# Towards Testing Stochastic Timed Systems[*]

Manuel Núñez and Ismael Rodríguez

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

**Abstract.** In this paper we present a first approach to the definition of conformance testing relations for systems presenting *stochastic timed* behavior. By stochastic time we mean that the probability of performing an event may vary according to the elapsed time. In particular, we will consider delays specified by means of random variables.

In order to define our formal model, we will provide a stochastic extension of the notion of finite state machine. We will give a first implementation relation and we will discuss its practical drawbacks. That is, we will show that this relation cannot be appropriately checked under a black/grey-box testing methodology. We will also present other alternative implementation relations that can be checked up to a certain degree of confidence. We will define test cases and how they are applied to implementations. Finally, we will give a test generation algorithm providing *complete*, up to a degree of confidence, test suites.

**Keywords:** Conformance testing, test theory, performance testing

## 1 Introduction

Testing consists in checking the correctness of an implementation by performing experiments on it. Usually, correctness is considered with respect to a specification describing the *desired* behavior of the system. In order to perform this task, several techniques, algorithms, and semantic frameworks have been introduced in the literature (see e.g. [6, 17, 18] for overviews on the topic). In recent years 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 has been several proposals for timed testing (e.g. [8, 10, 15, 21, 23]). In these works time is considered to be *deterministic*. In fact, in most of the cases, time is introduced by means of clocks following [3].

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 this notion of time. 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

[*] Research supported in part by the Spanish MCYT projects *AMEVA* and *MASTER* and the Junta de Castilla-La Mancha project *DISMEF*.

H. König, M. Heiner, and A. Wolisz (Eds.): FORTE 2003, LNCS 2767, pp. 335–350, 2003.

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, as process algebras and Petri Nets, have appeared in the literature (see e.g. [1, 2, 5, 7, 11, 12, 14, 16, 20]). 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 of timed systems, stochastic testing has received almost no attention. In fact, to the best of our knowledge, there are only two works in the field [4, 19], where extensions of the classical theory of testing [9, 13] are given. We think that this lack of research in this area 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, 19], we still have technical problems as the aggregation of delays. In the case of grey/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 times of execution. 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).

In this paper we give notions of conformance testing for systems presenting stochastic time information. In order to do so, we need to introduce a formal framework to specify this kind of systems. 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 (see e.g. [22]), 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 *delay* between the input is applied and the output is received is given by a random variable $\xi$. 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 rest of the paper is organized as follows. In Section 2 we present our stochastic finite state machine model. In Section 3 we define our first implementation relation and we show its weakness from the practical point of view. In Section 4 we give our notion of test case and we define how they are applied to implementations under test. In Section 5 we take up again the study of implementation relations. We give a first notion where an implementation conforms a specification up to a certain degree of confidence, with respect to a (multi)set of observations from the implementation. The idea, regarding time, is that we try to decide whether the times observed from the implementation correspond to the random variable indicated by the specification. Next we give other alternative notions where the *speed* of the implementation is taken into account. In Section 6 we give a test derivation algorithm and we show that, given a specification, the generated test suite is *complete*, for a certain confidence level, with respect to the first implementation relation given in Section 5. In Section 7 we present our conclusions and some lines for future work. In the appendix of this paper we formally introduce Pearson's contrast, which is (abstractly) used along the paper.

## 2 Stochastic Finite State Machines

In this section we introduce our notion of finite state machines with stochastic time. We use random variables to model *stochastic delays.* 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 the set of real numbers $\mathbb{R}$. Besides, random variables take positive values only in $\mathbb{R}^+$. 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 $\xi$ be a random variable. We define its *probability distribution function* as the function $F_\xi : \mathbb{R} \longrightarrow [0,1]$ such that $F_\xi(x) = P(\xi \leq x)$, where $P(\xi \leq x)$ is the probability that $\xi$ assumes values less than or equal to $x$. Let $\xi, \xi' \in \mathcal{V}$ be random variables. We write $\xi = \xi'$ if for any $x \in \mathbb{R}$ we have $F_\xi(x) = F_{\xi'}(x)$.

Given two random variables $\xi$ and $\psi$ we consider that $\xi + \psi$ denotes a random variable distributed as the addition of the two random variables $\xi$ and $\psi$.

We will call *sample* to any multiset of positive real numbers. We denote the set of multisets in $\mathbb{R}^+$ by $\wp(\mathbb{R}^+)$. Let $\xi$ be a random variable and $J$ be a sample. We denote by $\gamma(\xi, J)$ the *confidence* of $\xi$ on $J$. □

In the previous definition, a sample simply denotes an observation of values. In our setting, samples will be associated with times 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 $\xi$. That is, this function decides how *similar* the probability distribution function generated by $J$ and the one corresponding to the random variable $\xi$ are. In the appendix of this paper we show how confidence is formally defined. Next we introduce our notion of stochastic finite state machines.

