Specification, Testing and Implementation Relations for Symbolic-Probabilistic Systems ⋆

Natalia López, Manuel Núñez, and Ismael Rodríguez

Dept. Sistemas Informáticos y Programación
Facultad de Informática
Universidad Complutense de Madrid
E-28040 Madrid, Spain.
e-mail: {natalia,mn,isrodrig}@sip.ucm.es

Abstract

We consider the specification and testing of systems where probabilistic information is not given by means of fixed values but as intervals of probabilities. We will use an extension of the finite state machines model where choices among transitions labelled by the same input action are probabilistically resolved. We will introduce our notion of test and we will define how tests are applied to implementations under test. We will also present implementation relations to assess the conformance, *up to* a level of confidence, of an implementation to a specification. In order to define these relations we will take finite *samples* of executions of the implementation and compare them with the probabilistic constraints imposed by the specification. Finally, we will give an algorithm for deriving sound and complete test suites.

Key words: Symbolic-probabilistic finite state machines; Conformance testing; Test derivation and application.

1 Introduction

Formal methods try to keep a balanced trade-off between expressivity of the considered language and complexity of the underlying semantic framework. During the last years we have seen an evolution in the kind of systems that

⋆ This paper represents an extended, revised, and improved version of [1]. This research has been partially supported by the Spanish MCyT project *MASTER* (TIC2003-07848-C02-01), the Junta de Castilla-La Mancha project *DISMEF* (PAC-03-001), and the Marie Curie project *TAROT* (MRTN-CT-2003-505121).

formal methods are dealing with. In the beginning they mainly concentrated on the functional behavior of systems, that is, on what a system could/should do. In this regard, and considering specification formalism, we may mention the (original) notions of process algebras, Petri nets, and Moore/Mealy machines. Once the roots were well consolidated other considerations were taken into account. The next step was to deal with quantitative information such as the *time* underlying the performance of systems or the *probabilities* resolving the non-deterministic choices that a system may undertake. These characteristics gave raise to new models where time and/or probabilities were included (for example, [2,3,4,5,6,7,8,9,10,11,12] among many others).

One of the main criticisms about formal models including probabilistic information is the inability, in general, to accurately determine the actual values of the involved probabilities. In fact, using a process algebraic notation, the usual inclusion of probabilistic information is given by processes such as $P = a +_{\frac{1}{3}} b$. Intuitively, the process P indicates that if the choice between a and b must be resolved then a has probability $\frac{1}{3}$ of being chosen while b has probability $\frac{2}{3}$. However, there are situations where it is rather difficult to be so precise when specifying a probability. A very good example is the specification of *faulty channels* (e.g. the classical ABP [13]). These protocols contain information such as "the probability of losing the message is equal to 0.05." However, two questions may be immediately raised. First, why would one like to specify the exact probability of losing a message? Second, how can the specifier be sure that this is in fact the probability? In other words, to know the exact value seems as a very strong requirement. It would be more appropriate to say "the probability of losing the message is smaller than 0.05." Such a statement can be interpreted as "we know that the probability is low but we do not know the exact value." Moreover, this kind of probabilistic information, that we call *symbolic probabilities*, allows us to *refine* the model. For example, let us suppose that after experimentation with the system we have detected that some messages are lost, so that we can discard that the probability of losing a message is equal to zero, but *not many* messages were lost. By using the appropriate statistics machinery, mainly hypothesis contrasts and the Tchebyshev inequality, we may infer information such as "with probability 0.95 the *real* probability of losing the message belongs to the interval $[0.01, 0.03]$."

In order to specify probabilistic systems dealing with symbolic probabilities we will consider a probabilistic extension of *finite state machines* introduced in [14]. Intuitively, transitions in finite state machines indicate that if the machine is in a state s and receives an input i then it will produce an output o and it will change its state to s'. An appropriate notation for such a transition could be $s \xrightarrow{i/o} s'$. If we consider a probabilistic extension of finite state machines, a transition such as $s \xrightarrow{i/o}_p s'$ indicates that the probability with which the previous sequence of events happens is equal to p. By using symbolic

probabilities we go one step further. A transition such as $s \xrightarrow{i/o}_{\bar{p}} s'$ indicates that the probability with which the corresponding transition is performed belongs to the range given by $\bar{p}$. For instance, $\bar{p}$ may be the interval $[\frac{1}{4}, \frac{3}{4}]$ while the *real* probability is in fact 0.53.

An important issue when dealing with probabilities consists in fixing how different actions/transitions are probabilistically related. We consider a variant of the *reactive* interpretation of probabilities (see for example [4]) since it is the most suitable for our framework. Intuitively, a reactive interpretation imposes a probabilistic relation among transitions labelled by the same action. In contrast, choices between different actions are not quantified. In our setting we are able to express probabilistic relations between transitions outgoing from a given state and having the same input action (while the output action may vary). In the following example we illustrate this notion (a formal definition will be given in the next section).

Example 1.1 Let us consider that the unique transitions from a state s are

$$t_1 = s \xrightarrow{i_1/o_1}_{\bar{p}_1} s_1 \qquad t_2 = s \xrightarrow{i_1/o_2}_{\bar{p}_2} s_2 \qquad t_3 = s \xrightarrow{i_1/o_3}_{\bar{p}_3} s_2$$
$$t_4 = s \xrightarrow{i_2/o_1}_{\bar{p}_4} s_3 \qquad t_5 = s \xrightarrow{i_2/o_3}_{\bar{p}_5} s_1$$

If the environment offers the input action i_1 then the choice between t_1, t_2, and t_3 will be resolved according to some probabilities fulfilling the conditions $\bar{p}_1$, $\bar{p}_2$, and $\bar{p}_3$. All we know about these values is that they are within the imposed ranges, that they are non-negative, and that the sum of them equals 1. Something similar happens for the transitions t_4 and t_5. However, there does not exist any probabilistic relation between transitions labelled with different input actions (e.g. t_1 and t_4). □

Let us remark that, according to this framework, probabilistic information should not be the same for *specifications* of systems as for the systems themselves. In the first case we allow the specifier to use symbolic probabilities. In contrast, implementations will have fixed probabilities governing their behavior. For example, we may specify a *not-very-unfair coin* as a coin such that the probability of obtaining tails belongs to the interval $[0.4, 0.6]$ (and the same for faces). Given a *real* coin (i.e. an implementation) the probability p_t of obtaining tails (resp. p_f for faces) will be a fixed number (possibly unknown, but fixed). If $p_t, p_c \in [0.4, 0.6]$ and $p_t + p_c = 1$ then we will consider that the implementation conforms to the specification.

After introducing a proper formalism to deal with these concepts we will present a *testing methodology*. It follows a black-box testing approach (see e.g. [15,16]). That is, if we apply an input to an implementation under test (IUT) then we will observe an output and we may continue the testing procedure according to the observed result. However, we will not be able to *see*

the probabilities that the IUT has assigned to each of the choices. Thus, even though implementations will behave according to fixed probabilities we will not be able to *read* their values. In order to compute the probabilities associated with each choice of the implementation we will apply the same test several times and analyze the obtained responses. The set of tests used to check the suitability of an implementation will be constructed by using the given specification. By collecting the observations and comparing them with the symbolic probabilities of the specification, we will be able to assess the validity of the IUT. This comparison will be performed by using *hypothesis contrasts*. Hypothesis contrasts allow to (probabilistically) decide whether an observed sample follows the pattern given by a random variable. For example, even if we do not know the exact probabilities governing a *coin under test*, if we toss the coin 1000 times and we get 502 faces and 498 tails then we can infer, with a big probability, that the coin conforms with the specification of a *not-very-unfair coin*.

In addition to our testing methodology we will also introduce new implementation relations. First, we give two implementation relations where we require that the probabilities governing the implementation belong to the corresponding intervals of the specification. Unfortunately, these notions are useful only from a theoretical point of view. This is so because it is not possible to check, by using a finite number of observations, the correctness of the probabilistic behavior of an implementation with respect to a specification. Alternatively, we define implementation relations by following the ideas underlying our testing methodology. Intuitively, we do not request that the probabilities of the implementation belong to the corresponding intervals of the specification but that this fact happens *up to a certain probability*.

There is already significant work on testing preorders and equivalences for probabilistic processes [17,5,18,19,20,10,11]. However, most of these proposals follow the *de Nicola and Hennessy's style* [21,22], that is, the interaction between tests and processes is given by their concurrent execution, synchronizing on a set of actions. For example, we may say that two processes are *equivalent* if for any test T, out of a set of tests $\mathcal{T}$, the application of T to each of the processes returns an *equivalent* result. These frameworks are not very related to ours since our main task is to determine whether an implementation conforms to a specification. Even though some of the aforementioned preorders can be used for this purpose, our approach is more based on *pushing buttons*: The test applies an input to the IUT and we check whether the returned output is expected by the specification. Moreover, none of these papers use the kind of *statistical testing* that we use, that is, applying the same test several times and extracting conclusions about the probabilities governing the implementation. In this sense, the work closest to ours is reported in [23,24]. In fact, we take the statistical machinery from [24], where a testing framework to deal with systems presenting time information given by stochastic time is introduced.

In [23] the authors present a testing scenario for a notion of probabilistic automata. In order to replicate the same experiment several times they introduce a *reset* button. Since this button is the only way to influence the behavior of the IUT, they can only capture trace-like semantics. Actually, their equivalence coincides with a certain notion of trace distribution equivalence. In our case we can capture branching-like behavior since different tests can guide the IUT through different paths. Finally, it is worth to mention that all of the previous approaches use fixed probabilities.

The rest of the paper is organized as follows. In Section 2 we present the probabilistic finite state machines model. In Section 3 we introduce our first implementation relations and show that, regardless of their elegant definition, they are not adequate from the practical point of view. Some basic concepts about hypothesis contrasts that will be needed in the rest of the paper will be presented in Section 4. For the sake of readability, the definition of a specific hypothesis contrast mechanism will be presented in the appendix of the paper. In Section 5 we present the notion of test and define how they are applied to implementations. In Section 6 we introduce implementation relations based on samples. The underlying idea is that a hypothesis contrast is used to assess whether the observed behavior corresponds, *up to* a certain confidence, to the probabilistic behavior defined in the specification. In Section 7 we present an algorithm to derive sound and complete test suites with respect to two of the relations presented in the previous section. In fact, we cannot properly use the term *completeness*; we should use the most accurate expression *completeness up to a certain confidence level*. In Section 8 we present our conclusions. Finally, in the appendix of this paper we give some basic statistical concepts that are (abstractly) used along the paper, including a full definition of a specific hypothesis contrast mechanism.

2 Probabilistic Finite State Machines

In this section we introduce our notion of probabilistic finite state machines. As we have previously mentioned, probabilistic information will not be given by fixed values of probabilities but by introducing certain constraints on the considered probabilities. By taking into account the inherent nature of probabilities we will consider that these constraints are given by intervals contained in $(0, 1] \subseteq \mathbb{R}$.

Definition 2.1 We define the set of *symbolic probabilities*, denoted by `simbP`,

as the following set of intervals

$$\texttt{simbP} = \left\{ \$p_1, p_2\& \;\middle|\; \begin{array}{l} p_1, p_2 \in [0,1] \ \wedge \ p_1 \leq p_2 \ \wedge \\ \$ \in \{\ (, [\ \} \ \wedge \ \& \in \{\ ),]\ \} \ \wedge \\ 0 \notin \$p_1, p_2\& \ \wedge \ \$p_1, p_2\& \neq \emptyset \end{array} \right\}$$

If we have a symbolic probability such as $[p, p]$, with $0 < p \leq 1$, we simply write p.

Let $\overline{p}_1, \ldots, \overline{p}_n \in \texttt{simbP}$ be symbolic probabilities such that for all $1 \leq i \leq n$ we have $\overline{p}_i = \$_i p_i, q_i \&_i$, with $\$_i \in \{\ (, [\ \}$ and $\&_i \in \{\ ),]\ \}$. We define the *product* of $\overline{p}_1, \ldots, \overline{p}_n$, denoted by $\prod \overline{p}_i$, as the symbolic probability $\$ \prod p_i, \prod q_i \&$. The limits of the interval are defined as:

$$\$ = \begin{cases} (& \text{if } \exists 1 \leq i \leq n : \ \$_i = (\\ [& \text{otherwise} \end{cases} \qquad \& = \begin{cases}) & \text{if } \exists 1 \leq i \leq n : \ \&_i =) \\ ] & \text{otherwise} \end{cases}$$

We define the *addition* of $\overline{p}_1, \ldots, \overline{p}_n$, denoted by $\sum \overline{p}_i$, as the symbolic probability $\$ \sum p_i, \min\{1, \sum q_i\} \&$. The limits of the interval are defined as before.

