Implementation Relations for Stochastic Finite State Machines *

Mercedes G. Merayo, Manuel Núñez and Ismael Rodríguez

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

Abstract. We present a timed extension of the classical finite state machines model where time is introduced in two ways. On the one hand, *timeouts* can be specified, that is, we can express that if an input action is not received before a fix amount of time then the machine will change its state. On the other hand, we can associate time with the performance of actions. In this case, time will be given by means of *random variables*. Intuitively, we will not have conditions such as "the action a takes t time units to be performed" but conditions such as "the action a will be completed before time t with probability p." In addition to introducing the new language, we present several conformance relations to relate implementations and specifications that are defined in terms of our new notion of stochastic finite state machine.

1 Introduction

Formal analysis techniques rely on the idea of constructing a *formal model* that represents the critical aspects of the system under study. These models, simpler and more handleable than the original system, allow to perform a systematic analysis that would be harder, or ever impossible, in the system. For example, the model can be formally manipulated to find out whether a given property holds (for instance, by using *model checking* [CGP00]). The model can be also used to define the specification of a system being constructed. Then, we can check its correctness with respect to the specification by comparing its empirical behavior with that of the model (for instance, by using *formal testing* techniques [BU91,LY96]). In order to use a formal technique, we need that the systems under study can be expressed in terms of a formal language. These languages became more sophisticated as they provided more expressivity capabilities. The first languages represented only the functional behavior of systems (i.e., what must or must not be done). Then, a new generation of languages allowed to explicitly represent non-functional aspects of systems (the probability of performing a certain task [GSS95,CDSY99,SV03,CCV+03,Núñ03,LNR06], the

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

time consumed by the system while performing tasks, being it either given by fix amounts of time [RR88,NS91,HR95] or defined in probabilistic/stochastic terms [Hil96,BG98,Her98,LN01,BG02], the dependence of the system on the available resources [BL97,NR01,CdAHS03], etc).

A suitable representation of the temporal behavior is critical for constructing useful models of real-time systems. A language to represent these systems should enable the definition of temporal conditions that may direct the system behavior, as well as the time consumed by the execution of tasks. Moreover, global temporal requirements should be easily extracted from the requirements of each activity in the system. We can split the time consumed during the execution of a system into the following categories:

(a) The system consumes time while it performs its tasks.
(b) The time passes while the system waits for a reaction from the environment. In particular, the system can change its internal state if an interaction is not received before a certain amount of time.

A language focusing on temporal issues should allow models to explicitly define how the time of type (a) is consumed. Besides, it should allow to define how the system behavior is affected by both types of temporal aspects (e.g., a task is performed if executing the previous task took too much time, if the environment did not react for a long time, if the addition of both times exceeded a given threshold, etc). Finally, the twofold relation between functional activities and temporal aspects should be defined so that they influence each other in an easy way.

In this paper we present a formalism, based on *finite state machines*, allowing to take into account the subtle temporal aspects considered before. Even though there exists a myriad of timed extensions of classical frameworks, this number is not so big in the framework of finite state machines. Moreover, when considering that time is stochastically defined, there are almost no proposals ([NR03] is an exception). Besides, most approaches specialize only in one of the previous variants: Time is either associated with actions or associated with delays/timeouts. Our formalism allows to specify in a natural way both time aspects. In our framework, timeouts are specified by using fix amounts of time. In contrast, the duration of actions will be given by *random variables*. That is, instead of having expressions such as "the action o takes t units of time to be performed" we will have expressions such as "with probability p the action o will be performed before t units of time". We will consider a suitable extension of finite state machines where (stochastic) time information is included. Intuitively, transitions in finite state machines indicate that if the machine is in a state s and receives and input i then it will produce and 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 timed extension of finite state machines, transitions as $s \xrightarrow{i/o}_t s'$ indicate that the time between receiving the input i and returning the output o is equal to t. In the new model that we introduce in this paper for stochastic transitions, we will consider that the time consumed between the input is applied and the

output is received is given by a random variable ξ. Thus the interpretation of a transition $s \xrightarrow{i/o}_{\xi} s'$ is "if the machine is in state s and receives an input i then it will produce the output o before time t with probability $P(\xi \leq t)$ and it will change its state to s'". The definition of conformance testing relations is more difficult than usually. In particular, even in the absence of non-determinism, the same sequence of actions may take different time values to be performed in different runs of the system. While the definition of the new language is not difficult, mixing these temporal requirements strongly complicates the posterior theoretical analysis.

We propose several *stochastic-temporal conformance relations*: An implementation is correct with respect to a specification if it does not show any behavior that is forbidden by the specification, where both the functional behavior and the temporal behavior are considered (and, implicitly, how they affect each other). From the functional point of view, the idea underlying the definition of the conformance relations is that the implementation does not *invent* anything for those sequences of inputs that are *specified* in the specification. Moreover, regarding functional conformance we have to consider not only that the sequences of inputs/outpus produced by the implementation must be considered in the specification. We also have to take into account the possible timeouts. For example, a sequence of inputs/outputs could be accepted after different timeouts have been triggered, and not in the case of other combinations. Regarding stochastic-time, we might require that any trace of the specification that can be performed by the implementation must have the same associated delay, that is, an identically distributed random variable. Even though this is a very reasonable notion to define conformance, if we assume a black-box testing framework then we cannot check whether the corresponding random variables are identically distributed. In fact, we would need an infinite number of observations from a random variable of the implementation (with an unknown distribution) to assure that this random variable is distributed as another random variable from the specification (with a known distribution). Thus, we have to give more *realistic* implementation relations based on a finite set of observations. The idea will be to check that for any trace observed in the implementation that can be performed by the specification, the observed execution times *fit* the random variable indicated by the specification. This notion of *fitting* will be given by means of a hypothesis contrast.