**Definition 2.** A *Stochastic Finite State Machine*, in short SFSM, is a tuple $M = (S, I, O, \delta, s_{in})$ where $S$ is the set of states, $I$ and $O$ denote the sets of input and output actions, respectively, $\delta$ is the set of transitions, and $s_{in}$ is the initial state. Each transition belonging to $\delta$ is a tuple $(s, i, o, \xi, s')$ where $s, s' \in S$ are the initial and final states, $i \in I$ and $o \in O$ are the input and output ac-

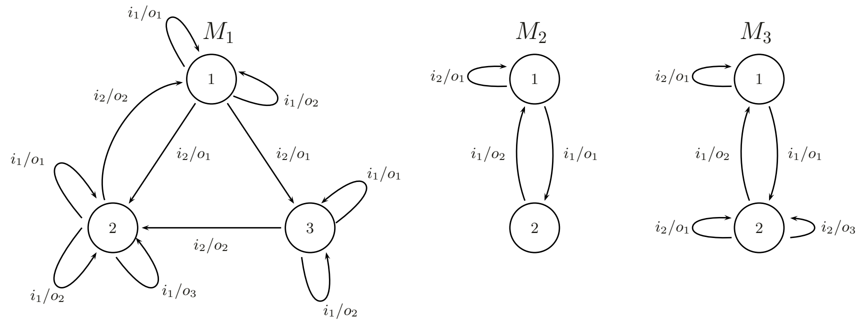

**Fig. 1.** Examples of SFSM.

tions, and $\xi \in \mathcal{V}$ is the random variable defining the delay associated with the transition. Transitions will be sometimes denoted as $s \xrightarrow{i/o}_{\xi} s'$. □

Intuitively, a transition $(s, i, o, \xi, s')$ 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 $s'$. 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. Next we define some conditions that are usually required to finite state machines.

**Definition 3.** Let $M = (S, I, O, \delta, s_{in})$ be a SFSM. We say that $M$ is *input-enabled* if for any state $s \in S$ and input $i \in I$ there exist $s'$, $o$, $\xi$, such that $(s, i, o, \xi, s') \in \delta$. We say that $M$ is *deterministically observable* if for any $s, i, o$ there do not exist two different transitions $(s, i, o, \xi_1, s_1), (s, i, o, \xi_2, s_2) \in \delta$. □

First, let us remark that the previous concepts are independent of the stochastic information appearing in SFSMs. Regarding the notion of deterministically observable SFSM, 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 same input action, as far as the outputs are different.

*Example 1.* Let us consider the finite state machines depicted in Figure 1. In order to *transform* these machines into stochastic finite state machines, we need to add random variables to all the transitions. Let us consider $M_2$. We have $M_2 = (\{1, 2\}, \{i_1, i_2\}, \{o_1, o_2, o_3\}, \delta, 1)$ where the set of transitions $\delta$ 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 $M_2$ 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 $\xi_1$ 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 $\xi_2$ 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 $\xi_3$ is exponentially distributed with parameter 3. Let us consider the transition $(1, i_2, o_1, \xi_1, 1)$. Intuitively, if $M_2$ is in state 1 and it receives the input $i_2$ then it will produce the output $o_2$ after a delay given by $\xi_1$. 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 notions of input-enabled and deterministically observable, we have that $M_1$ fulfills the first of the properties but not the second one (there are two transitions from the state 1 labelled by $i_2/o_1$). The first property does not hold in $M_2$ (there is no outgoing transition labelled by $i_2$ from the state 2) while the second one does. Finally, both properties hold for $M_3$. □

Usually we need to consider not only single evolutions of a SFSM but also sequences of transitions. Thus, we introduce the notion of trace.

**Definition 4.** Let $M = (S, I, O, \delta, s_{in})$ be a SFSM. We write the *generalized* transition $s \xRightarrow{(i_1/o_1,\ldots,i_n/o_n)}_{\xi} s'$ if there exist $s_1, \ldots, s_{n-1}, \xi_1, \ldots, \xi_n$ such that $s \xrightarrow{i_1/o_1}_{\xi_1} s_1 \xrightarrow{i_2/o_2}_{\xi_2} s_2 \cdots s_{n-1} \xrightarrow{i_n/o_n}_{\xi_n} s'$ and $\xi = \xi_1 + \ldots + \xi_n$.

We say that $\rho = (i_1/o_1, \ldots, i_n/o_n)$ is a *non-stochastic trace*, or simply a *trace*, of $M$ if there exist $s' \in S$ and $\xi \in \mathcal{V}$ such that $s_{in} \xRightarrow{\rho}_{\xi} s'$. We say that $\rho_s = ((i_1/o_1, \ldots, i_n/o_n), \xi)$ is a *stochastic trace* of $M$ if there exist $s' \in S$ such that $s_{in} \xRightarrow{\rho}_{\xi} s'$.

