$ (so, in this transition $p = 1$), and for all $s \in S_O$ there is at most one transition $s \xrightarrow{o} s'$ with a *continuation*, that is, $|\{s'' \mid \exists\, i \in I,\, o \in O,\, s''' \in S_O,\, p \in (0,1] :\ s \xrightarrow{o} s'' \xrightarrow{i}_p s'''\}| \leq 1$. □

Let us note that, contrarily to other frameworks, tests are not provided with diagnostic capabilities on *their own*. In other words, tests do not have fail/success states. Since our framework is probabilistic, the requirements defined by specifications are given in probabilistic terms. Moreover, the absence of transitions labeled by specific outputs in specification states is considered in probabilistic terms as well: If there exists a state s, an input i, and an output o such that there do not exist p, s' with $s \xrightarrow{i/o}_p s'$ then we consider that the probability of producing o in the state s after receiving the input i is 0. As we will see in the next section, deciding whether the IUT conforms to the specification will also be done in probabilistic terms. In particular, we will consider whether it is *feasible* that the IUT *behaves* as if it were defined as the specification indicates. We will check this fact by means of a suitable *hypothesis contrast*.

Our testing methodology consists in testing the behavior of a system under the assumption that it is stimulated by a given user model. Thus, the sequences we use to stimulate it, that is, the tests, will be extracted from the behavior of the user model. Next we show how a test is constructed from a probabilistic trace of a user model. The input and output states of the test are identified with natural numbers. All the input states (but the first and last ones) are also endowed with an output action. In order to distinguish between input and output states we decorate them with $^\bullet$ and *, respectively.

Definition 9. Let $\rho = (i_1/o_1, i_2/o_2, \ldots, i_r/o_r)$ be a trace, I be a set of input actions such that $\{i_1, \ldots i_r\} \subseteq I$, and O be a set of output actions such that $\{o_1, \ldots, o_r\} \subseteq O$. We define the *associated test* to ρ, denoted by assoc(ρ), as the test $(S_{IT}, S_{OT}, I, O, \delta_T, 0^\bullet)$, where

- $S_{IT} = \{0^\bullet, r^\bullet\} \cup \{(j,o)^\bullet | o \in O, 1 \leq j < r\}$ and $S_{OT} = \{j^\star | 1 \leq j \leq r\}$.
- For all $1 \leq j < r, o \in O$: $(j,o_j)^\bullet \xrightarrow{i_{j+1}}_1 (j+1)^\star$, $j^\star \xrightarrow{o} (j,o)^\bullet \in \delta_T$. We also have $0^\bullet \xrightarrow{i_1}_1 0^\star$, $r^\star \xrightarrow{o_r} r^\bullet \in \delta_T$.

Fig. 1. Normalization if composition of PFSMs and PLTSs

Let U be a PLTS. The *set of associated tests to* U, denoted by $\texttt{assoc}(U)$, is the set of tests associated to its traces, that is $\{\texttt{assoc}(\rho) \mid \rho \in \texttt{tr}(U)\}$. □

Next we define the composition of a PFSM (denoting either a specification or an IUT) with a PLTS (denoting either a user model or a test) in terms of its behavior, that is, in terms of traces and probabilistic traces. The set of traces is easily computed as the intersection of the traces produced by both components. In order to define the set of probabilistic traces, the ones provided by both components are considered. For a given input/output pair i/o, the probability of producing i will be taken from the corresponding transition of the PLTS, while the probability of producing o as a response to i will be given by a transition of the PFSM. Let us note that the states of a specification do not necessarily define outgoing transitions for all available inputs, that is, specifications are not necessarily *input-enabled*. So, a PFSM representing a specification could not provide a response for an input produced by a PLTS. Since the specification does not define any behavior in this case, we will assume that the PFSM is allowed to produce *any* behavior from this point on. The composition of a PLTS and a PFSM will be constructed to check whether the traces *defined* by the specification are correctly produced by the implementation (under the assumption that these machines are stimulated by the user model). Hence, undefined behaviors will not be considered relevant and will not provide any trace to the composition of the PLTS and the PFSM. In order to appropriately represent the probabilities of the relevant traces, their probabilities will be *normalized* if undefined behaviors appear. We illustrate this process in the following example.