In terms of related work, a lot of formalisms have been proposed to describe the temporal behavior of systems. If we restrict ourselves to time values given by fix amounts, instead of using random variables, our formalism is as expressive as the most popular one: *Timed automata* [AD94]. Thus, our formalism can compare in terms of expressivity with stochastic extensions of timed automata (e.g. [DK05]). However, our way to deal with time is different. As we said before, we can associate time with the performance of actions while timeouts can be easily represented as fix amounts of time. These features do not only improve the modularity of models, but they are also suitable for clearly identifying IUT requirements and responsibilities in a testing methodology. Besides, the formalism

underlying our language is not based on automata but on *finite state machines* (i.e., Mealy machines), which have been extensively used by the formal testing community. Regarding testing of temporal requirements, there exist several proposals (e.g., [CL97,HNTC99,SVD01,NR03,ED03]) but most of them, with the exception of our previous work, are based on timed automata.

The rest of the paper is structured as follows. In the next section we introduce our notion of stochastic finite state machine. In Section 3 we introduce an implementation relation that takes into account only functional aspects, that is, which actions can be performed and how timeouts are specified. This notion is extended in Section 4 to cope with performance time of actions. In Section 5 we present our conclusions and some lines for future work. Finally, in the appendix of the paper, we show how hypothesis contrasts can be performed.

2 A stochastic extension of the EFSM model

In this section we introduce our notion of finite state machines with stochastic time. We use random variables to model the (stochastic) time output actions take to be executed. Thus, we need to introduce some basic concepts on random variables. We will consider that the sample space, that is, the domain of random variables, is a set of numeric time values `Time`. Since this is a *generic* time domain, the specifier can choose whether the system will use a discrete/continuous time domain. We simply assume that $0 \in$ `Time`. Regarding passing of time, we will also consider that machines can evolve by raising *timeouts*. Intuitively, if after a given time, depending on the current state, we do not receive any input action then the machine will change its current state.

During the rest of the paper we will use the following notation. Tuples of elements $(e_1, e_2 \ldots, e_n)$ will be denoted by $\bar{e}$. â denotes an interval of elements $[a_1, a_2)$, with $a_1, a_2 \in$ `Time` and $a_1 < a_2$. We will use the projection function π_i such that given a tuple $\bar{t} = (t_1, \ldots, t_n)$, for all $1 \leq i \leq n$ we have $\pi_i(\bar{t}) = t_i$. Let $\bar{t} = (t_1, \ldots, t_n)$ and $\bar{t}' = (t'_1, \ldots, t'_n)$. We write $\bar{t} = \bar{t}'$ if for all $1 \leq j \leq n$ we have $t_j = t'_j$. We write $\bar{t} \leq \bar{t}'$ if for all $1 \leq j \leq n$ we have $t_j \leq t'_j$. We denote by $\sum \bar{t}$ the addition of all the elements belonging to the tuple $\bar{t}$, that is, $\sum_{j=1}^{n} t_j$. The number of elements of the tuple will be represented by $|\bar{t}|$. Finally, if $\bar{t} = (t_1 \ldots t_n)$, $\bar{p} = (\hat{t_1} \ldots \hat{t_n})$ and for all $1 \leq j \leq n$ we have $t_j \in \hat{t_j}$, we write $\bar{t} \in \bar{p}$.

Definition 1. We denote by $\mathcal{V}$ the set of random variables ($\xi, \psi, \ldots$ range over $\mathcal{V}$). Let ξ be a random variable. We define its *probability distribution function* as the function $F_\xi :$ `Time` $\longrightarrow [0, 1]$ such that $F_\xi(x) = P(\xi \leq x)$, where $P(\xi \leq x)$ is the probability that ξ assumes values less than or equal to x.

Given two random variables ξ and ψ we consider that $\xi + \psi$ denotes a random variable distributed as the addition of the two random variables ξ and ψ. We will call *sample* to any multiset of elements belonging to `Time`. We denote the set of multisets in `Time` by $\wp($`Time`$)$. Let ξ be a random variable and J be a sample. We denote by $\gamma(\xi, J)$ the *confidence* of ξ on J. □

In the previous definition, a sample simply contains an observation of values. In our setting, samples will be associated with the time values that implementations take to perform sequences of actions. We have that $\gamma(\xi, J)$ takes values in the interval $[0, 1]$. Intuitively, bigger values of $\gamma(\xi, J)$ indicate that the observed sample J is more likely to be produced by the random variable ξ. That is, this function decides how *similar* the probability distribution function generated by J and the one corresponding to the random variable ξ are. In the appendix of this paper we show one of the possibilities to formally define the notion of confidence by means of a hypothesis contrast.

Definition 2. A *Stochastic Finite State Machine*, in short `SFSM`, is a tuple $M = (S, I, O, \delta, TO, s_{in})$ where S is the set of states, with $s_{in} \in S$ being the *initial state*, I and O denote the sets of input and output actions, respectively, δ is the set of transitions, and $TO : S \longrightarrow S \times (\texttt{Time} \cup \{\infty\})$ is the *timeout function*. Each transition belonging to δ is a tuple (s, i, o, ξ, s') where $s, s' \in S$ are the initial and final states, $i \in I$ and $o \in O$ are the input and output actions, and $\xi \in \mathcal{V}$ is the random variable defining the time associated with the transition.

