A Simplified Approach for Testing Real-Time Systems Based on Action Refinement

Saddek Bensalem*, Moez Krichen*,
Lotfi Majdoub**, Riadh Robbana**,
Stavros Tripakis***

* Verimag Laboratory, Centre Equation 2, avenue de Vignate, 38610, Gières, France.
saddek.bensalem@imag.fr, moez.krichen@imag.fr.
** LIP2 Laboratory and Polytechnic School of Tunisia.
lotfi.majdoub@ensi.rnu.tn, riadh.robbana@fst.rnu.tn.
*** Verimag Laboratory and Cadence Berkeley Labs, 1995 University avenue,
Suite 460, Berkeley, CA 94704, USA.
tripakis@cadence.com.

Abstract. We propose a new method for generating digital-clock tests for real-time systems. That is, tests which can observe time with only a finite precision. Our goal from testing is to check the conformance of a given implementation with respect to a given specification (the model). The method is based on the so-called action refinement techniques. The main benefit of the method is to save memory space needed to build and to store tests. One important contribution of this work is a simplified way for both modelling and testing real-time systems. We first write a (high-level) simplified version of the model of the system, as an input-output transition system (IOTS) and then we refine it into a more detailed (low-level) model as a timed input-output transition system (TIOTS). This same mechanism applies to the test generation procedure.

1 Introduction

Action refinement techniques are well experimented techniques in the field of hierarchical design of wide classes of systems. They mainly consist in translating high-level actions into lower-level ones. That is to move from a high-level abstraction to a lower one until reaching the implementation level.

Applying action refinement techniques in the field of testing is quite promising. Current techniques typically suffer from state explosion problems: this includes test generation, storage and execution. The problem is more dramatic in the case of timed systems, where an extra level of complexity is introduced by handling time-measuring variables (clocks).

Our main goal in this paper is to reduce the size of generated tests. Our focus is on digital-clock test generation for real-time systems (Krichen and Tripakis, 2004, 2005) and our objective is to improve our previous method to generate such tests. To achieve this, we use an approach based on action refinement. In our timed setting, untimed actions are refined into timed actions. This helps reduce redundancy and results in optimized storage of useful data.

1.1 Overview of our approach

As already mentioned, we are interested in testing real-time systems. For this goal, we adopt the following approach. We assume that the model of the system we want to test is given as an untimed graph. The latter describes the behavior of the considered system using high-level actions. More precisely this untimed graph is, in our case, an *input-output transition system* (IOTS). For instance, an example of an IOTS is the following

$$\rightarrow q \xrightarrow{\mathsf{double?}} q' \xrightarrow{\mathsf{bright!}} q''.$$

That is the (*high-level*) model of a lighting device: "q" is the *initial state*; "double" is a high-level *input-action*; and "bright" a high-level *output-action*. It says that after receiving input "double" the lighting device will emit output "bright".

We then assume that each high-level action is refined into a *timed path*. For instance, the actions "double" and "bright" are refined, respectively, into

$$q \xrightarrow[[0,\infty]]{\mathsf{touch}_r?} p \xrightarrow[[0,1]]{\mathsf{touch}_r?} s \xrightarrow[[1,1]]{\tau_r} q' \text{ and } q' \xrightarrow[[2,3]]{\mathsf{bright}_r!} q'',$$

where touch_r is a *low-level input-action*, bright_r a *low-level output-action* and τ_r is an unobservable low-level action. That means that input double happens if two consecutive touch_r's occur within one time-unit interval. The transition $s \xrightarrow[[1,1]]{\tau_r} q'$ means that no (observable) action should happen during the one time-unit following the occurrence of the second touch_r. The interval $[0, \infty]$ means that there is no constraint on the timing of occurrence of the first touch_r. Similarly, the refinement of bright means that a time elapse between 2 and 3 time-units is needed before moving to the desired state.

Thus, the IOTS above is transformed into the following *timed input-output transition system* (TIOTS)

$$\rightarrow q \xrightarrow[[0,\infty]]{\mathsf{touch}_r?} p \xrightarrow[[0,1]]{\mathsf{touch}_r?} s \xrightarrow[[1,1]]{\tau_r} q' \xrightarrow[[2,3]]{\mathsf{bright}_r!} q''.$$

We define both *analog-clock* and *digital-clock semantics* of a given TIOTS. Analog-clock means that we observe time with an infinite-precision and digital-clock means that time is observed with only a finite-precision. Given an IOTS, the next step consists in using the algorithm of (Tretmans, 1999) to generate (untimed) tests. The generated tests are given as trees. Other refinement techniques are then used to transform these test trees into digital-timed test trees.

The remaining part of the paper is structured as follows. Section 2 recalls the untimed testing framework of (Tretmans, 1999). Section 3 defines the TIOTS model. Section 4 describes the timed testing framework. The action-refinement rules to transform an IOTS into a TIOTS are given is Section 5. The method for deriving refined digital-timed tests from untimed tests is described in Section 6. Section 7 gives an estimation of the memory saving we achieve by the refinement techniques we propose. Finally, Section 8 concludes the paper and gives directions for future-work.

