Formal Specification of 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 formal specification and validation of systems where probabilistic information is not given by means of fixed values but as sets of probabilities. These sets will be intervals contained in $(0, 1]$ indicating the possible value of the *real* probability. In order to specify this kind of systems we will introduce a suitable extension of the notion of finite state machine. Essentially, choices between transitions outgoing from a state and having the same input action are probabilistically resolved. We will also present some implementation relations to assess the conformance of an implementation to a specification. The first implementation relation will clearly present the probabilistic constraints of the specification, but it will be unfeasible from the practical point of view. The other relations will overcome the problems of the first one by introducing a notion of conformance *up to* a given level of confidence. These relations will assess the validity of an implementation with respect to a specification by considering a finite *sample* of executions of the implementation and comparing it with the probabilistic constraints imposed by the specification.

1 Introduction

During the last years we have seen an evolution in the kind of systems that formal methods are dealing with. In the beginning the main focus was on functional properties. The next step was to consider quantitative information such as the *time* underlying the performance of the system or the *probabilities* resolving the non-deterministic choices to be taken. Following this line, plenty of frameworks considering time and/or probabilistic aspects have been proposed (e.g. [13,8,9,16,6,7,1,4,10,3]).

In this paper we concentrate on the formal specification and validation of a novel notion of probabilistic systems. One of the main criticisms about formal models including probabilistic information is the inability, in general, to

* Work 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).

accurately determine the actual values of the involved probabilities. The usual inclusion of probabilities is given by (using a process algebraic notation) processes such as $P = a +_{\frac{1}{3}} b$. In this case, P may be thought as expressing 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 [2]). Usually, 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 that 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, our approach, that we call *symbolic probabilities*, allows us to use *successive refinements*. 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 Tchebyshev inequality) we may indicate information such as "with probability 0.95 the *real* probability of losing the message belongs to the interval $[0.01, 0.03]$."

In this paper we present a formalism to specify symbolic-probabilistic systems. We will consider a suitable extension of *finite state machines* where probabilistic information is included in the appropriate places. 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, transitions as $s \xrightarrow{i/o}_p s'$ indicate that the probability with which state s performs the output o after the input i is received and reaches state s' belongs to the range given by the expression p. For instance, 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 related according to the probabilistic information. In [6] a taxonomy, that has become classical, is given. In this paper we consider a variant of the *reactive* interpretation of probabilities since it is the most suitable for our framework. Intuitively, a reactive interpretation imposes a probabilistic relation among transitions labeled by the same action, but without quantifying how choices between different actions is resolved. In our case, our probabilistic finite state machines will be 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 of probabilities (the formal definition of our notation will be given in the next section).

Example 1. Let us consider that the unique transitions from a state s are

$$t_1 = s \xrightarrow{i_1/o_1}_{p_1} s_1 \qquad t_2 = s \xrightarrow{i_1/o_2}_{p_2} s_2 \qquad t_3 = s \xrightarrow{i_1/o_3}_{p_3} s_2$$
$$t_4 = s \xrightarrow{i_2/o_1}_{p_4} s_3 \qquad t_5 = s \xrightarrow{i_2/o_3}_{p_5} s_1$$

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 p_1, p_2, and p_3. All we know about these values is that they fulfill the imposed restrictions, 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 labeled with different input actions (e.g. t_1 and t_4). □

In addition to the definition of our formalism, we have also developed appropriate implementation relations to try and determine whether an implementation conforms to the behavior indicated by the corresponding specification. We have to take into account that, in general, we will not be able to *see* the probabilities that the implementation has assigned to each of the choices. Thus, even though implementations will behave according to fixed probabilities, in contrast with specifications, the problem is that we will not be able to *read* their values. Following the approach presented in [12], in order to compute the probabilities associated with each choice of the implementation we will analyze some samples collected from the implementation. By comparing them with the symbolic probabilities of the specification we will be able to assess the validity of the implementation. This comparison will be performed by using *hypothesis contrasts*. These contrasts allow to (probabilistically) decide whether an observed sample follows the pattern given by a random variable.