□

The previous definition expresses that a symbolic probability $\overline{p}$ is any non-empty (open or closed) interval contained in $(0, 1]$. In particular, we will not allow transitions with probability 0 because this value would complicate (even more) our model since we would have to deal with priorities. [1] We have also defined how to *multiply* and *add* symbolic probabilities. The maximal (resp. minimal) bound of the resulting interval is obtained by operating over the maximal (resp. minimal) bounds of the considered intervals. In the case of addition of symbolic probabilities the maximal bound of the interval is set to the minimum between 1 and the addition of the maximal bounds because this last number could be greater than 1. Multiplication of symbolic probabilities will be used to compute the symbolic probability of having several consecutive events. Intuitively, if $p_1 \in \overline{p}_1$ and $p_2 \in \overline{p}_2$, that is, p_1 and p_2 are possible values that two symbolic probabilities can have, then $p_1 \cdot p_2 \in \prod \overline{p}_i$. Addition of symbolic probabilities will be used to ensure that 1 is a possible probability when considering all the transitions fulfilling certain conditions. Let us remark that other possibilities to develop a model of symbolic probabilities can be taken just by modifying the previous definition and keeping the rest of the theory as it is. For example, in order to consider *fix* non-symbolic probabilities

[1] The interested reader can check [25] where different approaches for introducing priorities are reviewed.

it is enough to set $\texttt{simbP} = (0, 1]$, while a more elaborated notion of symbolic probabilities (e.g. sets of probabilities) would need that the new set simbP is close with respect to the corresponding notions of addition and multiplication. In the following definition we introduce our notion of probabilistic finite state machine. We assume that $\mid X \mid$ denotes the cardinality of the set X.

Definition 2.2 A *Probabilistic Finite State Machine*, in short PFSM, is a tuple $M = (S, I, O, \delta, s_0)$ where

- S is a finite set of states,
- I and O denote the sets of input and output actions, respectively,
- $\delta \subseteq S \times I \times O \times \texttt{simbP} \times S$ is the set of transitions, and
- s_0 is the initial state.

For all state $s \in S$ and input action $i \in I$, we consider that the set

$$\alpha_{s,i} = \{s \xrightarrow{i/o}_{\overline{p}} s' \mid \exists\, o \in O, \overline{p} \in \texttt{simbP}, s' \in S : s \xrightarrow{i/o}_{\overline{p}} s' \in \delta\}$$

fulfills the following two conditions:

- If $\mid \alpha_{s,i} \mid > 1$ then for all $s \xrightarrow{i/o}_{\overline{p}} s' \in \alpha_{s,i}$ we have that $1 \notin \overline{p}$ and
- $1 \in \sum\{\overline{p} \mid \exists\, o \in O, s' \in S : s \xrightarrow{i/o}_{\overline{p}} s' \in \alpha_{s,i}\}$.

□

We will usually denote transitions as $(s, i, o, \overline{p}, s')$ by $s \xrightarrow{i/o}_{\overline{p}} s'$. Intuitively, a transition $s \xrightarrow{i/o}_{\overline{p}} s'$ indicates that if the machine is in state s and receives the input i then, with a probability belonging to the interval $\overline{p}$, the machine emits the output o and evolves into s'. Let us comment the restrictions introduced at the end of the previous definition. The first constraint indicates that a symbolic probability such as $\overline{p} = \$p, 1]$ can appear in a transition $s \xrightarrow{i/o}_{\overline{p}} s' \in \delta$ only if it is the unique transition for s and i. Let us note that if there would exist two transitions $s \xrightarrow{i/o}_{\overline{p}} s', s \xrightarrow{i/o'}_{\overline{p}'} s'' \in \delta$ and the probability of one of them (say $\overline{p}$) includes 1, the only situation that makes sense is that the probability associated to the other transition (that is, $\overline{p}'$) includes 0, which is forbidden. Regarding the second condition, since the *real* probabilities for each state $s \in S$ and for each input $i \in I$ should add 1, then 1 must be within the lower and upper bounds of the associated symbolic probabilities.

Next we define some additional properties that our state machines will sometimes fulfill.

Definition 2.3 Let $M = (S, I, O, \delta, s_0)$ be a PFSM. We say that M is *input-enabled* if for all $s \in S$ and $i \in I$ there exist $s' \in S$, $o \in O$, and $\overline{p} \in \texttt{simbP}$ such that $(s, i, o, \overline{p}, s') \in \delta$. We say that M is *deterministically observable* if

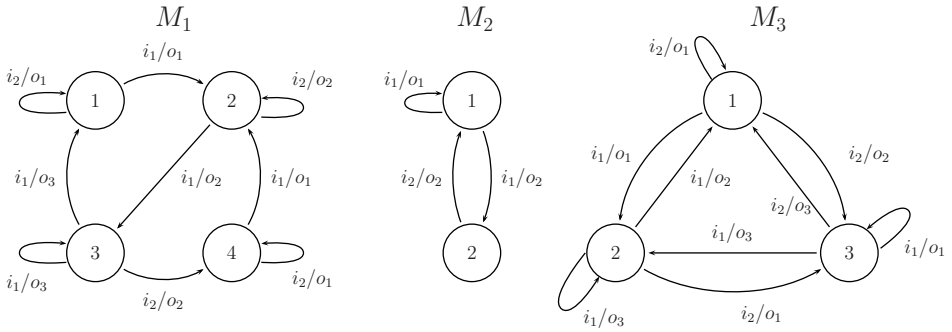

Figure 1. Examples of PFSM.

for all $s \in S$, $i \in I$, and $o \in O$ there do not exist two different transitions $(s, i, o, \overline{p}_1, s_1), (s, i, o, \overline{p}_2, s_2) \in \delta$. □

First, let us remark that the previous concepts are independent of the probabilistic information appearing in state machines. Besides, the notion of deterministically observable is different from the more restricted notion of deterministic finite state machine. In particular, we allow transitions from the same state labelled by the same input action, as far as the outputs are different.

Example 2.4 Let us consider the (probabilistic) finite state machines depicted in Figure 1. For the sake of clarity we have not included probabilistic information in the graphs. Let $M_3 = (\{1, 2, 3\}, \{i_1, i_2\}, \{o_1, o_2, o_3\}, \delta, 1)$. Next we define the set of transitions δ. For the first state we suppose the transitions $(1, i_1, o_1, 1, 2)$, $(1, i_2, o_1, \overline{p}_1, 1)$, and $(1, i_2, o_2, \overline{p}_2, 3)$, where $\overline{p}_1 = (0, \frac{1}{2}]$, $\overline{p}_2 = [\frac{1}{3}, 1)$. Let us remind that we denote the *interval* $[1, 1]$ simply by 1. We also know that the real probabilities associated with the last two transitions, say p_1 and p_2, are such that $p_1 + p_2 = 1$. A similar assignment of symbolic probabilities can be done to the rest of transitions appearing in the graph.

Regarding the notions of input-enabling and deterministically observable, we have that M_1 fulfills the first of the properties but not the second one (there are two transitions outgoing from the state 3 labelled by i_1/o_3). The first property does not hold in M_2 (there is no outgoing transition labelled by i_2 from the state 2) while the second one does. Finally, M_3 fulfills both properties. □

As usually, we need to consider not only single evolutions of a PFSM but also sequences of transitions. Thus, we introduce the notion of (probabilistic) trace. We will associate probabilities to traces. The probability of a trace will be obtained by multiplying the probabilities of all transitions involved in the trace.

Definition 2.5 Let $M = (S, I, O, \delta, s_0)$ be a PFSM. Let $s, s' \in S$, $i_1, \ldots, i_n \in I$,

$o_1, \ldots, o_n \in O$, and $\overline{p} \in \texttt{simbP}$. We write $s \xRightarrow{(i_1/o_1,\ldots,i_n/o_n)}_{\overline{p}} s'$ if there exist $s_1, \ldots, s_{n-1} \in S$ and $\overline{p}_1, \ldots, \overline{p}_n \in \texttt{simbP}$ such that

$$s \xrightarrow{i_1/o_1}_{\overline{p}_1} s_1 \xrightarrow{i_2/o_2}_{\overline{p}_2} s_2 \cdots s_{n-1} \xrightarrow{i_n/o_n}_{\overline{p}_n} s'$$

and $\overline{p} = \prod \overline{p}_i$.

We say that $\rho = (i_1/o_1, \ldots, i_n/o_n)$ is a *non-probabilistic trace*, or simply a *trace*, of M if there exist $s' \in S$ and $\overline{p} \in \texttt{simbP}$ such that $s_0 \xRightarrow{\rho}_{\overline{p}} s'$.

Let $\rho = (i_1/o_1, \ldots, i_n/o_n)$ and $\overline{p} \in \texttt{simbP}$. We say that $\overline{\rho} = (\rho, \overline{p})$ is a *probabilistic trace* of M if there exists $s' \in S$ such that $s_0 \xRightarrow{\rho}_{\overline{p}} s'$.

We denote by $\texttt{Traces}(M)$ and $\texttt{pTraces}(M)$ the sets of non-probabilistic and probabilistic traces of M, respectively. □

3 Probabilistic Implementation Relations

In this section we introduce our first two implementation relations. We will consider that both specifications and implementations are given by deterministically observable PFSMs. Moreover, we will assume that PFSMs representing implementations are input-enabled. The idea is that an implementation should not be able to refuse an input provided by a test. As we have commented before, we assume that IUTs are *black boxes*. Thus, no information can be known about their internal behavior/structure: We may only apply inputs and observe the returned outputs. In addition, let us remark that the *symbolic* probabilities appearing in implementations follow the pattern $[p, p]$ (or simply p), for some $p \in (0, 1]$. That is, they are indeed *fixed* probabilities. While specifications are abstract entities where symbolic probabilities allow us to represent different scenarios (one for each probability within the intervals) in a compact fashion, implementations represent concrete machines. Hence, even though observations will not give us the actual probability associated to a transition in an implementation, we may rely on the fact that the probability is indeed fixed.

Regarding the performance of actions, our implementation relations follow the classical pattern of formal conformance relations defined in systems distinguishing between inputs and outputs (see e.g. [26]). That is, an IUT conforms to a specification $\mathcal{S}$ if for any possible evolution of $\mathcal{S}$ the outputs that the IUT may perform after a given input are a subset of those for the specification. Besides, the first relation will require that the probability of any trace of the IUT is *within* the corresponding (symbolic) probability of the specification for this trace.

Definition 3.1 Let $\mathcal{S}$ and $\mathcal{I}$ be PFSMs. We say that $\mathcal{I}$ *non-probabilistically conforms* to $\mathcal{S}$, denoted by $\mathcal{I}$ $\mathtt{conf}$ $\mathcal{S}$, if for all $\rho = (i_1/o_1, \ldots, i_n/o_n) \in \mathtt{Traces}(\mathcal{S})$, with $n \geq 1$, we have

$$\rho' = (i_1/o_1, \ldots, i_{n-1}/o_{n-1}, i_n/o'_n) \in \mathtt{Traces}(\mathcal{I}) \text{ implies } \rho' \in \mathtt{Traces}(\mathcal{S})$$

We say that $\mathcal{I}$ *probabilistically conforms to* $\mathcal{S}$ *considering traces*, denoted by $\mathcal{I}$ $\mathtt{conf}_p$ $\mathcal{S}$, if $\mathcal{I}$ $\mathtt{conf}$ $\mathcal{S}$ and for each $\overline{\rho} = (\rho, p) \in \mathtt{pTraces}(\mathcal{I})$ we have

$$\rho \in \mathtt{Traces}(\mathcal{S}) \text{ implies } \exists\, \overline{q} \in \mathtt{simbP} :\ (\rho, \overline{q}) \in \mathtt{pTraces}(\mathcal{S}) \ \wedge\ p \in \overline{q}$$

□

Intuitively, the idea underlying the definition of the non-probabilistic conformance relation $\mathtt{conf}$ is that the implementation $\mathcal{I}$ does not *invent* anything for those inputs that are *specified* in the specification (this notion has been previously used in the context of finite state machines in [27,24]). This condition is also required in the probabilistic case: We check probabilistic traces only if they can be performed by the specification.