Let $M = (S, I, O, \delta, TO, s_{in})$ be a `SFSM`. We say that M is *input-enabled* if for all state $s \in S$ and input $i \in I$ there exist s', o, ξ, such that $(s, i, o, \xi, s') \in \delta$. We say that M is *deterministically observable* if for all s, i, o there do not exist two different transitions $(s, i, o, \xi_1, s_1), (s, i, o, \xi_2, s_2) \in \delta$. □

Intuitively, a transition (s, i, o, ξ, s') indicates that if the machine is in state s and receives the input i then the machine emits the output o before time t with probability $F_\xi(t)$ and the machine changes its current state to s'. Let us remark that non-deterministic choices will be resolved before the timers indicated by random variables start counting, that is, we follow a *pre-selection* policy. Thus, if we have several transitions, outgoing from a state s, associated with the same input i, and the system receives this input, then the system *at time* 0 non-deterministically chooses which one of them to perform. So, we do not have a *race* between the different timers to decide which one is faster. In order to avoid side-effects, we will assume that all the random variables appearing in the definition of a `SFSM` are independent. Let us note that this condition does not restrict the distributions to be used. In particular, there can be random variables identically distributed even though they are independent.

For each state $s \in S$, the application of the timeout function $TO(s)$ returns a pair (s', t) indicating the time that the machine can remain at the state s waiting for an input action and the state to which the machine evolves if no input is received on time. We indicate the absence of a timeout in a given state by setting the corresponding time value to ∞. In addition, we assume that $TO(s) = (s', t)$ implies $s \neq s'$, that is, timeouts always produce a change of the state. In fact, let us note that a definition such as $TO(s) = (s, t)$ is equivalent to set the timeout for the state s to infinite.

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 labelled by the

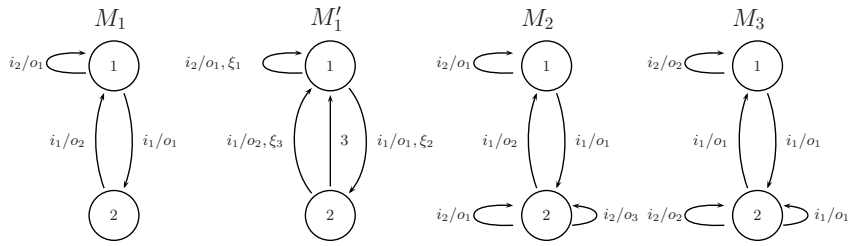

Fig. 1. Examples of (Stochastic) Finite State Machines.

same input action, as far as the outputs are different. Let us remark that both the concept of deterministically observable and input-enabled are independent of the stochastic information appearing in SFSMs.

Example 1. Let us consider the finite state machines M_1, M_2, and M_3 depicted in Figure 1. In order to *transform* these machines into stochastic finite state machines, we simply need to add random variables to all the transitions and timeouts to all the states. We assume that absent timeouts correspond to timeouts having ∞ as parameter. For example, if we transform M_1 into a stochastic state machine we obtain M_1'. Then we have $M_1' = (\{1,2\}, \{i_1, i_2\}, \{o_1, o_2\}, \delta, TO, 1)$ where the set of transitions δ is given by:

$$\delta = \{(1, i_2, o_1, \xi_1, 1), (1, i_1, o_1, \xi_2, 2), (2, i_1, o_2, \xi_3, 1)\}$$

In order to complete our specification of M_1' we need to say how random variables are distributed. Let us suppose the following distributions:

$$F_{\xi_1}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ \frac{x}{3} & \text{if } 0 < x < 3 \\ 1 & \text{if } x \geq 3 \end{cases}$$

$$F_{\xi_2}(x) = \begin{cases} 0 & \text{if } x < 4 \\ 1 & \text{if } x \geq 4 \end{cases}$$

$$F_{\xi_3}(x) = \begin{cases} 1 - e^{-3 \cdot x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

We say that ξ_1 is uniformly distributed in the interval $[0, 3]$. Uniform distributions allow us to keep compatibility with time intervals in (non-stochastic) timed models in the sense that the same *weight* is assigned to all the times in the interval. We say that ξ_2 is a Dirac distribution in 4. The idea is that the corresponding delay will be equal to 4 time units. Dirac distributions allow us to simulate deterministic delays appearing in timed models. We say that ξ_3 is exponentially

distributed with parameter 3. Let us consider the transition $(1, i_2, o_1, \xi_1, 1)$. Intuitively, if M_1' is in state 1 and it receives the input i_2 then it will produce the output o_2 after a delay given by ξ_1. For example, we know that this delay will be less than 1 unit of time with probability $\frac{1}{3}$, it will be less than 1.5 units of time with probability $\frac{1}{2}$, and so on. Finally, once 3 units of time has passed we know that the output o_1 has been performed (that is, we have probability 1). Regarding the timeout function, we have $TO(1) = (2, \infty)$ and $TO(2) = (1, 3)$. In this case, if the machine is in state 2 and no input is received before 3 units of time then the state is changed to 1.