1.2 Related Work

To our knowledge, only few works have attempted to investigate the link between testing and action refinement techniques. In (van der Bijl et al., 2005), the authors present an approach

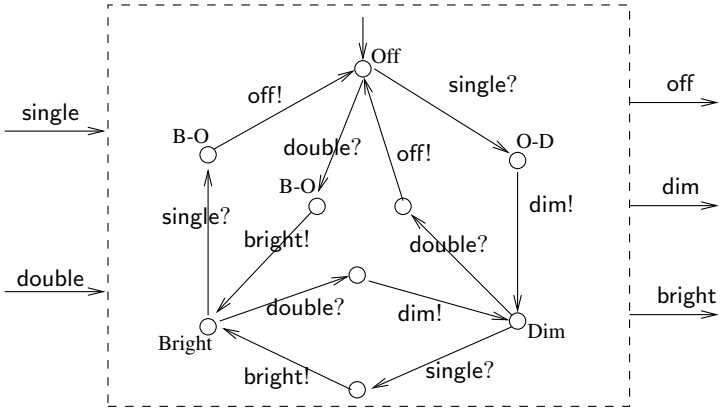


Fig. 1 – An example of an IOTS: a lighting device.

to automatically obtain (untimed) test cases at some required level of detail by a particular type of action refinement, so-called *atomic linear input-inputs refinement*.

2 Model-Based Untimed Conformance Testing

Definition 1 *An input-output transition system is a 5-tuple* $(Q, \mathsf{In}, \mathsf{Out}, T, q_0)$ *where:* Q *is a nonempty countable set of states.* In *is a countable set of input labels.* Out *is a countable set of output labels (such that* $\mathsf{In} \cap \mathsf{Out} = \emptyset$*).* $T \subseteq Q \times (\mathsf{In} \cup \mathsf{Out}) \times Q$ *the transition relation.* q_0 *the initial state.*

A triple $(q, \mu, q') \in T$ is denoted $q \xrightarrow{\mu} q'$. We write $q \xrightarrow{\mu}$ if $\exists q' \in Q : q \xrightarrow{\mu} q'$. An IOTS is said to be *input-complete* if $\forall q \in Q, \forall \lambda \in \mathsf{In} : q \xrightarrow{\lambda}$. A state is said to be *quiescent* if no output action is possible from it. A new output label δ is considered. For each quiescent state q, a self-loop $q \xrightarrow{\delta} q$ is added to the considered IOTS. Next, we consider only IOTS for which this operation is already achieved. Let $\mathsf{Out}_\delta = \mathsf{Out} \cup \{\delta\}$ and $L_\delta = \mathsf{In} \cup \mathsf{Out}_\delta$. For $\sigma = \mu_1 \cdots \mu_n \in L_\delta^+$, we write $q \xrightarrow{\sigma} q'$ if $\exists\, q_1, q_2, \cdots, q_{n-1} \in Q : q \xrightarrow{\mu_1} q_1 \xrightarrow{\mu_2} q_2 \cdots q_{n-1} \xrightarrow{\mu_n} q'$ and write $q \xrightarrow{\sigma}$ if $\exists\, q' \in Q : q \xrightarrow{\sigma} q'$. The sequence of transitions $q \xrightarrow{\mu_1} q_1 \xrightarrow{\mu_2} q_2 \cdots q_{n-1} \xrightarrow{\mu_n} q'$ is called a *path* of the IOTS. An IOTS is said to be *deterministic* if: $\forall q, q', q'' \in Q, \forall \mu \in L_\delta : q \xrightarrow{\mu} q' \wedge q \xrightarrow{\mu} q'' \Rightarrow q' = q''$. An example of an IOTS is given in Figure 1. It is a modification of the case study presented in (Krichen and Tripakis, 2004).

For $q \in Q, P \subseteq Q$ and $\sigma \in L_\delta^+$: $q \,\mathsf{after}\, \sigma =_{df} \{q' \mid q \xrightarrow{\sigma} q'\}$, $\mathsf{out}(q) =_{df} \{\mu \in \mathsf{Out}_\delta \mid q \xrightarrow{\mu}\}$, $\mathsf{STraces}(q) =_{df} \{\sigma \in L_\delta^+ \mid q \xrightarrow{\sigma}\}$, $\mathsf{out}(P) =_{df} \bigcup_{q \in P} \mathsf{out}(q)$. Let $Spec = (Q, \mathsf{In}, \mathsf{Out}, T, q_0)$ and $Imp = (Q', \mathsf{In}, \mathsf{Out}, T', q_0')$ be two IOTS. $Imp \,\mathsf{ioco}\, Spec =_{df} \forall \sigma \in \mathsf{STraces}(q_0) : \mathsf{out}(q_0' \,\mathsf{after}\, \sigma) \subseteq \mathsf{out}(q_0 \,\mathsf{after}\, \sigma)$.

