Extending EFSMs to specify and test timed systems with action durations and timeouts *

Mercedes G. Merayo, Manuel Núñez and Ismael Rodríguez

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

Abstract. In this paper we introduce a timed extension of the extended finite state machines model. On the one hand, we consider that output actions take time to be performed. This time may depend on several factors such as the value of variables. On the other hand, our formalism allows to specify timeouts. In addition to present our formalism, we develop a testing theory. First, we define ten timed conformance relations and relate them. Second, we introduce a notion of timed test and define how to apply tests to IUTs.

1 Introduction

Formal testing techniques [2,9,14,3,5] allow to test the correctness of a system with respect to a specification. Formal testing originally targeted the functional behavior of systems, such as determining whether the tested system can, on the one hand, perform certain actions and, on the other hand, does not perform some non-expected ones. In the last years formal testing techniques also deal with non-functional properties such as the time that it takes to perform a certain action. In order to test timed systems, more precisely, the timed behavior of a system, we need a suitable language to formally specify these systems. The time consumed during the execution of a system falls into one of the following categories:

(a) The system consumes time while it performs its operations. This time may depend on the values of certain parameters of the system, such as the available resources.
(b) The time passes while the system waits for a reaction from the environment. In particular, the system can change its internal state if an interaction is not received before a certain amount of time.

A language focussing on temporal issues should allow the specifier to define how the system behavior is affected by both kinds of temporal aspects (e.g., a task is performed if executing the previous task took too much time, if the environment does not react for a long time, if the addition of both times exceeds a

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

given threshold, etc). In this paper we present a formalism, based on *extended finite state machines*, allowing to take into account the subtle temporal aspects considered before. Even though there exists a myriad of timed extensions of classical frameworks, most of them specialize only in one of the previous variants: Time is either associated with actions or associated with delays/timeouts. Our formalism allows to specify in a natural way both time aspects. While the definition of the new language is not difficult, mixing these temporal requirements strongly complicates the posterior theoretical analysis. In particular, the definition of timed conformance testing relations is more difficult than usually. The theoretical framework is also complicated by two additional features. First, we consider that variables may influence the timing aspects of the specification. Thus, the execution of an action may take different time values if the value of the variables change. Second, we do not impose any restriction on the deterministic behavior of our machines. This implies again that the same sequence of actions may take different time values to be executed.

We also propose a formal testing methodology allowing to systematically test a system with respect to a specification. Regarding functional conformance we have to consider not only that the sequences of inputs/outpus produced by the implementation must be considered in the specification. We have to take into account the possible timeouts. For example, a sequence of inputs/outputs could be accepted only after different timeouts have been triggered. Let us consider the machines depicted in Figure 1. In these diagrams we use the following notation: A transition labelled by '$i/o, t$' denotes that the execution of the output action o takes time t to be performed after the input i is received; a transition with a label t indicates that a timeout will be applied at time t. That is, if after t time units no input is received then the timeout is executed. If we consider M_1 and M_2 we can observe that M_1 is not functionally conforming to M_2. The sequence i_1/o_2 that can be performed by M_1 is forbidden by M_2. On the other hand, if we consider the conformance of M_2 with respect to M_1 and we only check the possible sequences of inputs/outputs, M_2 would conform to M_1 due to the fact that the unique sequence that can be performed by M_2 is i_1/o_1. However, this sequence is allowed by M_1 only in the case that the input has been received after three time units. So, under our conformance framework, M_2 does not conform to M_1. We can say the same regarding the conformance of M_3 with respect to M_1. On the contrary, this is not the case when considering the conformance of M_1 with respect to M_3. The sequences performed by M_1 are accepted by M_3 at any time. So, M_1 functionally conforms to M_3.

Let us note that testing the temporal behavior of a system affected by nondeterminism requires to face some issues that are not considered by other testing approaches. In particular, contrary to usual approaches, providing an *incorrectness* diagnosis may require to consider the result of *all* tests in a test suite, because a single test result could be insufficient to claim the incorrectness of the IUT.[1] For instance, we may require that, among all the times the IUT may

[1] If we consider this statement the other way around then the resulting scenario is the usual one: Passing a test does not allow to claim that the IUT is correct.

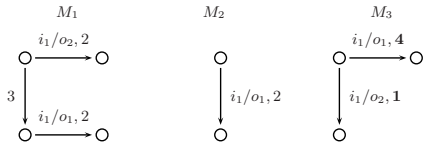


Fig. 1. Examples of functional conformance.

consume to perform a task, one of them is smaller than the corresponding specification time for this task. Hence, if during the application of a test we observe that the IUT takes a long time, then it does not necessarily mean that the IUT cannot do it faster. Regarding temporal performance requirements, our testing methodology will take into account that the system is only responsible for the (a) type consumed time, not for that of (b) type. That is, we have to distinguish between time associated with the performance of tasks and passing of time due to the possible inactivity of the operator of the system.

Our *timed conformance relations* follow the standard pattern: An implementation is correct with respect to a specification if it does not show any behavior that is forbidden by the specification, where both the functional behavior and the temporal behavior are considered (and, implicitly, how they affect each other). In this paper we present ten different conformance relations. The differences among them come from the effect non-determinism causes in specifications/implementations. We will relate all these notions and propose alternative characterizations for some of them.

In terms of related work, our way to deal with time is completely different to that in *timed automata* [1]. As we said before, we can associate time with the performance of actions while timeouts can be easily represented. These features do not only improve the modularity of models, but they are also suitable for clearly identifying IUT requirements and responsibilities in a testing methodology. This paper continues the work in [12]. The main advantage with respect to this previous work is that we can now express timeouts, we remove all the restrictions regarding non-determinism of the machines, and we consider more conformance relations. Regarding testing of temporal requirements, there exist several proposals (e.g., [4,8,15,6]) but most of them are based on timed automata. Moreover, in these approaches tests are independent and the diagnosis of a test does not depend on other tests. By considering that tests are interrelated, we can relate non-determinism and temporal requirements, as well as define and apply several conformance relations where non-determinism is explicitly considered. There are also some time extensions of `FSM`s(e.g., [13,7]) but they do not deal with conformance.

The rest of the paper is structured as follows. In Section 2 we introduce our model. In Section 3 we give our timed conformance relations and provide several examples to show the differences among them. In Section 4 we show how tests are defined and applied to IUTs.

2 A timed extension of the EFSM model

In this section we introduce our notion of *timed extended finite state machine*. As we have indicated in the introduction of the paper, we will add new features so that the timed behavior of a system can be properly specified. On the one hand, we consider that output actions take time to be executed. These time values will not only depend on the corresponding action to be performed and the state where the machine is placed. Actually, we will also consider that this time value takes into account the current value of the variables. In fact, with this approach we can simulate that the speed with which a task is performed depends on the available resources. On the other hand, we will also consider that the machine can evolve by raising *timeouts*. Intuitively, if after a given time, depending on the current state, we do not receive any input action then the machine will change its current state.