Example 1. Let us suppose that a user model can produce the inputs i_1, i_2, and i_3 with probabilities $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{4}$, respectively (see Figure 1, left). At the same time, the corresponding specification provides outgoing transitions with inputs i_1 and i_2, but not with i_3 (see Figure 1, right). Since the specification does not define any reaction to i_3, the probabilities of taking inputs i_1 or i_2 in the composition of the specification and the user model are normalized to denote that i_3 is not considered. So, the probability of i_1 becomes $\frac{1/2}{3/4} = \frac{2}{3}$ while the probability of i_2 is $\frac{1/4}{3/4} = \frac{1}{3}$. □

Definition 10. Let $M = (S_M, I, O, \delta_M, s_{0M})$ be a PFSM and let us consider a PLTS $U = (S_{IU}, S_{OU}, I, O, \delta_U, s_{0U})$ such that $s_{0M} \stackrel{\rho}{\Longrightarrow}_{\pi_1} s_1$ and $s_{0U} \stackrel{\rho}{\Longrightarrow}_{\pi_2} s_2$. We define:

– The sum of the probabilities of *continuing together after* ρ as

$$\mathtt{cont}_{M\|U}(\rho) = \sum \left\{ p \;\middle|\; \begin{array}{c} \exists i \in I,\, o \in O,\, s_2' \in S_{OU},\, s_1' \in S_M,\, r \in (0,1] : \\ s_2 \xrightarrow{i}_p s_2' \;\wedge\; s_1 \xrightarrow{i/o}_r s_1' \end{array} \right\}$$

– The *normalization factor of* $M \parallel U$ *after* ρ as the sum of the previous probability plus the probability of U to stop after ρ, that is $\mathtt{norm}_{M\|U}(\rho) = \mathtt{cont}_{M\|U}(\rho) + \mathtt{stop}_U(\rho)$. □

Definition 11. Let $M = (S_M, I, O, \delta_M, s_{0M})$ be a PFSM and let us consider a PLTS $U = (S_{IU}, S_{OU}, I, O, \delta_U, s_{0U})$. The *set of traces* generated by the *composition* of M and U, denoted by $\mathtt{tr}(M \parallel U)$, is defined as $\mathtt{tr}(M) \cap \mathtt{tr}(U)$. The *set of probabilistic traces* generated by the *composition* of M and U, denoted by $\mathtt{pTr}(M \parallel U)$, is defined as the smallest set such that

– $(\epsilon, \epsilon) \in \mathtt{pTr}(M \parallel U)$.

– If we have that $(\rho, \pi) \in \mathtt{pTr}(M \parallel U)$, $s_{0M} \overset{\rho}{\Longrightarrow}_{\pi_1} s_1' \xrightarrow{i/o}_{p_1} s_1$, and $s_{0U} \overset{\rho}{\Longrightarrow}_{\pi_2} s_2' \xrightarrow{i}_{p_2} s_2'' \xrightarrow{o} s_2$, then $(\rho \cdot i/o, \pi \cdot \langle p \rangle) \in \mathtt{pTr}(M \parallel U)$, where p is the product of p_1 and p_2 *normalized* with respect to the normalization factor of $M \parallel U$ after ρ, that is, $p = \frac{p_1 \cdot p_2}{\mathtt{norm}_{M \parallel P}(\rho)}$. □

Let us remark that the probabilistic behavior of the traces belonging to the composition of PFSMs and PLTSs is completely specified: The probabilities of inputs are provided by the PLTS while the probabilities of outputs are given by the PFSM. So, a random variable denoting the probability of *each* trace produced by the composition can be constructed. Moreover, the composition of the specification and the user model provides a source to *randomly* generate tests. In fact, tests are constructed by following a specific sequence of inputs and outputs of the user model. Hence, the random selection of tests can be represented by a random variable associating tests with the probability that the probabilistic trace guiding the test is taken in the composition of the specification and the user model.