The problem behind the previous implementation relation is that we have no access to the probabilities governing the transitions of the IUT. So, we are not able to check whether a given IUT probabilistically conforms to a specification. However, we may get *approximations* of the probabilities controlling the IUT by applying a test several times and computing the empirical ratio associated with the different decision points of the IUT. This is the purpose of the next sections. However, before introducing these new relations, that we call *based on samples*, we would like to discuss the complexity underlying the checking of implementation relations based on $\mathtt{conf}_p$.

As we have previously mentioned, the probabilistic features of implementations will be estimated on the basis of observed samples. Thus, we will need to perform a high amount of experimental samples to obtain good estimations. Actually, collecting useful samples to estimate the probability of a given trace becomes harder as its length grows. This is so because the probability of following a given path decays with the number of probabilistic decisions. So, if we have to obtain enough samples to infer reliable probabilistic information about a trace, the number of experiments will increase with the length of the trace. However, this problem could be bypassed if we assume the following hypothesis about implementations:

> *"In the case that the non-probabilistic requirements are fulfilled then we assume that the IUT is implemented by a* PFSM *equal to that of the specification, up to probabilistic information"*

Let us denote the previous hypothesis by Hyp (a formal presentation of this

requirement is presented in Definition 3.2). Given the fact that we are assuming non-probabilistic conformance, this condition is slightly stronger than the combination of the two usual conditions in testing: The IUT has not redundant states and the number of states of the IUT is not greater than that of the specification. Intuitively, if this hypothesis is assumed then a second implementation relation can be informally formulated as:

> *"The IUT conforms to the specification if the non-probabilistic requirements are fulfilled and the probability distributions, one for each input action, associated with each state of the implementation match the ones of the equivalent state in the specification"*

Thus, the samples collected by stimulating the implementation will be used to collect information about the state of the implementation that *matches* the corresponding state in the specification. Clearly, the efficiency of the experimental samples is dramatically boosted since *all* the samples, for all the traces, will collaborate together to estimate the probabilities associated with each state of the implementation.

Let us remark that the assumption of the previous hypothesis, when the IUT does not fulfill it, could result in mixing probabilistic information of the implementation that actually cannot be mixed. For instance, a loop in the specification could not appear in the implementation. Thus, it would be incorrect to add the experimental samples to the *same* state of the implementation every time a trace completes a new round according to the states of the specification.

Definition 3.2 Let $\mathcal{S} = (S, I, O, \delta, s_0)$ and $\mathcal{I} = (V, I, O, \delta', v_0)$ be PFSMs. We say that the hypothesis Hyp holds if $\mathcal{I}$ conf $\mathcal{S}$ and there exists a bijection $\mathtt{cSt} : S \longrightarrow V$ such that $(s_1, i, o, s_2) \in \delta$ iff $(\mathtt{cSt}(s_1), i, o, \mathtt{cSt}(s_2)) \in \delta'$. In this case we say that, for each state $s \in S$, $\mathtt{cSt}(s) \in V$ is the state *corresponding to s in the implementation.*

We say that $\mathcal{I}$ *probabilistically conforms to* $\mathcal{S}$ *considering* Hyp, denoted by $\mathcal{I}\ \mathtt{conf}_p^{\mathtt{Hyp}}\ \mathcal{S}$, if $\mathcal{I}$ conf $\mathcal{S}$ and for each state $s \in S$, $i \in I$, and $o \in O$ we have that if there exist $s' \in S$ and $\overline{p} \in \mathtt{simbP}$, with $(s, i, o, \overline{p}, s') \in \delta$, then there exists $p' \in (0..1]$ such that $(\mathtt{cSt}(s), i, o, p', \mathtt{cSt}(s')) \in \delta'$ and $p' \in \overline{p}$. □

Let us remark that this new notion presents the same problem as the previous one: It cannot be checked in practice. However, as we will show in the next sections, by following its pattern we will have a better confidence in the estimation of the probabilities controlling the IUT. The following result relates both implementation relations.

Lemma 3.3 Let $\mathcal{S} = (S, I, O, \delta, s_0)$ and $\mathcal{I} = (V, I, O, \delta', v_0)$ be PFSMs and let us assume that Hyp holds. We have that $\mathcal{I}\ \mathtt{conf}_p^{\mathtt{Hyp}}\ \mathcal{S}$ implies $\mathcal{I}\ \mathtt{conf}_p\ \mathcal{S}$.

Proof: Let us prove the implication by contrapositive. Let us suppose that $\mathcal{I}\ \mathtt{conf}_p\ \mathcal{S}$ does not hold. We have two possibilities: Either $\mathcal{I}\ \mathtt{conf}\ \mathcal{S}$ is false or the probabilistic conditions imposed by $\mathtt{conf}_p$ do not hold. In the first case, we have that $\mathcal{I}\ \mathtt{conf}_p^{\mathtt{Hyp}}\ \mathcal{S}$ does not hold, in contrast with our assumption, since $\mathcal{I}\ \mathtt{conf}\ \mathcal{S}$ is also a requirement for $\mathcal{I}\ \mathtt{conf}_p^{\mathtt{Hyp}}\ \mathcal{S}$. In the second case, let us assume $\mathcal{I}\ \mathtt{conf}\ \mathcal{S}$ and let us suppose that there exists a probabilistic trace $\overline{\rho} = (\rho, q) \in \mathtt{pTraces}(\mathcal{I})$ such that $\rho \in \mathtt{Traces}(\mathcal{S})$ but there does not exist a probabilistic trace $(\rho, \overline{p}) \in \mathtt{pTraces}(\mathcal{S})$ with $q \in \overline{p}$. Since $\mathtt{Hyp}$ holds, there exists $(\rho, \overline{p}') \in \mathtt{pTraces}(\mathcal{S})$ with $q \notin \overline{p}'$. Let us consider the trace $\rho = (i_1/o_1, \ldots, i_n/o_n)$, where

$$s_0 \xrightarrow{i_1/o_1}_{\overline{p}_1} s_1 \cdots\ s_{n-1} \xrightarrow{i_n/o_n}_{\overline{p}_n} s_n$$
$$v_0 \xrightarrow{i_1/o_1}_{q_1} \mathtt{cSt}(s_1) \cdots\ \mathtt{cSt}(s_{n-1}) \xrightarrow{i_n/o_n}_{q_n} \mathtt{cSt}(s_n)$$

with $\overline{p}' = \prod \overline{p}_i$ and $q = \prod q_i$. If for all $1 \leq j \leq n$ we have $q_j \in \overline{p}_j$ then $q \in \overline{p}'$, which represents a contradiction. Hence, there exists $1 \leq k \leq n$ such that $q_k \notin \overline{p}_k$. So, we have two transitions $(s_{k-1}, i_k, o_k, \overline{p}_k, s_k) \in \delta$ and $(\mathtt{cSt}(s_{k-1}), i_k, o_k, q_k, \mathtt{cSt}(s_k)) \in \delta'$ such that $q_k \notin \overline{p}_k$. Since both $\mathcal{S}$ and $\mathcal{I}$ are deterministically observable we can conclude that there do not exist two transitions $(s_{k-1}, i_k, o_k, \overline{p}'_k, s'') \in \delta$ and $(\mathtt{cSt}(s_{k-1}), i_k, o_k, q'_k, \mathtt{cSt}(s'')) \in \delta'$ with $q'_k \in \overline{p}'_k$. Thus, $\mathcal{I}\ \mathtt{conf}_p^{\mathtt{Hyp}}\ \mathcal{S}$ does not hold and we obtain a contradiction. □

It is worth to point out that the reverse implication is false, that is, $\mathcal{I}\ \mathtt{conf}_p\ \mathcal{S}$ does not imply $\mathcal{I}\ \mathtt{conf}_p^{\mathtt{Hyp}}\ \mathcal{S}$. This is because the effect of the *deviation* of the probability of a single transition in a trace may be *compensated* by other transitions of the trace. Hence, the relation $\mathtt{conf}_p^{\mathtt{Hyp}}$ is stronger than the relation $\mathtt{conf}_p$. Next, we present a counterexample for this reverse implication.

Example 3.4 Let $\mathcal{S} = (S, I, O, \delta, s_0)$ and $\mathcal{I} = (V, I, O, \delta', v_0)$, where $I = \{a\}$, $O = \{b, c, d\}$, $S = \{s_0, s_1, s_2\}$, $V = \{v_0, v_1, v_2\}$, and

$$\delta = \{s_0 \xrightarrow{a/b}_{[0.2,0.6]} s_1, s_0 \xrightarrow{a/d}_{[0.3,0.7]} s_2, s_1 \xrightarrow{a/b}_{[0.4,0.6]} s_2, s_1 \xrightarrow{a/d}_{[0.3,0.8]} s_2\}$$
$$\delta' = \{v_0 \xrightarrow{a/b}_{0.5} v_1, v_0 \xrightarrow{a/d}_{0.5} v_2, v_1 \xrightarrow{a/b}_{0.2} v_2, v_1 \xrightarrow{a/d}_{0.8} v_2\}$$

It is immediate to establish a bijection between states of $\mathcal{I}$ and states of $\mathcal{S}$ such that the availability of transitions in each pair is the same. As a result, it is clear that $\mathcal{I}\ \mathtt{conf}\ \mathcal{S}$. The probabilistic traces of $\mathcal{S}$ are

$$((a/b), [0.2, 0.6]),\ ((a/d), [0.3, 0.7]),\ ((a/b, a/b), [0.08, 0.36]),\ \text{and}$$
$$((a/b, a/d), [0.06, 0.48])$$

while the probabilistic traces of $\mathcal{I}$ are

$$((a/b), 0.5),\ ((a/d), 0.5),\ ((a/b, a/b), 0.1),\ \text{and}\ ((a/b, a/d), 0.4)$$

Since the probability of any trace in $\mathcal{I}$ fits into the symbolic probability of the corresponding trace in $\mathcal{S}$, we have that $\mathcal{I}\ \mathtt{conf}_p\ \mathcal{S}$. Let us note that the transition $s_1 \xrightarrow{a/b}_{[0.4,0.6]} s_2$ of $\mathcal{S}$ requires that the probability of the corresponding transition in $\mathcal{I}$ is between 0.4 and 0.6. However, that transition in $\mathcal{I}$ is $v_1 \xrightarrow{a/b}_{0.2} v_2$. So, $\mathcal{I}\ \mathtt{conf}_p^{\mathtt{Hyp}}\ \mathcal{S}$ does not hold. □

4 Hypothesis Contrasts

In this section we introduce some notation related to hypothesis contrasts. These concepts and definitions will be used along the rest of the paper. Hypothesis contrasts will be presented abstractly, that is, we will not consider the specific operations needed to perform a hypothesis contrast but we will focus only on using their output. Additionally, an operational definition of these concepts will be given in the Appendix of this paper.

In the following definition we present some concepts that are independent from the peculiarities of our framework. These concepts will be particularized to our framework later, in Definition 4.2. We call *event* to any reaction we can detect from a system or environment. A *sample* contains information about the number of times we have detected each event along a set of observations. Besides, we associate a random variable with each set of events. Its purpose is to provide the theoretical (*a priori*) probability of each event in the set. In our framework, these random variables will be inferred from the PFSMs denoting the (ideal) probabilistic behavior of systems, while samples will be collected by interacting with the implementation under test. We will consider a variant of random variable allowing to deal with *symbolic* probabilities. Besides, we provide a function that returns the confidence we have that a sample of events has been produced according to a given random variable. This function encapsulates the subjacent hypothesis contrast.

Definition 4.1 Let $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$ be a set of *events*. A *sample* of $\mathcal{A}$ is a set $J = \{(\alpha_1, m_1), \ldots, (\alpha_n, m_n)\}$ where for all $1 \leq i \leq n$ we have that $m_i \in \mathbb{N}_+$ represents the number of times the event α_i has been observed.

Let $\xi : \mathcal{A} \to \mathtt{simbP}$ be a function such that $1 \in \sum_{\alpha \in \mathcal{A}} \xi(\alpha)$. We say that ξ is a *symbolic random variable* for the set of events $\mathcal{A}$. We denote the set of symbolic random variables for the set of events $\mathcal{A}$ by $\mathcal{RV}(\mathcal{A})$. We denote the set of symbolic random variables for any set of events by $\mathcal{RV}$.

Given the symbolic random variable ξ and the sample J we denote the *confidence* of ξ on J by $\gamma(\xi, J)$. □

We assume that $\gamma(\xi, J)$ takes values in the interval $[0, 1]$. Intuitively, bigger val-

ues of $\gamma(\xi, J)$ denote that the observed sample J is more likely to be produced by the symbolic random variable ξ. There exist several hypothesis contrasts to compute these confidence levels. In the appendix of this paper we show one of them to indicate how the notion of confidence may be formally defined.