During the rest of the paper we will use the following notation. A tuple of elements $(e_1, e_2 \ldots, e_n)$ will be denoted by $\bar{e}$. â denotes an interval of elements $[a_1, a_2)$. A tuple of intervals is denoted by $\breve{t}$. Let $\breve{q} = (\hat{q_1}, \ldots, \hat{q_n})$ and $\bar{t} = (t_1, \ldots, t_n)$. We write $\bar{t} \in \breve{q}$ if for all $1 \leq j \leq n$ we have $t_j \in \hat{q_j}$. We will use the projection function π_i such that given a tuple $\bar{t} = (t_1, \ldots, t_n)$, for all $1 \leq i \leq n$ we have $\pi_i(\bar{t}) = t_i$. Let $\bar{t} = (t_1, \ldots, t_n)$ and $\bar{t'} = (t'_1, \ldots, t'_n)$. We write $\bar{t} = \bar{t'}$ if for all $1 \leq j \leq n$ we have $t_j = t'_j$. We write $\bar{t} \leq \bar{t'}$ if for all $1 \leq j \leq n$ we have $t_j \leq t'_j$. Finally, we will denote by $\sum \bar{t}$ the sum of all elements of the tuple $\bar{t}$, that is, $\sum_{j=1}^{n} t_j$.

Definition 1. Let `Time` be the domain to define time values, $D_1, \ldots, D_m$ be sets of values, and let us consider $\mathcal{D} = D_1 \times D_2 \times \cdots \times D_m$. A *Timed Extended Finite State Machine*, in the following `TEFSM`, is a tuple $M = (S, I, O, TO, Tr, s_{in}, \bar{y})$ where S is a finite set of *states*, I is the set of *input action*s, O is the set of *output actions*, $TO : S \longrightarrow S \times (\texttt{Time} \cup \infty)$ is the *timeout function*, Tr is the set of *action transitions*, s_{in} is the *initial state*, and $\bar{y} \in \mathcal{D}$ is the tuple of initial values of the *variables*.

An *action transition* is a tuple (s, s', i, o, Q, Z, C) where $s, s' \in S$ are the initial and final state of the transition, $i \in I$ and $o \in O$ are the input and output action associated with the transition, $Q : \mathcal{D} \longrightarrow \texttt{Bool}$ is a predicate on the set of variables, $Z : \mathcal{D} \longrightarrow \mathcal{D}$ is a transformation over the current variables, and $C : \mathcal{D} \longrightarrow \texttt{Time}$ is the time that the transition needs to be completed.

A *configuration* in M is a pair $(s, \bar{x})$ where $s \in S$ is the current state and $\bar{x} \in \mathcal{D}$ is the tuple containing the current value of the variables.

We say that M is *input-enabled* if for all state $s \in S$ and input $i \in I$ there exist s', o, Q, Z, C such that $(s, s', i, o, Q, Z, C) \in Tr$. □

Given a configuration $(s, \bar{x})$, an action transition (s, s', i, o, Q, Z, C) denotes that if the input i is received and $Q(\bar{x})$ holds then the output o will be produced after $C(\bar{x})$ units of time, and the configuration will be $(s', Z(\bar{x}))$. In this paper we consider that time can be discretized, that is, the time domain is isomorphic to $\mathbb{N}$. We will take advantage of this characteristic to simplify some definitions.

In particular, we will sometimes enumerate the elements of $\texttt{Time}$ simply as $0, 1, 2$ and so on.

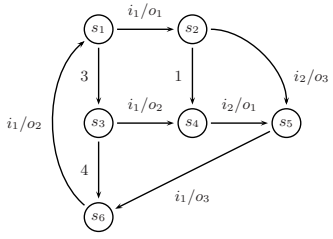
For each state $s \in S$, the application of the timeout function $TO(s)$ returns a pair (s', t) indicating the time that the machine can remain at the state s waiting for an input action, and the state to which the machine evolves if no input is received on time. We assume that $TO(s) = (s', t)$ implies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

Definition 2. Let $M = (S, I, O, TO, Tr, s_{in}, \bar{y})$ be a $\texttt{TEFSM}$ and $c_0 = (s_0, \bar{x}_0)$ be a configuration of M. A tuple $(s_0, s, i/o, \hat{t}, t_o, \bar{v})$ is a *step* of M for the configuration c_0 if there exist $k \geq 0$ states $s_1, \ldots, s_k \in S$ such that for all $1 \leq j \leq k$ we have $TO(s_{j-1}) = (s_j, t_j)$ and there exists a transition $(s_k, s, i, o, Q, Z, C) \in T_r$ such that $Z(\bar{x}_0) = \bar{v}$, $\hat{t} = \left[\sum_{j=1}^{k} t_j, \sum_{j=1}^{k} t_j + \pi_2(TO(s_k))\right)$, $t_o = C(\bar{x}_0)$ and $Q(\bar{x}_0)$ holds. We denote by $\texttt{Steps}(M, s, \bar{x})$ the set of steps of M for the configuration $(s, \bar{x})$.

We say that $(\hat{t}_1/i_1/t_{o1}/o_1, \ldots, \hat{t}_r/i_r/t_{or}/o_r)$ is a *timed evolution* of M if there exist r steps of M $(s_{in}, s_1, i_1/o_1, \hat{t}_1, t_{o1}, \bar{y}_1), \ldots, (s_{r-1}, s_r, i_r/o_r, \hat{t}_r, t_{or}, \bar{y}_r)$ for the configurations $(s_{in}, \bar{y}), \ldots, (s_{r-1}, \bar{y}_{r-1})$, respectively. We denote by $\texttt{TEvol}(M)$ the set of timed evolutions of M. In addition, we say that $(\hat{t}_1/i_1/o_1, \ldots, \hat{t}_r/i_r/o_r)$ is a *functional evolution* of M and we denote by $\texttt{FEvol}(M)$ the set of functional evolutions of M. □

Intuitively, a step is an action transition preceded by zero or more timeouts. The interval $\hat{t}$ indicates the time values where the input action could be received. An evolution is a sequence of inputs/outputs corresponding to the action transitions of a chain of steps where the first one begins with the initial configuration of the machine. In addition, timed evolutions include time values which inform us about possible timeouts (indicated by the intervals $\hat{t}_j$) and the time consumed to execute each output after receiving each input in each step of the evolution.