The rest of the paper is organized as follows. In Section 2 we present our probabilistic finite state machines model. In Section 3 we introduce some basic statistical concepts that are (abstractly) used along the paper. In Section 4 we give our first implementation relation and show that, regardless its elegant definition, it is not adequate from the practical point of view. In Section 5 we present new implementation relations that can be checked in practice. The underlying idea is that these relations perform a hypothesis contrast to assess whether the observed behavior corresponds to the probabilistic behavior defined in the specification *up to* a certain confidence. Finally, in Section 6 we present our conclusions and some lines for future work.

2 Probabilistic Finite State Machines

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

Definition 1. We define the set of *symbolic probabilities*, denoted by $\texttt{simbP}$, 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 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 any $1 \leq i \leq n$ 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 $\prod \overline{p}_i$, (respectively the *addition* of $\overline{p}_1, \ldots, \overline{p}_n$, denoted by $\sum \overline{p}_i$) as the symbolic probability $\& \prod p_i, \prod q_i \$$ (respectively $\& \sum p_i, \sum q_i \$$). The limits of the interval are defined in both cases as:

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

□

The previous definition expresses that a symbolic probability $\overline{p}$ is a 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 respective maximal (resp. minimal) bounds of the intervals considered. Next, we introduce our notion of probabilistic finite state machine. In the following we consider that $|X|$ denotes the cardinality of the set X.

Definition 2. A *Probabilistic Finite State Machine*, in short $\texttt{PFSM}$, is a tuple $M = (S, I, O, \delta, s_0)$ where S is the 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. Each transition belonging to δ is a tuple $(s, i, o, \overline{p}, s')$ where $s, s' \in S$ are the initial and final states, $i \in I$ is an input action, $o \in O$ is an output action, and $\overline{p} \in \texttt{simbP}$ is the symbolic probability associated with the transition. We will usually denote transitions as $(s, i, o, \overline{p}, s')$ by $s \xrightarrow{i/o}_{\overline{p}} s'$. Besides, we consider that for any $s \in S$, $i \in I$, and the set $\alpha_{s,i} = \{s \xrightarrow{i/o}_{\overline{p}} s' \mid \exists\, o \in O, \overline{p} \in \texttt{simbP}, s' \in S : s \xrightarrow{i/o}_{\overline{p}} s' \in \delta\}$ the following two conditions hold:

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

□

[1] The interested reader can check [5] where different approaches for introducing priorities are reviewed.

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'. As we pointed out in the introduction of the paper, this interpretation of the probabilistic information follows the *reactive* model as described in [6]. 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 like $s \xrightarrow{i/o}_{\overline{p}} s' \in \delta$ 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 $\overline{p}$) included 1, then the (real) probability associated to the other transition ($\overline{p}'$) could be 0, which is forbidden. Regarding the second condition, let us note that the *real* probabilities for each state $s \in S$ and for each input $i \in I$ should add 1. This is only possible if 1 is within the lower and upper bounds of the associated symbolic probabilities.

Next we define some additional conditions that we will sometimes impose on our probabilistic finite state machines.

Definition 3. Let $M = (S, I, O, \delta, s_0)$ be a `PFSM`. We say that M is *input-enabled* if for any state $s \in S$ and input $i \in I$ there exist $s' \in S$, $o \in O$, and $\overline{p} \in$ `simbP` such that $(s, i, o, \overline{p}, s') \in \delta$.

We say that M is *deterministically observable* if for any $s \in S$, $i \in I$, 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 the state machines. Regarding the notion of deterministically observable, it is worth to point out that it is different from the more restricted notion of deterministic finite state machine. In particular, we allow transitions from the same state labeled by the same input action, as far as the outputs are different.