We denote by $\texttt{NSTraces}(M)$ and $\texttt{STraces}(M)$ the sets of non-stochastic and stochastic traces of $M$, respectively. □

Traces are defined as sequences of transitions. In this case, the random variable associated with stochastic traces is computed from the corresponding to each transition of the sequence. In fact, we consider the addition of all the random variables appearing in the trace. Intuitively, the time that it will take for the trace to be performed is equal to the sum of the delays corresponding to each of the transitions.

## 3 Stochastic Implementation Relations: A First Attempt

In this section we introduce our first implementation relation. It follows the classical pattern: An implementation $I$ *conforms* to a specification $S$ if for any 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-timed conformance of the implementation, we will require some time conditions to be held.

Next, we formally define the sets of specifications and implementations: $\mathtt{Spec}$ and $\mathtt{Imp}$. We will consider that both specifications and implementations are given by deterministically observable $\mathtt{SFSM}$s. That is, we do not allow a machine to have two different transitions as $(s, i, o, \xi_1, s')$ and $(s, i, o, \xi_2, s'')$. Note that we do not restrict observable non-determinism, that is, we may have the transitions $(s, i, o_1, \xi_1, s_1)$ 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 3. This is a usual condition to assure that the implementation will react (somehow) to any input appearing in the specification.

Next, we introduce our first implementation relation.

**Definition 5.** Let $I$ and $S$ be $\mathtt{SFSM}$s. We say that $I$ *non-stochastically conforms* to $S$, denoted by $I\,\mathtt{conf}_{ns}\,S$, if for each $\rho = (i_1/o_1, \ldots, i_n/o_n) \in \mathtt{NSTraces}(S)$, with $n \geq 1$, we have

$$\rho' = (i_1/o_1, \ldots, i_n/o'_n) \in \mathtt{NSTraces}(I) \text{ implies } \rho' \in \mathtt{NSTraces}(S)$$

We say that $I$ *stochastically conforms* to $S$, denoted by $I\,\mathtt{confs}\,S$, if $I\,\mathtt{conf}_{ns}\,S$ and for any $\rho_s = (\rho, \xi) \in \mathtt{STraces}(I)$ we have

$$\rho \in \mathtt{NSTraces}(S) \text{ implies } (\rho, \xi') \in \mathtt{STraces}(S) \;\wedge\; \xi = \xi'$$

□

Intuitively, the idea underlying the definition of the non-stochastic conformance relation $I\,\mathtt{conf}_{ns}\,S$ is that the implementation $I$ does not *invent* anything for those inputs that are *specified* in the specification. Let us note that if the specification has also the property of input-enabled then we may remove the condition “for each $\rho = (i_1/o_1, \ldots, i_n/o_n) \in \mathtt{NSTraces}(S)$, with $n \geq 1$”.

*Example 2.* Consider the finite state machines $M_2$ and $M_3$ depicted in Figure 1 (center and right). We have $M_3\,\mathtt{conf}_{ns}\,M_2$. Note that the traces of $M_3$ having as prefix the sequence $i_1/o_1, i_2/o$, with $o \in \{o_1, o_3\}$, are not checked since $M_2$ (playing the role of specification) cannot perform those traces.

Let us now consider that $M_2$ is extended with the transition $(2, i_2, \mathtt{fail}, \xi_2, 2)$, for a random variable $\xi_2$, so that it fulfills the conditions required for implementations (input-enabled). Then, $M_2$ does not conform to $M_3$. For example, $M_3$

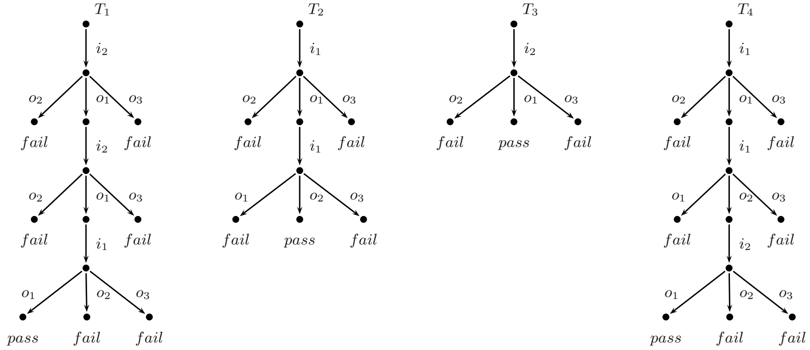

**Fig. 2.** Examples of Test Cases.

may perform the trace $\rho = i_1/o_1, i_2/o_3$, $M_2$ has the trace $\rho' = i_1/o_1, i_2/\texttt{fail}$, but $\rho'$ does not belong to the set of non-stochastic traces of $S_3$. Note that $\rho$ and $\rho'$ have the same prefix $i_1/o_1, i_2$. □