In the next definition we particularize the previous notions in the context of our framework. Basically, we will consider two different ways to collect samples from a system. They will follow the ideas underlying the two implementation relations defined in the previous section, that is, they will associate samples to either traces or inputs leaving each state, respectively. In the first case, given a sequence of inputs we consider the sequence of outputs that the system can return. Hence, the set of events are those sequences of outputs that could be produced in response. The random variable to denote the theoretical probability of each event is computed by considering the symbolic probability of the corresponding trace in the specification. Alternatively, by using our second approach, in order to collect samples we may observe the outputs produced by the system from a state in response to a single input. In this case, the random variable is constructed by considering the corresponding state and input in the specification.

Definition 4.2 Let $M = (S, I, O, \delta, s_0)$ be a `PFSM` and $\pi = (i_1, \ldots, i_n)$ be a sequence of inputs. The *set of trace events* associated to M with respect to π, denoted by $\texttt{TraceEvents}(M, \pi)$, is defined as

$$\texttt{TraceEvents}(M, \pi) = \{(o_1, \ldots, o_n) \,|\, (i_1/o_1, \ldots, i_n/o_n) \in \texttt{Traces}(M)\}$$

The *symbolic random variable* associated to the previous events, denoted by ξ^{π}_{M}, fulfills that for all sequence $(o_1, \ldots, o_n) \in \texttt{TraceEvents}(M, \pi)$ we have $\xi^{\pi}_{M}(o_1, \ldots, o_n) = \overline{p}$, being $((i_1/o_1, \ldots, i_n/o_n), \overline{p}) \in \texttt{pTraces}(M)$.

Let $s \in S$ and $i \in I$. The *set of state events* associated to M, starting from s by using input i, denoted by $\texttt{StateEvents}(M, s, i)$, is defined as

$$\texttt{StateEvents}(M, s, i) = \{o \mid \exists\, o \in O, s' \in S : s \xrightarrow{i/o}_{\overline{p}} s' \in \delta\}$$

The *symbolic random variable* associated to the previous events, denoted by $\xi^{s,i}_{M}$, fulfills that for all $o \in \texttt{StateEvents}(M, s, i)$ we have $\xi^{s,i}_{M}(o) = \overline{p}$, where $s \xrightarrow{i/o}_{\overline{p}} s' \in \delta$. □

5 Testing Probabilistic Systems

In this section we introduce the notion of test and we present how they are applied to implementations. We will follow the two approaches introduced in

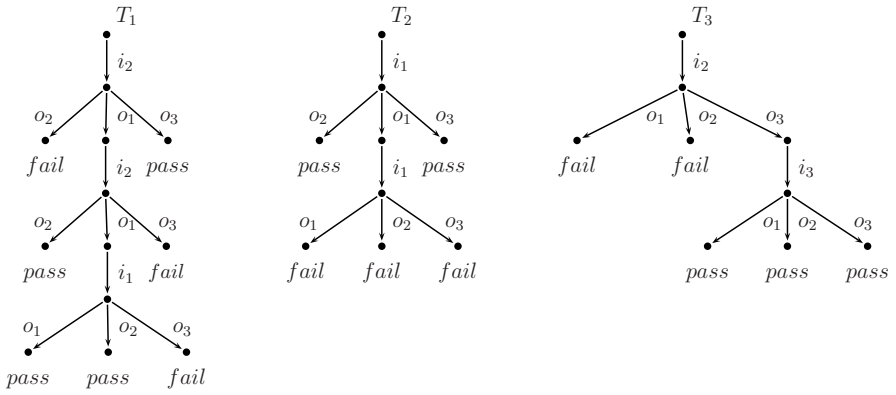

Figure 2. Examples of Tests.

Section 3 to define implementation relations, that is, to consider either traces or the behavior in each state. In our context, to test an IUT consists in applying a sequence of inputs to the IUT. After an input is offered to the IUT by the test, an output is received. We check whether the output belongs to the set of expected ones. If this is the case, either a pass signal is emitted (indicating successful termination) or the testing process continues by applying another input. Otherwise, a fail signal is produced, the testing process stops, and we conclude that the implementation does not conform to the specification.

As we indicated in the introduction, the methodology to *guess* the probabilities associated with each output action in response to some input action consists in applying several times the same test. If we are testing an IUT with input and output sets I and O, respectively, tests are deterministic acyclic I/O labelled transition systems (i.e. trees) with a strict alternation between an input action and the whole set of output actions. A branch labelled by an output action can be followed by a leaf or by another input action. Moreover, leaves of the tree represent either *successful* or *failure* states. We will collect a sample of successes and failures of the test (one for each test execution) and, in the successful case, the sequences of input/output actions performed. With this information we will experimentally compute the probabilities associated with input actions in the IUT. In addition, successful states will have a symbolic random variable associated with them. This random variable will denote the *probabilistic constraint* imposed in the test for the trace leading to that state. Basically, a hypothesis contrast will compare the samples collected for that event with the probabilistic constraint imposed by the test.

Definition 5.1 A *test* is a tuple $T = (S, I, O, \delta, s_0, S_I, S_O, S_F, S_P, \zeta)$ where S is a finite set of states, I and O, with $I \cap O = \emptyset$, are the sets of input and output actions, respectively, $\delta \subseteq S \times I \cup O \times S$ is the transition relation, $s_0 \in S$ is the initial state, and the sets $S_I, S_O, S_F, S_P \subseteq S$ are a partition of S. The transition relation and the sets of states fulfill the following conditions:

- S_I is the set of *input* states. We have that $s_0 \in S_I$. For each input state $s \in S_I$ there exists a unique outgoing transition $(s, i, s') \in \delta$. For this transition we have that $i \in I$ and $s' \in S_O$.
- S_O is the set of *output* states. For each output state $s \in S_O$ and each output action $o \in O$ there exists a unique state $s' \in S$ such that $(s, o, s') \in \delta$; in each case, $s' \notin S_O$. Besides, for each output state $s \in S_O$ there exists at most one state $s' \in S_I$ such that $(s, o, s') \in \delta$, for some $o \in O$. Moreover, there do not exist $i \in I$ and $s' \in S$ such that $(s, i, s') \in \delta$.
- S_F and S_P are the sets of *fail* and *pass* states, respectively. We say that these states are *terminal*, that is, $s \in S_F \cup S_P$ implies that there do not exist $a \in I \cup O$ and $s' \in S$ such that $(s, a, s') \in \delta$.

Finally, $\zeta : S_P \longrightarrow \mathcal{R}V$ is a function associating passing states with (symbolic) random variables.

We say that the test T is *valid* if the graph induced by T is a tree with root at its initial state s_0. □

Next we define the set of traces that a test can perform. These traces are sequences of input/output actions reaching terminal states. Depending on the final state we will classify them as either *successful* or *failure* traces.

Definition 5.2 Let $T = (S, I, O, \delta, s_0, S_I, S_O, S_F, S_P, \zeta)$ be a test, $s \in S$, and $\rho = (i_1/o_1, \ldots, i_r/o_r)$ be a sequence of input/output actions. We say that ρ is a *trace of* T *reaching* s, denoted by $T \stackrel{\rho}{\Longrightarrow} s$, if $s \in S_F \cup S_P$ and there exist states $s_{12}, s_{21}, s_{22}, \ldots s_{r1}, s_{r2} \in S$ such that $(s_0, i_1, s_{12}), (s_{r2}, o_r, s) \in \delta$, and for all $2 \leq j \leq r$ we have $(s_{j1}, i_j, s_{j2}), (s_{(j-1)2}, o_{j-1}, s_{j1}) \in \delta$. □

The next definition presents some auxiliary predicates that we will use during the rest of the paper. While the first two notions are easy to understand, the last one needs some additional explanation. Given a sequence of input actions σ and a set H of pairs (trace, natural number), the function $\texttt{IPrefix}(H, \sigma)$ returns another set of pairs including all subtraces belonging to H such that their sequence of input actions matches σ. The number attached to each trace corresponds to the number of traces belonging to H *beginning* with the given sequence of inputs σ. Before defining this concept, we present a simple example to show how this function works.

Example 5.3 Let us consider the sequence of input actions $\sigma = (i_1)$ and the set

$$H = \left\{ \begin{array}{l} ((i_1/o_1, i_2/o_1), 1), ((i_1/o_2, i_1/o_2), 2), \\ ((i_1/o_2, i_2/o_1), 3), ((i_2/o_1, i_2/o_2), 4) \end{array} \right\}$$

The application of the function $\texttt{IPrefix}(H, \sigma)$ returns the set of pairs $H' = \{((i_1/o_1), 1), ((i_1/o_2), 5)\}$. □

Given a sample of executions from an implementation, we will use this function to compute the number of times that the implementation has performed each sequence of outputs in response to some sequence of inputs. Let us note that if the sequence of outputs $(o_1, \ldots, o_n)$ has been produced in response to the sequence of inputs $(i_1, \ldots, i_n)$ then, for all $j \leq n$, we know that the sequence of outputs $(o_1, \ldots, o_j)$ has been produced in response to $(i_1, \ldots, i_j)$. Hence, the observation of a trace is useful to compute the number of instances of its prefixes. In the next definition, the symbols $\{\!|$ and $|\!\}$ are used to denote multisets.

Definition 5.4 Let $\sigma = (u_1, \ldots, u_n)$ and $\sigma' = (u'_1, \ldots, u'_m)$ be two sequences. We say that σ is *a prefix of* σ', denoted by $\texttt{Prefix}(\sigma, \sigma')$, if $n \leq m$ and for all $1 \leq i \leq n$ we have $u_i = u'_i$.

Let $\rho = (i_1/o_1, \ldots, i_m/o_m)$ be a sequence of input/output actions. We define the *input actions of the sequence* ρ, denoted by $\texttt{inputs}(\rho)$, as the sequence $(i_1, \ldots, i_m)$ and the *output actions of the sequence* ρ, denoted by $\texttt{outputs}(\rho)$, as the sequence $(o_1, \ldots, o_m)$. We denote the set of all sequences of output actions by ϕ.

Let $H = \{(\rho_1, r_1), \ldots, (\rho_m, r_m)\}$ be a set of pairs (trace, natural number) and $\sigma = (i_1, \ldots, i_n)$ be a sequence of input actions. The *set of input prefixes* of σ in H, denoted by $\texttt{IPrefix}(H, \sigma)$, is defined as

$$\texttt{IPrefix}(H,\sigma) = \left\{ (\rho', r') \;\middle|\; \begin{array}{l} \sigma = \texttt{inputs}(\rho') \;\wedge\; r' > 0 \;\wedge \\ r' = \sum \{\!| r'' \mid (\rho'', r'') \in H \;\wedge\; \texttt{Prefix}(\rho', \rho'') |\!\} \end{array} \right\}$$

□

Next we present the notions that we will use to denote that a given event has been detected in an IUT. We will also compute the sequences of actions that the implementation performs when a test is applied.

Definition 5.5 Let $\mathcal{I} = (S, I, O, \delta, s_0)$ be a PFSM representing an IUT. We say that $(i_1/o_1, \ldots, i_n/o_n)$ is an *execution* of $\mathcal{I}$ if the sequence $(i_1/o_1, \ldots, i_n/o_n)$ can be performed by $\mathcal{I}$. Let $\rho_1, \ldots, \rho_n$ be executions of $\mathcal{I}$ and $r_1, \ldots, r_n \in \mathbb{N}$. We say that the set $H = \{(\rho_1, r_1), \ldots, (\rho_n, r_n)\}$ is an *execution sample* of $\mathcal{I}$.

Let $T = (S', I, O, \delta', s'_0, S_I, S_O, S_F, S_P, \zeta)$ be a valid test. We say that $H = \{(\rho_1, r_1), \ldots, (\rho_n, r_n)\}$ is an *execution sample of* $\mathcal{I}$ *under the test* T if H is an execution sample of $\mathcal{I}$ and for all $(\rho, r) \in H$ we have that $T \stackrel{\rho}{\Longrightarrow} s$, with $s \in S'$.

