Fifth IEEE International Conference on Software Engineering and Formal Methods

Testing conformance on Stochastic Stream X-Machines*

Mercedes G. Merayo and Manuel Núñez
Dept. Sistemas Informáticos y Computación
Universidad Complutense de Madrid
E-28040 Madrid. Spain.
mgmerayo@fdi.ucm.es mn@sip.ucm.es

Abstract

Stream X-machines have been used to specify real systems requiring to represent complex data structures. One of the advantages of using stream X-machines to specify a system is that it is possible to produce a test set that, under certain conditions, detects all the faults of an implementation. In this paper we present a formal framework to test temporal behaviors in systems where temporal aspects are critical. Temporal requirements are expressed by means of random variables and affect the duration of actions. Implementation relations are presented as well as a method to determine the conformance of an implementation with respect to a specification by applying a test set.

1. Introduction

Testing consists in checking the conformity of an implementation by performing experiments on it. The application of formal testing techniques [7, 26, 25, 35] to check the correctness of a system requires to identify the *critical* aspects of the system, that is, those aspects that will make the difference between correct and incorrect behavior. While the relevant aspects of some systems only concern *what* they do, in some other systems it is equally relevant *how* they do what they do. For instance, the time consumed by each operation should be considered critical in real-time systems.

In order to perform this task, several techniques, algorithms, and semantic frameworks have been introduced in the literature. The testing community has shown a growing interest in extending these frameworks so that not only functional properties but also quantitative ones could be tested. Thus, there have been several proposals for timed testing (e.g. [31, 11, 17, 36, 12, 34, 24, 8]).

*Research partially supported by the Spanish MEC projects MASTER/TERMAS (TIC2003-07848-C02-01) and WEST/FAST (TIN2006-15578-C02-01), and the Marie Curie project MRTN-CT-2003-505121/TAROT.

Even though the inclusion of time allows the specifier to give a more precise description of the system to be implemented, there are frequent situations that cannot be accurately described by using a *simple* notion of time such as a task takes t seconds to be performed". For example, we may desire to specify a system where a message is expected to be received with probability $\frac{1}{2}$ in the interval $(0, 1]$, with probability $\frac{1}{4}$ in $(1, 2]$, and so on. This is an advantage with respect to usual *deterministic* time where we could only specify that the message arrives in the interval $(0, \infty)$. Thus, *stochastic* extensions of classical formal models have appeared in the literature (see e.g. [14, 1, 18, 5, 15, 9, 29, 28]). As we have shown in the previous example, the main idea underlying stochastic models is that time information is incremented with some kind of probabilistic information.

In contrast with testing timed systems, testing stochastic systems has received almost no attention. In fact, to the best of our knowledge, there are only three works in the field [4, 27, 33]. We think that this lack of research is due to the technical difficulties in the definition of the corresponding notions of testing. For example, even if we consider white-box testing, as it is the case of [4, 27], we still have technical problems as the aggregation of delays. In the case of black-box testing, that we consider in this paper, the problem of detecting *time faults* is much more involved than in the case of timed testing. This is so because we will suppose that the test(er) has no access to the probability distribution functions associated with delays. Let us remark that due to the probabilistic nature of time in stochastic timed systems, several observations of a system may lead to different execution times. Actually, this may happen even if the implementation performs in all the cases the same sequence of actions (and through the same sequence of states).

One of the possibilities to formally specify systems is to use X-machines [19]. They can be seen as a form of finite state machine where transitions are labelled with relations over a basic data set. This set of relations is called the type of the machine and represents the operations that can perform. Different types of machines derived from the

0-7695-2884-8/07 $25.00 © 2007 IEEE
DOI 10.1109/SEFM.2007.15

original model, in particular, stream X-machines. Stream X-machines have an internal memory; the input, the current state, and the current value of the internal memory determine the next state, the output, and how the memory is updated. The main advantages of using them for representing systems is that they can be adapted to the different necessities the systems can present as well as they allow to integrate control and data processing. This formalism has been used to specify systems in different areas [20, 3, 23] and several testing techniques have been developed.

The standard stream X-Machines test generation algorithm is based in the W-method introduced by [10] in the context of finite state machines. The method presents some restrictions: (a) specification and implementation must be deterministic (b) the functions associated to the transitions are correctly implemented and (c) it assumes that certain conditions, called design for test conditions, hold. Under these assumptions the set of tests generated by this method is guaranteed to determine the correctness of the system [21, 20]. This integration of specification and test generation is one of the most significant benefits of using deterministic stream X-machines in software development. Since this method was developed, it has been extended for reducing the imposed conditions (for a survey see [6]).

In this paper we present a formal testing methodology where the temporal behavior of systems is considered. Specifically, we will consider the approach sketched in [33] in the context of stream X-machines. A suitable extension of the classical concept of *stream X-machine* will allow a specifier to explicitly denote temporal requirements for each action of a system, taking advantage of the power that this model provides. In the new formalism we will consider that the *delay* between the input is applied and the output is received is given by a random variable ξ. In order to cope with this extension, the semantic frameworks have to be strongly modified. In this line, we propose a conformance relation according to our notion of correct implementation respect to a specification for systems that present stochastic information.

Regarding the derivation of test sets, we adapt the standard algorithm used for deterministic stream X-machines. The inclusion of stochastic information is included in the tests, allowing to check that the implementations fulfill the time restrictions present in the specification in accordance with the notion of conformance considered.

The rest of the paper is organized as follows. In Section 2 we present basic concepts regarding random variables and finite automata. In Section 3 we introduce our model to represent stochastic systems by means of stochastic stream X-machines. In Section 4 we introduce a notion of conformance for our framework where temporal behavior is taken into account. In Section 5 we show how stochastic stream X-machines can be tested and how to derive test sets from stochastic stream X-machines, and the application of the test set with our notion of conformance. Finally, in Section 6 we present our conclusions.

2. Preliminaries

In this section we introduce some basic concepts that will be used along the paper.

2.1. Random variables

We will consider that the sample space (that is, the domain of random variables) is the set of real numbers $\mathbf{R}$. Besides, random variables take positive values only in $\mathbf{R}^+$, that is, in the set of non-negative real numbers. The reason for this restriction is that they will always be associated with time distributions, so they cannot take a negative value.