In addition to requiring this notion of *non-timed* conformance, we have to ask for some conditions on delays. Thus, $I$ $\texttt{confs}$ $S$ also requires that any stochastic trace of the specification that can be 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 grey/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, we have to give more *realistic* implementation relations based on a finite set of observations. In the next sections we will present other implementation relations that are less *accurate* but that are *checkable.*

## 4 Tests Cases for Stochastic Systems

In this section we introduce test cases and we present how they are applied to implementations under test. We consider that a test case is simply a sequence of inputs applied to the implementation. Once an output is received, we check whether it is the expected one or not. In the latter case, a fail signal is produced. In the former case, either a pass signal is emitted (indicating successful termination) or the testing process continues by applying another input. If we are testing an implementation with input and output sets $I$ and $O$, respectively, tests are deterministic acyclic $I/O$ labelled transition systems (i.e. trees) with a strict alternation between an input action and the whole set of output actions. After an output action we may find either a leaf or another input action. Leaves can be labelled either by *pass* or by *fail*. In the first case we add a *random variable*. The idea is that we will record the time that it takes for the implementation to arrive to that point. We will collect a sample of times (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 6.** A *test case* is a tuple $T = (S, I, O, \delta, s_0, S_I, S_O, S_F, S_P, \zeta)$ where $S$ is the set of states, $I$ and $O$, with $I \cap O = \emptyset$ are the sets of input and output actions, respectively, $\delta \subseteq S \times I \cup O \times S$ is the transition relation, $s_0 \in S$ is the initial state, and the sets $S_I, S_O, S_F, S_P \subseteq S$ are a partition of $S$. The transition relation and the sets of states fulfill the following conditions:

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

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

Let $\rho = (i_1/o_1, \ldots, i_r/o_r)$. We write $T \overset{\rho}{\Longrightarrow} s$, if $s \in S_F \cup S_P$ and there exist states $s_{12}, s_{21}, s_{22}, \ldots s_{r1}, s_{r2} \in S$ such that $\{(s_0, i_1, s_{12}), (s_{r2}, o_r, s)\} \subseteq \delta$, for any $2 \leq j \leq r$ we have $(s_{j1}, i_j, s_{j2}) \in \delta$, and for any $1 \leq j \leq r-1$ we have $(s_{j2}, o_j, s_{(j+1)1}) \in \delta$.

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

In Figure 2 we give some test cases, where random variables associated with passing states have been omitted. Before we define the application of a test to an implementation, we need to introduce an auxiliary definition. As we said before, we will compare the time that the implementation needs to perform the test with the random variable appearing in the reached passing state of the test. Thus, we suppose that there exists a mechanism allowing to observe the time that the implementation needed to perform the sequence offered by the test and that leads the test to the corresponding passing state.

**Definition 7.** Let $I$ be a `SFSM`. We say that $((i_1/o_1, \ldots, i_n/o_n), t)$ is the *observed timed execution* of $I$, or simply *timed execution*, if the observation of $I$ shows that the sequence $(i_1/o_1, \ldots, i_n/o_n)$ is performed in time $t$.

Let $\Phi = \{\rho_1, \ldots, \rho_m\}$ be a set of traces and $H = \{\!|(\rho'_1, t_1), \ldots, (\rho'_n, t_n)|\!\}$ be a multiset of timed executions. We say that $\texttt{Sampling}_{(H,\Phi)} : \Phi \longrightarrow \wp(\mathbb{R}^+)$ is a *sampling application* of $H$ for $\Phi$ if $\texttt{Sampling}_{(H,\Phi)}(\rho) = \{\!| t \mid (\rho, t) \in H |\!\}$, for any $\rho \in \Phi$. □

Regarding the definition of sampling applications, we just associate with each trace the observed times of execution. Next we define the application of a test to an implementation.

**Definition 8.** Let $I = (S_1, I, O, \delta_1, s_{in})$ be an implementation under test and $T = (S_2, I, O, \delta_2, s_0, S_I, S_O, S_F, S_P, \zeta)$ be a valid test. We denote the application of the test $T$ to the implementation $I$ as $I \,\|\, T$.

We write $I \,\|\, T \overset{\rho}{\Longrightarrow}_t s^T$ if $T \overset{\rho}{\Longrightarrow} s^T$ and $(\rho, t)$ is the observed timed execution of $I$. In this case we say that $(\rho, t)$ is a *test execution* of $I$ and $T$.