Untimed tests can be represented as either total functions or IOTS. A possible test for the lighting-device example is given in Figure 2.

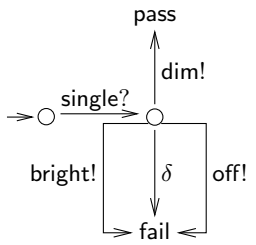


Fig. 2 – An untimed test represented as an IOTS.

The *execution of the test* T on the implementation Imp can be defined as the *parallel composition* of the IOTS defined by T and Imp, with the usual *synchronization* rules for transitions carrying the same label. We say that Imp *passes* the test, denoted Imp passes T, if state fail is not reachable in the product $Imp \| T$. We say that an implementation passes (resp. fails) a set of tests $\mathcal{T}$ if it passes all tests (resp. fails at least one test) in $\mathcal{T}$. We say that $\mathcal{T}$ is *sound* with respect to $Spec$ if $\forall Imp \ : \ Imp$ ioco $Spec \Rightarrow Imp$ passes $\mathcal{T}$.

3 Timed Input-Output Transition Systems

Let R be the set of non-negative reals and N the set of non-negative integers. For $t \in \mathsf{R}$ and $j \in \mathsf{N}$, we write $j = \lfloor t \rfloor$ if $t \in [j, j+1)$. A *timed sequence* over the set of actions $L = \mathsf{In} \cup \mathsf{Out}$ is a sequence $\rho = (\mu_0, t_0)(\mu_1, t_1) \cdots (\mu_n, t_n)$ where $\mu_i \in L$ and $t_i \in \mathsf{R}$ and such that $\forall i \ : \ t_i \leq t_{i+1}$. The sequence ρ is observed if each μ_i occurs at time t_i. If we use a periodic digital-clock with a stepwidth equal to 1 then ρ will be observed as $[\rho] = (\mu_0, j_0)(\mu_1, j_1) \cdots (\mu_n, j_n)$ such that $\forall i \ : \ j_i = \lfloor t_i \rfloor$. For instance, if $\rho = (\mu, 0.3)(\nu, 5.6)(\nu, 8.4)$ then $[\rho] = (\mu, 0)(\nu, 5)(\nu, 8)$. ρ and $[\rho]$ are an *analog-timed* and a *digital-timed sequence*, respectively. We will write $(\mu, 0_{tick})(\nu, 5_{tick})(\nu, 8_{tick})$ instead of $(\mu, 0)(\nu, 5)(\nu, 8)$ to avoid confusion.

Definition 2 *A timed input-output transition system is a 7-tuple* $(Q, \mathsf{In}, \mathsf{Out}, T, q_0, l, u)$ *where:* $(Q, \mathsf{In}, \mathsf{Out}, T, q_0)$ *is an IOTS.* $l \in \mathsf{N}^T$ *is the minimal-delay function.* $u \in (\mathsf{N} \cup \{\infty\})^T$ *is the maximal-delay function such that* $\forall \tau \in T \ : \ l(\tau) \leq u(\tau)$. *By convention we assume that* $\forall n \in \mathsf{N} : \ n \leq \infty$.

Each TIOTS defines a set of accepted timed sequences. The analog-timed trace $(\mu_1, t_1) \cdots (\mu_n, t_n)$ is *accepted* by $S = (Q, \mathsf{In}, \mathsf{Out}, T, q_0, l, u)$ if there exists a path $q_{i_0} \stackrel{\mu_1}{\rightarrow} q_{i_1} \cdots q_{i_{n-1}} \stackrel{\mu_n}{\rightarrow} q_{i_n}$ in S such that $q_{i_0} = q_0$ and $\forall j = 1, \cdots, n : \ l(\tau) \leq t_j - t_{j-1} \leq u(\tau)$ where $\tau = q_{i_{j-1}} \stackrel{\mu_j}{\rightarrow} q_{i_j}$ and $t_0 = 0$.

The digital-timed sequence ρ_{tick} is said to be *accepted* by the TIOTS S if there exists an analog-timed sequence ρ accepted by S such that $\rho_{tick} = [\rho]$. For each state q of S, the duration of authorized stay at q must not exceed the maximum duration: $\mathsf{max_d}(q) = \text{Max}\,\{u(\tau) \,|\, \tau = (q, \mu, q') \in T\}$ where "Max" is the maximum operator.

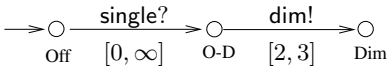


Fig. 3 – An example of a TIOTS.

Let cl be a variable ranging over R which measures the time elapsed since the system S has reached its current state. Variable cl is reset as soon as a new state is visited. We say that S is occupying the *global state* $\langle q, \mathsf{cl}\rangle$. Clearly for each global state $\langle q, \mathsf{cl}\rangle$, we have $\mathsf{cl} \leq \mathsf{max_d}(q)$.