Definition 1 We denote by $\mathcal{V}$ the set of random variables ($\xi, \psi, \ldots$ to range over $\mathcal{V}$). Let ξ be a random variable. We define its *probability distribution function* as the function $F_\xi : \mathbf{R}_+ \longrightarrow [0,1]$ 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. Let $\xi, \xi' \in \mathcal{V}$ be random variables. We write $\xi = \xi'$ if for any $x \in \mathbf{R}$ we have $F_\xi(x) = F_{\xi'}(x)$. We will denote by θ the random variable which probability distribution function is defined by $F(x) = 1$ for all $x \in \mathbf{R}_+$.

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 a set. Let A be a set. We denote the *powerset* of A by $\mathcal{P}(A)$. We denote the set of multisets in A by $\wp(A)$. Let ξ be a random variable and J be a sample of positive real numbers. We denote by $\gamma(\xi, J)$ the *confidence* of ξ on J. □

In the previous definition, a sample simply denotes a collection of observed values by means of an experiment. In our setting, samples will be associated with 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 how confidence is formally defined.

2.2. Finite automata

We denote by C^* the set of all finite sequences with elements in C, $\bar{c}$ denotes a sequence with length greater than

0 while ϵ denotes the empty sequence. For all $\bar{a}, \bar{b} \in C^*$, $\overline{ab}$ denotes the concatenation of the sequences $\bar{a}$ and $\bar{b}$. For $R, W \subseteq C^*$, $R \cdot W$ is $\{\overline{ab} \mid \bar{a} \in R, \bar{b} \in W\}$. For a (partial function) $f : A \rightarrow B$, $dom(f)$ denotes the domain of f. Given $D \subseteq A$ we denote by $f|D$ the restriction of f to D.

Definition 2 A *finite automaton*, in short FA, is defined by a tuple $A = (S, s_0, I, F, \Gamma)$ in which S is a finite set of states, $s_0 \in S$ is the initial state, I is the finite input alphabet, $F : S \times I \longrightarrow \mathcal{P}(S)$ is the transition function, and $\Gamma \subseteq S$ is the set of final states. □

Intuitively, if A receives an input $i \in I$ when in state $s \in S$ it moves to some state in the set $F(s, i)$.

Definition 3 Let $A = (S, s_0, I, F, \Gamma)$ be a FA. The relation $F^* : S \times I^* \longrightarrow \mathcal{P}(S)$ is defined as follows:

$$\begin{array}{lcl} F^*(s, \epsilon) & = & \{s\} \\ F^*(s, \bar{i}i) & = & \{s' \mid \exists s'' \in F^*(s, \bar{i}) : s' \in F(s'', i)\} \end{array}$$

The *language accepted by* A, denoted by $L(A)$ is defined as $\{\bar{i} \in I^* \mid F^*(s_0, \bar{i}) \cap \Gamma \neq \emptyset\}$. In the same way, for a state $s \in S$ *the language accepted by A in s*, denoted by $L_A(s)$, is the set $\{\bar{i} \in I^* \mid F^*(s, \bar{i}) \cap \Gamma \neq \emptyset\}$.

An FA is *deterministic* if for all $s \in S$ and $i \in I$ we have $|F(s, i)| \leq 1$. A pair of states $s_i, s_j \in S$ are called *equivalent* if $L_A(s_i) = L_A(s_j)$; otherwise s_i and s_j are called *distinguishable*. A is *reduced* if for all $s_i, s_j \in S$ we have that $s_i \neq s_j$ implies s_i and s_j are distinguishable.

Given two FAs A and A' over the same input alphabet, we say that A and A' are *equivalent* if they accept the same language, that is, $L(A) = L(A')$. A deterministic FA A, is *minimal* if there does not exist any other equivalent deterministic FA with fewer states than A. □

Lemma 1 [32] For any FA there exists an equivalent minimal deterministic FA. □

In what follows we will refer to deterministic and minimal FAs where all states are terminal, that is, $\Gamma = S$, and we will denote them by a tuple $A = (S, s_0, I, F)$.

2.3. Finite automaton testing

A *test set* is a set of input sequences obtained from a specification such that can be applied to an implementation to establish whether these two machines are equivalent, that is, if their behaviors are the same. If they are not equivalent then there will exist an input sequence in the test set that will detect the difference. Several methods have been presented for generating test sets from an FA specification [10, 13, 30, 2]. In this paper we will apply the W-method that was initially introduced by [10] in the context of completely specified finite state machines and extended by [2] to partially specified finite state machines and finite automata.

Definition 4 Let $\mathcal{S}$ and $\mathcal{I}$ be two FAs over the same input alphabet I. A finite set $\mathcal{T} \subseteq I^*$ is a *test set* of $\mathcal{S}$ with respect to $\mathcal{I}$ if $L(\mathcal{S}) \cap \mathcal{T} = L(\mathcal{I}) \cap \mathcal{T}$ implies $L(\mathcal{S}) = L(\mathcal{I})$. □

We will define some basic concepts that are the basis of the W-method: *State cover, transition cover, and characterization set* of a minimal FA.

Definition 5 Let $A = (S, s_0, I, Tr)$ be a minimal FA. A set $R \subseteq I^*$ is a *state cover* of A if for all $s \in S$ there exists $\bar{i} \in R$ such that $Tr^*(s_0, \bar{i}) = s$. We say that $P \subseteq I^*$ is a *transition cover* of A if for all $s \in S$ there exists $\bar{i} \in P$ such that $Tr^*(s_0, \bar{i}) = s$ and for all $i \in I$ we have $\bar{i}i \in P$. Finally, $W \subseteq I^*$ is a *characterization set* of A if W distinguishes between any two distinct states of A. □

That is, a state cover is a set of input sequences that allows us to access from the initial state to the states of the machine. Let us note that if R is a state cover of A then $P = R \cup (R \cdot I)$ is a transition cover of A.

The following theorem is the theoretical basis of the W-method in the context of deterministic finite automata.

Theorem 1 [2] Let A and A' be two minimal deterministic FAs over the same input alphabet I. Let n be the number of states of A and m be the number of states of A', with $m \geq n$. Let P and W, respectively, be a transition cover and a characterization set of A and $Z = (I^{m-n} \cup \ldots \cup I \cup \{\epsilon\})(W \cup \{\epsilon\})$. Then, $\mathcal{T} = P \cdot Z$ is a test set of A with respect to A'. □

3. A stochastic extension of stream X-machines

In this section we introduce our stochastic extension of the classical stream X-machine model. The main difference with respect to usual stream X-machines consists in the addition of *time*. We use random variables to model *stochastic delays*. This extension will allow a specifier to explicitly denote temporal requirements for each action of a system.