Regarding the notions of input-enabled and deterministically observable, the first property does not hold in M_1 (there is no outgoing transition labelled by i_2 from the state 2) while the second one does. We have that M_3 fulfills the first of the properties but not the second one (there are two transitions from the state 2 labelled by i_1/o_1). Finally, both properties hold for M_2. □

Definition 3. Let $M = (S, I, O, \delta, TO, s_{in})$ be a `SFSM`. We say that a tuple $(s_0, s, i/o, \hat{t}, \xi)$ is a *step* for the state s_0 of M if there exist k states $s_1, \ldots, s_k \in S$, with $k \geq 0$, such that $\hat{t} = \left[\sum_{j=0}^{k-1} \pi_2(TO(s_j)), \sum_{j=0}^{k} \pi_2(TO(s_j))\right)$ and there exists a transition $(s_k, i, o, \xi, s) \in \delta$.

We say that $(\hat{t}_1/i_1/\xi_1/o_1, \ldots, \hat{t}_r/i_r/\xi_r/o_r)$ is a *stochastic evolution* of M if there exist r steps of M $(s_{in}, s_1, i_1/o_1, \hat{t}_1, \xi_1), \ldots, (s_{r-1}, s_r, i_r/o_r, \hat{t}_r, \xi_r)$ for the states $s_{in} \ldots s_{r-1}$, respectively. We denote by $\texttt{SEvol}(M)$ the set of stochastic evolutions of M. In addition, we say that $(\hat{t}_1/i_1/o_1, \ldots, \hat{t}_r/i_r/o_r)$ is a *functional evolution* of M. We denote by $\texttt{FEvol}(M)$ the set of functional evolutions of M. We will use the shortenings $(\sigma, \bar{p})$ and $(\sigma, \bar{p}, \bar{\xi})$ to denote a functional and a stochastic evolution, respectively, where $\sigma = (i_1/o_1 \ldots i_r/o_r)$, $\bar{p} = (\hat{t_1} \ldots \hat{t_r})$ and $\bar{\xi} = (\xi_1 \ldots \xi_r)$. □

Intuitively, a step is a sequence of transitions that contains an action transition preceded by zero or more timeouts. The interval $\hat{t}$ indicates the time values where the transition could be performed. In particular, if the sequence of timeouts is empty then we have the interval $\hat{t} = [0, TO(s_0))$. An evolution is a sequence of inputs/outputs corresponding to the transitions of a chain of steps, where the first one begins with the initial state of the machine. In addition, stochastic evolutions also include time information which inform us about possible timeouts (indicated by the intervals $\hat{t}_j$) and random variables associated to the execution of each output after receiving each input in each step of the evolution. In the following definition we introduce the concept of *instanced evolution*. Intuitively, instanced evolutions are constructed from evolutions by instantiating to a concrete value each timeout, given by an interval, of the evolution.

Definition 4. Let $M = (S, I, O, \delta, TO, s_{in})$ be a `SFSM` and let us consider a *stochastic evolution* $e = (\hat{t}_1/i_1/\xi_1/o_1, \ldots, \hat{t}_r/i_r/\xi_r/o_r)$. We say that the tuple $(t_1/i_1/\xi_1/o_1, \ldots, t_r/i_r/\xi_r/o_r)$ is an *instanced stochastic evolution of* e if for all $1 \leq j \leq r$ we have $t_j \in \hat{t}_j$. Besides, we say that the tuple $(t_1/i_1/o_1, \ldots, t_r/i_r/o_r)$ is an *instanced functional evolution* of e.

We denote by $\texttt{InsSEvol}(M)$ the set of instanced stochastic evolutions of M and by $\texttt{InsFEvol}(M)$ the set of instanced functional evolutions of M. □

3 A first Implementation Relation

In this section we introduce an implementation relation to deal with functional aspects. It follows the pattern borrowed from $\texttt{conf}_{nt}$ [NR02]: An implementation I *conforms* to a specification S if for all possible evolution of S the outputs that the implementation I may perform after a given input are a subset of those for the specification. In addition to the non-stochastic conformance of the implementation, we require other additional conditions, related to time, to hold. Specifically, we require that the implementation always complies with the timeouts established by the specification.

Next, we fix the sets of specifications and implementations. A specification is a stochastic finite state machine. Regarding implementations, we consider that they are also given by means of SFSMs. We will consider that both specifications and implementations are given by deterministically observable SFSMs. That is, we do not allow a machine to have two different transitions such as (s, i, o, ξ_1, s') and (s, i, o, ξ_2, s''). Let us note that we do not restrict observable non-determinism, that is, we may have the transitions (s, i, o_1, ξ_1, s_1) and $(s, i, o_2, \xi_2, s_2,)$ as far as $o_1 \neq o_2$. Besides, we assume that input actions are always enabled in any state of the implementation, that is, implementations are input-enabled according to Definition 2. This is a usual condition to assure that the implementation will react (somehow) to any input appearing in the specification.

First, we introduce the implementation relation $\texttt{conf}_f$, where only functional aspects of the system (i.e., which outputs are allowed/forbidden and how timeouts are defined) are considered while the performance of the system (i.e., how fast outputs are executed) is ignored. Let us note that the time spent by a system waiting for the environment to react has the capability of affecting the set of available outputs of the system. This is because this time may trigger a change of the state. So, a relation focusing on functional aspects must explicitly take into account the maximal time the system may stay in each state. This time is given by the *timeout* of each state.