Let $H = \{\!|(\rho_1, t_1), \ldots, (\rho_n, t_n)|\!\}$ be a test executions sample of $I$ and $T$, $0 \leq \alpha \leq 1$, $\Phi = \{\rho \mid \exists\, s^T, t : I \,\|\, T \overset{\rho}{\Longrightarrow}_t s^T\}$, and let us consider $\texttt{Sampling}_{(H,\Phi)}$. We say that the implementation $I$ $(\alpha, H)-$*passes* the test $T$ if for any trace $\rho \in \texttt{NSTraces}(I)$ with $I \,\|\, T \overset{\rho}{\Longrightarrow}_t s^T$ we have both that $s^T \notin S_F$ and if $s^T \in S_P$ then $\gamma(\zeta(s^T), \texttt{Sampling}_{(H,\Phi)}(\rho)) > \alpha$. □

Intuitively, an implementation passes a test if two conditions hold. First, there is no evolution leading to a fail state (see the condition $s^T \notin S_F$). Once we know that the functional behavior of the implementation is correct with respect to the test, we need to check time conditions. The set $H$ corresponds to the observations of the (several) applications of $T$ to $I$. Thus, we have to decide whether, for each trace $\rho$, the observed times (that is, $\texttt{Sampling}_{(H,\Phi)}(\rho)$)) *match* the definition of the random variable appearing in the successful state of the test corresponding to the execution of that trace (that is, $\zeta(s^T)$). As we commented previously, we assume a function $\gamma$ 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 $\chi^2$).

Next we extend the notion of passing a test to deal with test suites. In order to increase the degree of reliability, we will not take the classical approach where passing a test suite is defined according only to the results for each test. In our approach, we will put together all the observations, for each test, so that we have more samples for each trace. In particular, some observations will be used several times. In other words, an observation from a given test may be used to check the validity of another test sharing the same observed trace.

**Definition 9.** Let $I$ be a $\texttt{SFSM}$ and $\Omega = \{T_1, \ldots, T_n\}$ be a set of tests. Let $H_1, \ldots, H_n$ be test execution samples of $I$ and $T_1, \ldots, T_n$, respectively. Finally, let $H = \bigcup_{i=1}^{n} H_i$. We say that the implementation $I$ $(\alpha, H)-$*passes* the test suite $\Omega$ if for any $1 \leq i \leq n$ we have that $I$ $(\alpha, H_i')-$passes $T_i$, where $H_i' = \{\!|(\rho, t) \mid (\rho, t) \in H \,\wedge\, \exists\, s^T, t : I \,\|\, T_i \overset{\rho}{\Longrightarrow}_t s^T|\!\}$. □

## 5 Implementation Relations Based on Samples

In Section 3 we presented an implementation relation that fulfilled a nice definition and it seemed appropriate for our framework. Unfortunately, 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. In this section we introduce a new implementation relation that, inspired by our testing framework, takes into account the observations that we may get from the implementation.

**Definition 10.** Let $I$ and $S$ be two SFSMs. Let $H$ be a multiset of timed executions of $I$, $0 \leq \alpha \leq 1$, $\Phi = \texttt{NSTraces}(I) \cap \texttt{NSTraces}(S)$, and let us consider $\texttt{Sampling}_{(H,\Phi)}$. We say that $I$ $(\alpha, H)-$*stochastically conforms* to $S$, denoted by $I \; \texttt{confs}^{(\alpha,H)} \; S$, if $I \; \texttt{conf}_{ns} \; S$ and for any non-stochastic trace $\rho = (i_1/o_1, \ldots, i_n/o_n) \in \texttt{NSTraces}(I)$ we have that if there exists a stochastic trace $((i_1/o_1, \ldots, i_n/o_n), \xi) \in \texttt{STraces}(S)$ then $\gamma(\xi, \texttt{Sampling}_{(H,\Phi)}(\rho)) > \alpha$. □

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 that it can be performed by the specification, the observed execution times *fit* the random variable indicated by the specification. Again, this notion of *fitting* is given by the function $\gamma$ that it is formally defined in the appendix of this paper. A first direct result says that if we decrease the confidence level then we keep conformance.

**Lemma 1.** Let $I$ and $S$ be two SFSMs such that $I \; \texttt{confs}^{(\alpha_1,H)} \; S$. If $\alpha_2 < \alpha_1$ then we have $I \; \texttt{confs}^{(\alpha_2,H)} \; S$. □

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

**Lemma 2.** Let $I$ and $S$ be two SFSMs, $H_1, H_2$ be multisets of timed executions for $I$, and let $b_i = \{\!|(\rho, t) \mid (\rho, t) \in H_i \;\wedge\; \rho \in \texttt{NSTraces}(I) \cap \texttt{NSTraces}(S)|\!\}$, for $i = \{1, 2\}$. If $b_1 = b_2$ then we have $I \; \texttt{confs}^{(\alpha,H_1)} \; S$ iff $I \; \texttt{confs}^{(\alpha,H_2)} \; S$. □

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 $\xi$ following a Dirac distribution in 4 (see Example 1 for the formal definition). If we consider a new random variable $\xi'$ following a Dirac distribution in 3, we say that $\xi'$ represents a shift of $\xi$. Moreover, we also say that $\xi$ and $\xi'$ belong to the same family.