Example 1. Consider the $\texttt{TEFSM}$ depicted in Figure 2. We suppose that the variables of the $\texttt{TEFSM}$ are given by a tuple $\bar{x} \in \mathbb{R}_+^4$ and we denote by x_i the *i-th* component of $\bar{x}$. Let us assume that the value of the variables is $\bar{x} = (1, 2, 2, 1)$. Next, we give some of the *steps* that the machine can generate. For example, $(s_1, s_2, i_1/o_1, [0, 3), 3, (2, 2, 2, 0))$, represents the transition t_{12} when no timeouts precede it. The input i_1 can be accepted before 3 units of time pass (this is indicated by the interval $[0, 3)$). In addition, o_1 takes $C_1((1, 2, 2, 1)) = 3$ time units to be performed and the new tuple of variables is $Z_1((1, 2, 2, 1)) = (2, 2, 2, 0)$. The second one, $(s_1, s_4, i_1/o_2, [3, 7), 4, (1, 3, 1, 1))$ is built from the timeout transition associated to the state s_1 and the action transition t_{34}. The step represents that if after 3 units of time no input is received, the timeout transition associated with that state will be triggered and the state will change to s_3. After this, the machine can accept the input i_1 before 4 units of time pass, that is, the timeout assigned to the state s_3. So during the time interval $[3, 7)$ if the machine receives an input i_1 it will emit an output o_2 and the state will change to s_4. Similarly,



$$
\begin{array}{l}
t_{12} = (s_1, s_2, i_1, o_1, Q_1, Z_1, C_1) \\
t_{34} = (s_3, s_4, i_2, o_2, Q_2, Z_2, C_2) \\
t_{25} = (s_2, s_5, i_2, o_3, Q_3, Z_3, C_3) \\
t_{45} = (s_4, s_5, i_2, o_1, Q_4, Z_4, C_4) \\
t_{56} = (s_5, s_6, i_1, o_3, Q_5, Z_5, C_5) \\
t_{61} = (s_6, s_1, i_1, o_2, Q_6, Z_6, C_6)
\end{array}
$$

$$TO(s_1) = (s_3, 3),\ TO(s_3) = (s_6, 4),\ TO(s_2) = (s_4, 1)$$

$$Z_i(\bar{x}) = \bar{x} + \begin{cases} (1,0,0,-1) \text{ if } i \in \{1,3,5\} \\ (0,1,-1,0) \text{ if } i \in \{2,4,6\} \end{cases}$$

$$Q_i(\bar{x}) \equiv Z_i(\bar{x}) \geq \bar{0} \ \wedge \ \begin{cases} x_i > 0 \text{ if } i \in \{1,2,3,4\} \\ x_1 > 0 \text{ if } i \in \{5,6\} \end{cases}$$

$$C_i(\bar{x}) = \begin{cases} x_i + 2 & \text{if } i \in \{1,2,3,4\} \ \wedge \ x_i \neq 0 \\ x_1 & \text{if } i \in \{5,6\} \ \wedge \ x_1 \neq 0 \\ 3 & \text{otherwise} \end{cases}$$

Fig. 2. Example of TEFSMs.

we can obtain the step $(s_1, s_1, i_1/o_2, [7,\infty), 1, (1,3,1,1))$, using the timeout transitions corresponding to s_1 and s_3 and the action transition t_{61}. All the steps presented, correspond to the configuration $(s_1, (1,2,2,1))$.

Now, we present an example of a temporal evolution built from two steps, and assuming that $s1$ is the initial state: $([7,\infty)/i_1/1/o_2, [3,7)/i_1/3/o_2)$. The configuration that has been considered for the first step is again $(s_1, (1,2,2,1))$. The configuration that corresponds to the second step is the one obtained after the first step has been performed, that is, $(s_1, (1,3,1,1))$. □

Let us note that different instances of the same evolution may appear in a specification as result of the different configurations obtained after traversing the corresponding TEFSM.

In the following definition we introduce the concept of *instanced evolution*. Intuitively, instanced evolutions are constructed from evolutions by instanciating to a concrete value each timeout, given by an interval, of the evolution.

Definition 3. Let $M = (S, I, O, TO, Tr, s_{in}, \bar{y})$ be a TEFSM and let us consider a *timed evolution* $e = (\hat{t}_1/i_1/t_{o1}/o_1, \ldots, \hat{t}_r/i_r/t_{or}/o_r)$. We say that the tuple $(t_1/i_1/t_{o1}/o_1, \ldots, t_r/i_r/t_{or}/o_r)$ is an *instanced timed evolution of* e if for all $1 \leq j \leq r$ we have $t_j \in \hat{t}_j$. In addition, we say that the tuple $(t_1/i_1/o_1, \ldots, t_r/i_r/o_r)$ is an *instanced functional evolution* of e.

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

By abusing the notation, we will sometimes refer to instanced time evolutions such as $(t_1/i_1/t_{o1}/o_1, \ldots, t_r/i_r/t_{or}/o_r)$ as $(\bar{t}, \sigma, \bar{t}_o)$, where $t = (t_1, \ldots, t_r)$, $\sigma = (i_1/o_1, \ldots, i_r/o_r)$, and $t_o = (t_{o1}, \ldots, t_{or})$. Similarly, we will also refer to instanced functional evolutions as $(\bar{t}, \sigma)$.

Example 2. As example, if we consider the temporal evolution showed previously, $([7,\infty)/i_1/1/o_2, [3,7)/i_1/3/o_2)$, we have that $(8, /i_1/1/o_2, 5/i_1/3/o_2)$ and $(12, /i_1/1/o_2, 3/i_1/3/o_2)$ are *instanced temporal evolutions.* □

3 (Timed) Implementation Relations

In this section we introduce our implementation relations. All of them follow the same pattern: An implementation I *conforms* to a specification S if for all possible evolution of S the outputs that the implementation I may perform after a given input are a subset of those for the specification. This pattern is borrowed from $\mathtt{conf}_{nt}$ [10] and it is inspired in *ioco* [16]. In addition to the non-timed conformance of the implementation, we require some time conditions to hold. For example, we may ask an implementation to be always faster than the time constraints imposed by the specification. Additionaly, we require that the implementation always complies with the timeouts established by the specification.

Next, we formally define the sets of specifications and implementations. A specification is a timed extended finite state machine. Regarding implementations, we consider that they are also given by means of TEFSMs. In this case, we assume, as usual, that all the input actions are always enabled in any state of the implementation. Thus, we can assume that for any input i and any state of the implementation s there always exists a transition $(s, s, i, \mathtt{null}, Q, Z, C)$ where null is a special (empty) output symbol, the predicate $Q(\bar{x})$ is defined as $\neg \bigvee\{Q'(\bar{x}) \mid \exists$ a transition $(s, s', i, o, Q', Z', C')\}$, $Z(\bar{x}) = \bar{x}$, and $C(\bar{x}) = 0$. Let us note that such a transition will be performed when (and only if) no other transition is available for input i (that is, either there are no transitions outgoing from s labelled by i or none of the corresponding predicates hold). Let us note that we do not restrict the machines to be deterministic. Thus, both implementations and specifications may present non-deterministic behavior. This is an important advantage with respect to previous work [12].