We define two types of *valid transitions* between global states: (1) *Timed transitions*: for $d \in \mathsf{R}$, if $\mathsf{cl} + d \leq \mathsf{max_d}(q)$ then $\langle q, \mathsf{cl}\rangle \xrightarrow{d} \langle q, \mathsf{cl} + d\rangle$. (2) *Discrete transitions*: for $\mu \in L$ and $\tau = (q, \mu, q') \in T$, if $l(\tau) \leq \mathsf{cl} \leq u(\tau)$ then $\langle q, \mathsf{cl}\rangle \xrightarrow{\mu} \langle q', 0\rangle$. An example of a TIOTS is given in Figure 3. That is the timed version of one possible path of the IOTS example shown in Figure 1.

4 Digital-Clock Testing

An *analog-timed trace* is an element of $(L \cup \mathsf{R})^*$ and a *digital-timed trace* an element of $(L \cup \mathsf{N})^*$. Clearly a unique analog-timed trace and a unique digital-timed trace correspond to each analog-timed sequence. For instance for $\rho = (\mu, 0.3)(\nu, 5.6)(\nu, 8.4)$, the corresponding analog and digital traces are $\sigma = 0.3\,\mu\,5.3\,\nu\,2.8\,\nu$ and $\sigma_{tick} = 0_{tick}\,\mu\,5_{tick}\,\nu\,3_{tick}\,\nu$, respectively.

We extend the operator "$[\,\cdot\,]$" to the case of timed traces. For instance, $2_{tick}\,\mu\,1_{tick}\,\nu\,1_{tick} = [2.1\,\mu\,1.2\,\nu\,0.8]$. The analog-timed trace $\sigma = t_0\,\mu_1\,t_1 \cdots \mu_n\,t_n$ is said to be a *valid analog-timed trace* of the TIOTS $S = (Q, \mathsf{In}, \mathsf{Out}, T, q_0, l, u)$ if there exist $q_{i_0}, \cdots, q_{i_n} \in Q$ such that $q_{i_0} = q_0$ and $\langle q_{i_0}, 0\rangle \xrightarrow{t_0} \langle q_{i_0}, t_0\rangle \xrightarrow{\mu_1} \langle q_{i_1}, 0\rangle \xrightarrow{t_1} \langle q_{i_1}, t_1\rangle \cdots \xrightarrow{t_n} \langle q_{i_n}, t_n\rangle$ is a sequence of valid transitions of S. In this case, we will use the following notation $\langle q_{i_0}, 0\rangle \xrightarrow{\sigma} \langle q_{i_n}, t_n\rangle$ and $\langle q_{i_0}, 0\rangle \xrightarrow{\sigma}$. The digital-timed trace σ_{tick} is said to be a *valid digital-timed trace* of S if there exists a valid analog-timed trace σ of S such that $\sigma_{tick} = [\sigma]$. For $q \in Q, \mathsf{cl} \in [0, \mathsf{max_d}(q)], \mathcal{P}$ a set of global states and $\sigma \in (\mathsf{R} \cdot L)^* \cdot \mathsf{R}$: $\langle q, \mathsf{cl}\rangle\ \mathsf{after}\ \sigma =_{df} \{\langle q', \mathsf{cl}'\rangle \mid \langle q, \mathsf{cl}\rangle \xrightarrow{\sigma} \langle q', \mathsf{cl}'\rangle\}$, $\mathsf{out}(\langle q, \mathsf{cl}\rangle) =_{df} \{\mu \in \mathsf{Out} \mid \langle q, \mathsf{cl}\rangle \xrightarrow{\mu}\} \cup \{d \in \mathsf{R} \mid \langle q, \mathsf{cl}\rangle \xrightarrow{d}\}$, $\mathsf{TTraces}(\langle q, \mathsf{cl}\rangle) =_{df} \{\sigma \in (\mathsf{R} \cdot L)^* \cdot \mathsf{R} \mid \langle q, \mathsf{cl}\rangle \xrightarrow{\sigma}\}$, $\mathsf{DTraces}(\langle q, \mathsf{cl}\rangle) =_{df} \{[\sigma] \mid \sigma \in \mathsf{TTraces}(\langle q, \mathsf{cl}\rangle)\}$, $\mathsf{out}(\mathcal{P}) =_{df} \bigcup_{\langle q, \mathsf{cl}\rangle \in \mathcal{P}} \mathsf{out}(\langle q, \mathsf{cl}\rangle)$.

Let $Spec = (Q, \mathsf{In}, \mathsf{Out}, T, q_0, l, u)$ (specification) and $Imp = (Q', \mathsf{In}, \mathsf{Out}, T', q'_0, l', u')$ (implementation) be two TIOTS. $Imp\ \mathsf{tioco}\ Spec =_{df} \forall \sigma \in \mathsf{TTraces}(\langle q_0, 0\rangle) : \mathsf{out}(\langle q'_0, 0\rangle\ \mathsf{after}\ \sigma) \subseteq \mathsf{out}(\langle q_0, 0\rangle\ \mathsf{after}\ \sigma)$.