Let $\Omega = \{T_1, \ldots, T_n\}$ be a test suite and $H_1, \ldots, H_n$ be sets such that each H_i is an execution sample of $\mathcal{I}$ under T_i. We say that $H = \{(\rho_1, r_1), \ldots, (\rho_n, r_n)\}$ is an *execution sample of* $\mathcal{I}$ *under the test suite* Ω if for all $(\rho, r) \in H$ we have

that $r = \sum_i \{\!|r' \mid (\rho, r') \in H_i|\!\}$. □

In the definition of execution sample under a test we have that each number r, with $(\rho, r) \in H$, denotes the number of times we have observed the execution ρ in $\mathcal{I}$ under the repeated application of T.

5.1 Passing a test on the basis of the behavior in traces

In this section we introduce a notion of passing a test where the behavior of traces is concerned. A different notion, concerning the behavior of states, will be presented in the next section. Passing a test consists in fulfilling two constraints. First, we require that the test never reaches a failure state as a result of its interaction with the implementation. This condition concerns what is *possible*. Second, we require that the random variables attached to successful states conform to the samples collected during the repeated application of the test to the IUT. This condition concerns what is *probable*. We will consider that the set of executions analyzed to pass a test does not only include those executions obtained by applying that test, but also the executions obtained by applying other tests. Let us remark that the very same traces that are available in a test could be part of other tests as well. Let us also note that the validity of any hypothesis contrast increases with the number of samples. Hence, it would not be efficient to apply each hypothesis contrast to the *limited* collection of samples obtained by a single test. On the contrary, samples collected from the application of different tests will be shared so that our statistical information grows and the hypothesis contrast procedure improves. Thus, this testing methodology represents a real novelty with respect to usual techniques where the application of each test is independent from the application of other tests.

Definition 5.6 Let $\mathcal{I}$ be an IUT, H be an execution sample of $\mathcal{I}$ under the test suite Ω, $0 \leq \alpha \leq 1$, and $T = (S', I, O, \delta', s'_0, S_I, S_O, S_F, S_P, \zeta) \in \Omega$ be a test. We say that $\mathcal{I}$ $(\alpha, H)-$*passes* the test T if for all trace $\rho \in \mathtt{Traces}(\mathcal{I})$, with $T \stackrel{\rho}{\Longrightarrow} s$, we have that $s \notin S_F$ and if $s \in S_P$ then $\gamma(\zeta(s), R) > \alpha$, where

$$R = \{(\mathtt{outputs}(\rho'), r) \mid (\rho', r) \in \mathtt{IPrefix}(H, \mathtt{inputs}(\rho))\}$$

We say that $\mathcal{I}$ $(\alpha, H)-$*passes* the test suite Ω if for all test $T \in \Omega$ we have $\mathcal{I}$ $(\alpha, H)-$*passes* T. □

Let us remark that the previous definition can be applied only if for each test T the function ζ associating random variables to passing states returns the random variables appearing in the first part of Definition 4.2, that is, we consider *trace events*. If events described by these random variables denote

state events then they would not fit into the kind of samples considered in the previous relation.

5.2 Passing tests on the basis of the behavior in states

As we pointed out in Section 3, it is possible to check the correctness of an implementation by means of its reactions in each *state*. This new framework requires to assume the additional hypothesis `Hyp` indicating that the graph underlying the IUT is isomorphic, up to probabilities, to the one representing the specification. Equivalently, we may assume that we can *see* the internal ramification of the model of the implementation, that is, we may consider that a function provides us the state of the implementation after observing a given trace. This function will be used to convert our *trace-oriented* samples into *state-oriented* ones. Finally, we still assume that both specifications and implementations are given by deterministically observable `PFSM`*s*.

Definition 5.7 Let $\rho = (i_0/o_0, \ldots, i_n/o_n)$ and $\rho' = (i'_0/o'_0, \ldots, i'_m/o'_m)$ be two sequences of input/output actions. We define the concatenation of ρ and ρ', denoted by $\rho \circ \rho'$, as $(i_0/o_0, \ldots, i_n/o_n, i'_0/o'_0, \ldots, i'_m/o'_m)$.

Let $\mathcal{I} = (S, I, O, \delta, s_0)$ be a `PFSM`. The function $\mathcal{G}_{\mathcal{I}} : \texttt{Traces}(\mathcal{I}) \longrightarrow S$ fulfills that for any $\rho \in \texttt{Traces}(\mathcal{I})$ we have $s_0 \stackrel{\rho}{\Longrightarrow} s$ implies $\mathcal{G}_{\mathcal{I}}(\rho) = s$. We say that $\mathcal{G}_{\mathcal{I}}$ is the *guiding function* of $\mathcal{I}$.

Let $H = \{(\rho_1, r_1), \ldots, (\rho_n, r_n)\}$ be an execution sample of $\mathcal{I}$ under a test suite Ω. The *states execution sample* of H for $\mathcal{I}$, denoted by $\texttt{SSample}(H, \mathcal{I})$, is defined as the set

$$\left\{ ((s,i,o), r') \;\middle|\; r' = \sum \left\{ r \;\middle|\; \begin{array}{r} \exists\, \rho, \rho' : \texttt{Prefix}(\rho' \circ i/o, \rho) \wedge \\ \mathcal{G}_{\mathcal{I}}(\rho') = s \;\wedge\; (\rho, r) \in H \end{array} \right\} > 0 \right\}$$

□

Let us comment on the previous definition. In order to generate information concerning states from information concerning traces, we have to consider all the times that each trace belonging to the sample has arrived at each state $s \in S$ and has produced an output $o \in O$ in response to an input $i \in I$. This is done by taking into account those traces appearing in the sample as well as their *prefixes*. Let us suppose that a prefix of a trace reaches s and it is still a prefix of that trace after adding i/o. Then, all the observations of that prefix are in fact observations of events where the implementation was in the state s and produced o after receiving i. Let us remark that a single trace could perform the pair i/o from the state s several times if that trace reaches

a state s several times. In this case, different prefixes of the trace would fulfill the previous condition. Each time one of these prefixes does so, the number of observations of that trace will be added *again* to the number of repetitions of that event.

Next we introduce our second definition of passing a test. In this case, acceptance states of the test will be provided with random variables describing the ideal probabilistic behavior concerning the last transition of the actual execution of the implementation and the test. For each trace leading the test to a passing state, the hypothesis contrast is applied to the state the implementation stayed *before* the last input/output pair of the trace was performed. In fact, this is the state where the decision of performing that output in response to that input was taken.

Definition 5.8 Let $\mathcal{I}$ be an IUT, H be an execution sample of $\mathcal{I}$ under the test suite Ω, $0 \leq \alpha \leq 1$, and $T = (S', I, O, \delta', s'_0, S_I, S_O, S_F, S_P, \zeta) \in \Omega$ be a test. We say that the implementation $\mathcal{I}$ $(\alpha, H)^{st}-$*passes* the test T *for states* if for all trace $\rho \in \texttt{Traces}(\mathcal{I})$, with $T \stackrel{\rho}{\Longrightarrow} s$, we have that $s \notin S_F$ and if $s \in S_P$ then $\gamma(\zeta(s), R) > \alpha$, where

- $\rho = \rho' \circ i'/o'$ with $\mathcal{G}_{\mathcal{I}}(\rho') = s'$, and
- $R = \{(o, r) \mid ((s', i', o), r) \in \texttt{SSample}(H, \mathcal{I})\}$.

We say that $\mathcal{I}$ $(\alpha, H)^{st}-$*passes for states* the test suite Ω if for any $T \in \Omega$ we have $\mathcal{I}$ $(\alpha, H)^{st}-$*passes* T for states. □

Let us remark that the previous definition can be applied only if the functions associating random variables to passing states in tests concern *state events* (see Definition 4.2).

5.3 Relating Notions of Passing Tests

In this section we show how the notions of passing tests introduced in Definitions 5.6 and 5.8 are related. The next definition introduces a simple method to *convert* a test where random variables concern trace events into a new test where random variables concern state events. Unfortunately, random variables in passing states do not provide enough probabilistic information to perform that conversion. In order to obtain this additional information, the transformation will be supported by a `PFSM` which will be supposed to be the *model* of the test. When a passing state is reached in the test after performing a trace, the random variable in that passing state will be constructed according to the symbolic probability of that trace in the referenced `PFSM`. In the following definition, let us remind that ϕ denotes the set of all sequences of output actions (see Definition 5.4).

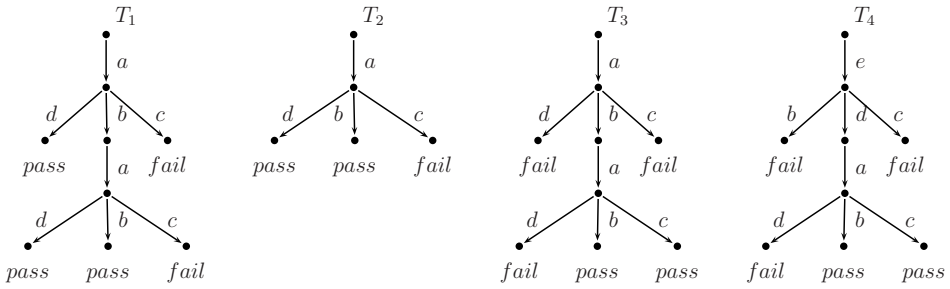

Figure 3. Tests to be used in Lemma 5.10.

Definition 5.9 Let $T = (S, I, O, \delta, s_0, S_I, S_O, S_F, S_P, \zeta)$ be a test for a certain function $\zeta : S \to (\phi \to \texttt{simbP})$. Let $\mathcal{S}$ be a PFSM. We say that the test T *fits into* $\mathcal{S}$ if for any $s \in S_P$, with $T \stackrel{\rho}{\Longrightarrow} s$, we have that $\zeta(s)(\texttt{outputs}(\rho)) = \overline{p}$, where $(\rho, \overline{p}) \in \texttt{pTraces}(\mathcal{S})$.

If T fits into $\mathcal{S}$ then the *conversion to states* of T with respect to $\mathcal{S}$, which is denoted by $\texttt{ConvToSt}(T, \mathcal{S})$, is a test $T' = (S, I, O, \delta, s_0, S_I, S_O, S_F, S_P, \zeta')$, for a certain function $\zeta : S \to (O \to \texttt{simbP})$ where for any $s \in S_P$, with $T \stackrel{\rho}{\Longrightarrow} s$, and $\rho = (i_1/o_1, \ldots, i_n/o_n)$ we have that $\zeta'(s)(o_n) = \overline{p}''$, where $((i_1/o_1, \ldots, i_{n-1}/o_{n-1}), \overline{p}'), (\rho, \overline{p}) \in \texttt{pTraces}(\mathcal{S})$ and $\overline{p} = \overline{p}' \cdot \overline{p}''$. □

The following property shows that the notions of passing a test suite given in definitions 5.6 and 5.8 are quite different. In particular, neither of them implies the other. Intuitively, the relation based on traces is not contained in the relation based on states because, as we pointed out in Example 3.4, an unexpected probabilistic behavior in a transition could be compensated by the traces where it is included. On the other hand, the relation based on states is not contained in the relation based on traces because a correct probabilistic behavior in a state could be the result of some traces whose behavior is unexpected. This could happen if the empirical bias of each of them balance in the overall in that state.

Lemma 5.10 Let $T_1, \ldots, T_n$ be tests fitting into a PFSM $\mathcal{S}$. Let $\mathcal{I}$ be an implementation under test, H be an execution sample of $\mathcal{I}$ under the test suite $\{T_1, \ldots, T_n\}$, and $0 \leq \alpha \leq 1$. We have that $\mathcal{I}$ $(\alpha, H)-$*passes for traces* $\{T_1, \ldots, T_n\}$ does not imply that $\mathcal{I}$ $(\alpha, H)^{st}-$*passes for states* the test suite $\{\texttt{ConvToSt}(T_1, \mathcal{S}), \ldots, \texttt{ConvToSt}(T_n, \mathcal{S})\}$, neither the reverse is true.