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

Definition 4. Let S, I be TEFSMs. We say that I *functionally conforms* to S, denoted by $I\ \mathtt{conf}_f\ S$, if for each functional evolution $e \in \mathtt{FEvol}(S)$, with $e = (\hat{t}_1/i_1/o_1, \ldots, \hat{t}_r/i_r/o_r)$ and $r \geq 1$, we have that for all $t_1 \in \hat{t}_1, \ldots, t_r \in \hat{t}_r$ and o'_r, $e' = (t_1/i_1/o_1, \ldots, t_r/i_r/o'_r) \in \mathtt{InsFEvol}(I)$ implies $e' \in \mathtt{InsFEvol}(S)$. □

The idea underlying the definition of $\mathtt{conf}_f$ is that the implementation does not *invent* anything for those sequences of 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 functional evolution $e \in \mathtt{FEvol}(S)$, with $e = (\hat{t}_{t1}/i_1/o_1, \ldots, \hat{t}_{tr}/i_r/o_r)$ and $r \geq 1$". Next, we introduce our *timed* implementation relations. We will distinguish two classes of conformance relations:

Weak and *strong*. The family of weak conformance relations demands conditions only over the total time associated to timed evolutions of the implementation with respect to the corresponding timed evolutions of the specification. In contrast, strong conformance relations establish requests over the time values corresponding to the performance of each transition separately. For each of these approaches we define five relations. In the $\texttt{conf}_a^s$ and $\texttt{conf}_a^w$ relations (conforms *always*) we consider, for any timed evolution σ of the implementation, that if its associated functional evolution σ' is a functional evolution of the specification then σ is also a timed evolution of the specification. In the $\texttt{conf}_w^s$ and $\texttt{conf}_w^w$ relations (conformance in the *worst* case) the implementation is forced, for each timed evolution fulfilling the previous conditions, to be faster than the slowest instance of the same evolution in the specification. The $\texttt{conf}_b^s$ and $\texttt{conf}_b^w$ relations (conforms in the *best* case) are similar but considering only the fastest instance of the specification. Finally, the relations $\texttt{conf}_{sw}^s$ and $\texttt{conf}_{sw}^w$ (*sometimes worst*), and $\texttt{conf}_{sb}^s$ and $\texttt{conf}_{sb}^w$ (*sometimes best*), are similar to the previous relations, but in each case only *one* instance of each temporal trace of the implementation is required to be as fast as the worst/best instance in the specification.

Definition 5. Let $\bar{t}_o = (t_{o1} \dots t_{or}) \in \texttt{Time}^r$. For all instanced functional evolution $insfevol = (t_1/i_1/o_1, \dots, t_r/i_r/o_r) \in (\texttt{Time} \times I \times O)^r$, we denote by $insfevol \nabla \bar{t}_o$ the instanced timed evolution $(t_1/i_1/t_{o1}/o_1, \dots, t_r/i_r/t_{or}/o_r)$. Let S and I be TEFSMs. The *timed conformance relations* are defined as follows:

- *(strong always)* $I\ \texttt{conf}_a^s\ S$ iff $I\ \texttt{conf}_f\ S$ and for all instanced functional evolution $insfevol \in \texttt{InsFEvol}(I) \cap \texttt{InsFEvol}(S)$ we have that $\forall\ \bar{t}_i$

$$insfevol \nabla \bar{t}_i \in \texttt{InsTEvol}(I) \Longrightarrow insfevol \nabla \bar{t}_i \in \texttt{InsTEvol}(S)$$

- *(strong best)* $I\ \texttt{conf}_b^s\ S$ iff $I\ \texttt{conf}_f\ S$ and for all instanced functional evolution $insfevol \in \texttt{InsFEvol}(I) \cap \texttt{InsFEvol}(S)$ we have that $\forall\ \bar{t}_i$

$$insfevol \nabla \bar{t}_i \in \texttt{InsTEvol}(I) \Longrightarrow \forall\ \bar{t}_s : \begin{pmatrix} insfevol \nabla \bar{t}_s \in \texttt{InsTEvol}(S) \\ \Downarrow \\ \bar{t}_i \leq \bar{t}_s \end{pmatrix}$$

- *(strong worst)* $I\ \texttt{conf}_w^s\ S$ iff $I\ \texttt{conf}_f\ S$ and for all instanced functional evolution $insfevol \in \texttt{InsFEvol}(I) \cap \texttt{InsFEvol}(S)$ we have that $\forall\ \bar{t}_i$

$$insfevol \nabla \bar{t}_i \in \texttt{InsTEvol}(I) \Longrightarrow \exists\ \bar{t}_s : \begin{pmatrix} insfevol \nabla \bar{t}_s \in \texttt{InsTEvol}(S) \\ \wedge \\ \bar{t}_i \leq \bar{t}_s \end{pmatrix}$$

- *(strong sometimes best)* $I\ \texttt{conf}_{sb}^s\ S$ iff $I\ \texttt{conf}_f\ S$ and for all instanced functional evolution $insfevol \in \texttt{InsFEvol}(I) \cap \texttt{InsFEvol}(S)$ we have that $\exists\ \bar{t}_i$ such that

$$insfevol \nabla \bar{t}_i \in \texttt{InsTEvol}(I)\ \wedge\ \forall\ \bar{t}_s : \begin{pmatrix} insfevol \nabla \bar{t}_s \in \texttt{InsTEvol}(S) \\ \Downarrow \\ \bar{t}_i \leq \bar{t}_s \end{pmatrix}$$

– *(strong sometimes worst)* $I \; \texttt{conf}^{s}_{sw} \; S$ iff $I \; \texttt{conf}_f \; S$ and for all instanced functional evolution $insfevol \in \texttt{InsFEvol}(I) \cap \texttt{InsFEvol}(S)$ we have that $\exists \; \bar{t}_i, \; \bar{t}_s$ such that

$$\begin{pmatrix} insfevol \nabla \bar{t}_i \in \texttt{InsTEvol}(I) \\ \wedge \\ insfevol \nabla \bar{t}_s \in \texttt{InsTEvol}(S) \\ \wedge \\ \bar{t}_i \leq \bar{t}_s \end{pmatrix}$$

– *(weak always)* $I \, \texttt{conf}^{w}_{a} \, S$ iff $I \, \texttt{conf}_f \, S$ and for all instanced functional evolution $insfevol \in \texttt{InsFEvol}(I) \cap \texttt{InsFEvol}(S)$ we have that $\forall \; \bar{t}_i$

$$insfevol \nabla \bar{t}_i \in \texttt{InsTEvol}(I) \Longrightarrow \exists \; \bar{t}_s : \begin{pmatrix} insfevol \nabla \bar{t}_s \in \texttt{InsTEvol}(S) \\ \wedge \\ \sum \bar{t}_i = \sum \bar{t}_s \end{pmatrix}$$

– *(weak best)* $I \; \texttt{conf}^{w}_{b} \; S$ iff $I \; \texttt{conf}_f \; S$ and for all instanced functional evolution $insfevol \in \texttt{InsFEvol}(I) \cap \texttt{InsFEvol}(S)$ we have that $\forall \; \bar{t}_i$

$$insfevol \nabla \bar{t}_i \in \texttt{InsTEvol}(I) \Longrightarrow \forall \; \bar{t}_s : \begin{pmatrix} insfevol \nabla \bar{t}_s \in \texttt{InsTEvol}(S) \\ \Downarrow \\ \sum \bar{t}_i \leq \sum \bar{t}_s \end{pmatrix}$$