A *digital-timed test* for a specification $Spec$ over L is a total function

$$D : (L \cup \mathsf{N})^* \rightarrow \mathsf{In} \cup \{\mathsf{wait}, \mathsf{pass}, \mathsf{fail}\}$$

which can be represented as an IOTS. Given an implementation Imp and a digital-timed test D, we have: $Imp\ \mathsf{passes}\ D \Leftrightarrow \mathsf{DTraces}(Imp) \subseteq \mathsf{DTraces}(D)$, and $Imp\ \mathsf{fails}\ D \Leftrightarrow \exists \sigma_{tick} \in \mathsf{DTraces}(Imp) : \sigma_{tick} \notin \mathsf{DTraces}(D)$. Given two digital-timed tests D and D', D' is said to be a *prefix* of D if $\mathsf{DTraces}(D') \subseteq \mathsf{DTraces}(D)$. Consider a suite $\mathcal{D}$ of digital-timed tests. We say that $\mathcal{D}$ is *sound* with respect to $Spec$ if $\forall Imp\ :\ Imp\ \mathsf{tioco}\ Spec \Rightarrow Imp\ \mathsf{passes}\ \mathcal{D}$.

5 Action Refinement

5.1 Analog-time Action Refinement

The high-level actions to be refined are in $L_\delta = \mathsf{In} \cup \mathsf{Out}_\delta$ and the low-level actions are in $L_r = \mathsf{In}_r \cup \mathsf{Out}_{r,\delta_r} \cup \{\tau_r\}$, where $\mathsf{Out}_{r,\delta_r} = \mathsf{Out}_r \cup \{\delta_r\}$ and τ_r is a low-level unobservable action. Each high-level action μ is refined into a timed-path of the form

$$\mathsf{analog}_{\mathsf{ref}}(\mu) = q_1 \xrightarrow[[l_1,u_1]]{\mu_1^r} q_2 \xrightarrow[[l_2,u_2]]{\mu_2^r} \cdots q_n \xrightarrow[[l_n,u_n]]{\mu_n^r} q_{n+1}$$

where $\mu_i^r \in L_r$ and $l_i, u_i \in \mathsf{N}$ are, respectively, the minimal and maximal delays of the corresponding transition. The sequence $\mathsf{analog}_{\mathsf{ref}}(\mu)$ is called the *analog-time action refinement* of the high level action μ. Each high-level input-action in In is refined into a timed-path over low-level input-actions in $\mathsf{In}_r \cup \{\tau_r\}$. Similarly, each high-level output-action in Out is refined into a timed-path over low-level output-actions in $\mathsf{Out}_r \cup \{\tau_r\}$. The length of the analog-timed refinement of a given low-level action is the number of edges that compose it. For instance the length of the analog-timed refinement of the action μ above is n.

The refinement of the particular action δ is made as follows:

$$\mathsf{analog}_{\mathsf{ref}}(\delta) = q \xrightarrow[[\Delta,\Delta]]{\delta_r} q'$$

where $\Delta \in \mathsf{N}$ is the maximal waiting time constant (e.g., set either by the system designer or the tester). That is Δ time-units must elapse before announcing that a timeout occurs.

The refinement technique above carries over in a natural way to the refinement of an IOTS S into a TIOTS S_r. Given a transition $\tau = q \xrightarrow{\mu} q'$ of S, if $\mathsf{analog}_{\mathsf{ref}}(\mu) = q_0 \xrightarrow[[l_0,u_0]]{\mu_0^r} q_1 \xrightarrow[[l_1,u_1]]{\mu_1^r} \cdots q_n \xrightarrow[[l_n,u_n]]{\mu_n^r} q_{n+1}$ then τ is replaced within S_r with the sequence $\mathsf{analog}_{\mathsf{ref}}(\tau) = q \xrightarrow[[l_0,u_0]]{\mu_0^r} q_{\tau,1} \xrightarrow[[l_1,u_1]]{\mu_1^r} \cdots q_{\tau,n} \xrightarrow[[l_n,u_n]]{\mu_n^r} q'$ where q and q' are the same states as in τ and $q_{\tau,1}, \cdots, q_{\tau,n}$ are the names of the new added states (they must be distinct from the already added ones). Notice that q and q' may be the same. If $S = (Q, \mathsf{In}, \mathsf{Out}, T, q_0)$ then $S_r = \mathsf{analog}_{\mathsf{ref}}(S) = (Q_r, \mathsf{In}_r, \mathsf{Out}_{r,\delta_r}, T_r, q_0^r, l_r, u_r)$ such that: $Q_r = Q \cup \{q_{\tau,i} \mid \tau \in T\}$. $T_r = \bigcup_{\tau \in T} \mathsf{analog}_{\mathsf{ref}}(\tau)$; $q_0^r = q_0$. l_r and u_r are the functions encoding, respectively, the minimal and maximal delays appearing in $\mathsf{analog}_{\mathsf{ref}}(\tau)$ for all $\tau \in T$. We consider again the lighting-device example. The model given in Figure 1 is indeed a simplified version of the real implementation. In fact, the lighting-device has only one input action touch. The input high-level actions are introduced for ease of modeling. The lighting device disposes of a touch-sensitive pad. The user interface logic is as follows: a simple unique touch corresponds to the high-level input-action single and two "quick" consecutive touches correspond to the high-level input-action double. The maximum delay between two consecutive touches considered as a double touch is fixed to 1 time-unit.