Definition 6 A *stochastic stream X-machine*, in short SSXM, is a tuple $X = (I, O, S, Mem, \Phi, F, s_0, m_0)$ where I is the set of input actions, O is the set of output actions, S is a finite set of states, Mem is the memory, Φ, called the type of M, is a finite set of partial functions, $\phi : Mem \times I \longrightarrow O \times Mem \times \mathcal{V}$ represents timed processing functions, $F : S \times \Phi \longrightarrow S$ is the *next state function*, $s_0 \in S$ is the initial state, and $m_0 \in Mem$ is the initial memory value.

We can represent a SSXM by a finite automata where the elements of the type Φ are used as input alphabet, and the internal memory is not considered. The FA $A(X) = (S, s_0, \Phi, F)$ over the alphabet Φ is called *the associated* FA of X. □

Intuitively, we can think of a SSXM as a state transition diagram where arcs are labelled by processing functions. Each function receives an input and current memory values and produces an output while may modify the memory. Additionally, a random variable $\xi \in \mathcal{V}$ indicates that the time invested by the system for performing these actions will be equal to t time units with probability $F_\xi(t)$.

Example 1 Let us consider the machine depicted in Figure 1 in which the initial state is s_1. Each processing function has associated random variables depending on the input and memory value that they receive. Next we explain how the random variables are distributed. Let us consider that ξ_1 is *uniformly distributed* in the interval $[0, 2]$. Uniform distributions assign equal probability to all the time values in the interval. The random variable ξ_2 follows a Dirac distribution in 4. The idea is that the corresponding delay will be equal to 4 time units. Finally, ξ_3 is *exponentially* distributed with parameter 2. Let us consider that the initial memory value is $m_0 = 0$. Intuitively, if the machine is in state s_1 and it receives the input a then it will produce the output b and modify the value of the memory to 1 after a time given by ξ_1. For example, we know that this time will be less than 1 time unit with probability $\frac{1}{2}$, it will be less than 1.5 time units with probability $\frac{3}{4}$, and so on. Finally, once 2 time units have passed we know that the output b has been performed (that is, we have probability 1). □

Next we present some notions regarding stream X-machines that will be used throughout the paper and some results that will be adapted to be applied to our formalism.

Definition 7 A SSXM $X = (I, O, S, Mem, \Phi, F, s_0, m_0)$ is *deterministic* if for all $\phi, \phi' \in \Phi$ if there exists $s \in S$ such that $(s, \phi), (s, \phi') \in dom(F)$ then $\phi = \phi'$ or $dom(\phi) \cap dom(\phi') = \emptyset$. We say that X is *completely specified* if for all $s \in S, m \in Mem$, and $i \in I$, there exists $\phi \in \Phi$ such that $(m, i) \in dom(\phi)$ and $(s, \phi) \in dom(F)$. □

A SSXM is deterministic if given a state s, for any input value and any memory value there is only one processing function that can be applied. In turn, a SSXM is *completely specified* if for all $s \in S, m \in Mem$ and $i \in I$ there is always a possible transition. If a deterministic SSXM is completely specified then there is only one transition for any $s \in S, m \in Mem$, and $i \in I$.

Next we introduce a partial function that establishes the relation between a pair (memory values, input sequence) and a triplet (output sequence, updated memory values, random variable) produced by the application of a sequence of timed processing functions.

Definition 8 Given a sequence $\bar{\phi} \in \Phi^*$, we consider $\| \bar{\phi} \|$: $Mem \times I^* \rightarrow O^* \times Mem \times \mathcal{V}$ to be a partial function inductively defined in the following way:

$$\| \epsilon \| = \{((m, \epsilon), (\epsilon, m, \theta)) \mid m \in Mem\}$$

$$\| \bar{\phi}\phi \| = \left\{ ((m, \bar{i}i), (\bar{o}o, m', \xi)) \;\middle|\; \begin{array}{l} \exists\, m'' \in Mem : \\ ((m, \bar{i}), (\bar{o}, m'', \xi'')) \in \| \bar{\phi} \| \\ \wedge\, ((m'', i), (o, m', \xi')) \in \phi \\ \wedge\, \xi = \xi' + \xi'' \end{array} \right\}$$

□

In the previous definition we have used the notation $(a, b) \in f$ instead of the more standard $f(a) = b$.

Let us note that when we consider a deterministic SSXM each computation from the initial state to any other state is completely determined by the input sequence and the initial memory value.

A SSXM gives rise to a relation between the input sequences applied to the machine and the output sequences that it produces. This relation is given by the execution of a sequence of processing functions, from the initial state of the machine, that allows to obtain an output sequence in response to an input sequence. In our formalism, we will require, for dealing with the specified temporal restrictions, to extend this relation with the time that the machine needs for performing the processing functions. Thus, we introduce a new notion of correspondence between input sequences and pairs of output sequences and random variables.

Definition 9 Let $X = (I, O, S, Mem, \Phi, F, s_0, m_0)$ be a deterministic SSXM. The *timed function computed* by X $ft_X : I^* \rightarrow O^* \times \mathcal{V}$ is defined as follows:

$$ft_X = \left\{ (\bar{i}, \bar{o}, \xi) \;\middle|\; \begin{array}{l} \exists\, m \in Mem, \xi \in \mathcal{V}, \bar{\phi} \in \Phi^* : \\ (s_0, \bar{\phi}) \in dom(F^*) \wedge \\ ((m_0, \bar{i}), (\bar{o}, m, \xi)) \in \| \bar{\phi} \| \end{array} \right\}$$

The *function computed* by X $f_X : I^* \rightarrow O^*$ is defined as follows:

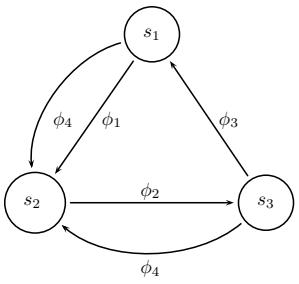
$$f_X = \{(\bar{i}, \bar{o}) | \exists\, \xi \in \mathcal{V} : (\bar{i}, \bar{o}, \xi) \in ft_X\}$$

□

Let us note that if the machine X is deterministic and completely specified then f_X and ft_X are total.