– *(weak worst)* $I \, \texttt{conf}^{w}_{w} \, S$ iff $I \, \texttt{conf}_f \, S$ and for all instanced functional evolution $insfevol \in \texttt{InsFEvol}(I) \cap \texttt{InsFEvol}(S)$ we have that $\forall \; \bar{t}_i$

$$insfevol \nabla \bar{t}_i \in \texttt{InsTEvol}(I) \Longrightarrow \exists \; \bar{t}_s : \begin{pmatrix} insfevol \nabla \bar{t}_s \in \texttt{InsTEvol}(S) \\ \wedge \\ \sum \bar{t}_i \leq \sum \bar{t}_s \end{pmatrix}$$

– *(weak sometimes best)* $I \, \texttt{conf}^{w}_{sb} \, S$ iff $I \, \texttt{conf}_f \, S$ and for all instanced functional evolution $insfevol \in \texttt{InsFEvol}(I) \cap \texttt{InsFEvol}(S)$ we have that $\exists \; \bar{t}_i$ such that

$$insfevol \nabla \bar{t}_i \in \texttt{InsTEvol}(I) \; \wedge \; \forall \; \bar{t}_s : \begin{pmatrix} insfevol \nabla \bar{t}_s \in \texttt{InsTEvol}(S) \\ \Downarrow \\ \sum \bar{t}_i \leq \sum \bar{t}_s \end{pmatrix}$$

– *(weak sometimes worst)* $I \; \texttt{conf}^{w}_{sw} \; S$ iff $I \; \texttt{conf}_f \; S$ and for all instanced functional evolution $insfevol \in \texttt{InsFEvol}(I) \cap \texttt{InsFEvol}(S)$ we have that $\exists \; \bar{t}_i, \; \bar{t}_s$ such that

$$\begin{pmatrix} insfevol \nabla \bar{t}_i \in \texttt{InsTEvol}(I) \\ \wedge \\ insfevol \nabla \bar{t}_s \in \texttt{InsTEvol}(S) \\ \wedge \\ \sum \bar{t}_i \leq \sum \bar{t}_s \end{pmatrix}$$

□

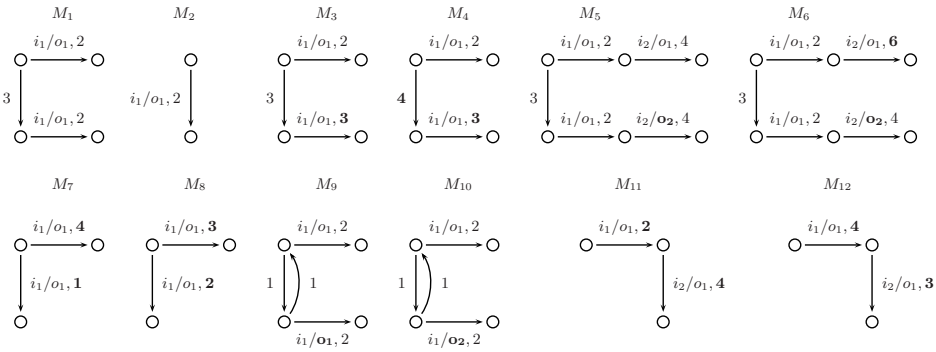


Fig. 3. Example of TEFSMs.

3.1 Illustrating Examples

In this section we show how our implementation relations capture the functional and temporal behavior of systems. In particular, we give some examples where several TEFSMs are related. For the sake of simplicity, we will use some additional conformance binary operators. We will assume that $I\ \mathtt{conf}_*\ S$ denotes that *all* implementation relations given in Definition 5 hold between I and S. If *none* of these relations holds then we denote it by $I\ \mathtt{co\not nf}_*\ S$. Besides, $I\ \mathtt{conf}_\Box\ S$ denotes that all relations but $\mathtt{conf}_a^s$ and $\mathtt{conf}_a^w$ hold. We will consider the TEFSMs depicted in Figure 3. Finally, let us note that if a TEFSM is very similar to the ones presented before, then we stress the differences by using a boldface font.

Equivalent machines. We have $M_1\,\mathtt{conf}_*\,M_2$. Actually, we also have $M_2\,\mathtt{conf}_*\,M_1$. Let us note that the behavior of both machines is exactly the same regardless of whether 3 units of time pass: All transitions available for M_1 after taking a timeout are also available in M_2 from its first state. For similar reasons, we have $M_1\ \mathtt{conf}_*\ M_9$, $M_9\ \mathtt{conf}_*\ M_1$, $M_2\ \mathtt{conf}_*\ M_9$, and $M_9\ \mathtt{conf}_*\ M_2$.

Non-Conformance due to different time values to perform output actions. However, we have $M_3\ \mathtt{co\not nf}_*\ M_2$. Let us note that M_3 may take 3 time units to perform the output o_1 if it receives the input i_1 after 3 time units, $(3/i_1/3/o_1)$, while M_2 only needs 2 time units, $(3/i_1/2/o_1)$. Moreover, in these machines the only way to perform i_1/o_1 after a timeout 3 consists in taking these traces, respectively (the same applies for traces with a timeout higher than 3). Since M_3 is, in *any* case, slower than M_2 for these sequences of inputs/outputs, no conformance relation where M_3 is the IUT and M_2 is the specification holds. However, we have $M_2\,\mathtt{conf}_\Box\,M_3$: Despite M_2 does not take the same time values as M_3 for each sequence, its time is always smaller than (timeouts ≥ 3) or equal to (timeouts < 3) the times of M_3.

Non-conformance due to different timeouts. As we have seen, reducing the time consumed by actions can benefit a TEFSM with respect to another. In spite of the fact that requirements on timeouts are *strict*, sometimes having differ-

ent timeouts can benefit a TEFSM as well. Most traces in M_3 and M_4 take the same times. There is an exception: The trace with timeout 3. In M_3 we have $(3/i_1/3/o_1)$, but in M_4 we have $(3/i_1/2/o_1)$ because after 3 time units pass the state does not change yet in M_4. Hence, we have $M_4 \, \mathtt{conf}_{\Box} \, M_3$ but $M_3 \not\mathtt{conf}_* M_4$.

Non-conformance due to $\mathtt{conf}_f$. Let us consider how the availability of outputs affects the relations. We have $M_5 \not\mathtt{conf}_* M_{11}$. Let us note that if i_2 is offered in M_{11} after executing i_1/o_1 then only o_1 can be produced. However, M_5 can produce this output as well as o_2, which is forbidden by M_{11}. So, we do not have $M_5 \, \mathtt{conf}_f \, M_{11}$, and no temporal relation holds without fulfilling this condition. If M_5 is substituted by M_6 then the same considerations apply: We have $M_6 \not\mathtt{conf}_* M_{11}$. However, we have $M_{11} \, \mathtt{conf}_{\Box} \, M_6$ because all sequences concerned by M_{11} that appear in M_6 (in fact only the sequence $i_1/o_1, i_2/o_1$) are performed faster than or equal to the corresponding trace in M_6 (but we do not have that all are equal). Let us note that $M_9 \not\mathtt{conf}_* M_{10}$ and $M_{10} \not\mathtt{conf}_* M_9$. The reason is that $\mathtt{conf}_f$ does not hold, though, in this case, it does not hold in either direction. Let us note that, after 1 time units passes and the timeout is raised, if i_1 is offered then M_9 must answer o_1, and o_2 is forbidden. However, it is the other way around for M_{10}. Hence, their answers are mutually incompatible.