Example 2. 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 us consider $M_3 = (\{1, 2, 3\}, \{i_1, i_2\}, \{o_1, o_2, o_3\}, \delta, 1)$. Next we define the set of transitions δ. For the first state, we have the transitions $(1, i_1, o_1, 1, 2)$, $(1, i_2, o_1, \overline{p}_1, 1)$, and $(1, i_2, o_2, \overline{p}_2, 3)$. Let us suppose that $\overline{p}_1 = (0, \frac{1}{2}]$ and $\overline{p}_2 = [\frac{1}{3}, 1)$, and 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 labeled by i_1/o_3). The first property does not hold in M_2 (there is no outgoing transition labeled by i_2 from the state 2) while the second one does. Finally, M_3 holds both properties. □

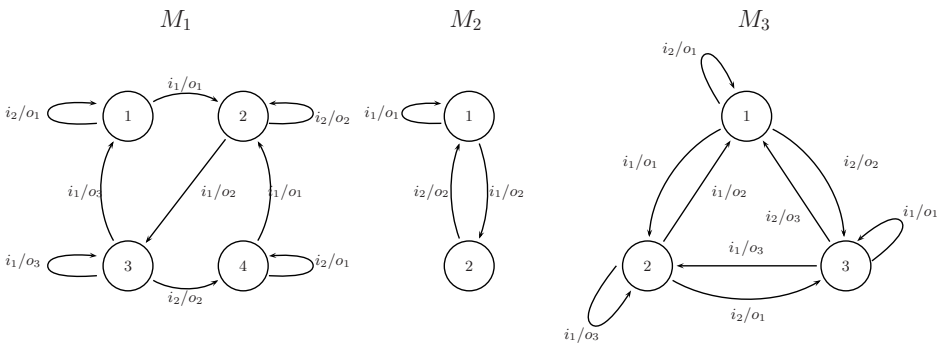

Fig. 1. Examples of PFSM.

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 4. Let $M = (S, I, O, \delta, s_0)$ be a PFSM. We write the *generalized* transition $s \xrightarrow{(i_1/o_1,\ldots,i_n/o_n)}_{\overline{p}} s'$ if there exist $s_1, \ldots, s_{n-1} \in S, \overline{p}_1, \ldots, \overline{p}_n \in \texttt{simbP}$ 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 $s_0 \overset{\rho}{\Longrightarrow}_{\overline{p}} s'$.

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 $s_0 \overset{\rho}{\Longrightarrow}_{\overline{p}} s'$.

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

3 Statistical Concepts

In this section we introduce some statistical notions that will be used in our framework and a procedure to perform hypothesis contrasts. In the following definition we call *event* to any reaction we can detect from a system. For any set of events, a *sample* denotes 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 belonging to the set. In our framework, these random variables will be inferred from the PFSMs denoting the (ideal) probabilistic behavior of systems, while the samples will be collected by interacting with the implementation to be validated. 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 contrast hypothesis in our framework.

Definition 5. Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be a set of *events*. A *sample* of $\mathcal{A}$ is a set $J = \{(a_1, m_1), \ldots, (a_n, m_n)\}$ where for any $1 \leq i \leq n$ we have that m_i represents the number of times that we have observed the event α_i.

Let $\xi : \mathcal{A} \to \texttt{simbP}$ be a function such that $1 \in \sum_{\alpha \in \mathcal{A}} \xi(\alpha)$. In this case 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 values of $\gamma(\xi, J)$ indicate that the sample J is more likely to be produced by the symbolic random variable ξ. In the next definition we particularize the previous notions in the context of our framework. Basically, we give a sequence of inputs and 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 denoting the theoretical probability of each event is computed by considering the symbolic probability of the corresponding trace in the specification.

Definition 6. Let $M = (S, I, O, \delta, s_0)$ be a $\texttt{PFSM}$. Let $\pi = (i_1, \ldots, i_n)$ 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

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

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

Now we introduce one of the standard ways to measure the confidence that a random variable has on a sample. In order to do so we will 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 $k-1$.
– We will *accept* that the sample follows the proposed random variable if the probability to obtain a discrepancy greater than or equal to the detected discrepancy is high enough, that is, if $X^2 < \chi^2_\alpha(k-1)$ for some α high enough. Actually, as such margin to accept the sample decreases as α increases, we can obtain a measure of the validity of the sample as $\max\{\alpha | X^2 \leq \chi^2_\alpha(k-1)\}$.