**Definition 11.** We say that $\xi'$ is a *mean shift* of $\xi$ with mean $M'$, and we denote it by $\xi' = \texttt{MShift}(\xi, M')$, if $\xi, \xi'$ belong to the same family and the mean of $\xi'$, denoted by $\mu_{\xi'}$, is equal to $M'$.

Let $I$ and $S$ be two SFSMs. Let $H$ be a multiset of timed executions of $I$, $0 \leq \alpha \leq 1$, $\Phi = \texttt{NSTraces}(I) \cap \texttt{NSTraces}(S)$, and let us consider $\texttt{Sampling}_{(H,\Phi)}$. We say that $I$ $(\alpha, H)-$*stochastically conforms* to $S$ *with speed* $\pi$, denoted by $I \texttt{confm}_{\pi}^{(\alpha,H)} S$, if $I \, \texttt{conf}_{ns} S$ and for any $\rho = (i_1/o_1, \ldots, i_n/o_n) \in \texttt{NSTraces}(I)$ we have that if there exists a stochastic trace $((i_1/o_1, \ldots, i_n/o_n), \xi) \in \texttt{STraces}(S)$ then $\gamma(\texttt{MShift}(\xi, \mu_\xi \cdot \pi), \texttt{Sampling}_{(H,\Phi)}(\rho)) > \alpha$. □

An interesting remark regarding this new relation is that when $\alpha$ is *small enough* and/or $\pi$ is *close enough* to 1, then it may happen that we have both

$I\,\mathtt{confs}^{(\alpha,H)}\,S$ and $I\mathtt{confm}_{\pi}^{(\alpha,H)}S$. Nevertheless, it is enough to increase $\alpha$, as far as $\pi \neq 1$, so that we do not have both results simultaneously. Let us note that in the previous notion, a value of $\pi$ greater than 1 indicates that the new delay is *slower*. This observation induces the following relation.

**Definition 12.** Let $I$ and $S$ be two SFSMs. Let $H$ be a multiset of timed executions of $I$. We say that $I$ is *generally faster* (respectively *generally slower*) than $S$ for $H$ if there exist $0 \leq \alpha \leq 1$, and $0 < \pi < 1$ (respectively $\pi > 1$) such that $I\mathtt{confm}_{\pi}^{(\alpha,H)}S$ but $I\,\mathtt{confs}^{(\alpha,H)}\,S$ 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 $\alpha$.

**Definition 13.** Let $I$ and $S$ be two SFSMs. Let $H$ be a multiset of timed executions of $I$. Let us consider $0 \leq \alpha \leq 1$ and $\pi \in \mathbb{R}^+$ such that $I\mathtt{confm}_{\pi}^{(\alpha,H)}S$ and there do not exist $\alpha' > \alpha$ and $\pi' \in \mathbb{R}^+$ with $I\mathtt{confm}_{\pi'}^{(\alpha',H)}S$. Then, we say that $\pi$ 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. However, let us remark that this new notion correspond neither to our first implementation relation (see Definition 5) nor to have a confidence level $\alpha$ equal to 1.

**Definition 14.** Let $I$ and $S$ be two SFSMs. Let $H$ be a multiset of timed executions of $I$ and $0 \leq \alpha \leq 1$. We say that $I$ $(\alpha, H)-$*stochastically and precisely conforms* to $S$, denoted by $I\,\mathtt{confp}^{(\alpha,H)}\,S$, if $I\,\mathtt{confs}^{(\alpha,H)}\,S$ 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 3.** Let $I$ and $S$ be two SFSMs. We have $I\,\mathtt{confp}^{(\alpha,H)}\,S$ iff $I\,\mathtt{confs}^{(\alpha,H)}\,S$ and neither $I$ is generally faster than $S$ for $H$ nor $I$ is generally slower than $S$ for $H$. □