Non-conformance due to different time requirements. Let us consider a case where the IUTs and specifications can spent different time values in executing pairs of input/outputs included in traces. We consider M_7 and M_8. Since they only perform traces of length 1, any strong relation coincides with its respective weak version. Next we will refer to strong relations. Both M_7 and M_8 can execute i_1/o_1 in a time that cannot be taken in the other, so we do not have $M_7 \, \mathtt{conf}^s_a \, M_8$. The worst time values to execute i_1/o_1 in M_7 and M_8 are 4 and 3, respectively, while the best time values are 1 and 2, respectively. The worst time of M_7 is not better than the worst or the best time in M_8, so we have neither $M_7 \, \mathtt{conf}^s_w \, M_8$ nor $M_7 \, \mathtt{conf}^s_b \, M_8$. However, the best time in M_7 is better to both the worst and the best time of M_8. So, both $M_7 \, \mathtt{conf}^s_{sw} \, M_8$ and $M_7 \, \mathtt{conf}^s_{sb} \, M_8$ hold. On the other hand, the worst time in M_8 is better than the worst of M_7 but not than the best of M_7. Hence, $M_8 \, \mathtt{conf}^s_w \, M_7$ holds but $M_8 \, \mathtt{conf}^s_b \, M_7$ does not. Finally, the best time in M_8 is better than the worst of M_7, but not better than its best one. Thus, $M_8 \, \mathtt{conf}^s_{sw} \, M_7$ holds, but $M_8 \, \mathtt{conf}^s_{sb} \, M_7$ does not.

Differences between weak and strong. Next we show how temporal requirements are dealt by strong and weak relations. Let us consider M_{11} and M_{12}. No strong relation holds between these TEFSMs in any direction. The reason is that M_{11} performs, i_1/o_1, faster than M_{12}, but M_{12} performs the next transition i_2/o_1 faster than M_{11}. The result is that none of these machines is always at least as fast as the other (concerning transitions). However, if we consider traces (i.e., weak relations) then some relations arise. Let us note that M_{11} performs both available sequences of inputs/outputs (i_1/o_1 and $i_1/o_1, i_2/o_1$) faster than M_{12}: In M_{11} they spend 2 and 6 time units, respectively, while these time values are 4 and 7, respectively, in M_{12}. So, all *weak* relations (but $\mathtt{conf}^w_a$) hold: We have $M_{11} \, \mathtt{conf}^w_w \, M_{12}$, $M_{11} \, \mathtt{conf}^w_b \, M_{12}$, $M_{11} \, \mathtt{conf}^w_{sw} \, M_{12}$, and $M_{11} \, \mathtt{conf}^w_{sb} \, M_{12}$. None of them holds if we exchange the roles of both machines.

3.2 Relating Conformance Relations

Theorem 1. The relations given in Definition 5 are related as follows:

$$\begin{array}{ccccc} & & I\ \mathtt{conf}^w_{sw}\ S & \Leftarrow & I\ \mathtt{conf}^w_{sb}\ S \\ & & \Uparrow & & \Uparrow \\ I\ \mathtt{conf}^w_{a}\ S & \Rightarrow & I\ \mathtt{conf}^w_{w}\ S & \Leftarrow & I\ \mathtt{conf}^w_{b}\ S \\ \Uparrow & & \Uparrow & & \Uparrow \\ I\ \mathtt{conf}^s_{a}\ S & \Rightarrow & I\ \mathtt{conf}^s_{w}\ S & \Leftarrow & I\ \mathtt{conf}^s_{b}\ S \\ & & \Downarrow & & \Downarrow \\ & & I\ \mathtt{conf}^s_{sw}\ S & \Leftarrow & I\ \mathtt{conf}^s_{sb}\ S \end{array}$$

Besides, we have $I\ \mathtt{conf}^s_{sw}\ S \Rightarrow I\ \mathtt{conf}^w_{sw}\ S$ and $I\ \mathtt{conf}^s_{sb}\ S \Rightarrow I\ \mathtt{conf}^w_{sb}\ S$. □

Let us remark that the implications inferred in the previous result are, obviously, transitive. For instance, we also have $I\ \mathtt{conf}^w_{a}\ S \Rightarrow I\ \mathtt{conf}^w_{sw}\ S$.

It is interesting to note that if specifications are restricted to take always the same time for each given evolution (independently from the possible derivation taken for such evolution) then, on the one hand, the relations $\mathtt{conf}^w_b$ and $\mathtt{conf}^w_w$ would coincide while, on the other hand, $\mathtt{conf}^s_b$ and $\mathtt{conf}^s_w$ also coincide. However these relations would be still different from the $\mathtt{conf}^w_a$ and $\mathtt{conf}^s_a$ relations. Similarly, if this property holds in implementations then all relations concerning the best temporal traces of the implementation (*sometimes* relations) coincide with the corresponding relation where all the temporal traces of the implementation are regarded.

Lemma 1. Let us consider two TEFSMs $I = (S_I, I_I, O_I, TO_I, Tr_I, s_{in_I}, \bar{y}_I)$ and $S = (S_S, I_S, O_S, TO_S, Tr_S, s_{in_S}, \bar{y}_S)$. If there do not exist different transitions $(s, s', i, o, Q, Z, C), (s, s'', i, o, Q', Z', C') \in Tr_I$ then

$$\begin{array}{ll} I\ \mathtt{conf}^s_b\ S \Leftrightarrow I\ \mathtt{conf}^s_{sb}\ S & I\ \mathtt{conf}^s_w\ S \Leftrightarrow I\ \mathtt{conf}^s_{sw}\ S \\ I\ \mathtt{conf}^w_b\ S \Leftrightarrow I\ \mathtt{conf}^w_{sb}\ S & I\ \mathtt{conf}^w_w\ S \Leftrightarrow I\ \mathtt{conf}^w_{sw}\ S \end{array}$$

If there do not exist different transitions $(s, s', i, o, Q, Z, C), (s, s'', i, o, Q', Z', C') \in Tr_S$ then

$$\begin{array}{ll} I\ \mathtt{conf}^s_b\ S \Leftrightarrow I\ \mathtt{conf}^s_w\ S & I\ \mathtt{conf}^s_{sw}\ S \Leftrightarrow I\ \mathtt{conf}^s_{sb}\ S \\ I\ \mathtt{conf}^w_b\ S \Leftrightarrow I\ \mathtt{conf}^w_w\ S & I\ \mathtt{conf}^w_{sw}\ S \Leftrightarrow I\ \mathtt{conf}^w_{sb}\ S \end{array}$$

□

The hierarchy of relations induced in Theorem 1 allows to compare implementations in the following way: I_1 is *preferable* to I_2 to implement S if it meets with S a relation that is *stricter* according to this hierarchy.