Definition 12. Let $M = (S_M, I, O, \delta_M, s_{0M})$ be a PFSM and let us consider a PLTS $U = (S_{IU}, S_{OU}, I, O, \delta_U, s_{0U})$. We define the *traces random variable of the composition* of M and U as the function $\xi_{M\|U} : \mathtt{tr}(M \parallel U) \longrightarrow (0, 1]$ such that for all $(\rho, \pi) \in \mathtt{pTr}(M \parallel U)$ we have $\xi_{M\|U}(\rho) = \prod \pi * (1 - \mathtt{stop}_U(\rho))$.

Let $\mathcal{T} = \{T \mid \rho \in \mathtt{tr}(M \parallel U) \;\wedge\; T = \mathtt{assoc}(\rho)\}$. We define the *tests random variable of the composition of* M *and* U as the function $\zeta_{M\|U} : \mathcal{T} \longrightarrow (0, 1]$ such that for all test $T \in \mathcal{T}$ we have $\zeta_{M\|U}(T) = p$ iff $(\rho, \pi) \in \mathtt{pTr}(U)$, $(\rho, \pi') \in \mathtt{pTr}(M)$, $T = \mathtt{assoc}(\rho)$, and $p = \prod \pi * \prod \pi' * (1 - \mathtt{stop}_U(\rho))$. □

Let us note that the sum of the probabilities of all traces may be strictly less than 1. This is because random variables have to take into account some events that are not directly considered in the traces: The choice of a user to stop in a state. Next we identify some properties of our framework.

Proposition 1. Let S be a PFSM, U be a PLTS, $(\rho, \pi) \in \mathtt{pTr}(U)$, and $T = \mathtt{assoc}(\rho)$. The following properties hold:

- $\mathtt{tr}(T) \subseteq \mathtt{tr}(U)$ and $\mathtt{tr}(S \parallel T) \subseteq \mathtt{tr}(S \parallel U)$.
- if $(\rho, \pi') \in \mathtt{pTr}(S \parallel T)$ then $(\rho, \pi \cdot \pi') \in \mathtt{pTr}(S \parallel U)$.
- $\mathtt{tr}(U) = \bigcup \{\mathtt{tr}(T) \mid T \in \mathtt{assoc}(U)\}$.
- $\mathtt{tr}(S \parallel U) = \bigcup \{\mathtt{tr}(S \parallel T) \mid T \in \mathtt{assoc}(U)\}$. □

4 Probabilistic Relations

In this section we introduce our probabilistic conformance relations. Following our user customized approach, they relate an IUT *and* a user model with a specification *and* the same user model. These three elements will be related if the probabilistic behavior shown by the IUT when stimulated by the user model appropriately follows the corresponding behavior of the specification. In particular, we will compare the *probabilistic traces* of the composition of the IUT and the user with those corresponding to the composition of the specification and the user. Let us remind that IUTs are input-enabled but specifications might not be so. So, the IUT could define probabilistic traces including sequences of inputs that are not defined in the specification. Since there are no specification requirements for them, these behaviors will be ignored by the relation. In order to do it, an appropriate subset of the traces of the composition of the IUT and the user must be taken. In the following relation, we require that the probabilities of the corresponding traces are *exactly* the same in both compositions. Later we will see another relation where, due to practical reasons, this requirement will be relaxed.

Definition 13. Let S, I be PFSMs, U be a PLTS, and s_{0S}, s_{0I}, and s_{0U} be their initial states, respectively. We define the *set of probabilistic traces generated by the implementation I and the user model U modulo the specification S*, denoted by $\mathtt{pTr}(I \parallel U)_S$, as the smallest set such that:

- $(\epsilon, \epsilon) \in \mathtt{pTr}(I \parallel U)_S$.
- If $(\rho, \pi) \in \mathtt{pTr}(I \parallel U)_S$ and we have the following sequences of transitions:
  - $s_{0U} \overset{\rho}{\Longrightarrow}_{\pi_2} s'_1 \xrightarrow{i}_{p_2} s''_1 \xrightarrow{o} s_1$, and
  - $s_{0I} \overset{\rho}{\Longrightarrow}_{\pi_1} s'_2 \xrightarrow{i/o}_{p_1} s_2$,
  - $s_{0S} \overset{\rho}{\Longrightarrow}_{\pi_3} s'_3 \xrightarrow{i/o}_{p_3} s_3$,

  then $(\rho \cdot i/o, \pi \cdot \langle p \rangle) \in \mathtt{pTr}(I \parallel U)_S$, where p is the product of p_1 and p_2 *normalized* with respect to the normalization factor of $S \parallel U$ after ρ, that is $p = \frac{p_1 \cdot p_2}{\mathtt{norm}_{S \parallel U}(\rho)}$.