According to the previous steps, we can now present an operative definition of the function γ which has been presented before in Definition 5. 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 carefull 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 a (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 7. Let $\mathcal{A} = \{a_1, \ldots, a_k\}$ be a set of events, $\xi : \mathcal{A} \to \texttt{simbP}$ be a symbolic random variable, $\xi' : \mathcal{A} \to (0, 1]$ be a random variable, and 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 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:

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

□

4 Probabilistic Implementation Relation

In this section we introduce our first implementation relation. We will consider that both specifications and implementations are given by deterministically ob-

servable PFSMs. Moreover, we will assume that PFSMs representing implementations are input-enabled. We assume that implementations are *black-boxes*. Thus, no information can be known about their internal behavior/structure. 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. [14,15]). That is, an implementation $\mathcal{I}$ conforms to a specification $\mathcal{S}$ if for any possible evolution of $\mathcal{S}$ the outputs that the implementation may perform after a given input are a subset of those for the specification. Besides, this relation will require that the probability of any trace of the implementation is *within* the corresponding (symbolic) probability of the specification for this trace.

Definition 8. 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 any $\rho = (i_1/o_1, \ldots, i_n/o_n) \in \mathtt{Traces}(\mathcal{S})$, 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* $\mathcal{S}$, 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

$$\rho \in \mathtt{Traces}(\mathcal{S}) \text{ implies } (\rho, \overline{p}) \in \mathtt{pTraces}(\mathcal{S}) \text{ for some } \overline{p} \text{ with } p \in \overline{p}.$$

□

Intuitively, the idea underlying the definition of the non-probabilistic conformance relation 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, with some variants, in [11,12]). This condition is also required in the probabilistic case: We check probabilistic traces only if they can be performed by the specification.

The problem underlying this implementation relation is that we have no access to the probabilities governing the transitions of the implementations. So, we are not able to check whether a given implementation probabilistically conforms to a specification. However, we may get *approximations* of the probabilities controlling the behavior of the implementation by collecting *samples* of its execution and computing the empirical ratio associated with the different decision points of the implementation.

5 Implementation Relations based on Samples

In the previous section we presented an implementation relation that clearly expressed the probabilistic constraints an implementation must fulfill to conform to a specification. Unfortunately, this notion is useful only from a theoretical point of view since the correctness of the probabilistic behavior of an implementation with respect to a specification cannot be checked by using a finite number of observations. In this section we introduce new implementation relations that take into account the practical limitations to collect probabilistic information from a given implementation. We will present new versions of the relation defined in Section 4. 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, obtained from the implementation, we will apply a hypothesis contrast to check whether the probabilistic choices taken by the implementation follow the patterns given by the specification.

In order to introduce our new relations, we need some notions to deal with samples collected from an implementation. The next definition presents some auxiliar predicates that we will use during the rest of the section. While the first two notions are easy to understand, the last one needs some additional explanation. Given a trace ρ and a set H of pairs (trace,natural number), $\texttt{IPrefix}(H, \rho)$ is another set of pairs including all traces such that its sequence of input actions matches that of ρ. The number attached to each trace corresponds with the number of traces belonging to H *beginning* with that trace. For instance, let us consider the set of pairs $H = \{((i_1/o_1, i_2/o_1), 1), ((i_1/o_2, i_1/o_2), 2), ((i_1/o_2, i_2/o_1), 3), ((i_2/o_1, i_2/o_2), 4)\}$ and let us apply the function to the trace (i_1/o_1). Then, the resulting set is $H' = \{((i_1/o_1), 1), ((i_1/o_2), 5)\}$. Let us remark that we discard the output of the trace considered as parameter. For instance, in the previous example we considered the trace (i_1/o_1) but the resulting set contained information for both (i_1/o_1) and (i_1/o_2). Given a sample of executions from an implementation, we will use this function to calculate 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 we observe that 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 any $j \leq n$, 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 any prefix of it.