## 6 Test Derivation

In this section we provide an algorithm to derive tests from specifications. In addition, we will show that the derived test suite is *complete*, up to a given confidence $\alpha$ and for a sample $H$, with respect to the conformance relation $\mathtt{confs}^{(\alpha,H)}$ given in Definition 10. As usually, the idea consists in traversing the specification to get all the possible traces in the adequate way. So, each test is generated so that it *focuses* on chasing a concrete trace of the specification.

```
Input: $M = (S, I, O, \delta, s_{in})$.
Output: $T = (S', I, O, \delta', s_0, S_I, S_O, S_F, S_P, \zeta)$.

Initialization:

  – $S' := \{s_0\}, \delta' := S_I := S_O := S_F := S_P := \zeta := \emptyset$.
  – $S_{aux} := \{(s_{in}, \xi_0, s_0)\}$.

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

  1. If $(s^M, \xi, s^T) \in S_{aux}$ then perform the following steps:
     (a) $S_{aux} := S_{aux} - \{(s^M, \xi, s^T)\}$.
     (b) $S_P := S_P \cup \{s^T\};\ \zeta(s^T) := \xi$.
  2. If $S_{aux} = \{(s^M, \xi, s^T)\}$ is a unitary set and there exists $i \in I$ such that
     $\mathtt{out}(s^M, i) \neq \emptyset$ then perform the following steps:
     (a) $S_{aux} := \emptyset$.
     (b) Choose $i$ such that $\mathtt{out}(s^M, i) \neq \emptyset$.
     (c) Create a fresh state $s' \notin S'$ and perform $S' := S' \cup \{s'\}$.
     (d) $S_I := S_I \cup \{s^T\};\ S_O := S_O \cup \{s'\};\ \delta' := \delta' \cup \{(s^T, i, s')\}$.
     (e) For each $o \notin \mathtt{out}(s^M, i)$ do
           – Create a fresh state $s'' \notin S'$ and perform $S' := S' \cup \{s''\}$.
           – $S_F := S_F \cup \{s''\};\ \delta' := \delta' \cup \{(s', o, s'')\}$.
     (f) For each $o \in \mathtt{out}(s^M, i)$ do
           – Create a fresh state $s'' \notin S'$ and perform $S' := S' \cup \{s''\}$.
           – $\delta' := \delta' \cup \{(s', o, s'')\}$.
           – $s_1^M := \mathtt{after}(s^M, i, o)$.
           – Let $(s^M, i, o, \xi_1, s_1^M) \in \delta$. $S_{aux} := S_{aux} \cup \{(s_1^M, \xi + \xi_1, s'')\}$.
```

**Fig. 3.** Test Derivation Algorithm.

However, if during the execution of the test it is detected an output which does not correspond to such a trace, then the testing process stops and finishes in either a pass or a fail state. The status of the final state depends on whether such an output is *allowed* by the specification at this point. First, we give some auxiliary functions.

**Definition 15.** Let $M = (S, I, O, \delta, s_{in})$ be a `SFSM`. We define the set of possible outputs in state $s$ after input $i$ as $\mathtt{out}(s, i) = \{o \,|\, \exists s' : (s, i, o, \xi, s') \in \delta\}$. For any transition $(s, i, o, \xi, s') \in \delta$ we write $\mathtt{after}(s, i, o) = s'$. □

Let us remark that due to the assumption that `SFSM`s are deterministically observable we have that $\mathtt{after}(s, i, o)$ is uniquely determined.

Our derivation algorithm is presented in Figure 3. By considering the possible available choices we get a set of tests extracted from the specification $M$. We denote this set of tests by $\mathtt{tests}(M)$.

In this algorithm, the set of pending states $S_{aux}$ keeps track of the states of the test under construction whose definition has not been *finished* yet. A tuple $(s^M, \xi, s^T) \in S_{aux}$ indicates that the current state in the traversal of the specification is $s^M$, that the random variable accounting for the elapsed time in the specification from the initial state is $\xi$, and that we did not conclude yet the description of the state $s^T$ in the test.

The set $S_{aux}$ initially contains a tuple with the initial states (of both specification and test) and a random variable $\xi_0$ following a Dirac distribution in 0 (i.e. no time has yet elapsed). For each tuple in $S_{aux}$ we may choose one possibility. It is important to remark that the second possibility is applied at most to one of the possible tuples. Thus, our derived tests correspond to valid tests as given in Definition 6. The first possibility simply indicates that the state of the test becomes a passing state. The second possibility takes an input and generate a transition in the test labelled by this input. Then, the whole sets of outputs is considered. If the output is not expected by the implementation, a transition leading to a failing state is created. This could be simulated by a single branch in the test, labelled by `else`, leading to a failing state (in the algorithm we suppose that *all* the possible outputs appear in the test). For the rest of outputs, we create a transition with the corresponding output and add the appropriate tuple to the set $S_{aux}$.

Finally, let us remark that finite test cases are constructed simply by considering a step where the second inductive case is not applied. For example, the test cases appearing in Figure 2 can be generated by applying the previous algorithm to the `SFSM` $M_2$ given in Example 1. In addition, we have to define the random variables corresponding to the passing states. Given that each of these tests has a unique passing state, we will denote by $\psi_i$ the random variable associated with the passing state of the test $T_i$. We have $\psi_1 = \xi_1 + \xi_1 + \xi_2$, $\psi_2 = \xi_2 + \xi_3$, $\psi_3 = \xi_1$, and $\psi_4 = \xi_2 + \xi_3 + \xi_1$.

The next result states that for any specification $S$ we have that the test suite $\texttt{tests}(S)$ can be used to distinguish those (and only those) implementations that conforms with respect to `confs`. However, we cannot say that the test suite is complete since both passing tests and the considered implementation relation have a probabilistic component. So, we can talk of *completeness* up to a certain confidence level.