Let S, I be PFSMs and U be a PLTS. We say that I *conforms to S with respect to U*, denoted by $I \ \mathtt{conf}_U \ S$, if $\mathtt{pTr}(I \parallel U)_S = \mathtt{pTr}(S \parallel U)$. □

Although the previous relation properly defines our probabilistic requirements, it cannot be used in practice because we cannot *read* the probability attached

to a transition in a black-box IUT. So, a more applicable version of the relation is required. Let us note that even though a single observation does not provide valuable information about the probability of an IUT trace, an *approximation* to this value can be calculated by interacting a high number of times with the IUT and analyzing its reactions. In particular, we can compare the empirical behavior of the IUT with the ideal behavior defined by the specification and check whether it is *feasible* that the IUT would have behaved like this if, internally, it were defined conforming to the specification. Depending on the empirical observations, this feasibility may be different. The feasibility degree of a set of samples with respect to its ideal probabilistic behavior (defined by a random variable) will be provided by a suitable contrast hypothesis. We will rewrite the previous relation in these terms. The new relation will be parameterized by two values: The samples collected by means of interactions with the IUT and a feasibility threshold. Then, by using an indirect approach, the new relation will impose the same probabilistic constraints as the relation defined before.

We must establish the way samples are collected. First, we generate the tests associated to the user model U and then we let these tests to interact with the IUT. Then, we must check if the obtained sample conforms to the random variable corresponding to the user and the specification, that is, $\xi_{S\|U}$, as introduced in Definition 12. This last point will be done via the hypothesis contrast. We will require that the feasibility of the hypothesis contrast reaches a required threshold. Before we present the new relation, we introduce the notion of *sampling*.

Definition 14. Let M be a `PFSM` and U be a `PLTS`. We say that a sequence $\langle \rho_1, \rho_2, \ldots, \rho_n \rangle$ is a *trace sample* of $M \parallel U$ if it is generated by $\xi_{M\|U}$. We say that a sequence $\langle T_1, T_2, \ldots T_n \rangle$ is a *test sample of* $M \parallel U$ if it is generated by $\zeta_{M\|U}$. Let $\langle T_1, T_2, \ldots T_n \rangle$ be a test sample of $M \parallel U$. We say that a sequence $\langle \rho_1, \rho_2, \ldots, \rho_n \rangle$ is a *trace-test sample* of $M \parallel U$ if for all $1 \leq i \leq n$ we have that ρ_i is the result of a probabilistic execution of $M \parallel T_i$. □

Next we introduce the new conformance relation defined in probabilistic terms. As before, we will ignore any implementation behavior involving sequences of inputs not considered by the specification. This will be done by removing them from the *trace-test sample* we use to compare the IUT and the specification. In the next definition, H_S represents the sequence of traces resulting after removing those traces from the original trace-test sample H.

Definition 15. Let S be a `PFSM` and $H = \langle \rho_1, \rho_2, \ldots, \rho_n \rangle$ be a sequence of traces. H_S denotes the sub-sequence $\langle \rho_{r1}, \rho_{r2}, \ldots, \rho_{rn} \rangle$ of H that contains all the probabilistic traces whose input sequences can be produced by S, that is, $\rho = (i_1/o_1, \ldots i_m/o_m) \in H_S$ iff $\rho \in H$ and there exist $o'_1, \ldots o'_n \in O$ such that $(i_1/o'_1, \ldots i_m/o'_m) \in \mathtt{tr}(S)$.