Definition 6. Let I_1, I_2 and S be TEFSMs and $\mathtt{conf}_x$ and $\mathtt{conf}_y$ be timed conformance relations such that $I_1\ \mathtt{conf}_x\ S$, $I_2\ \mathtt{conf}_y\ S$, $\mathtt{conf}_x \Rightarrow \mathtt{conf}_y$, and $\mathtt{conf}_y \not\Rightarrow \mathtt{conf}_x$. We say that I_1 is *preferred* to I_2 to implement S and we denote it by $I_1 >_S I_2$. □

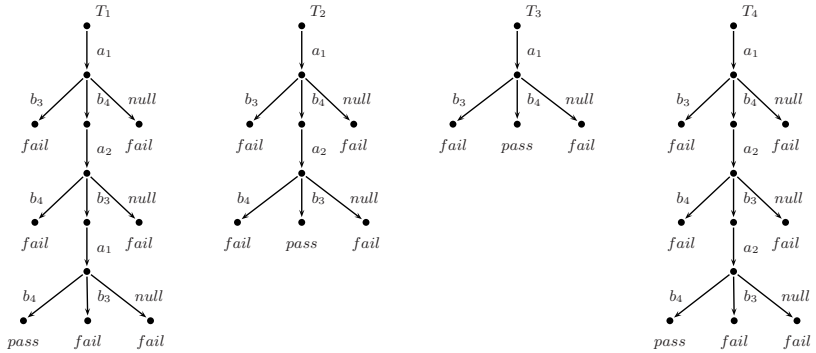


Fig. 4. Examples of Tests.

4 Definition and Application of Tests

We consider that tests represent sequences of inputs applied to an IUT. Once an output is received, the tester checks whether it belongs to the set of expected ones 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 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 addition to check the functional behavior of the IUT, test have also to detect whether wrong timed behaviors appear. Thus, tests have to include capabilities to deal with the two ways of specifying time. On the one hand, we will include *time stamps* to record the time that each sequence of output actions takes to be executed. The time values recorded from the IUT while applying the test will be compared with the ones expected by the specification. Each time stamp will contain a set of *time sequences* corresponding to the time values that the specification establishes for each transition of a trace. Since we do not restrict non-deterministic behavior, we will have as many time sequences as possible timed evolutions can exist for a trace. Moreover, depending on the number of inputs applied so far, we will have different lengths for the associated time sequences in the time stamps of the test. On the other hand, tests will include *delays* before offering input actions. The idea is that delays in tests will induce timeouts in IUTs. Thus, we may indirectly check whether the timeouts imposed by the specification are reflected in the IUT by offering input actions after a specific delay. Let us note that a tester can not observe when the IUT takes a timeout. However, she can check the IUT behavior after different delays.

Definition 7. A *test* is a tuple $T = (S, I, O, Tr, s_0, S_I, S_O, S_F, S_P, C, W)$ where S is the set of states, I and O are disjoint sets of input and output actions,

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

Finally, we have two timed functions. $C : S_P \longrightarrow \bigcup_{j=1}^{\infty} \mathcal{P}(\texttt{Time}^j)$ is a function associating time stamps with passing states. $W : S_I \longrightarrow \texttt{Time}$ is a function associating delays with input states.

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

We say that a test T is an *instance* of the test T' if they only differ in the associated timed functions C and W.

Let $\sigma = i_1/o_1, \ldots, i_r/o_r$. We write $T \stackrel{\sigma}{\Longrightarrow} s$ 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 $\{(s_0, i_1, s_{12}), (s_{r2}, o_r, s)\} \subseteq Tr$, for all $2 \leq j \leq r$ we have $(s_{j1}, i_j, s_{j2}) \in Tr$, and for all $1 \leq j \leq r-1$ we have $(s_{j2}, o_j, s_{(j+1)1}) \in Tr$.

Let T be a test, $\sigma = i_1/o_1, \ldots, i_r/o_r$, s^T be a state of T, and $\overline{t}$, $\bar{t_o} \in \texttt{Time}^r$. We write $T \stackrel{\sigma}{\Longrightarrow_{\overline{t}}} s^T$ if $T \stackrel{\sigma}{\Longrightarrow} s^T$, $t_1 = D(s_0)$ and for all $1 < j \leq r$ we have $t_j = D(s_{j1})$. □

Let us remark that $T \stackrel{\sigma}{\Longrightarrow} s^T$, and its variant $T \stackrel{\sigma}{\Longrightarrow_{\overline{t}}} s^T$, imply that s is a terminal state. Next we define the application of a test suite to an implementation. We say that the test suite $\mathcal{T}$ is *passed* if for all test the terminal states reached by the composition of implementation and test are *pass* states. Besides, we give different timing conditions in a similar way to what we did for implementation relations.

Definition 8. Let I be a `TEFSM`, T be a valid test, $\sigma = i_1/o_1, \ldots, i_r/o_r$, s^T be a state of T, $\overline{t} = (t_1, \ldots, t_r)$, and $\bar{t}_o = (t_{o1}, \ldots, t_{or})$. We write $I \parallel T \stackrel{\sigma}{\Longrightarrow_{\overline{t}}} s^T$ if $T \stackrel{\sigma}{\Longrightarrow_{\overline{t}}} s^T$ and $(\bar{t}, \sigma) \in \texttt{InsFEvol}(I)$. We write $I \parallel T \stackrel{\sigma}{\Longrightarrow_{\overline{t},\bar{t_o}}} s^T$ if $I \parallel T \stackrel{\sigma}{\Longrightarrow_{\overline{t}}} s^T$ and $(\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$. Let $e = (\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$. We define the set $\texttt{Test}(e, \mathcal{T}) = \{T \mid T \in \mathcal{T} \ \wedge \ I \parallel T \stackrel{\sigma}{\Longrightarrow_{\overline{t},\bar{t_o}}} s^T\}$.

We say that I *passes* the set of valid tests $\mathcal{T}$, denoted by $\texttt{pass}(I, \mathcal{T})$, if for all test $T \in \mathcal{T}$ there do not exist σ, s^T, $\overline{t}$ such that $I \parallel T \stackrel{\sigma}{\Longrightarrow_{\overline{t}}} s^T$ and $s^T \in S_F$.

We say that I *strongly passes* the set of valid tests $\mathcal{T}$ *for any time* if $\texttt{pass}(I, \mathcal{T})$ and for all $e = (\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$ we have that for all $T \in \texttt{Test}(e, \mathcal{T})$ such that $I \parallel T \stackrel{\sigma}{\Longrightarrow_{\overline{t},\bar{t_o}}} s^T$ it holds $\bar{t}_o \in C(s^T)$.

We say that I *strongly passes* the set of valid tests $\mathcal{T}$ *in the best time* if $\texttt{pass}(I, \mathcal{T})$ and for all $e = (\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$ we have that for all $T \in \texttt{Test}(e, \mathcal{T})$ such that $I \parallel T \stackrel{\sigma}{\Longrightarrow}_{\bar{t},\bar{t}_o} s^T$, for all $\bar{t}_c \in C(s^T)$ it holds $\bar{t}_o \leq \bar{t}_c$.