Definition 9. 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 $n < m$ and for any $1 \leq i \leq n$ 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 $(o_1, \ldots, o_m)$.

Let $H = \{(\rho_1, r_1), \ldots, (\rho_m, r_m)\}$ be a set of pairs (trace, natural number) and $\rho = (i_1/o_1, \ldots, i_n/o_n)$ be a trace. The *set of input prefixes* of ρ in H, denoted by $\texttt{IPrefix}(H, \rho)$, is defined as

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

□

Next we present the notions that we will use to denote that a given event has been detected in an implementation.

Definition 10. Let $\mathcal{I} = (S, I, O, \delta, s_0)$ be a PFSM. 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 $\mathcal{I}$. □

Intuitively, a pair (ρ, r) denotes that the trace ρ has been observed r times.

Now we have the necessary notions to introduce our new implementation relations based on samples. In these relations, the non probabilistic constraint is exactly that of the previous one given in Definition 8. Thus, we require that the implementation does not *invent* any behavior that does not exist in the specification (i.e. the implementation does not show any output that cannot be performed by the specification). Let us remark that this constraint could be rewritten in probabilistic terms: The confidence we have on the fact that the implementation will not perform forbidden behaviors is 1. However, let us note that no hypothesis contrast can provide full confidence on the complete absence of a given event. So, it is preferable to keep the constraints over actions separated from the probabilistic constraints and deal with them in the classic way, that is, an implementation is incorrect with respect to forbidden behavior if such a behavior is detected.

Regarding the probabilistic constraints of the specification, the new relations will express them differently to the way we used in Definition 8. In the new setting we put together all the observations of the implementation. Then, the set of samples corresponding to each trace of the specification will be composed 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 to that sequence of inputs in the specification. In the next definition we introduce our first implementation relation based on samples.

Definition 11. Let $\mathcal{S}$ be a specification and $\mathcal{I}$ be an implementation. Let H be an execution sample of $\mathcal{I}$ and let $0 \leq \alpha \leq 1$. 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 any $\rho \in \mathtt{Traces}(\mathcal{S})$ we have $\gamma(\xi^{\pi}_{\mathcal{S}}, R) > \alpha$, where $\pi = \mathtt{inputs}(\rho)$ and $R = \{(\mathtt{outputs}(\rho'), r) \,|\, (\rho', r) \in \mathtt{IPrefix}(H, \rho)\}$. □

In the previous relation, $\xi^{\pi}_{\mathcal{S}}$ denotes the symbolic random variable associated to the PFSM $\mathcal{S}$ and the sequence of input actions π given in Definition 6. Let us remind that this symbolic random variable shows the symbolic probability associated to the traces with the sequence of input actions π in the specification. Besides, each trace observed in the implementation will add one instance to the accounting of each trace being a prefix of it. 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 affect negatively the quality of the hypothesis contrasts. Let us note 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, in particular, $(o_1, \ldots, o_j)$ has been produced in response to $(i_1, \ldots, i_j)$, with $j \leq n$. So, by accounting 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 creation of a new implementation relation that is a refinement of the previous one. 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 more than the number of long 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. For example, let us suppose that the sequence of outputs (o_1, o_2) has been observed 51 times in response to (i_1, i_2), while the sequence (o_5, o_6) has been observed 49 times in response to the same input sequence (i_1, i_2). Then, it would be feasible that the number of times (o_1, o_2, o_3, o_4) was detected in response to (i_1, i_2, i_3, i_4) is 11, while (o_5, o_6, o_7, o_8) was detected 9 times. Similarly, the number of instances of other sequences of four outputs would be lower than those of two outputs. Let us note that statistical noise effects are higher when smaller sets of samples are considered. If we consider the previous example, a hypothesis contrast is more likely to be imprecise in the case of the traces having length four. Moreover, if we consider extremely long traces we could obtain a couple of instances or even none in each class of events to be considered by a hypothesis contrast, which would ruin the result of such 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 growths. This reduction is defined by a non-increasing function associating confidence levels to the length of traces.