Proof: To prove that the first implication is false it is enough to create a counterexample inspired in Example 3.4. Let us take a test suite $\Omega = \{T_1, T_2\}$, where T_1 and T_2 are depicted in Figure 3. In both tests, the passing states reached after (a/d) will be endowed with a random variable ξ_1 such that $\xi_1(b) = [0.2, 0.6]$ and $\xi_1(d) = [0.3, 0.7]$. In T_2, the passing state reached after executing (a/b) will be equipped with the same random variable. Besides, states reached in T_1 after performing the sequences $(a/b, a/b)$ and $(a/b, a/d)$

will be equipped with a random variable ξ_2 such that $\xi_2(b,b) = [0.08, 0.36]$ and $\xi_2(b,d) = [0.06, 0.48]$. Tests T_1 and T_2 fit into $\mathcal{S}$ (see Example 3.4). To convert these tests into new tests that concern *state events* (according to $\mathcal{S}$) it is enough to change the previous random variables by ξ_1' and ξ_2' respectively, where $\xi_1'(b) = [0.2, 0.6]$, $\xi_1'(d) = [0.3, 0.7]$, $\xi_2'(b) = [0.4, 0.6]$, and $\xi_2'(d) = [0.3, 0.8]$. Besides, let us suppose that H, the execution sample of $\mathcal{I}$ under Ω, is the set $H = \{((a/d), 5), ((a/b, a/b), 1), ((a/b, a/d), 4)\}$. Finally, the confidence function $\gamma : (\mathcal{A} \to \mathtt{simbP}) \times ((\mathcal{A} \times \mathbb{N}) \times \ldots (\mathcal{A} \times \mathbb{N})) \to \mathbb{R}$ is defined such that

$$\gamma(\mathcal{V}, ((\rho_1, r_1), \ldots, (\rho_k, r_k))) = \begin{cases} 1 \text{ if for all } \rho_i \in \mathcal{A} : \frac{r_i}{\sum r_j} \in \mathcal{V}(\rho_i) \\ 0 \text{ otherwise} \end{cases}$$

where the type of $\mathcal{V}$ is given by $\mathcal{V} : \mathcal{A} \to \mathtt{simbP}$. Let us consider $0 < \alpha \leq 1$. Under these conditions, we have that $\mathcal{I}$ $(\alpha, H)-$*passes for traces* $\{T_1, T_2\}$ holds because all observed ratios fit into the probabilistic constraints. Let us remind that the observed ratio of (a/b) is 0.5, because it has been detected $1 + 4$ times out of 10. On the other hand, we have that $\mathcal{I}$ $(\alpha, H)^{st}-$*passes for states* $\{\mathtt{ConvToSt}(T_1, \mathcal{S}), \mathtt{ConvToSt}(T_2, \mathcal{S})\}$ is false because the observed ratio of output b in response to input a after (a/b) has been performed is 0.2, which does not belong to the interval $[0.4, 0.6]$.

In order to show that the opposite implication is also false, we need to give a new counterexample. Let $\mathcal{S} = (S, I, O, \delta, s_0)$, where $I = \{a, e\}$, $O = \{b, c, d\}$, $S = \{s_0, s_1, s_2\}$, and the set of transitions is given by

$$\delta = \{s_0 \xrightarrow{a/b}_1 s_1, s_0 \xrightarrow{e/d}_1 s_1, s_1 \xrightarrow{a/b}_{0.5} s_2, s_1 \xrightarrow{a/c}_{0.5} s_2\}$$

Let us remind that an interval $[p, p]$ is denoted simply by p. Let us consider a test suite $\Omega = \{T_3, T_4\}$, where T_3 and T_4 are depicted in Figure 3. In T_3, the passing states reached after $(a/b, a/b)$ and $(a/b, a/c)$ will be endowed with the same random variable, ξ_1, taking values $\xi_1(b) = 0.5$ and $\xi_1(c) = 0.5$. Besides, states reached in T_4 after $(e/d, a/b)$ and $(e/d, a/c)$ will be also equipped with the same random variable ξ_1.

In order to transform these tests into new tests that concern *trace events* and fit into $\mathcal{S}$, we substitute the previous random variable ξ_1 by the random variables ξ_1' (in T_3) and ξ_2' (in T_4). These new variables are defined by $\xi_1'(b,b) = \xi_1'(b,c) = \xi_2'(d,b) = \xi_2'(d,c) = 0.5$. Let T_3' and T_4' be these new tests. Besides, let us suppose that the execution sample of $\mathcal{I}$ under Ω is $H = \{((a/b, a/b), 1), ((e/d, a/c), 1)\}$. Finally, let us suppose that the confidence function γ is defined as before, and let $0 < \alpha \leq 1$. Under these conditions we have that $\mathcal{I}$ $(\alpha, H)^{st}-$*passes for states* $\{T_1, T_2\}$ holds. Since $\mathtt{Hyp}$ is assumed in this relation, both samples in H reach the same implementation state after (a/b) and (e/d), respectively. At this state, after the input a is received, half

of the times the output b is produced, while the other half the output c is produced. This fits into the probabilistic constraint given in both tests for that state. However, $\mathcal{I}$ $(\alpha, H)-$*passes for traces* $\{T_1, T_2\}$ does not hold. Let us note that the empirical ratio of the trace $(a/b, a/b)$ when the sequence of inputs (a, a) is produced is equal to 1, while it should be equal to 0.5. □

6 Implementation Relations based on Samples

In Section 3 we presented two implementation relations that clearly expressed the probabilistic constraints an implementation must fulfill to conform to a specification. Unfortunately, these notions are useful only from a theoretical point of view since it cannot be tested, by using a finite number of test executions, the correctness, in this sense, of the probabilistic behavior of an implementation with respect to a specification. In this section we introduce new implementation relations that take into account the practical limitations to collect probabilistic information from an implementation. These new relations allow us to claim the accurateness of the probabilistic behavior of an implementation with respect to a specification *up to* a given confidence level. Given a set of execution samples, we will apply a hypothesis contrast to check whether the probabilistic choices taken by the implementation follow the patterns given by the specification.

Our new implementation relations follow again the classical pattern. Regarding input/output actions, the constraints imposed by these implementation relations are given by the relation `conf` (see Definition 3.1). This condition over the sequences that the implementation may perform could be rewritten in probabilistic terms as follows: The confidence we have on the fact that the implementation will not perform forbidden behaviors is 1 (i.e. *complete*). However, since no hypothesis contrast can provide full confidence, it is preferable to keep the constraints over actions separated from the probabilistic constraints. In fact, the reverse is not true: We cannot claim that the implementation is correct even if no forbidden behavior is detected after a finite number of interactions with it.

Regarding the probabilistic constraints of the specification, the new relations will express them in a different way to the one used in Definitions 5.6 and 5.8. In the first two relations (forthcoming Definitions 6.1 and 6.2) we put together all the observations of the implementation. Then, the set of samples corresponding to each trace of the specification will be generated by taking all the observations such that the trace is a *prefix* of them. By doing so, we will be able to compare the number of times the implementation has performed the chosen trace with the number of times the implementation has performed any other behavior. We will use hypothesis contrasts to decide whether the

probabilistic choices of the implementation conform to the probabilistic constraints imposed by the specification. In particular, a hypothesis contrast will be applied to each sequence of inputs considered by the specification. This contrast will check whether the different sequences of outputs associated with these inputs are distributed according to the probability distribution of the random variable associated with that sequence of inputs in the specification.

Definition 6.1 Let $\mathcal{S}$ be a specification, $\mathcal{I}$ be an IUT, H be an execution sample of $\mathcal{I}$, and $0 \leq \alpha \leq 1$. We say that $\mathcal{I}$ $(\alpha, H)-$*probabilistically conforms to* $\mathcal{S}$, denoted by $\mathcal{I}$ $\mathtt{confp}^{(\alpha,H)}$ $\mathcal{S}$, if $\mathcal{I}$ $\mathtt{conf}$ $\mathcal{S}$ and for all $\rho \in \mathtt{Traces}(\mathcal{S})$ we have $\gamma(\xi^{\pi}_{\mathcal{S}}, R) > \alpha$, where $\pi = \mathtt{inputs}(\rho)$ and

$$R = \{(\mathtt{outputs}(\rho'), r) \mid (\rho', r) \in \mathtt{IPrefix}(H, \pi)\}$$

□

In the previous relation, $\xi^{\pi}_{\mathcal{S}}$ denotes the symbolic random variable associated with the sequence of input actions π for the $\mathtt{PFSM}$ $\mathcal{S}$ (see Definition 4.2). Intuitively, each trace observed in the implementation will add one instance to the accounting of its prefixes. We could consider an alternative procedure where traces are independently accounted and each observed trace does not affect the number of instances of other traces being prefix of it. However, this method would lose valuable information that might negatively affect the quality of the hypothesis contrasts. Let us remind that the reliability of any hypothesis contrasts increases with the number of instances included in the samples. Besides, as we said before, an observation where $(o_1, \ldots, o_n)$ has been produced in response to $(i_1, \ldots, i_n)$ is indeed an observation where, for all $1 \leq j \leq n$, $(o_1, \ldots, o_j)$ has been produced in response to $(i_1, \ldots, i_j)$. So, by joining prefixes we properly increase the number of instances processed by hypothesis contrasts, which makes them more precise (as well as the probabilistic implementation relation that takes them into account).

The previous idea induces the definition of a refinement of the previous implementation relation. Let us note that the probability of observing a given trace decreases, in general, as the length of the trace increases. This is so because more probabilistic choices are taken in long traces. Besides, taking prefixes into account increases the number of instances of *short* traces. Thus, it is likely that the number of short traces applied to the hypothesis contrasts of the previous relation will outnumber that of longer traces. Let us note that statistical noise effects are higher when smaller sets of samples are considered. Moreover, if we consider extremely long traces we could obtain a few instances, or even none, in each class of events to be considered by a hypothesis contrast. This fact would ruin the result of such a contrast. Taking these factors into account, in the next definition we introduce a new implementation relation where the confidence requirement is *relaxed* as the length of the trace grows. This reduction is defined by a non-increasing function associating confidence

levels to the length of traces.

Definition 6.2 Let $f : \mathbb{N} \to [0,1]$ be a strictly non-increasing function, $\mathcal{S}$ be a specification, $\mathcal{I}$ be an implementation, and H be an execution sample of $\mathcal{I}$. We say that $\mathcal{I}$ $(f,H)-$*probabilistically conforms to* $\mathcal{S}$, denoted by $\mathcal{I}\ \mathtt{confp}^{(f,H)}\ \mathcal{S}$, if $\mathcal{I}\ \mathtt{conf}\ \mathcal{S}$ and for all trace $\rho \in \mathtt{Traces}(\mathcal{S})$ we have $\gamma(\xi_{\mathcal{S}}^{\pi}, R) > f(l)$, where $\pi = \mathtt{inputs}(\rho)$, l is the length of π, and

$$R = \{(\mathtt{outputs}(\rho'), r) \mid (\rho', r) \in \mathtt{IPrefix}(H, \pi)\}$$

□

It is straightforward to see that when the function f of the previous definition is defined as $f(i) = \alpha$, for all $i \in \mathbb{N}$, then we have $\mathtt{confp}^{(\alpha,H)} = \mathtt{confp}^{(f,H)}$.

Next, we will define the relation based on samples for states. This relation works under the additional implementation hypothesis `Hyp` explained in Section 3. This hypothesis allows us to collect the probabilistic behavior of each state of the implementation by considering all the implementation observations whose trace would traverse the corresponding state in the specification. Thus, each time that the interaction between a test and the IUT produces a trace that would reach a given state, a new sample is added to the probabilistic information corresponding to that state. Hence, for each state of the implementation and each input, a different set of samples is created. Each of these sets provides the number of times each output was performed in that state in response to that input. By taking into account these samples, we can check the probabilistic constraint of the second implementation relation. It forces the sets of samples to match the probabilities given in the specification for the corresponding states and inputs.

Definition 6.3 Let $\mathcal{S} = (S, I, O, \delta, s_0)$ and $\mathcal{I} = (S', I, O, \delta', s'_0)$ be a specification and an IUT, respectively, H be an execution sample of $\mathcal{I}$, and $0 \leq \alpha \leq 1$. Let us suppose that `Hyp` holds. We say that $\mathcal{I}$ $(\alpha,H)-$*probabilistically conforms to* $\mathcal{S}$ *considering states*, denoted by $\mathcal{I}\ \mathtt{confp}_{\mathtt{Hyp}}^{(\alpha,H)}\ \mathcal{S}$, if $\mathcal{I}\ \mathtt{conf}\ \mathcal{S}$ and for each $s \in S$ and $i \in I$ we have $\gamma(\xi_{\mathcal{S}}^{s,i}, R) > \alpha$, where

$$R = \{(o', r) \mid ((s', i, o'), r) \in \mathtt{SSample}(H, \mathcal{I}) \ \wedge\ \mathtt{cSt}(s) = s'\}$$

where for each state $s \in S$, $\mathtt{cSt}(s)$ denotes the corresponding state in S'. □

Let us remind that $\xi_{\mathcal{S}}^{s,i}$ is a random variable constructed from the specification $\mathcal{S}$ (see Definition 4.2). In the following result we show the relation between the implementation relations $\mathtt{confp}^{(\alpha,H)}$ and $\mathtt{confp}_{\mathtt{Hyp}}^{(\alpha,H)}$. Let us note that, similarly to the relations presented in Section 5, both relations are based on the idea of *sampling*. Hence, the reasons considered when we related the notion of passing a test suite for *traces* and the concept of passing it for *states*, shown

in Lemma 5.10, apply also to the implementation relations $\texttt{confp}^{(\alpha,H)}$ and $\texttt{confp}_{\texttt{Hyp}}^{(\alpha,H)}$. In fact, to adapt the counterexamples presented in Lemma 5.10 to the context of these relations is easy. In particular, let us note that samples in $\texttt{confp}^{(\alpha,H)}$ and in the relation introduced in Definition 5.6 are grouped in the same way. Similarly, samples are dealt in the same manner in $\texttt{confp}_{\texttt{Hyp}}^{(\alpha,H)}$ and in the relation presented in Definition 5.8. Moreover, in both cases the confidence function is applied to the execution sample in the same way.