Definition 5. Let S and I be SFSMs. We say that I *functionally conforms* to S, denoted by $I\ \texttt{conf}_f\ S$, if for each functional evolution $e \in \texttt{FEvol}(S)$, with $e = (\hat{t}_1/i_1/o_1, \ldots, \hat{t}_r/i_r/o_r)$ and $r \geq 1$, we have that for all $t_1 \in \hat{t}_1, \ldots, t_r \in \hat{t}_r$ and o'_r

$$e' = (t_1/i_1/o_1, \ldots, t_r/i_r/o'_r) \in \texttt{InsFEvol}(I) \text{ implies } e' \in \texttt{InsFEvol}(S)$$

□

Intuitively, the idea underlying the definition of the functional conformance relation $I\ \texttt{conf}_f\ S$ is that the implementation I does not *invent* anything for those sequences of inputs that are *specified* in the specification S. Let us note

that if the specification has also the property of input-enabled then we may remove the condition "for each functional evolution $e \in \texttt{FEvol}(S)$, with $e = (\hat{t}_{t1}/i_1/o_1, \ldots, \hat{t}_{tr}/i_r/o_r)$ and $r \geq 1$".

In addition to requiring this notion of *functional* conformance, we have to ask for some conditions on delays. As indicated in the introduction, a first approach would be to require that the random variables associated with evolutions of the implementation are identically distributed as the ones corresponding to the specification. However, the fact that we assume a black-box testing framework disallows us to check whether these random variables are indeed identically distributed. Thus, we have to give more *realistic* implementation relations based on finite sets of observations. Next, we present implementation relations that are less *accurate* but that are *checkable*.

4 Implementation Relations based on Samples

In the previous section we discussed how an appropriate implementation relation can be defined. Unfortunately, this notion is useful only from a theoretical point of view. In this section we introduce implementation relations that take into account the observations that we may get from the implementation. We will collect a sample of time values and we will *compare* this sample with the random variables appearing in the specification. By comparison we mean that we will apply a contrast to decide, with a certain confidence, whether the sample could be generated by the corresponding random variable.

Definition 6. Let I be a SFSM. We say that $(\sigma, \bar{t}, \bar{t}')$, with $\sigma = i_1/o_1, \ldots, i_n/o_n$, $\bar{t} = (t_1 \ldots t_n)$, and $\bar{t}' = (t'_1 \ldots t'_n)$, is an *observed time execution* of I, or simply *time execution*, if the observation of I shows that for all $1 \leq j \leq n$ we have that the time elapsed between the acceptance of the input i_j and the observation of the output o_j is t'_j units of time, being the input i_j accepted t_j units of time after the last output was observed.

Let $\Phi = \{(\sigma_1, \bar{p}_1), \ldots, (\sigma_m, \bar{p}_n)\}$ where for all $1 \leq j \leq n$ we have $\bar{p}_j = (\hat{t_1} \ldots \hat{t_n})$, and let $H = \{\!\!\{(\sigma'_1, \bar{t}_1, \bar{t}'_1), \ldots, (\sigma'_n, \bar{t}_n, \bar{t}'_n)\}\!\!\}$ be a multiset of timed executions. We say that $\texttt{Sampling}^k_{(H,\Phi)} : \Phi \longrightarrow \wp(\texttt{Time})$ is a *k-sampling application* of H for Φ if $\texttt{Sampling}^k_{(H,\Phi)}(\sigma, \bar{p}) = \{\!\!\{\pi_k(\bar{t}') \mid (\sigma, \bar{t}, \bar{t}') \in H \ \wedge \ |\sigma| \geq k \ \wedge \ \bar{t} \in \bar{p}\}\!\!\}$, for all $(\sigma, \bar{p}) \in \Phi$. We say that $\texttt{Sampling}_{(H,\Phi)} : \Phi \longrightarrow \wp(\texttt{Time})$ is a *sampling application* of H for Φ if $\texttt{Sampling}_{(H,\Phi)}(\sigma, \bar{p})) = \{\!\!\{\sum \bar{t}' \mid (\sigma, \bar{t}, \bar{t}') \in H \ \wedge \ \bar{t} \in \bar{p}\}\!\!\}$, for all $(\sigma, \bar{p}) \in \Phi$. □

Regarding the definition of k-sampling applications, we just associate with each subtrace of length k the observed time of each transition of the execution at length k. In the definition of sampling applications, we assign to each trace the total observed time corresponding to the whole execution.

Definition 7. Let I and S be SFSMs, H be a multiset of timed executions of I, $0 \leq \alpha \leq 1$, $\Phi = \texttt{FEvol}(S)$, and let us consider $\texttt{Sampling}_{(H,\Phi)}$ and $\texttt{Sampling}^k_{(H,\Phi)}$, for all $1 \leq k \leq max\{|\sigma| \mid (\sigma, \bar{p}) \in \Phi\}$.

We say that I $(\alpha, H)-$*strong stochastically conforms* to S, and we denote it by $I\ \mathtt{confs}_s^{(\alpha,H)}\ S$, if $I\ \mathtt{conf}_f\ S$ and for all $(\sigma, \bar{t}, \bar{t}') \in H$ we have

$$\exists (\sigma, \bar{p}, \bar{\xi}) \in \mathtt{SEvol}(S) : \bar{t} \in \bar{p}$$
$$\Downarrow$$
$$\forall\ 1 \leq j \leq |\sigma| : \gamma(\pi_j(\bar{\xi}), \mathtt{Sampling}^j_{(H,\Phi)}(\sigma, \bar{p})) > \alpha$$