Definition 12. Let $f : \mathbb{N} \to \mathbb{R}^+$ be a strictly non-increasing function. Let $\mathcal{S}$ be an specification and $\mathcal{I}$ an implementation. Let 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 any $\rho \in \mathtt{Traces}(\mathcal{S})$ we have $\gamma(\xi^{\pi}_{\mathcal{S}}, R) > f(l)$, where $\pi = \mathtt{inputs}(\rho)$, l is the length of $\mathtt{inputs}(\rho)$, and $R = \{(\mathtt{outputs}(\rho'), r) \mid (\rho', r) \in \mathtt{IPrefix}(H, \rho)\}$. □

6 Conclusions and Future Work

In this paper we have presented a validation methodology to check whether an implemention properly follows the behavior described by a specification. We have considered a framework where the specifications of systems contain probabilistic information. In order to improve the expressivity of specifications, this quantitative information is given in terms of symbolic probabilities. That is, *probabilities* are intervals of values instead of fixed values. This feature increases the complexity of the validation 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 we have defined two implementation relations based on samples. For any given trace performed by the implementation we compute the symbolic probability of the specification to perform it and we compare this value, by using hypothesis contrasts, with the number of times that a particular trace appears in the execution sample. The main difference between both implementation relations is how the hypothesis contrasts are applied.

We are developing a testing methodology so that our implementation relations can be checked by applying a set of tests derived from the specification to the implementation under test. Besides, we would like to study the integration of this framework within that presented in [12], where a testing methodology for stochastic-timed processes is introduced.

Acknowledgments

We would like to thank the anonymous referees of this paper for the careful reading and interesting suggestions. Besides, we would also like to thank Fernando Rubio for his comments and technical support during the development of this paper.

References

1. J.C.M. Baeten and C.A. Middelburg. *Process algebra with timing.* EATCS Monograph. Springer, 2002.
2. K.A. Bartlett, R.A. Scantlebury, and P.T. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Communications of the ACM*, 12(5):260–261, 1969.
3. M. Bravetti and A. Aldini. Discrete time generative-reactive probabilistic processes with different advancing speeds. *Theoretical Computer Science*, 290(1):355–406, 2003.

4. D. Cazorla, F. Cuartero, V. Valero, F.L. Pelayo, and J.J. Pardo. Algebraic theory of probabilistic and non-deterministic processes. *Journal of Logic and Algebraic Programming*, 55(1–2):57–103, 2003.
5. R. Cleaveland, G. Lüttgen, and V. Natarajan. Priority in process algebra. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of process algebra*, chapter 12. North Holland, 2001.
6. R. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative and stratified models of probabilistic processes. *Information and Computation*, 121(1):59–80, 1995.
7. B. Jonsson, W. Yi, and K.G. Larsen. Probabilistic extensions of process algebras. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of process algebra*, chapter 11. North Holland, 2001.
8. K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.
9. X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In *Computer Aided Verification'91, LNCS 575*, pages 376–398, 1991.
10. M. Núñez. Algebraic theory of probabilistic processes. *Journal of Logic and Algebraic Programming*, 56(1–2):117–177, 2003.
11. M. Núñez and I. Rodríguez. Encoding `PAMR` into (timed) `EFSMs`. In *FORTE 2002, LNCS 2529*, pages 1–16. Springer, 2002.
12. M. Núñez and I. Rodríguez. Towards testing stochastic timed systems. In *FORTE 2003, LNCS 2767*, pages 335–350. Springer, 2003.
13. G.M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.
14. J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software – Concepts and Tools*, 17(3):103–120, 1996.
15. J. Tretmans. Testing concurrent systems: A formal approach. In *CONCUR'99, LNCS 1664*, pages 46–65. Springer, 1999.
16. W. Yi and K.G. Larsen. Testing probabilistic and nondeterministic processes. In *Protocol Specification, Testing and Verification XII*, pages 47–61. North Holland, 1992.