4. Conformance

In order to properly define how to test an implementation against a specification it is necessary to state what it means for an implementation to be correct. It is usual when testing from a stream X-machine to assume that the implementation under test (IUT) behaves like an unknown stream X-machine. For the sake of simplicity we will assume that



$I = \{a, b\}$
$O = \{a, b\}$
$Mem = \{1, 0\}$

$\phi_1(0, a) = (b, 1, \xi_1)$
$\phi_1(0, b) = (a, 1, \xi_2)$
$\phi_2(1, a) = (b, 0, \xi_2)$
$\phi_2(1, b) = (b, 0, \xi_2)$
$\phi_3(0, a) = (b, 0, \xi_1)$
$\phi_4(0, b) = (a, 1, \xi_3)$
$\phi_4(1, a) = (a, 1, \xi_3)$
$\phi_4(1, b) = (b, 1, \xi_3)$

$$F_{\xi_1}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ \frac{x}{2} & \text{if } 0 < x < 2 \\ 1 & \text{if } x \geq 2 \end{cases}$$

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

$$F_{\xi_3}(x) = \begin{cases} 1 - e^{-2 \cdot x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Figure 1. Example of Stochastic Stream X-machine.

both, implementation and specification, are given by deterministic SSXMs. This framework can be easily extended to cope with non-determinism by considering [16, 22]. First we introduce some auxiliary definitions.

Definition 10 Let $X = (I, O, S, Mem, \Phi, F, s_0, m_0)$ be a deterministic SSXM.

We say that $(s, s', (\phi_1, \ldots, \phi_r))$ is a *trace* of X if there exist $f_1, \ldots, f_r \in F$ such that $f_1(s, \phi_1) = s_1, \ldots, f_r(s_{r-1}, \phi_r) = s'$.

We say that the pair $((i_1/o_1, \ldots, i_r/o_r), \xi)$ is a *stochastic trace* of X if there exists a trace $(s_0, s', (\phi_1, \ldots, \phi_r))$ of X such that for all $1 \leq j \leq r$ we have $(m_{j-1}, i_j) \in dom(\phi_j)$, $\phi_j(m_{j-1}, i_j) = (o_j, m_j, \xi_j)$ and $\xi = \sum_{j=1}^{r} \xi_j$. In addition, we say that $(i_1/o_1, \ldots, i_r/o_r)$ is a *non-stochastic trace*. We denote by $\texttt{STraces}(X)$ and $\texttt{NSTraces}(X)$ the sets of stochastic and non-stochastic traces of X, respectively. □

The usual approach is that an implementation conforms to a specification if they describe the same behavior, that is, if the functions computed by them are equivalent. Additionally, we need to describe what means for a implementation to be temporally correct with respect to a specification. It would be reasonable to require that any sequence of the implementation have associated the same delay as the one specified by the specification. This reasoning leads us to define our first notion of conformance.

Definition 11 Let $\mathcal{S}$ and $\mathcal{I}$ be two deterministic SSXMs. We say that $\mathcal{I}$ *non-stochastically conforms* to $\mathcal{S}$, denoted by $\mathcal{I}\ \texttt{conf}_{ns}\ \mathcal{S}$, if $f_{\mathcal{I}} = f_{\mathcal{S}}$.

We say that $\mathcal{I}$ *stochastically conforms* to $\mathcal{S}$, denoted by $\mathcal{I}\ \texttt{conf}_{s}\ \mathcal{S}$, if $ft_{\mathcal{I}} = ft_{\mathcal{S}}$ □

In addition to requiring non-stochastic conformance we have to ask for some conditions on delays. Thus, $\mathcal{I}\ \texttt{conf}_{s}\ \mathcal{S}$ also requires that any stochastic trace of the specification that is performed by the implementation must have the same associated delay. Even though this is a very reasonable notion of conformance, the fact that we assume a black-box testing framework disallows us to check whether the corresponding random variables are identically distributed. In fact, we would need and 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, this notion is useful only from a theoretical point of view since, under our assumptions, it cannot be tested in finite time that an implementation conforms with respect to a specification. We have to give more *realistic* notion of temporal conformance based on a finite set of observations that is less *accurate* but that is *checkable*.

We introduce a new implementation relation that takes into account a *sample* of time values. The idea is to compare these time values with the random variable associated to it.

Definition 12 We say that a pair (ρ, t) is a *timed sequence* if ρ is a sequence of inputs/outputs and $t \in \mathbf{R}_+$.

Let $\Psi = \{\rho_1, \ldots, \rho_m\}$ be a set of input/output sequences and $H = \{\!\{(\rho'_1, t_1), \ldots, (\rho'_n, t_n)\}\!\}$ be a multiset of timed sequences. We say that $\mathit{Sampling}_{(H,\Psi)} : \Psi \longrightarrow \wp(\mathbf{R}^+)$ is a *sampling application* of H for Ψ if for all $\rho \in \Psi$ we have $\mathit{Sampling}_{(H,\Psi)}(\rho) = \{\!\{t \mid (\rho, t) \in H\}\!\}$.

Let $\mathcal{I}$ and $\mathcal{S}$ be two deterministic SSXMs. Let $H = \{\!\{(\rho_1, t_1), \ldots, (\rho_p, t_p)\}\!\}$ be a multiset of timed sequences with $\rho_j \in \texttt{NSTraces}(\mathcal{I})$ for all $1 \leq j \leq p$. Let $0 \leq \alpha \leq 1$, $\Psi = \{\rho \mid \exists\ t : (\rho, t) \in H\} \cap \texttt{NSTraces}(\mathcal{S})$, and let us consider $\mathit{Sampling}_{(H,\Psi)}$. We say that $\mathcal{I}$ $(\alpha, H)-$*stochastically conforms* to $\mathcal{S}$, denoted by $\mathcal{I}\texttt{conf}_{s}^{(\alpha,H)}\mathcal{S}$, if $\mathcal{I}\texttt{conf}_{ns}\mathcal{S}$ and for all $\rho \in \Psi$ we have that $(\rho, \xi) \in \texttt{STraces}(\mathcal{S})$ then $\gamma(\xi, \mathit{Sampling}_{(H,\Psi)}(\rho)) > \alpha$. □

The idea underlying the new relation is that the implementation must conform to the specification in the usual

way, that is, I $\texttt{conf}_{ns}$ S. Besides, for any trace of the implementation performed by the specification, the time values associated to it in the sample *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.