We say that I $(\alpha, H)-$*weak stochastically conforms* to S, and we denote it by $I\ \mathtt{confs}_w^{(\alpha,H)}\ S$, if $I\ \mathtt{conf}_f\ S$ and for all $(\sigma, \bar{t}, \bar{t}') \in H$ we have

$$\exists (\sigma, \bar{p}, \bar{\xi}) \in \mathtt{SEvol}(S) : \bar{t} \in \bar{p}$$
$$\Downarrow$$
$$\gamma\left(\sum_{j=1}^{|\sigma|} \pi_j(\bar{\xi}), \mathtt{Sampling}_{(H,\Phi)}(\sigma, \bar{p})\right) > \alpha$$

□

The idea underlying the new relations is that the implementation must conform to the specification in the usual way (that is, $I\ \mathtt{conf}_f\ S$). Besides, for all observation of the implementation that can be performed by the specification, the observed execution time values *fit* the random variable indicated by the specification. This notion of *fitting* is given by the function γ that it is formally defined in the appendix of this paper. While the *weak* notion only compares the total time, the *strong* notion checks that the time values are appropiate for each performed output. A first direct result says that if we decrease the confidence level then we keep conformance.

Lemma 1. Let I and S be SFSMs. If $I\ \mathtt{confs}_s^{(\alpha_1,H)}\ S$ and $\alpha_2 < \alpha_1$ then we have $I\ \mathtt{confs}_s^{(\alpha_2,H)}\ S$. If $I\ \mathtt{confs}_w^{(\alpha_1,H)}\ S$ and $\alpha_2 < \alpha_1$ then we have $I\ \mathtt{confs}_w^{(\alpha_2,H)}\ S$. □

The next result, whose proof is straightforward, states that if we have two samples sharing some properties then our conformance relations give the same result for both of them.

Lemma 2. Let I and S be SFSMs, H_1 and H_2 be multisets of timed executions for I, and let $b_i = \{\!\{(\sigma, \bar{t}, \bar{t}') \,|\, (\sigma, \bar{t}, \bar{t}') \in H_i \wedge (\sigma, \bar{t}) \in \mathtt{InsFEvol}(I) \cap \mathtt{InsFEvol}(S)\}\!\}$, for $i \in \{1, 2\}$. If $b_1 = b_2$ then we have $I\,\mathtt{confs}_s^{(\alpha,H_1)}\,S$ iff $I\,\mathtt{confs}_s^{(\alpha,H_2)}\,S$. Similarly, if $b_1 = b_2$ then we have $I\ \mathtt{confs}_w^{(\alpha,H_1)}\ S$ iff $I\ \mathtt{confs}_w^{(\alpha,H_2)}\ S$. □

Lemma 3. Let I and S be SFSMs. We have $I\,\mathtt{confs}_s^{(\alpha,H)}\,S$ implies $I\,\mathtt{confs}_w^{(\alpha,H)}\,S$. □

Next we present different variations of the previous implementation relation. First, we define the concept of *shifting* a random variable with respect to its mean. For example, let us consider a random variable ξ following a Dirac distribution in 4 (see Example 1 for the formal definition). If we consider a new random variable ξ' following a Dirac distribution in 3, we say that ξ' represents a shift of ξ. Moreover, we also say that ξ and ξ' belong to the same family.

Definition 8. We say that ξ' is a *mean shift* of ξ with mean M', and we denote it by $\xi' = \texttt{MShift}(\xi, M')$, if ξ, ξ' belong to the same family and the mean of ξ', denoted by $\mu_{\xi'}$, is equal to M'.

Let I and S be SFSMs, H be a multiset of timed executions of I, $0 \leq \alpha \leq 1$, $\Phi = \texttt{FEvol}(S)$, and let us consider $\texttt{Sampling}_{(H,\Phi)}$ and $\texttt{Sampling}^k_{(H,\Phi)}$ for all $1 \leq k \leq max\{|\sigma| \mid (\sigma, \bar{p}) \in \Phi\}$. We say that I $(\alpha, H)-$*strongly stochastically conforms* to S *with speed* π, denoted by $I\,\texttt{confm}^{(\alpha,H)}_{s\pi} S$, if $I\ \texttt{conf}_f\ S$ and for all $(\sigma, \bar{t}, \bar{t}') \in H$ we have

$$\begin{array}{c} \exists\ (\sigma, \bar{p}, \bar{\xi}) \in \texttt{SEvol}(S) : \bar{t} \in \bar{p} \\ \Downarrow \\ \forall\ 1 \leq j \leq |\sigma| : \gamma(\texttt{MShift}(\pi_j(\xi), \mu_{\pi_j(\xi)} \cdot \pi), \texttt{Sampling}^j_{(H,\Phi)}(\sigma, \bar{p})) > \alpha \end{array}$$

We say that I $(\alpha, H)-$*weakly stochastically conforms* to S *with speed* π, denoted by $I\,\texttt{confm}^{(\alpha,H)}_{w\pi} S$, if $I\ \texttt{conf}_f\ S$ and for all $(\sigma, \bar{t}, \bar{t}') \in H$ we have

$$\begin{array}{c} \exists\ (\sigma, \bar{p}, \bar{\xi}) \in \texttt{SEvol}(S) : \bar{t} \in \bar{p} \\ \Downarrow \\ \gamma(\texttt{MShift}(\xi, \mu_\xi \cdot \pi), \texttt{Sampling}_{(H,\Phi)}(\sigma, \bar{p})) > \alpha \end{array}$$

where we have considered $\xi = \sum_{j=1}^{|\sigma|} \pi_j(\bar{\xi})$. □

An interesting remark regarding these new relations is that when α is *small enough* and/or π is *close enough* to 1, then it may happen that we have both $I\,\texttt{confs}^{(\alpha,H)}_s S$ and $I\,\texttt{confm}^{(\alpha,H)}_{s\pi} S$, and similarly for the case of $I\,\texttt{confs}^{(\alpha,H)}_w S$ and $I\,\texttt{confm}^{(\alpha,H)}_{w\pi} S$. Nevertheless, it is enough to increase α, as far as $\pi \neq 1$, so that we do not have both results strong/weak conformance notions simultaneously. Let us note that in the previous definition, a value of π greater than 1 indicates that the new delay is *slower*. This observation induces the following relation.