As already shown in Figure 3, minimum and maximum delays for the lamp to change intensity need to be introduced as well. We assume that moving from one intensity level to another takes between 2 and 3 time units. Furthermore, we fix the maximal waiting time constant Δ to 4 time-units. The (analog-time) refinement rules for this example are summed up within the table shown in Figure 4.

μ	$\mathsf{analog}_{\mathsf{ref}}(\mu)$
single?	$q \xrightarrow[[0,\infty]]{\mathsf{touch}_r?} q' \xrightarrow[[2,2]]{\tau_r} q''$
double?	$p \xrightarrow[[0,\infty]]{\mathsf{touch}_r?} p' \xrightarrow[[0,1]]{\mathsf{touch}_r?} p'' \xrightarrow[[1,1]]{\tau_r} p'''$
dim!	$r \xrightarrow[[2,3]]{\mathsf{dim}_r!} r'$
bright!	$s \xrightarrow[[2,3]]{\mathsf{bright}_r!} s'$
off!	$t \xrightarrow[[2,3]]{\mathsf{off}_r!} t'$
δ!	$u \xrightarrow[[4,4]]{\delta_r!} u'$

Fig. 4 – Analog-time refinement rules for the lighting device example.

5.2 Digital-time Action Refinement

We first start with the refinement of input-actions. Consider a high-level input-action μ such that $\mathsf{analog}_{\mathsf{ref}}(\mu) = q_0 \xrightarrow[[l_0,u_0]]{\mu_0^r} q_1 \xrightarrow[[l_1,u_1]]{\mu_1^r} \cdots q_n \xrightarrow[[l_n,u_n]]{\mu_n^r} q_{n+1}$. For the moment, we assume that $\forall i : u_i \in \mathsf{N}$ (i.e., $u_i \neq \infty$). The *digital-time action refinement* of μ is defined as follows:

$$\mathsf{digital}_{\mathsf{ref}}(\mu) = \{t_0\, \mu_0 \cdots t_n\, \mu_n \mid \forall i : t_i \in \mathsf{N} \wedge l_i \leq t_i \leq u_i\}.$$

That is, if we consider $\mathsf{analog}_{\mathsf{ref}}(\mu)$ as a TIOTS (with initial state q_0) then $\mathsf{digital}_{\mathsf{ref}}(\mu)$ is the set of valid digital-timed traces of this TIOTS.

Since we are interested in testing, considering unbounded time delays is not of any help (i.e., by definition, the tester must not wait for ever). Thus, we may consider a maximal time delay $\Delta' \in \mathsf{N} \setminus \{0\}$ that we will not exceed while generating consecutive inputs. For each $i = 1, \cdots, n$, if $u_i = \infty$ then the upper bound of the corresponding interval is replaced with Δ'.[1]

For instance for the high-level input-action "double", if we take $\Delta' = 2$ then:

$$\begin{aligned}\mathsf{digital}_{\mathsf{ref}}(\mathsf{double}) \;=\; \{\;& 0_{tick}\,\mathsf{touch}_r\,0_{tick}\,\mathsf{touch}_r\,1_{tick},\\ & 0_{tick}\,\mathsf{touch}_r\,1_{tick}\,\mathsf{touch}_r\,1_{tick},\\ & 1_{tick}\,\mathsf{touch}_r\,0_{tick}\,\mathsf{touch}_r\,1_{tick},\\ & 1_{tick}\,\mathsf{touch}_r\,1_{tick}\,\mathsf{touch}_r\,1_{tick},\\ & 2_{tick}\,\mathsf{touch}_r\,0_{tick}\,\mathsf{touch}_r\,1_{tick},\\ & 2_{tick}\,\mathsf{touch}_r\,1_{tick}\,\mathsf{touch}_r\,1_{tick}\quad\}.\end{aligned}$$

Notice that we erased on purpose the low-level action τ_r from the refined digital-timed traces since τ_r is unobservable anyway. Also notice that for simplicity, we can take $\Delta' = \Delta$.

For testing purposes, we assume that the analog-time refinement of any low-level input-action starts with $q_0 \xrightarrow[[l_0,u_0]]{\mu_0^r} q_1$ such that $l_0 = u_0 = 0$ and μ_0^r is an observable action. Thus,

[1] We assume that for each $i = 1, \cdots, n$, we have $l_i \leq \Delta'$.

for instance for the high-level input-action "double", we will have

$$\mathsf{analog}_{\mathsf{ref}}(\mathsf{double}) = p \xrightarrow[[0,0]]{\mathsf{touch}_r?} p' \xrightarrow[[0,1]]{\mathsf{touch}_r?} p'' \xrightarrow[[1,1]]{\tau_r} p'''$$

and

$$\begin{aligned}\mathsf{digital}_{\mathsf{ref}}(\mathsf{double}) \quad = \{ \quad & 0_{tick}\,\mathsf{touch}_r\,0_{tick}\,\mathsf{touch}_r\,1_{tick}, \\ & 0_{tick}\,\mathsf{touch}_r\,1_{tick}\,\mathsf{touch}_r\,1_{tick} \quad \}\end{aligned}$$

instead of the digital-time refinement above. This assumption guarantees that the tester may start the execution of the refined digital-time input-trace before any low-level action occurs.