Let S and I be `PFSMs`, U be a `PLTS`, $H = \langle \rho_1, \rho_2, \ldots, \rho_n \rangle$ be a trace-test sample of $I \parallel U$, and $0 \leq \alpha \leq 1$. We write $S \ \mathtt{conf}_{(H,\alpha)} \ I$ if $\gamma(\xi_{S\|U}, H_S) \geq \alpha$. □

5 Upper Bound of Probability of Failure for a User

In this section we provide an alternative measure of the correctness of an IUT. Similarly to the conformance relation given in Definition 15, it will be calculated by using a sample collected from the interaction with the IUT. This measure is an *upper bound* of the probability that the user obtains from the IUT a trace whose probabilistic behavior is *wrong* with respect to the specification. That is, it provides an upper bound of the probability of finding an error in the IUT. Since this measure will be computed from a specific sample, it will also be parameterized by a feasibility degree α. We assess the measure as we sketched in the introduction: From a given interaction sample with the IUT, we consider the feasibility degree α that this sample was generated according to the probabilistic behavior defined by the specification. Next we consider the probability of producing a behavior that is *not* included in the sample. Since we can assume that the probabilities of the traces in the sample are correct with feasibility α, we can use these probabilities to compute the probability of producing any other trace by adding the probabilities of all traces in the sample and by considering the *complementary* probability. Then, we consider the worst case of these traces, that is, we suppose that the probabilistic behavior of all of them is *wrong*. Hence, we obtain an upper bound of the probability that the user interacts with the IUT and observes a trace whose propensity is not that given by the specification (with feasibility α). First, we define the *prefixes* of a probabilistic trace that will allow to structure samples in a suitable form.

Definition 16. Let $(\rho, \pi) = ((i_1/o_1, \ldots, i_n/o_n), \langle p_1, \ldots, p_n \rangle)$ be a probabilistic trace. We say that a probabilistic trace (ρ', π') is a *prefix* of (ρ, π) if either $(\rho', \pi') = (\epsilon, \epsilon)$ or $(\rho', \pi') = ((i_1/o_1, \ldots, i_j/o_j), \langle p_1, \ldots, p_j \rangle)$, for some $1 \leq j \leq n$. We denote by $\texttt{prefix}(\rho, \pi)$ the set of all prefixes of the probabilistic trace (ρ, π). □

Let us consider a finite set of probabilistic traces such that all their prefixes are also included in the set. In fact, if these traces are a sample produced by the IUT then we can represent our *knowledge* about the IUT by means of a *suitable machine* producing these traces. Since we assume that the IUT does not have non-observable non-determinism, if two observed samples share a common prefix then we can consider that the common parts of both traces traverse the same path of states in the IUT. This fact can be reflected in the machine we construct by making both traces to share the same states until they diverge. Besides, let us note that we cannot detect whether a *loop* of states is taken in the IUT during our interaction with it, since we consider that the IUT is a black box. So, the machine representing our knowledge about the IUT, extracted from a sample, will be a *tree*: All traces in the sample depart from the initial state and traces share paths while they traverse common prefixes.

Definition 17. Let tr be a prefix-closed set of probabilistic traces. We say that tr is a *probabilistic tree* if the following conditions hold:

- The nodes of the tree are labeled by probabilistic traces belonging to tr.
- The arcs between nodes are labeled by pairs $(i/o, p)$, where i is an input, o is an output, and p is a probability. There exists an arc between two nodes (ρ, π) and (ρ', π'), denoted by $(\rho, \pi) \xrightarrow{i/o}_p (\rho', \pi')$, if $(\rho', \pi') = (\rho \cdot (i/o), \pi \cdot \langle p \rangle)$.
- For each node (ρ, π), the probabilities of all the outgoing arcs is less than or equal to 1, that is, $L = \sum\{p \mid \exists\, i, o, p : (\rho, \pi) \xrightarrow{i/o}_p (\rho \cdot (i/o), \pi \cdot \langle p \rangle)\} \leq 1$. Hence, the probability of *stopping in* (ρ, π) is given by $\mathtt{stop}_\pi(\rho) = 1 - L$.
- The probability of reaching a node (ρ, π) is equal to $\prod \pi$. □

After a sample of traces H is extracted from the IUT, the $\mathtt{conf}_{(H,\alpha)}$ relation given in Definition 15 allows to check whether the feasibility that H is produced by the specification is at least α. If this is the case then we can construct a set of *probabilistic traces* from H by attaching each trace in H with the probability given in the specification for that trace. The feasibility that the probabilities we attach are actually correct is equal to α. Then, the *probabilistic tree* representing this set shows, with feasibility α, the behavior of the IUT regarding the traces belonging to H. In order to compute the upper bound of the error probability, we will consider that any trace leaving this tree behaves *incorrectly*. We will identify these traces by considering a *higher* tree denoting all the traces that can be produced, not only those we observed in the sample. Then, the probability of producing any unobserved trace will be computed by considering the probability of performing a trace that leaves the lower tree. This probability is computed by adding the probabilities of all the traces reaching the border of the lower tree and performing an additional transition to leave it.