Definition 9. Let I and S be SFSMs. Let H be a multiset of timed executions of I. We say that I is *strong-generally faster* (respectively *strong-generally slower*) than S for H if there exist $0 \leq \alpha \leq 1$ and $0 < \pi < 1$ (respectively $\pi > 1$) such that $I\,\texttt{confm}^{(\alpha,H)}_{s\pi} S$ but $I\ \texttt{confs}^{(\alpha,H)}_s\ S$ does not hold. We say that I is *weak-generally faster* (respectively *weak-generally slower*) than S for H if there exist $0 \leq \alpha \leq 1$ and $0 < \pi < 1$ (respectively $\pi > 1$) such that $I\,\texttt{confm}^{(\alpha,H)}_{w\pi} S$ but $I\ \texttt{confs}^{(\alpha,H)}_w\ S$ does not hold. □

Given the fact that, in our framework, an implementation could *fit better* to a specification with higher or lower speed, it will be interesting to detect which variations of speed would make the implementation to fit better the specification. Intuitively, the best variation will be the one allowing the implementation to conform to the specification with a *higher* level of confidence α.

Definition 10. Let I and S be SFSMs. Let H be a multiset of timed executions of I. Let us consider $0 \leq \alpha \leq 1$ such that $I\,\mathtt{confm}_{s\pi}^{(\alpha,H)}S$, $I\,\mathtt{confm}_{w\pi}^{(\alpha,H)}S$, and there do not exist $\alpha' > \alpha$ and $\pi' \in \mathbb{R}^+$ with $I\,\mathtt{confm}_{s\pi'}^{(\alpha',H)}S$. Then, we say that π is a *relative speed* of I with respect to S for H. □

The concept of relative speed allows us to define another implementation relation which is more restrictive than those presented so far. Basically, the implementation must both $(\alpha, H)-$stochastically conform to the specification and have 1 as a relative speed. Let us note that the latter condition means that the implementation fits perfectly in its current speed.

Definition 11. Let I and S be SFSMs. Let H be a multiset of timed executions of I and let us consider $0 \leq \alpha \leq 1$. We say that I $(\alpha, H)-$*stochastically and precisely strong conforms* to S, denoted by $I\,\mathtt{confp}_s^{(\alpha,H)}\,S$, if $I\,\mathtt{confs}_s^{(\alpha,H)}\,S$ and we have that 1 is a relative speed of I with respect to S for H. Similarly, we say that I $(\alpha, H)-$*stochastically and precisely weak conforms* to S, denoted by $I\,\mathtt{confp}_S^{(\alpha,H)w}$, if $I\,\mathtt{confs}_w^{(\alpha,H)}\,S$ and we have that 1 is a relative speed of I with respect to S for H. □

The following result relates some of the notions presented in this section.

Lemma 4. Let I and S be SFSMs. We have $I\,\mathtt{confp}_s^{(\alpha,H)}\,S$ iff $I\,\mathtt{confs}_s^{(\alpha,H)}\,S$ and neither I is strong-generally faster than S for H nor I is strong-generally slower than S for H.

We have $I\,\mathtt{confp}_w^{(\alpha,H)}\,S$ iff $I\,\mathtt{confs}_w^{(\alpha,H)}\,S$ and neither I is weak-generally faster than S for H nor I is weak-generally slower than S for H.

□

5 Conclusions and Future Work

In this paper we have presented a new notion of finite state machine. In contrast with most timed extensions, our formalism allows to specify in an easy way both the passing of time due to timeouts and the time due to the performance of actions. In the first case, we consider that timeouts are given by fix amounts of time. For each state of the machine, if after a certain time no input action is received then the machine changes the state. In the second case, time is introduced by means of random variables. Thus, we are able to specify the time elapsed from the reception of an input until the observation of an output. These time values are specified by means of random variables. Finally, we have presented several implementation relations based on the notion of conformance. These relations share a common pattern: The implementation must conform to the specification regarding functional aspects. In addition to require that different sequences of actions are performed in the implementation as indicated by the specification, the timeouts of the implementation have also to be placed according to the ones of the specification. Our implementation relations also impose

some conditions regarding the random variables appearing in both specifications and implementations.

As future work we plan to introduce an appropriate notion of test and to define how tests are applied to implementations. In this sense, we will give a notion of passing a test suite *up to a certain probability*. The final goal will be to relate our implementation relations with this notion of passing tests. This will be done by providing a test derivation algorithm to obtain sound and complete test suites with respect to some of the implementation relations given in this paper.

References

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[BG98] M. Bernardo and R. Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202:1–54, 1998.

[BG02] M. Bravetti and R. Gorrieri. The theory of interactive generalized semi-Markov processes. *Theoretical Computer Science*, 282(1):5–32, 2002.

[BL97] P. Brémond-Grégoire and I. Lee. A process algebra of communicating shared resources with dense time and priorities. *Theoretical Computer Science*, 189(1-2):179–219, 1997.