Lemma 6.4 $\texttt{confp}^{(\alpha,H)}$ does not imply $\texttt{confp}_{\texttt{Hyp}}^{(\alpha,H)}$, neither the reverse implication holds. □

Besides, the relations presented in this section cannot be related with those introduced in Section 3. In particular, computing these relations on the basis on samples would require to collect *infinite* samples, which is not feasible. Hence, the relations presented in this section can be seen as a way to put in practice the concepts presented in Section 3.

7 Test Derivation

In this section we provide an algorithm to derive test suites from specifications. We will show that the derived test suites are *complete* with respect to two of the conformance relations introduced in the previous section. As usually, the idea consists in traversing the specification to get all the possible traces in the adequate way. Thus, each test is generated to *focus* on chasing a concrete trace of the specification. Test cases will also contain probabilistic constraints so that they can detect faulty probabilistic behaviors in the IUT. The probabilistic constraints of the specification will be encoded in the tests. Specifically, passing states will have attached symbolic random variables that impose some constraints concerning either the trace executed so far in the test or the transition taken from the previous state of the test. This choice will depend on whether we want to test implementations with respect to traces or states, respectively. First, we introduce some auxiliary functions.

Definition 7.1 Let $M = (S, I, O, \delta, s_0)$ be a PFSM. We define the set of possible outputs in state s after input i, denoted by $\texttt{out}(s, i)$, as the set $\texttt{out}(s, i) = \{o \mid \exists s' : (s, i, o, \overline{p}, s') \in \delta\}$. For each transition $(s, i, o, \overline{p}, s') \in \delta$ we write $\texttt{after}(s, i, o) = s'$. □

Let us remark that, due to the assumption that PFSMs are deterministically observable, $\texttt{after}(s, i, o)$ is uniquely determined.

Our derivation algorithm is presented in Figure 4. This is a non-deterministic algorithm that, given a specification, returns a single test case. However, by

Input: $M = (S, I, O, \delta, s_0)$.
Output: $T = (S', I, O, \delta', s'_0, S_I, S_O, S_F, S_P, \zeta)$.

Initialization:
- $S' := \{s'_0\}, \delta' := S_I := S_O := S_F := S_P := \emptyset$.
- $S_{aux} := \{(s_0, s'_0, (\,), \texttt{Nothing})\}$.

Inductive Cases: Apply one of the following two possibilities until $S_{aux} = \emptyset$.

(1) If $(s^M, s^T, \pi, s^{prev}) \in S_{aux}$ then perform the following steps:
   (a) $S_{aux} := S_{aux} - \{(s^M, s^T, \pi, s^{prev})\}$.
   (b) $S_P := S_P \cup \{s^T\}$.
   (c) Let $\pi = \pi' \circ i$. If we wish to check conformance considering traces, do
      - $\zeta(s^T) := \xi^{\pi}_M$

   else, if we wish to check conformance considering states, do
      - $\zeta(s^T) := \xi^{s^{prev},i}_{M,st}$

(2) If $S_{aux} = \{(s^M, s^T, \pi, s^{prev})\}$ is a unitary set and there exists $i \in I$ such that $\texttt{out}(s^M, i) \neq \emptyset$ then perform the following steps:
   (a) $S_{aux} := \emptyset$.
   (b) Choose i such that $\texttt{out}(s^M, i) \neq \emptyset$.
   (c) Create a fresh state $s' \notin S'$ and perform $S' := S' \cup \{s'\}$.
   (d) $S_I := S_I \cup \{s^T\};\ S_O := S_O \cup \{s'\};\ \delta' := \delta' \cup \{(s^T, i, s')\}$.
   (e) For each $o \notin \texttt{out}(s^M, i)$ do
      - Create a fresh state $s'' \notin S'$ and perform $S' := S' \cup \{s''\}$.
      - $S_F := S_F \cup \{s''\};\ \delta' := \delta' \cup \{(s', o, s'')\}$.

   (f) For each $o \in \texttt{out}(s^M, i)$ do
      - Create a fresh state $s'' \notin S'$ and perform $S' := S' \cup \{s''\}$.
      - $\delta' := \delta' \cup \{(s', o, s'')\}$.
      - $s^M_1 := \texttt{after}(s^M, i, o)$.
      - Let $(s^M, i, o, \overline{p}, s^M_1) \in \delta$.
        $S_{aux} := S_{aux} \cup \{(s^M_1, s'', \pi \circ i, s^M)\}$.

Figure 4. Test Derivation Algorithm.

considering the tests returned by the algorithm for each possible combination of non-deterministic choices, we get the (possible infinite) set of tests extracted from the specification M. In this algorithm the set of pending states S_{aux} keeps track of the states of the test whose definition has not been *finished* yet. A tuple $(s^M, s^T, \pi, s^{prev}) \in S_{aux}$ indicates that the current state in the traversal of the specification is s^M, that we did not conclude yet the description of the state s^T in the test, that the sequence of inputs traversed from s_0 to s^T is π, and that the state before s^M in the traversal of the specification was s^{prev}. The set S_{aux} initially contains a tuple with the initial states (of both specification and test), an empty sequence of inputs, and a symbol denoting that there does not exist a previous state yet. For each tuple in S_{aux} we may choose between two different choices. It is important to remark that possibility (2) is applied

at most to one of the possible tuples. Thus, our derived tests correspond to valid tests as introduced in Definition 5.2.

The possibility (1) simply indicates that the state of the test becomes a passing state. In this case, we attach a symbolic random variable to the passing state. On the one hand, if we are developing the test suite to check conformance considering traces (see Definition 5.6) then this random variable must encode the probability distribution, according to the specification, for all possible traces containing the sequence of inputs π. In this case we denote by $\texttt{tests}(M)$ the derived test suite. On the other hand, if we have to check conformance considering states (see Definition 5.8) then the random variable will encode the probability of performing each output in response to the last input from the last state visited in the specification. In this case we denote by $\texttt{tests}_{\texttt{st}}(M)$ the derived test suite.

The possibility (2) of the algorithm takes an input and generates a transition in the test labelled by this input. Then, the whole sets of outputs is considered. If the output is not expected by the implementation then a transition leading to a failing state is created. This could be simulated by a single branch in the test, labelled by `else`, leading to a failing state (in the algorithm we suppose that *all* the possible outputs appear in the test). For the rest of outputs, we create a transition with the corresponding output and add the appropriate tuple to the set S_{aux}.

Finally, let us remark that finite test cases are constructed simply by considering a step where the second inductive case is not applied.

The next result states that, for a given specification $\mathcal{S}$, the test suites $\texttt{tests}(\mathcal{S})$ and $\texttt{tests}_{\texttt{st}}(\mathcal{S})$ can be used to distinguish those (and only those) implementations conforming with respect to `confp` or $\texttt{confp}_{\texttt{Hyp}}$, respectively. However, we cannot properly say that the test suite is complete since both passing tests and the considered implementation relation have a probabilistic component. So, we can speak about *completeness* up to a certain confidence level.

Theorem 7.2 Let $\mathcal{S}$ and $\mathcal{I}$ be PFSMs. For any $0 \leq \alpha \leq 1$ and execution sample H of $\mathcal{I}$ we have

(a) $\mathcal{I}\ \texttt{confp}^{(\alpha,H)}\ \mathcal{S}$ iff $\mathcal{I}$ $(\alpha, H)-$*passes* $\texttt{tests}(\mathcal{S})$.
(b) $\mathcal{I}\ \texttt{confp}_{\texttt{Hyp}}^{(\alpha,H)}\ \mathcal{S}$ iff $\mathcal{I}$ $(\alpha, H)^{st}-$*passes* $\texttt{tests}_{\texttt{st}}(\mathcal{S})$.

Proof: We will prove the statement (a); the proof of (b) is similar.

First, let us show that $\mathcal{I}$ $(\alpha, H)-$*passes* the test suite $\texttt{tests}(\mathcal{S})$ implies that we also have $\mathcal{I}\ \texttt{confp}^{(\alpha,H)}\ \mathcal{S}$. In order to do that, we will use contrapositive, that is, we assume that $\mathcal{I}\ \texttt{confp}^{(\alpha,H)}\ \mathcal{S}$ does not hold and we show that $\mathcal{I}$ $(\alpha, H)-$*passes* the test suite $\texttt{tests}(\mathcal{S})$ is false as well. First, let us suppose

that $\mathcal{I}$ $\mathtt{confp}^{(\alpha,H)}$ $\mathcal{S}$ is false because $\mathcal{I}$ $\mathtt{conf}$ $\mathcal{S}$ does not hold. This means that there exist two non-probabilistic traces $\rho = (i_1/o_1, \ldots, i_{r-1}/o_{r-1}, i_r/o_r)$ and $\rho' = (i_1/o_1, \ldots, i_{r-1}/o_{r-1}, i_r/o'_r)$, with $r \geq 1$, such that $\rho \in \mathtt{Traces}(\mathcal{S})$, $\rho' \in \mathtt{Traces}(\mathcal{I})$, and $\rho' \notin \mathtt{Traces}(\mathcal{S})$. Let us show that if $\rho \in \mathtt{Traces}(\mathcal{S})$ then there exists a test $T = (S, I, O, \delta', s, S_I, S_O, S_F, S_P, \zeta) \in \mathtt{tests}(\mathcal{S})$ such that $T \stackrel{\rho}{\Longrightarrow} s$ and $s \in S_P$. That test is built by applying the algorithm presented in Figure 4 in such a way that we resolve the non-determinism as follows.

for $1 \leq j \leq r$ **do**
    apply inductive case (2) for input i_j
    apply inductive case (1) for all $(s^{\mathcal{S}}, s^T, \pi, s^{prev}) \in S_{aux}$ obtained
        by processing an output different from o_j
end
apply inductive case (1) for the last $(s^{\mathcal{S}}, s^T, \pi, s^{prev}) \in S_{aux}$

Intuitively, the generated test T traverses all inputs i_j and outputs o_j as specified in trace ρ. We have that the test T is such that $T \stackrel{\rho'}{\Longrightarrow} u$, with $u \in S_F$. This is because the last application of the inductive case (2) deals with the output o'_r in step (e), since we have $\rho' \notin \mathtt{Traces}(\mathcal{S})$. Due to the fact that $\rho' \in \mathtt{Traces}(\mathcal{I})$, we have that $\mathcal{I}$ $(\alpha, H)-$*passes* the test T does not hold. Since $T \in \mathtt{tests}(\mathcal{S})$, we conclude that $\mathcal{I}$ $(\alpha, H)-$*passes* the test suite $\mathtt{tests}(\mathcal{S})$ does not hold.

Let us suppose now that $\mathcal{I}$ $\mathtt{confp}^{(\alpha,H)}$ $\mathcal{S}$ is false because there exists a trace $\rho \in \mathtt{Traces}(\mathcal{S})$ such that $\gamma(\xi^{\pi}_{\mathcal{S}}, R) \leq \alpha$, where we have $\pi = \mathtt{inputs}(\rho)$ and $R = \{(\mathtt{outputs}(\rho'), r) \,|\, (\rho', r) \in \mathtt{IPrefix}(H, \pi)\}$. Then, there exists a test $T = (S, I, O, \delta', s, S_I, S_O, S_F, S_P, \zeta) \in \mathtt{tests}(\mathcal{S})$ such that $T \stackrel{\rho}{\Longrightarrow} s$, with $s \in S_P$ and $\zeta(s) = \xi^{\pi}_{\mathcal{S}}$. We have that $\gamma(\zeta(s), R) \leq \alpha$. So, $\mathcal{I}$ $(\alpha, H)-$*passes* the test T does not hold. Thus, we conclude that neither $\mathcal{I}$ $(\alpha, H)-$*passes* the test suite $\mathtt{tests}(\mathcal{S})$ holds.