We also assume that for any two distinct high-level input-actions μ and μ', for all $\mu_r \in \mathsf{digital}_{\mathsf{ref}}(\mu)$ and $\mu'_r \in \mathsf{digital}_{\mathsf{ref}}(\mu')$, μ_r is not a prefix of μ'_r. The latter is a quite natural assumption. It says that there is some kind of determinism between the refined low-level digital-timed input-traces. We give a counterexample. Consider the two distinct high-level input actions μ and μ' and the two low-level digital-timed input-traces μ_r^1 and μ_r^2 such that $\mu_r^1 \in \mathsf{digital}_{\mathsf{ref}}(\mu)$ and $\mu_r^1 \cdot \mu_r^2 \in \mathsf{digital}_{\mathsf{ref}}(\mu')$. After executing μ_r^1, the tester will not know whether it should wait for outputs (the ones due to the execution of μ) or to continue executing the low-level actions corresponding to μ_r^2 (in order to finish the execution of the high-level input-action μ'). It is not difficult to check that this assumption is true for the two high-level input-actions single and double of our running example.

Now we move to digital-time action refinement of high-level output-actions. It is quite similar to input-actions. The main difference is that we need to represent refinements as (deterministic) trees and no longer as sets of traces. For instance the digital-time action-refinement of the high-level output-action dim is shown in Figure 5 (a). The leaves of the tree are labeled with the name of the corresponding high-level output-action (the labels within rectangular boxes in the figure).

The operator "$\mathsf{digital}_{\mathsf{ref}}(\,\cdot\,)$" is extended in the natural way to the case of a set of output-actions. For instance, the digital-time action-refinement of the set $\{\mathsf{dim}, \mathsf{bright}\}$ of output-actions is shown in Figure 5 (b). As for input-actions, we assume that the refinement of a given output-action is not the prefix of the refinement of an other output-action. Each leaf of the tree is labeled with the name of the corresponding high-level output-action. These labels will help us remember which high-level output-action each path of the tree corresponds to.

The reason why we should refine low-level output-actions as trees is the fact that we have to consider only deterministic test cases. The set of low-level output-actions $\{\mathsf{dim}, \mathsf{bright}\}$ is a good example which illustrates this point. Suppose we refine the two output-actions separately. In that case, we will have two edges emanating from the initial node of Figure 5 (b) which are labeled with $2_{tick}!$ instead of only one and any corresponding test case will no longer be deterministic anymore.

Notice that, the integer time-delays are marked as output-actions in the figure since the tester considers them so.

Consider the digital-timed trace $\sigma_{tick} \in (L_r \cup \mathsf{N})^* \times L_r$. We denote $P(\sigma_{tick})$ the *projection* of σ_{tick} to L_r, obtained by "erasing" from σ_{tick} all actions in N. For example, if $\sigma_{tick} = \mu 1 \nu 2 \mu 3 \mu$, then $P(\sigma_{tick}) = \mu\nu\mu\mu$.

For a better optimization, we proceed as follows. We collapse all nodes reached by digital-timed traces with the same projection on L_r into a single node. For instance in Figure 5 (b), we may collapse the two leaves labeled with the output-action bright into a single node and the two nodes labeled with dim as well.

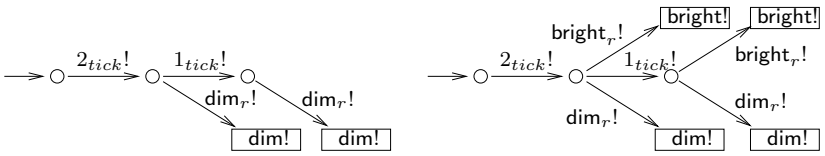


FIG. 5 – The two trees representing (a) the digital-time action-refinements of the high-level output-action dim and (b) the set of high-level output-actions $\{\mathsf{dim}, \mathsf{bright}\}$, respectively.

6 Digital-Timed Test Generation

The method we propose for generating digital-timed tests consists in transforming an initially untimed test "T" into a timed one "T_r" using the refinement techniques introduced so far. We assume that T is given as a tree. An internal node of T is called an *input-node* if it has a unique outgoing edge labeled with an input-action. Similarly, an internal node of T is called an *output-node* if all its outgoing edges are labeled with output-actions. To obtain T_r we proceed as follows:

S 1. We make a copy of T (we already call it T_r);

S 2. We omit all edges of T_r leading to fail;

S 3. We refine all edges of T_r and add progressively the new edges leading to fail.