5. Definition and application of tests

A *test* consists of a sequence of inputs that will be applied to an IUT, a sequence of expected outputs and a random variable that will be *compared* with the different time values that the implementation spends for giving the response. The ultimate goal of testing is to determine whether the IUT is a correct implementation of the specification. On the one hand, we must prove that the functions computed by both SSXMs are identical, that is, we need to determine the *functional correctness* of the IUT with respect to the specification. On the other hand, we have to check the *non-functional correctness* of the system, that is, the IUT fulfills the temporal requirements established in the specification. We will check that the time values the implementation takes for performing each test *match* the random variable associated with the test. We will collect a sample of time values (one for each test execution) and we will *compare* this sample with the random variable. 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 13 A *test* is a triplet $T = (\bar{i}, \bar{o}, \xi)$ where $\bar{i}$ is a sequence of inputs, $\bar{o}$ is a sequence of outputs, having $\bar{i}$ and $\bar{o}$ the same length, and ξ is a random variable.

Let $T = (i_1, \ldots, i_r, o_1, \ldots, o_r, \xi)$ be a test and $\rho = (i_1/o_1, \ldots, i_r/o_r)$ a sequence of iputs/outputs. We write $T \overset{\rho}{\longmapsto} \xi$.

We say that $\mathcal{I}$ *passes* a set of tests $\mathcal{T}$, denoted by $\texttt{pass}(\mathcal{I}, \mathcal{T})$, if for all test $T = (\bar{i}, \bar{o}, \xi) \in \mathcal{T}$ we have $f_{\mathcal{I}}(\bar{i}) = \bar{o}$.

We say that $((i_1/o_1, \ldots, i_n/o_n), t)$ is a *timed execution* of $\mathcal{I}$ if the observation of $\mathcal{I}$ shows that the sequence $(i_1/o_1, \ldots, i_n/o_n)$ is performed in time t.

We denote the application of the test T to the implementation $\mathcal{I}$ as $\mathcal{I} \parallel T$. We say that (ρ, t) is a *test execution* of $\mathcal{I}$ and T if $T \overset{\rho}{\longmapsto} \xi$ for some $\xi \in \mathcal{V}$ and (ρ, t) is a timed execution of $\mathcal{I}$. □

Definition 14 Let $\mathcal{I}$ be a SSXM and $\mathcal{T} = \{T_1, \ldots, T_n\}$ be a set of tests. Let $H_1, \ldots, H_n$ be test execution samples of $\mathcal{I}$ and $T_1, \ldots, T_n$, respectively, and $H = \bigcup_{i=1}^{n} H_i$. Let $\Psi = \{\rho \mid \exists\, t : (\rho, t) \in H\}$ and let us consider $\mathit{Sampling}_{(H,\Psi)}$. Finally, let $0 \leq \alpha \leq 1$.

We say that the implementation $\mathcal{I}$ $(\alpha, H)-$*passes* the set of tests $\mathcal{T}$ if $\texttt{pass}(\mathcal{I}, \mathcal{T})$ and for all $\rho \in \Psi$ and all $T \in \mathcal{T}$ such that $T \overset{\rho}{\longmapsto} \xi$ we have $\gamma(\xi, \mathit{Sampling}_{(H,\Psi)}(\rho)) > \alpha$. □

Intuitively, an implementation passes a set of tests if two conditions hold. First, the output obtained by means of the interaction with the implementation are those expected. Once we know that the functional behavior of the implementation with respect to the set of tests is correct, we need to check time conditions. The set H corresponds to the observations of the (several) applications of the tests belonging to $\mathcal{T}$ to I. Thus, we have to decide whether, the observed time values (that is, $\mathit{Sampling}_{(H,\Phi)}(\rho)$) *match* the definition of the random variable appearing in the test corresponding to the execution of that sequence (that is, ξ). As we commented previously, we assume a function γ that can perform this hypothesis contrast. In the appendix of this paper we give the technical details about the definition of this function (we will use Pearson's χ^2 hypothesis contrast).

Next, we will adapt the testing method for deterministic stream X-machines, developed by [21], that provides us with a *test set* derived from the specification and allows to determine the functional correctness of the implementation. This method assumes the implementation has the same type as the specification and that the processing functions are correctly implemented, that is, they have been tested in advanced and are equivalent to the ones in the specification. Testing is simplified then to determine whether these functions have been combined correctly. This method is proved to detect all the faults of the implementations considering the system must satisfy some requirements, called *design for test*. In fact, the method reduces the testing of SSXMs to testing that their associated automata are equivalent. After we obtain a test set for the associated automata, we will translate, by means of the *fundamental timed test function*, the processing functions sequences belonging to the test set into triplets (input sequence, output sequence, random variable) that will be used to test the SSXM. Let us note that in the standard approach this function only translates the test set for the associated automata to a set of input sequences. When we deal with SSXMs we also need to check the correctness of the temporal requirements established for each trace. Thus, the new fundamental timed test function must provide us with the associated random variables in the specification that allow us to compare them with the time values obtained from the interaction with the implementation.

Definition 15 Let X and X' be two deterministic SSXMs with the same input alphabet I and output alphabet O. Then, a finite set $R \subseteq I^*$ is a *test set* of X with respect to X' if $f_X|R = f_{X'}|R$ implies $f_X = f_{X'}$.

We say that $R_s \subseteq I^* \times O^* \times \mathcal{V}$ is a *timed test set* of X with respect to X' if $\{\bar{i} \mid \exists\, \bar{o}, \xi : (\bar{i}, \bar{o}, \xi) \in R_s\}$ is a test set of X with respect to X'.

We say that X and X' are *testing compatible* if they have the same input actions, output actions, memory, initial memory values, and identical types. □

The processing functions have to fulfill some requirements, called *design for test conditions*, that allow us to translate the test set obtained from the associated FA into a timed test set of the SSXM. These conditions are output-distinguishability and input-completeness.

Definition 16 A type Φ is called *output distinguishable* if for all $\phi_1, \phi_2 \in \Phi$ we have that there exists $m \in Mem$ and $i \in I$ such that $\phi_1(m,i) = (o_1, m_1, \xi_1)$ and $\phi_2(m,i) = (o_2, m_2, \xi_2)$, implies that either $\phi_1 = \phi_2$ or $o_1 \neq o_2$.