**Proposition 1.** Let $I$ and $S$ be `SFSM`s. For any $0 \leq \alpha \leq 1$ and multiset of timed executions $H$ we have $I\,\texttt{confs}^{(\alpha,H)}\,S$ iff $I$ $(\alpha, H)$*−passes* $\texttt{tests}(S)$. *Proof Sketch*: $I\,\texttt{confs}^{(\alpha,H)}\,S$ requires that $I\,\texttt{conf}_{ns}\,S$ and that for any trace belonging to both the specification and the implementation, the confidence of the random variable of the specification on the samples observed in the implementation is higher than $\alpha$. The way our algorithm deals with non-stochastic information of the traces is quite similar to other test derivation algorithms. Regarding stochastic information, let us remark that the samples collected from the tests will be exactly the samples we apply to check whether the implementation relation holds. So, the conditions we require about the confidence of the random variables on the samples will be the same both to pass the test and to make the implementation relation to hold. □

The previous idea can be easily extended to some of the implementation relations presented in Section 5. For example, we may suppose that $\texttt{tests}(S, \pi)$

denotes the test suite generated by replacing the last instruction of the algorithm in Figure 3 by $S_{aux} := S_{aux} \cup \{(s_1^M, \xi + \texttt{MShift}(\xi_1, \mu_{\xi_1} \cdot \pi), s'')\}$, and then applying the resulting algorithm.

**Proposition 2.** Let $I$ and $S$ be SFSMs. For any $0 \leq \alpha \leq 1$, $\pi \in \mathbb{R}^+$, and multiset of timed executions $H$ we have $I \, \texttt{confm}_\pi^{(\alpha,H)} \, S$ iff $I$ $(\alpha, H)-$passes $\texttt{tests}(S, \pi)$. □

## 7 Conclusions and Future Work

We have introduced a model of stochastic timed processes and we have presented several implementation relations for this model. Each of them has some advantages and drawbacks, so experiments on them are necessary to find out their practical usefulness. In particular, the most *appropriate* relation is not *checkable* if we consider a grey/black-box testing framework. We have included time by means of the delay between the reception of an input and the generation of an output. This delay is given by a random variable. Regarding testing, we have defined an appropriate notion of test case and its application to the implementation under test. We have given a derivation algorithm producing, for each specification, a test suite. Finally, we have presented a notion of pseudo-completeness of a test suite, for a certain degree of confidence.

A question that we do not address in this paper is test coverage as this is beyond the scope of a first approach to this difficult topic. However, we think that this is a relevant issue and we plan to confront it in the near future.

## 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. M. Ajmone Marsan, G. Conte, and G. Balbo. A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 5(2):93–122, 1984.
3. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
4. M. Bernardo and W.R. Cleaveland. A theory of testing for markovian processes. In *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*, 202:1–54, 1998.
6. 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.
7. M. Bravetti and R. Gorrieri. The theory of interactive generalized semi-Markov processes. *Theoretical Computer Science*, 282(1):5–32, 2002.
8. 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.

9. R. de Nicola and M.C.B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.
10. 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.
11. 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.
12. P.G. Harrison and B. Strulo. SPADES – a process algebra for discrete event simulation. *Journal of Logic Computation*, 10(1):3–42, 2000.
13. M. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988.
14. H. Hermanns, U. Herzog, and J.-P. Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 274(1-2):43–87, 2002.
15. 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.
16. J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.
17. R. Lai. A survey of communication protocol testing. *Journal of Systems and Software*, 62:21–46, 2002.
18. D. Lee and M. Yannakakis. Principles and methods of testing finite state machines: A survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.
19. N. López and M. Núñez. A testing theory for generally distributed stochastic processes. In *CONCUR 2001, LNCS 2154*, pages 321–335. Springer, 2001.
20. N. López, M. Núñez, and F. Rubio. Stochastic process algebras meet Eden. In *Integrated Formal Methods 2002, LNCS 2335*, pages 29–48. Springer, 2002.
21. 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.
22. M. Núñez and I. Rodríguez. Encoding `PAMR` into (timed) `EFSMs`. In *FORTE 2002, LNCS 2529*, pages 1–16. Springer, 2002.
23. 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* $\chi^2$ *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 $\chi^2$ .
- We estimate the number of freedom degrees of $\chi^2$ 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 $\alpha$ low enough. Actually, as such margin to accept the sample decreases as $\alpha$ decreases, we can obtain a measure of the validity of the sample as $min\{\alpha \mid X^2 < \chi^2_\alpha(k - r - 1)\}$.

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

**Definition 16.** Let $\xi$ 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 $\xi$ 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 $\xi$ on $J$ with classes $S$, denoted by $\gamma(\xi, J)$, as $min\{\alpha \mid X^2 < \chi^2_\alpha(k - 1)\}$. □

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.