Let us prove now that $\mathcal{I}$ $\mathtt{confp}^{(\alpha,H)}$ $\mathcal{S}$ implies $\mathcal{I}$ $(\alpha, H)-$*passes* $\mathtt{tests}(\mathcal{S})$. In order to do that, we assume that $\mathcal{I}$ $(\alpha, H)-$*passes* the test suite $\mathtt{tests}(\mathcal{S})$ is not true and we show that $\mathcal{I}$ $\mathtt{confp}^{(\alpha,H)}$ $\mathcal{S}$ is also false. First, let us suppose that $\mathcal{I}$ $(\alpha, H)-$*passes* the test suite $\mathtt{tests}(\mathcal{S})$ does not hold because there exist $T \in \mathtt{tests}(\mathcal{S})$, $\rho' = (i_1/o_1, \ldots, i_{r-1}/o_{r-1}, i_r/o'_r) \in \mathtt{Traces}(\mathcal{I})$, and a state s such that $T \stackrel{\rho'}{\Longrightarrow} s$ and $s \in S_F$. According to our test derivation algorithm, a branch of a test leads to a failure state only if the associated output cannot be performed in the specification. Thus, $\rho' \notin \mathtt{Traces}(\mathcal{S})$. Let us note that our algorithm only allows to create a failure state if it is the result of the application of the inductive case (2). A requirement to perform this case is $\mathtt{out}(s^{\mathcal{S}}, i) \neq \emptyset$, that is, the specification must produce some output after the reception of the chosen input. Therefore, there exist an output o_r and a non-probabilistic trace $\rho = (i_1/o_1, \ldots, i_{r-1}/o_{r-1}, i_r/o_r)$ such that $\rho \in \mathtt{Traces}(\mathcal{S})$.

Since we have $\rho' \in \texttt{Traces}(\mathcal{I})$, $\rho' \notin \texttt{Traces}(\mathcal{S})$ and $\rho \in \texttt{Traces}(\mathcal{S})$, the predicate $\mathcal{I}$ conf $\mathcal{S}$ does not hold. Thus, $\mathcal{I}$ $\texttt{confp}^{(\alpha,H)}$ $\mathcal{S}$ is false.

Finally, let us suppose that $\mathcal{I}$ $(\alpha, H)-$*passes* $\texttt{tests}(\mathcal{S})$ does not hold because there exist $T = (S, I, O, \delta', s, S_I, S_O, S_F, S_P, \zeta) \in \texttt{tests}(\mathcal{S})$ and $\rho \in \texttt{Traces}(\mathcal{I})$ such that $\mathcal{I}$ $(\alpha, H)-$*passes* T does not hold, $T \stackrel{\rho}{\Longrightarrow} s$, with $s \in S_P$, $\texttt{inputs}(\rho) = \pi$, and $\gamma(\zeta(s), R) \leq \alpha$, where $R = \{(\texttt{outputs}(\rho'), r) \,|\, (\rho', r) \in \texttt{IPrefix}(H, \pi)\}$. According to the test derivation algorithm we know that if $s \in S_P$ then $\rho \in \texttt{Traces}(\mathcal{S})$. Besides, we have $\zeta(s) = \xi^{\pi}_{\mathcal{S}}$. Then, $\gamma(\xi^{\pi}_{\mathcal{S}}, R) \leq \alpha$. Thus, we conclude that $\mathcal{I}$ $\texttt{confp}^{(\alpha,H)}$ $\mathcal{S}$ does not hold. □

8 Conclusions

In this paper we have presented a testing methodology to check whether an implementation properly follows the behavior described by a given specification. The particularity of our framework is that specifications can explicitly express the desired *propensity* of each option in each non-deterministic choice of the system. This propensity is denoted in terms of probabilities. Moreover, in order to improve the expressivity of specifications, symbolic probabilities are introduced. Let us note that, in several situations, it is not desirable or feasible to impose that the probability of an event is some fix value. Symbolic probabilities help to overcome this problem.

We have presented a testing methodology for these kind of systems. The necessity of assessing the probabilistic behavior of a system increases the complexity of the methodology, as it is impossible to infer the actual probabilities associated with implementations from a set of interaction samples. In order to cope with this problem, the probabilistic behavior of the implementation is *probabilistically* assessed on the basis of a finite set of samples, which is obtained by applying tests several times. Hypothesis contrasts allow to obtain a (probabilistic) diagnosis result by comparing the set of samples with the ideal probabilistic requirements.

We have explored several possibilities to apply this methodology. In particular, we have studied the practical differences between associating probabilistic constraints to traces and associating them to the transitions leaving each state. Finally, we have presented a test derivation algorithm. The resulting test suite is sound and complete with respect to our implementation relations, that is, the test suite is passed if and only if the corresponding implementation relation holds.

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

Appendix. Statistics Background: Hypothesis Contrasts

In this appendix we introduce one of the standard ways to measure the confidence that a certain sample has been generated by a given random variable. In order to do so we present a methodology to perform *hypothesis contrasts*. Intuitively, a sample will be *rejected* if the probability of observing that sample from a given random variable is low. In practice, we will check whether the probability to observe a *discrepancy* lower than or equal to the one that we have detected is low enough. We will present *Pearson's* χ^2 *contrast*. This contrast can be applied both to continuous and discrete random variables. 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 random variable χ^2.
- 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 $\max\{\alpha \mid X^2 \leq \chi^2_\alpha(k-1)\}$.

According to the previous steps, we can now present an operative definition of the function γ which was introduced in Definition 4.1. Since we will use hypothesis contrasts to compare samples with *symbolic* random variables but the previous procedure refers to *standard* random variables, we must be careful when applying the previous ideas in our framework. Let us note that symbolic random variables encapsulate a set of standard random variables (this set is in general infinite). For instance, let us consider the set of events $\mathcal{A} = \{a, b\}$ and the symbolic random variable $\xi : \mathcal{A} \to \texttt{simbP}$ with $\xi(a) = \xi(b) = (\frac{1}{4}, \frac{3}{4})$. Then, a possible standard random variable fitting into ξ is $\xi' : \mathcal{A} \to (0, 1]$ with $\xi'(a) = \frac{1}{3}$ and $\xi'(b) = \frac{2}{3}$. Another possibility is $\xi'' : \mathcal{A} \to (0, 1]$ with $\xi''(a) =$

$\xi''(b) = \frac{1}{2}$. Since ξ embraces both possibilities, assessing the confidence of ξ on a sample should consider both of them. Actually, we will consider that the sample is adequate for ξ if it would be so for some standard random variable fitting into ξ. More generally, an *instance* of a symbolic random variable is any (standard) random variable where each probability fits into the margins of the symbolic random variable for the corresponding class. Besides, the addition of the probabilities must be equal to 1. In order to compute the confidence of a symbolic random variable on a sample we consider the instance of it that returns the highest confidence on that sample.

Definition Let us consider a set of events $\mathcal{A} = \{a_1, \ldots, a_k\}$, a symbolic random variable $\xi : \mathcal{A} \to \texttt{simbP}$, and a random variable $\xi' : \mathcal{A} \to (0, 1]$. Let J be a sample of $\mathcal{A}$. We say that the random variable ξ' is an *instantiation* of ξ, denoted by $\texttt{Instance}(\xi', \xi)$, if for any $a \in \mathcal{A}$ we have $\xi'(a) \in \xi(a)$ and $\sum_{a \in \mathcal{A}} \xi'(a) = 1$.

For any random variable $\xi' : \mathcal{A} \to (0, 1]$ let $X^2_{\xi'}$ denote the discrepancy level of J on ξ' calculated as explained above by splitting the sampling space into the set of events $\mathcal{A}$. Let $\xi : \mathcal{A} \to \texttt{simbP}$ denote a symbolic random variable. We define the confidence of ξ on J, denoted by $\gamma(\xi, J)$, as follows:

$$\gamma(\xi, J) = \max \left\{ \alpha \;\middle|\; \begin{array}{l} \exists\, \xi' : \texttt{Instance}(\xi', \xi) \wedge \\ \qquad \alpha = \max\{\alpha' \mid X^2_{\xi'} \leq \chi^2_{\alpha'}(k-1)\} \end{array} \right\}$$

□

References

[1] N. López, M. Núñez, I. Rodríguez, Testing of symbolic-probabilistic systems, in: 4th Int. Workshop on Formal Approaches to Testing of Software (FATES 2004), LNCS 3395, Springer, 2004, pp. 49–63.

[2] G. Reed, A. Roscoe, A timed model for communicating sequential processes, Theoretical Computer Science 58 (1988) 249–261.

[3] X. Nicollin, J. Sifakis, An overview and synthesis on timed process algebras, in: Computer Aided Verification'91, LNCS 575, Springer, 1991, pp. 376–398.

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

[5] W. Yi, K. Larsen, Testing probabilistic and nondeterministic processes, in: Protocol Specification, Testing and Verification XII, North Holland, 1992, pp. 47–61.

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

[7] B. Jonsson, W. Yi, K. Larsen, Probabilistic extensions of process algebras, in: J. Bergstra, A. Ponse, S. Smolka (Eds.), Handbook of process algebra, North Holland, 2001, Ch. 11.

[8] J. Baeten, C. Middelburg, Process algebra with timing, EATCS Monograph, Springer, 2002.

[9] M. Bravetti, A. Aldini, Discrete time generative-reactive probabilistic processes with different advancing speeds, Theoretical Computer Science 290 (1) (2003) 355–406.

[10] D. Cazorla, F. Cuartero, V. Valero, F. Pelayo, J. Pardo, Algebraic theory of probabilistic and non-deterministic processes, Journal of Logic and Algebraic Programming 55 (1–2) (2003) 57–103.

[11] M. Núñez, Algebraic theory of probabilistic processes, Journal of Logic and Algebraic Programming 56 (1–2) (2003) 117–177.

[12] N. López, M. Núñez, An overview of probabilistic process algebras and their equivalences, in: Validation of Stochastic Systems, LNCS 2925, Springer, 2004, pp. 89–123.

[13] K. Bartlett, R. Scantlebury, P. Wilkinson, A note on reliable full-duplex transmission over half-duplex links, Communications of the ACM 12 (5) (1969) 260–261.

[14] N. López, M. Núñez, I. Rodríguez, Formal specification of symbolic-probabilistic systems, in: European Performance Engineering Workshop (EPEW'04), LNCS 3236, Springer, 2004, pp. 114–127.

[15] G. Myers, The Art of Software Testing, John Wiley and Sons, 1979.

[16] B. Beizer, Black Box Testing, John Wiley and Sons, 1995.

[17] I. Christoff, Testing equivalences and fully abstract models for probabilistic processes, in: CONCUR'90, LNCS 458, Springer, 1990, pp. 126–140.

[18] M. Núñez, D. de Frutos, Testing semantics for probabilistic LOTOS, in: Formal Description Techniques VIII, Chapman & Hall, 1995, pp. 365–380.

[19] R. Segala, Testing probabilistic automata, in: CONCUR'96, LNCS 1119, Springer, 1996, pp. 299–314.

[20] R. Cleaveland, Z. Dayar, S. Smolka, S. Yuen, Testing preorders for probabilistic processes, Information and Computation 154 (2) (1999) 93–148.

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

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

[23] M. Stoelinga, F. Vaandrager, A testing scenario for probabilistic automata, in: ICALP 2003, LNCS 2719, Springer, 2003, pp. 464–477.

[24] M. Núñez, I. Rodríguez, Towards testing stochastic timed systems, in: FORTE 2003, LNCS 2767, Springer, 2003, pp. 335–350.

[25] R. Cleaveland, G. Lüttgen, V. Natarajan, Priority in process algebra, in: J. Bergstra, A. Ponse, S. Smolka (Eds.), Handbook of process algebra, North Holland, 2001, Ch. 12.

[26] J. Tretmans, Test generation with inputs, outputs and repetitive quiescence, Software – Concepts and Tools 17 (3) (1996) 103–120.

[27] M. Núñez, I. Rodríguez, Encoding `PAMR` into (timed) `EFSMs`, in: FORTE 2002, LNCS 2529, Springer, 2002, pp. 1–16.