A type Φ is called *input completed* if for all $m \in Mem$ and all $\phi \in \Phi$ there exists $i \in I$ such that $(m,i) \in dom(\phi)$. □

The first condition means that two different functions of Φ cannot generate the same output for a given input i and memory value m. We will be able to distinguish between any two different processing functions observing outputs. The second property ensures that any processing function can be exercised from any memory value using the suitable input.

We need a method that translates sequences of processing functions, corresponding to the test set derived from the FA, to sequences that can be applied to our systems. Next, we introduce this function.

Definition 17 Let $X = (I, O, S, Mem, \Phi, F, s_0, m_0)$ be a deterministic SSXM with type Φ input-complete. A function $\Omega : \Phi^* \rightarrow I^* \times O^* \times \mathcal{V}$ is called a *fundamental timed test function* if its definitions fulfills:

$$\Omega(\epsilon) = (\epsilon, \epsilon, \theta)$$

$$\Omega(\bar{\phi}\phi) = \begin{cases} (\bar{i}i, \bar{o}o, \xi) & \text{if } \bar{\phi} \in L(A(X)) \wedge \\ & \Omega(\bar{\phi}) = (\bar{i}, \bar{o}, \xi') \wedge \\ & ((m,\bar{i}),(\bar{o},m',\xi')) \in \| \bar{\phi} \| \wedge \\ & (m',i) \in dom(\phi) \wedge \\ & ((m',i),(o,m'',\xi'')) \in \| \phi \| \wedge \\ & \xi = \xi' + \xi'' \\ \Omega(\bar{\phi}) & \text{if } \bar{\phi} \notin L(A(X)) \end{cases}$$

□

Let us remark that Ω is input-complete then, there always exist $\bar{i}$.

The next theorem is the basis for deterministic SSXM testing. It is a simple adaptation of a similar result appearing in [21].

Theorem 2 Let X be a deterministic SSXM with type Φ input-complete and output-distinguishable and X' a deterministic SSXM testing compatible with X. If Ω is a fundamental timed test function of X and $Z \subseteq \Phi^*$ is a test set of $A(X)$ with respect to $A(X')$ then $T_s = \Omega(Z)$ is a timed test set of X with respect to X'. □

Now we can use Theorems 1 and Theorem 2 to generate a test set of a specification with respect to an implementation.

Theorem 3 Let $\mathcal{S}$ be a deterministic SSXM and $\mathcal{I}$ be a deterministic SSXM testing compatible with $\mathcal{S}$, with type input-complete and output-distinguishable. Let n be the number of states of $\mathcal{S}$ and m be the number of states of $\mathcal{I}$. Let $A_{\mathcal{S}}$ and $A_{\mathcal{I}}$ be their associated automata, respectively, and Ω be a fundamental timed test function of $\mathcal{S}$. Let P and W, respectively, be a transition cover and a characterisation set of $\mathcal{S}$ and $Z = (I^{m-n} \cup \ldots \cup I \cup \{\epsilon\})(W \cup \{\epsilon\})$. Then, $T_s = \Omega(P \cdot Z)$ is a timed test set of $\mathcal{S}$ with respect to $\mathcal{I}$. □

In the following, for a given specification $\mathcal{S}$, any timed test set obtained by applying the method described in the previous result will be denoted by $\texttt{tests}(\mathcal{S})$.

Theorem 4 Let $\mathcal{S}$ be a deterministic SSXMs and $\mathcal{I}$ a deterministic SSXM testing compatible with $\mathcal{S}$, with type input-complete and output-distinguishable. Let $0 \leq \alpha \leq 1$ and H be a multiset of test executions of $\mathcal{I}$ and the tests belonging to $\texttt{tests}(\mathcal{S})$. We have that:

$$\mathcal{I}\ (\alpha, H)\text{−}\textit{stochastically conforms}\ \text{to}\ \mathcal{S}$$
$$\Updownarrow$$
$$\mathcal{I}\ (\alpha, H)\text{−}\textit{passes}\ \texttt{tests}(\mathcal{S})$$

Proof: First, let us show that $\mathcal{I}\ (\alpha, H)$−*passes* $\texttt{tests}(\mathcal{S})$ implies $\mathcal{I}\ (\alpha, H)$−*stochastically conforms* to $\mathcal{S}$. We will use the contrapositive, that is, we will suppose that $\mathcal{I}$ (α, H)−*stochastically conforms* to $\mathcal{S}$ does not hold and we will prove that $\mathcal{I}$ does not (α, H)−*passes* $\texttt{tests}(\mathcal{S})$. If $\mathcal{I}$ (α, H)−*stochastically conforms* to $\mathcal{S}$ does not hold then we have two possibilities:

1. Either $\mathcal{I}\ \texttt{conf}_{ns}\ \mathcal{S}$ does not hold, or
2. there exists $\rho \in \Psi = \{\rho : (\rho, t) \in H\} \cap \texttt{NSTraces}(\mathcal{S})$ such that $(\rho, \xi) \in \texttt{STraces}(\mathcal{S})$ and $\gamma(\xi, \textit{Sampling}_{(H,\Psi)}(\rho)) > \alpha$ does not hold.

Let us consider the first case, that is, we suppose that $I\ \texttt{conf}_{ns}\ S$ does not hold. Then, we automatically infer $f_{\mathcal{I}} \neq f_{\mathcal{S}}$. By the previous theorem we have that $\texttt{tests}(\mathcal{S})$ is a timed test set of $\mathcal{I}$ with respect to $\mathcal{S}$. By definition of timed test set, there exists $(\bar{i}, \bar{o}, \xi) \in \texttt{tests}(\mathcal{S})$ such that $f_{\mathcal{I}}(\bar{i}) \neq f_{\mathcal{S}}(\bar{i})$. Thus, we conclude $\mathcal{I}$ does not (α, H)−*passes* $\texttt{tests}(\mathcal{S})$ since $f_{\mathcal{I}}(\bar{i}) \neq \bar{o}$.