[BU91] B.S. Bosik and M.U. Uyar. Finite state machine based formal methods in protocol conformance testing. *Computer Networks & ISDN Systems*, 22:7–33, 1991.

[CCV+03] 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.

[CdAHS03] A. Chakrabarti, L. de Alfaro, T.A. Henzinger, and M. Stoelinga. Resource interfaces. In *3rd Int. Conf. on Embedded Software, EMSOFT 2003, LNCS 2855*, pages 117–133. Springer, 2003.

[CDSY99] R. Cleaveland, Z. Dayar, S.A. Smolka, and S. Yuen. Testing preorders for probabilistic processes. *Information and Computation*, 154(2):93–148, 1999.

[CGP00] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.

[CL97] D. Clarke and I. Lee. Automatic generation of tests for timing constraints from requirements. In *3rd Workshop on Object-Oriented Real-Time Dependable Systems*, 1997.

[DK05] P.R. D'Argenio and J.-P. Katoen. A theory of stochastic systems part I: Stochastic automata. *Information and Computation*, 203(1):1–38, 2005.

[ED03] A. En-Nouaary and R. Dssouli. A guided method for testing timed input output automata. In *TestCom 2003, LNCS 2644*, pages 211–225. Springer, 2003.

[GSS95] 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.

[Her98] H. Hermanns. *Interactive Markov Chains*. PhD thesis, Universität Erlangen-Nürnberg, 1998.

[Hil96] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[HNTC99] T. Higashino, A. Nakata, K. Taniguchi, and A. Cavalli. Generating test cases for a timed I/O automaton model. In *12th Workshop on Testing of Communicating Systems*, pages 197–214. Kluwer Academic Publishers, 1999.

[HR95] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117(2):221–239, 1995.

[LN01] N. López and M. Núñez. A testing theory for generally distributed stochastic processes. In *CONCUR 2001, LNCS 2154*, pages 321–335. Springer, 2001.

[LNR06] N. López, M. Núñez, and I. Rodríguez. Specification, testing and implementation relations for symbolic-probabilistic systems. *Theoretical Computer Science*, 353(1–3):228–248, 2006.

[LY96] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines: A survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.

[NR01] M. Núñez and I. Rodríguez. `PAMR`: A process algebra for the management of resources in concurrent systems. In *FORTE 2001*, pages 169–185. Kluwer Academic Publishers, 2001.

[NR02] M. Núñez and I. Rodríguez. Encoding `PAMR` into (timed) `EFSMs`. In *FORTE 2002, LNCS 2529*, pages 1–16. Springer, 2002.

[NR03] M. Núñez and I. Rodríguez. Towards testing stochastic timed systems. In *FORTE 2003, LNCS 2767*, pages 335–350. Springer, 2003.

[NS91] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In *Computer Aided Verification'91, LNCS 575*, pages 376–398. Springer, 1991.

[Núñ03] M. Núñez. Algebraic theory of probabilistic processes. *Journal of Logic and Algebraic Programming*, 56(1–2):117–177, 2003.

[RR88] G.M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.

[SV03] M. Stoelinga and F. Vaandrager. A testing scenario for probabilistic automata. In *ICALP 2003, LNCS 2719*, pages 464–477. Springer, 2003.

[SVD01] J. Springintveld, F. Vaandrager, and P.R. D'Argenio. Testing timed automata. *Theoretical Computer Science*, 254(1-2):225–257, 2001.

Appendix. Statistics Background: Hypothesis Contrasts

In this appendix we introduce one of the standard ways to measure the confidence degree that a random variable has on a sample. In order to do so, we will present a methodology to perform *hypothesis contrasts*. The underlying idea is that 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 we have observed 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 which cover all the possible range of values. We denote by O_i the *observed frequency* at class i (i.e. the number of elements belonging to the class i).

- We calculate the probability p_i of each class, according to the proposed random variable. We denote by E_i the *expected frequency*, which is given by $E_i = np_i$.
- We calculate the *discrepancy* between observed frequencies 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 .
- We estimate the number of freedom degrees of χ^2 as $k - r - 1$. In this case, 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 (i.e. $r = 0$ if 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 or equal to the discrepancy observed is high enough, that is, if $X^2 < \chi^2_\alpha(k - r - 1)$ for some α low enough. Actually, as such margin to accept the sample decreases as α decreases, we can obtain a measure of the validity of the sample as $max\{\alpha \mid X^2 < \chi^2_\alpha(k - r - 1)\}$.

According to the previous steps, we can now present an operative definition of the function γ which is used in this paper to compute the confidence of a random variable on a sample.

Definition 12. Let ξ be a random variable and let J be a multiset of real numbers representing a sample. Let X^2 be the discrepancy level of J on ξ calculated as explained above by splitting the sampling space into the set of classes $C = \{[0, a_1), [a_1, a_2), \ldots, [a_{k-1}, a_k), [a_k, \infty)\}$, where k is a given constant and for all $1 \leq i \leq k$ we have $a_i = q$ where $P(\xi \leq q) = \frac{i}{k+1}$. We define the confidence of ξ on J with classes S, denoted by $\gamma(\xi, J)$, as $max\{\alpha \mid X^2 < \chi^2_\alpha(k - 1)\}$. □

Let us comment some important details. First, given the fact that the random variables that we use in our framework denote the passing of time, we do not need classes to cover negative values. Thus, we will suppose that the class containing 0 will also contain all the negative values. Second, let us remark that in order to apply this contrast it is strongly recommended that the sample has at least 30 elements while each class must contain at least 3 elements.