We say that I *strongly passes* the set of valid tests $\mathcal{T}$ *in the worst time* if $\texttt{pass}(I, \mathcal{T})$ and for all $e = (\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$ we have that for all $T \in \texttt{Test}(e, \mathcal{T})$ such that $I \parallel T \stackrel{\sigma}{\Longrightarrow}_{\bar{t},\bar{t}_o} s^T$ there exists $\bar{t}_c \in C(s^T)$ such that $\bar{t}_o \leq \bar{t}_c$.

We say that I *strongly passes* the set of valid tests $\mathcal{T}$ *sometimes in best time* if $\texttt{pass}(I, \mathcal{T})$ and there exists $e = (\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$ such that for all $T \in \texttt{Test}(e, \mathcal{T})$ with $I \parallel T \stackrel{\sigma}{\Longrightarrow}_{\bar{t},\bar{t}_o} s^T$ we have that for all $\bar{t}_c \in C(s^T)$ it holds $\bar{t}_o \leq \bar{t}_c$.

We say that I *strongly passes* the set of valid tests $\mathcal{T}$ *sometimes in worst time* if $\texttt{pass}(I, \mathcal{T})$ and there exists $e = (\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$ such that for all $T \in \texttt{Test}(e, \mathcal{T})$ with $I \parallel T \stackrel{\sigma}{\Longrightarrow}_{\bar{t},\bar{t}_o} s^T$ we have that there exists $\bar{t}_c \in C(s^T)$ such that $\bar{t}_o \leq \bar{t}_c$.

We say that I *weakly passes* the set of valid tests $\mathcal{T}$ *for any time* if $\texttt{pass}(I, \mathcal{T})$ and for all $e = (\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$ we have that for all $T \in \texttt{Test}(e, \mathcal{T})$ such that $I \parallel T \stackrel{\sigma}{\Longrightarrow}_{\bar{t},\bar{t}_o} s^T$ it holds $\sum \bar{t}_o = \sum \bar{t}_c$ for some $\bar{t}_c \in C(s^T)$.

We say that I *weakly passes* the set of valid tests $\mathcal{T}$ *in the best time* if $\texttt{pass}(I, \mathcal{T})$ and for all $e = (\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$ we have that for all $T \in \texttt{Test}(e, \mathcal{T})$ such that $I \parallel T \stackrel{\sigma}{\Longrightarrow}_{\bar{t},\bar{t}_o} s^T$, for all $\bar{t}_c \in C(s^T)$ it holds $\sum \bar{t}_o \leq \sum \bar{t}_c$.

We say that I *weakly passes* the set of valid tests $\mathcal{T}$ *in the worst time* if $\texttt{pass}(I, \mathcal{T})$ and for all $e = (\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$ we have that for all $T \in \texttt{Test}(e, \mathcal{T})$ such that $I \parallel T \stackrel{\sigma}{\Longrightarrow}_{\bar{t},\bar{t}_o} s^T$ there exists $\bar{t}_c \in C(s^T)$ such that $\sum \bar{t}_o \leq \sum \bar{t}_c$.

We say that I *weakly passes* the set of valid tests $\mathcal{T}$ *sometimes in best time* if $\texttt{pass}(I, \mathcal{T})$ and there exists $e = (\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$ such that for all $T \in \texttt{Test}(e, \mathcal{T})$ with $I \parallel T \stackrel{\sigma}{\Longrightarrow}_{\bar{t},\bar{t}_o} s^T$ we have that for all $\bar{t}_c \in C(s^T)$ it holds $\sum \bar{t}_o \leq \sum \bar{t}_c$.

We say that I *weakly passes* the set of valid tests $\mathcal{T}$ *sometimes in worst time* if $\texttt{pass}(I, \mathcal{T})$ and there exists $e = (\bar{t}, \sigma, \bar{t}_o) \in \texttt{InsTEvol}(I)$ such that and for all $T \in \texttt{Test}(e, \mathcal{T})$ with $I \parallel T \stackrel{\sigma}{\Longrightarrow}_{\bar{t},\bar{t}_o} s^T$ we have that there exists $\bar{t}_c \in C(s^T)$ such that $\sum \bar{t}_o \leq \sum \bar{t}_c$. □

5 Conclusions and Future Work

In this paper we have introduced a new model to specify timed systems. In contrast with most approaches, our formalism allows to define time in two different ways: Duration of actions and timeouts of the system. Thus, by separating these two notions, it is easier to specify temporal properties of systems than if we use a formalism where only one of the possibilities is available. We have also developed a testing theory. On the one hand, we have defined ten conformance relations that take into account the influence of non-determinism in the behavior of systems. On the other hand, we have introduced a notion of test. In order to

capture the timed behavior of the IUT, test can both delay the execution of the IUT and record the time that it took to perform a given action.

In terms of future work, we would like to take this paper as a first step, together with [11], to define a testing theory for systems presenting stochastic time together with timeouts.

References

1. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
2. 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.
3. E. Brinksma and J. Tretmans. Testing transition systems: An annotated bibliography. In *4th Summer School, MOVEP 2000, LNCS 2067*, pages 187–195. Springer, 2001.
4. 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.
5. K. El-Fakih, N. Yevtushenko, and G. von Bochmann. FSM-based incremental conformance testing methods. *IEEE Transactions on Software Engineering*, 30(7):425–436, 2004.
6. A. En-Nouaary and R. Dssouli. A guided method for testing timed input output automata. In *TestCom 2003, LNCS 2644*, pages 211–225. Springer, 2003.
7. M.A. Fecko, M.Ü. Uyar, A.Y. Duale, and P.D. Amer. A technique to generate feasible tests for communications systems with multiple timers. *IEEE/ACM Transactions on Networking*, 11(5):796–809, 2003.
8. 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.
9. D. Lee and M. Yannakakis. Principles and methods of testing finite state machines: A survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.
10. M. Núñez and I. Rodríguez. Encoding `PAMR` into (timed) `EFSMs`. In *FORTE 2002, LNCS 2529*, pages 1–16. Springer, 2002.
11. M. Núñez and I. Rodríguez. Towards testing stochastic timed systems. In *FORTE 2003, LNCS 2767*, pages 335–350. Springer, 2003.
12. M. Núñez and I. Rodríguez. Conformance testing relations for timed systems. In *5th Int. Workshop on Formal Approaches to Software Testing (FATES 2005), LNCS 3997*, pages 103–117. Springer, 2006.
13. J.C. Park and R.E. Miller. Synthesizing protocol specifications from service specifications in timed extended finite state machines. In *17th IEEE Int. Conf. on Distributed Computing Systems, ICDCS'97*, pages 253–260. IEEE Computer Society, 1997.
14. A. Petrenko. Fault model-driven test derivation from finite state models: Annotated bibliography. In *4th Summer School, MOVEP 2000, LNCS 2067*, pages 196–205. Springer, 2001.
15. J. Springintveld, F. Vaandrager, and P.R. D'Argenio. Testing timed automata. *Theoretical Computer Science*, 254(1-2):225–257, 2001.
16. J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software – Concepts and Tools*, 17(3):103–120, 1996.