Now, let us suppose that $\mathcal{I}\ (\alpha, H)$−*stochastically conforms* to $\mathcal{S}$ does not hold due to the second case. If $\rho \in \Psi$ then there exists $(\rho, t) \in H$ for some $t \in$

$\mathbf{R}_+$. Since H is a test execution sample of $\mathcal{I}$ and the tests belonging to $\texttt{tests}(\mathcal{S})$, there exists $T = (\rho, \xi) \in \texttt{tests}(\mathcal{S})$ such that $T \stackrel{\rho}{\longmapsto} \xi$. Since, we assumed that $\gamma(\xi, \textit{Sampling}_{(H,\Psi)}(\rho)) > \alpha$ does not hold, we conclude that $\mathcal{I}$ does not $(\alpha, H)-$*passes* $\texttt{tests}(\mathcal{S})$.

Let us prove now that $\mathcal{I}$ $(\alpha, H)-$*stochastically conforms* to $\mathcal{S}$ implies $\mathcal{I}$ $(\alpha, H)-$*passes* $\texttt{tests}(\mathcal{S})$. We will use again the contrapositive, that is, we will assume that $\mathcal{I}$ does not $(\alpha, H)-$*passes* $\texttt{tests}(\mathcal{S})$ and we will conclude that $\mathcal{I}$ does not $(\alpha, H)-$*stochastically conforms* to $\mathcal{S}$. If $\mathcal{I}$ does not $(\alpha, H)-$*passes* $\texttt{tests}(\mathcal{S})$ then we have two possibilities:

1. Either $\texttt{pass}(\mathcal{I}, \texttt{tests}(\mathcal{S}))$ does not hold, or
2. there exists $\rho \in \Psi$ and $T \in \texttt{tests}(\mathcal{S})$ such that $T \stackrel{\rho}{\longmapsto} \xi$ and $\gamma(\xi, \textit{Sampling}_{(H,\Psi)}(\rho)) > \alpha$ does not hold.

Let us consider the first case, that is, we suppose that $\texttt{pass}(\mathcal{I}, \texttt{tests}(\mathcal{S}))$ does not hold. Then, there exists $(\bar{i}, \bar{o}, \xi) \in \texttt{tests}(\mathcal{S})$ such that $f_{\mathcal{I}}(\bar{i}) \neq \bar{o}$. Thus, $f|\mathcal{I} \neq f|\mathcal{S}$ and $\mathcal{I}$ $\texttt{conf}_{ns}$ $\mathcal{S}$ does not hold. We conclude $\mathcal{I}$ does not $(\alpha, H)-$*stochastically conforms* to $\mathcal{S}$.

Now, we consider the second possibility. If $\rho \in \Psi$ then $\rho \in \texttt{NSTraces}(\mathcal{S})$. If there exists $T \in \texttt{tests}(\mathcal{S})$ such that $T \stackrel{\rho}{\longmapsto} \xi$ then there exists $(\rho, \xi) \in \texttt{STraces}(\mathcal{S})$. Besides, we have that $\gamma(\xi, \textit{Sampling}_{(H,\Psi)}(\rho)) > \alpha$ does not hold. So, $\mathcal{I}$ does not $(\alpha, H)-$*stochastically conforms* to $\mathcal{S}$. □

6. Conclusions

We have presented a methodology for testing temporal aspects of systems where temporal behavior is critical. The model introduced for specifying the systems is a suitable extension of the classical concept of stream X-machine, where temporal requirements have been represented by means of random variables. Implementation relations have been introduced for describing the notion of conformance of an implementation with respect to a specification up to a certain degree of confidence.

The task of testing an implementation for conformance to a specification, both modeled by means of deterministic stochastic stream X-machines, has been tackled by adapting the standard technique for stream X-machines based in the W-method.

7 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 $X^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$. 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 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 $\texttt{max}\{\alpha \mid X^2 < \chi^2_\alpha(k - r - 1)\}$.

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 18 Let ξ be a random variable and 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 k classes

$$C = \{[0, a_1), [a_1, a_2), \ldots, [a_{k-2}, a_{k-1}), [a_{k-1}, \infty)\}$$

where k is a given constant and for all $1 \leq i \leq k-1$ we have $P(\xi \leq a_i) = \frac{i}{k}$. We define the confidence of ξ on J with

classes C, denoted by $\gamma(\xi, J)$, as $\texttt{max}\{\alpha | X^2 < \chi^2_\alpha(k-1)\}$. □

The previous definition indicates that in order to perform a contrast hypothesis, we split the collected values in several intervals having the same expected probability. We compute the value for X^2 as previously described and check this figure with the tabulated tables (see, for example, `www.statsoft.com/textbook/sttable.html`) corresponding to χ^2 with $k-1$ freedom degrees.

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.

Example 2 Let us consider a device that produces real numbers belonging to the interval $[0, 1]$. We would like to test whether the device produces these numbers randomly, that is, it does not have a number or sets of numbers that are more probable to be produced than other ones. Thus, we obtain a sample from the machine and we apply the contrast hypothesis to determine whether the machine follows a uniform distribution in the interval $[0, 1]$. First, we have to decide how many classes we will use. Let us suppose that we take $k = 10$ classes. Thus, for all $1 \leq i \leq 9$ we have $a_i = 0.i$ and $P(\xi \leq a_i) = \frac{i}{10}$. So, $C = \{[0, 0.1), [0.1, 0.2) \ldots [0.8, 0.9), [0.9, \infty)\}$.

Let us suppose that the multiset of observed values, after we sort them, is:

$$J = \left\{ \begin{array}{l} 0.00001, 0.002, 0.0876, 0.8, \\ 0.1, 0.11, 0.123, \\ 0.21, 0.22, 0.22, 0.2228, 0.23, 0.24, 0.28, \\ 0.32, 0.388, 0.389, 0.391 \\ 0.4, 0.41, 0.42, 0.4333 \\ 0.543, 0.55, 0.57, \\ 0.62, 0.643, 0.65, 0.67, 0.68, 0.689, 0.694 \\ 0.71, 0.711, 0.743, 0.756, 0.78, 0.788, \\ 0.81, 0.811, 0.82, 0.845, 0.8999992, \\ 0.91, 0.93, 0.94, 0.945, 0.9998 \end{array} \right\}$$

Since the sample has 48 elements we have that the expected frequency in each class, E_i, is equal to 4.8. In contrast, the observed frequencies, O_i, are $4, 3, 7, 4, 4, 3, 7, 6, 5, 5$. Next, we have to compute

$$X^2 = \sum_{i=1}^{10} \frac{(O_i - E_i)^2}{E_i} = 4.08333$$

Finally, we have to consider the table corresponding to χ^2 with 9 freedom degrees and find the maximum α such that $4.08333 < \chi^2_\alpha(9)$. We conclude that, with probability α, the machine produces indeed random values. □

References

[1] M. Ajmone Marsan, A. Bianco, L. Ciminiera, R. Sisto, and A. Valenzano. A LOTOS extension for the performance analysis of distributed systems. *IEEE/ACM Transactions on Networking*, 2(2):151–165, 1994.