Definition 18. Let tr_1 and tr_2 be probabilistic trees such that $tr_1 \subseteq tr_2$ and tr_1 is finite. The probability of *reaching* tr_2 *from* tr_1, denoted by $\mathtt{rch}(tr_1, tr_2)$, is defined as $\sum\{(1 - \mathtt{stop}_\pi(\rho)) * \prod \pi \mid (\rho, \pi)$ maximal probabilistic trace in $tr_1\}$. □

The *higher* tree tr_2 will be given by the set of probabilistic traces that are produced by the composition of the specification and the user model. This tree is used to compute the probability of leaving the lower tree. In particular, only the probabilities of transitions departing from *leaves* of the lower tree are considered. The following result shows how higher trees can be constructed.

Proposition 2. Let S be a PFSM and U be a PLTS. We have that the set of probabilistic traces $\mathtt{pTr}(S \parallel U)$ is a probabilistic tree. Moreover, there exists an arc labeled by $(i/o, p)$ between two nodes (ρ, π) and $(\rho \cdot (i/o), \pi \cdot \langle p \rangle)$ iff we have the sequences $s_{0S} \overset{\rho}{\Longrightarrow}_{\pi_1} s_1 \xrightarrow{i/o}_{p_1} s_1'$ and $s_{0U} \overset{\rho}{\Longrightarrow}_{\pi_2} s_2 \xrightarrow{i}_{p_2} s_2'$, where p is the normalized product of p_1 and p_2 after ρ, that is, $p = \frac{p_1 \cdot p_2}{\mathtt{norm}_{S \parallel U}(\rho)}$. □

Next we show how the lower tree is created. A tree containing the traces of a given *sample* is constructed by considering both the sample and the composition of the specification and the user model. Let us note that, despite the sample being produced by the interaction with the implementation, the probabilities of traces will be taken from the specification. Let us also note that we will be able

to do this if the sample passes the *hypothesis contrast* that compares it with the behavior of the specification. This hypothesis contrast is implicitly applied by the relation $\texttt{conf}_{(H,\alpha)}$.

Definition 19. Let S, I be PFSMs such that $S\ \texttt{conf}_{(H,\alpha)}\ I$, U be a PLTS and $H = \langle \rho_1, \rho_2, \ldots, \rho_r \rangle$ be a trace-test sample of $I \parallel U$. The *probabilistic tree of* H, denoted by $\texttt{pTree}(H)$, is defined as $\bigcup_{\rho \in H,\ (\rho,\pi) \in \texttt{pTr}(S \parallel U)} \texttt{prefix}(\rho, \pi)$. □

Due to the way probabilistic trees are constructed from implementations, specifications, and user models, the following result holds.

Proposition 3. Let S and I be PFSMs, U be a PLTS, and $H = \langle \rho_1, \rho_2, \ldots, \rho_n \rangle$ be a trace-test sample of $I \parallel U$ such that $S\ \texttt{conf}_{(H,\alpha)}\ I$. We have that $\texttt{pTree}(H)$ is finite and $\texttt{pTree}(H) \subseteq \texttt{pTr}(S \parallel U)$. □

Now we are provided with all the needed machinery to define the upper bound of the probability that a user interacting with the IUT observes a probabilistic behavior that does not conform to the specification.