[2] T. Balanescu, M. Gheorghe, F. Ipate, and M. Holcombe. Formal black box testing for partially specified deterministic finite state machines. *Foundations of Computer Decision Systems*, 28(1):17–28, 2003.

[3] J. Barnard. COMX: A design methodology using communicating X–machines. *Information and Software Technology*, 40(5–6):271–280, 1998.

[4] M. Bernardo and W. Cleaveland. A theory of testing for markovian processes. In *11th Int. Conf. on Concurrency Theory, CONCUR'2000, LNCS 1877*, pages 305–319. Springer, 2000.

[5] M. Bernardo and R. Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202(1-2):1–54, 1998.

[6] K. Bogdanov, M. Holcombe, F. Ipate, L. Seed, and S. Vanak. Testing methods for X-machines: a review. *Formal Aspects of Computing*, 18:3–30, 2006.

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

[8] L. Brandán Briones and E. Brinksma. Testing real-time multi input-output systems. In *7th Int. Conf. on Formal Engineering Methods, ICFEM'05, LNCS 3785*, pages 264–279. Springer, 2005.

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

[10] T. Chow. Testing software design modelled by finite state machines. *IEEE Transactions on Software Engineering*, 4:178–187, 1978.

[11] D. Clarke and I. Lee. Automatic generation of tests for timing constraints from requirements. In *3rd Workshop on Object-Oriented Real-Time Dependable Systems, WORDS'97*, pages 199–206. IEEE Computer Society Press, 1997.

[12] A. En-Nouaary, R. Dssouli, and F. Khendek. Timed Wp-method: Testing real time systems. *IEEE Transactions on Software Engineering*, 28(11):1024–1039, 2002.

[13] S. Fujiwara, G. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test selection based on finite-state models. *IEEE Transactions on Software Engineering*, 17(6):591–603, 1991.

[14] N. Götz, U. Herzog, and M. Rettelbach. Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras. In *16th Int. Symp. on Computer Performance Modelling, Measurement and Evaluation, PERFORMANCE'93, LNCS 729*, pages 121–146. Springer, 1993.

[15] H. Hermanns, U. Herzog, and J.-P. Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 274(1-2):43–87, 2002.

[16] R. Hierons and M. Harman. Testing conformance of a deterministic implementation to a non-deterministic stream X-machine. *Theoretical Computer Science*, 323(1–3):191–233, 2004.

[17] T. Higashino, A. Nakata, K. Taniguchi, and A. Cavalli. Generating test cases for a timed I/O automaton model. In *12th Int. Workshop on Testing of Communicating Systems, IWTCS'99*, pages 197–214. Kluwer Academic Publishers, 1999.

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

[19] M. Holcombe. X–machines as a basis for dynamic system specification. *Software Engineering Journal*, 3(2):69–76, 1988.

[20] M. Holcombe and F. Ipate. *Correct Systems: Building a Business Process Solution*. Springer, 1998.

[21] F. Ipate and M. Holcombe. An integration testing method that is proved to find all faults. *International Journal of Computer Mathematics*, 63(3-4):159–178, 1997.

[22] F. Ipate and M. Holcombe. Generating test sets from non-deterministic stream X-machines. *Formal Aspects of Computing*, 12(6):443–458, 2000.

[23] E. Kehris, G. Eleftherakis, and P. Kefalas. Using X–machines to model and test discrete event simulation programs. In N. Mastorakis, editor, *Systems and Control: Theory and Applications*, pages 163–171. World Scientific and Engineering Society Press, 2000.

[24] M. Krichen and S. Tripakis. An expressive and implementable formal framework for testing real-time systems. In *17th Int. Conf. on Testing of Communicating Systems, TestCom'05, LNCS 3502*, pages 209–225. Springer, 2005.

[25] R. Lai. A survey of communication protocol testing. *Journal of Systems and Software*, 62:21–46, 2002.

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

[27] N. López and M. Núñez. A testing theory for generally distributed stochastic processes. In *12th Int. Conf. on Concurrency Theory, CONCUR'01, LNCS 2154*, pages 321–335. Springer, 2001.

[28] N. López and M. Núñez. Weak stochastic bisimulation for non-markovian processes. In *2nd ICTAC, LNCS 3722*, pages 454–468. Springer, 2005.

[29] N. López, M. Núñez, and F. Rubio. An integrated framework for the analysis of asynchronous communicating stochastic processes. *Formal Aspects of Computing*, 16(3):238–262, 2004.

[30] G. Luo, A. Petrenko, and G. v. Bochmann. Selecting test sequences for partially-specified nondeterministic finite state machines. In *7th IFIP Workshop on Protocol Test Systems, IWPTS'94*, pages 95–110. Chapman & Hall, 1994.

[31] D. Mandrioli, S. Morasca, and A. Morzenti. Generating test cases for real time systems from logic specifications. *ACM Transactions on Computer Systems*, 13(4):356–398, 1995.

[32] E. Moore. Gedanken-experiments. In C. Shannon and J. McCarthy, editors, *Automata Studies*. Princeton University Press, 1956.

[33] M. Núñez and I. Rodríguez. Towards testing stochastic timed systems. In *23rd IFIP WG 6.1 Int. Conf. on Formal Methods for Networked and Distributed Systems, FORTE'03, LNCS 2767*, pages 335–350. Springer, 2003.

[34] M. Núñez and I. Rodríguez. Conformance testing relations for timed systems. In *5th Int. Workshop on Formal Approaches to Software Testing, FATES'05, LNCS 3997*, pages 103–117. Springer, 2006.

[35] I. Rodríguez, M. Merayo, and M. Núñez. $\mathcal{HOTL}$: Hypotheses and observations testing logic. *Journal of Logic and Algebraic Programming (in press)*, 2007.

[36] J. Springintveld, F. Vaandrager, and P. D'Argenio. Testing timed automata. *Theoretical Computer Science*, 254(1-2):225–257, 2001. Previously appeared as Technical Report CTIT-97-17, University of Twente, 1997.