Definition 20. Let S and I be PFSMs, U be a PLTS, and $H = \langle \rho_1, \rho_2, \ldots, \rho_n \rangle$ be a trace-test sample of $I \parallel U$ such that $S\ \texttt{conf}_{(H,\alpha)}\ I$. The *upper bound of the probability that the user* U *observes a wrong probabilistic behavior in* I *with feasibility* α, denoted by $\texttt{wrong}(U, I, \alpha)$, is given by $\texttt{rch}(\texttt{pTree}(H), \texttt{pTr}(S \parallel U))$. □

6 Conclusions and Future Work

In this paper we have presented a probabilistic testing methodology that allows to consider user models. On the one hand, by applying user models, we can focus on testing a specific critical behavior. On the other hand, since we explicitly consider the propensity of each non-deterministic choice of systems, we can study systems not only on the basis of what they do but also on how often they do it. Since actual probabilities cannot be extracted from a black-box system, the probabilistic behavior of implementations and specifications is compared by means of suitable hypothesis contrasts. In addition, the combination of user models and the probabilistic approach allows to compute a relevant measure that cannot be computed in other frameworks: For a given feasibility degree, we can provide an upper bound of the probability of finding an error in the IUT. After a finite test suite is applied to an IUT, this measure allows to assess how confident we are the IUT is correct. Moreover, it implicitly provides a method to evaluate the quality of a test suite to evaluate an IUT with respect to a specification: If a given test suite is passed and it provides a *lower* upper bound (or an upper bound with a *higher* feasibility) than another suite that is also passed, then the former suite is preferred.

As future work, we plan to extend our framework to deal with *symbolic* probabilities that allow to denote ranges of probabilities instead of fix probabilities [5]. Besides, we will also introduce *stochastic* temporal delays to denote the time consumed by actions, that is, temporal delays defined in probabilistic terms [9].

Acknowledgements. We would like to thank the anonymous referees of this paper for their suggestions and valuable comments.

References

1. D. Cazorla, F. Cuartero, V. Valero, F. Pelayo, and J. Pardo. Algebraic theory of probabilistic and non-deterministic processes. *Journal of Logic and Algebraic Programming*, 55(1–2):57–103, 2003.
2. I. Christoff. Testing equivalences and fully abstract models for probabilistic processes. In *CONCUR'90, LNCS 458*, pages 126–140. Springer, 1990.
3. R. Cleaveland, Z. Dayar, S.A. Smolka, and S. Yuen. Testing preorders for probabilistic processes. *Information and Computation*, 154(2):93–148, 1999.
4. M. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.
5. N. López, M. Núñez, and I. Rodríguez. Specification, testing and implementation relations for symbolic-probabilistic systems. *Theoretical Computer Science*, 353(1–3):228–248, 2006.
6. R. de Nicola and M.C.B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.
7. M. Núñez. Algebraic theory of probabilistic processes. *Journal of Logic and Algebraic Programming*, 56(1–2):117–177, 2003.
8. M. Núñez and I. Rodríguez. Encoding `PAMR` into (timed) `EFSMs`. In *FORTE 2002, LNCS 2529*, pages 1–16. Springer, 2002.
9. M. Núñez and I. Rodríguez. Towards testing stochastic timed systems. In *FORTE 2003, LNCS 2767*, pages 335–350. Springer, 2003.
10. M. Núñez and D. de Frutos. Testing semantics for probabilistic LOTOS. In *Formal Description Techniques VIII*, pages 365–380. Chapman & Hall, 1995.
11. K. Sayre. Usage model-based automated testing of C++ templates. In *International Conference on Software Engineering. Proceedings of the first international workshop on Advances in model-based testing*, pages 1–5. ACM Press, 2005.
12. R. Segala. Testing probabilistic automata. In *CONCUR'96, LNCS 1119*, pages 299–314. Springer, 1996.
13. M. Stoelinga and F.W. Vaandrager. A testing scenario for probabilistic automata. In *ICALP 2003, LNCS 2719*, pages 464–477. Springer, 2003.
14. J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software – Concepts and Tools*, 17(3):103–120, 1996.
15. G.H. Walton, J.H. Poore, and C.J. Trammell. Statistical testing of software based on a usage model. *Software - Practice & Experience*, 25(1):97–108, 1995.
16. J.A. Whittaker and J.H. Poore. Markov analysis of software specifications. *ACM Transactions on Software Engineering and Methodology*, 2(1):93–106, 1993.