Step "S 3" is achieved as follows:

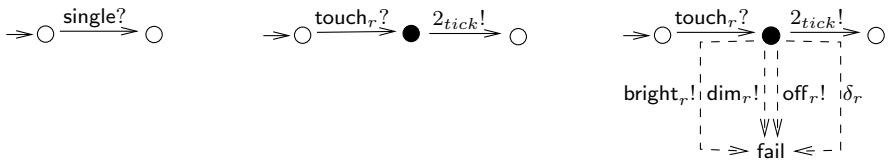
(I) For an input-node q, let $e = q \xrightarrow{\mu} q'$ be the outgoing edge from q:[2]

1. We choose a possible digital-time action-refinement $\mu_r = t_0\mu_0 \cdots \mu_n t_{n+1}$ of μ (i.e., $\mu_r \in \mathsf{digital}_{\mathsf{ref}}(\mu)$).
2. We replace e in T_r with the path $q_{i_0} \xrightarrow{t_0} q_{i_1} \xrightarrow{\mu_0} q_{i_2} \cdots \xrightarrow{\mu_n} q_{i_{2n+2}} \xrightarrow{t_{n+1}} q_{i_{2n+3}}$, where $q_{i_0} = q$, $q_{i_{2n+3}} = q'$ and the other q_{i_j} are new names not used in the past.
3. For each $t_j \neq 0$ and $\nu \in \mathsf{Out}_{r,\delta_r}$, we add a new edge $q_{i_{2j}} \xrightarrow{\nu} \mathsf{fail}$ to T_r.[3]

We consider the test case of Figure 2. We refine the edge emanating from the initial node of the test tree. This edge is labelled with the high-level input-action single. The low-level digital-timed input-trace we choose to refine this input-action with is $\mathsf{touch}_r?\, 2_{tick}!$. The different steps of the refinement of this edge are given in Figure 6. To make it clearer the new added nodes are filled in with black and the new added edges leading to fail are drawn as dashed arrows.

[2] We may have $q' = \mathsf{pass}$.

[3] This step allows to reject all the low-level output-actions that may be observed during the execution of refined digital-timed input-trace μ_r.



(a) (b) (c)

FIG. 6 – The different steps for refining an edge of a test case labelled with the low-level input-action single: (a) the original high-level edge, (b) the corresponding low-level digital-timed path, (c) adding the edges leading to fail.

(II) For an output-node q, let $\beta = \{\mu \in \mathsf{Out}_\delta \mid \exists q' : q \xrightarrow{\mu} q' \text{ is an edge of } T_r\}$ be the set of labels of the outgoing edges from q:

1. We build the tree $\mathsf{digital}_{\mathsf{ref}}(\beta)$.
2. For each edge $q \xrightarrow{\mu} q'$, we replace the leaves of $\mathsf{digital}_{\mathsf{ref}}(\beta)$ labeled with μ with the one node q'.[4]
3. We replace the set of edges $\{q \xrightarrow{\mu} q' \mid \mu \in \beta\}$ of T_r with $\mathsf{digital}_{\mathsf{ref}}(\beta)$.
4. For each internal node p of $\mathsf{digital}_{\mathsf{ref}}(\beta)$ and $\nu \in \mathsf{Out}_{r,\delta_r}$, if no outgoing edge from p is labeled with ν then we add a new edge $p \xrightarrow{\nu} \mathsf{fail}$ to $\mathsf{digital}_{\mathsf{ref}}(\beta)$.
5. For each internal node p of $\mathsf{digital}_{\mathsf{ref}}(\beta)$, If no outgoing edge from p is labelled with some $t \in \mathsf{N} \setminus \{0\}$ then we add a new edge $p \xrightarrow{1_{tick}} \mathsf{fail}$ to $\mathsf{digital}_{\mathsf{ref}}(\beta)$.

Again, we consider the test case of Figure 2. Now, we refine the edge labelled with the high-level output-action dim. We call q the node from which this edge emanates. Clearly, q is an output-node. The set β associated with q is the singleton set $\{\mathsf{dim}\}$. The digital-timed refinement of the set $\{\mathsf{dim}\}$ is already given in Figure 5 (a). The different steps of the refinement of the edge labelled with dim are given in Figure 7. As for Figure 6, the new added nodes are the black ones and the edges leading to fail are drawn as dashed arrows. [5]

If we concatenate the two refinements of Figure 6 and Figure 7, we obtain a possible digital-timed refinement of the test case of Figure 2. The refined test case is given in Figure 8.

The method we propose to generate digital-timed tests is sound in the following sense. Consider an IOTS $Spec$ and an untimed test T.

Proposition 1 *If T is generated by the untimed test genertaion algorithm of (Tretmans, 1999) then $\mathsf{digital}_{\mathsf{ref}}(T)$ is sound with respect to $\mathsf{analog}_{\mathsf{ref}}(Spec)$.*

[4] That means that the tester should behave in the same manner after any possible refinement of μ. Notice that, due to this, the refined test cases are DAG (Directed Acyclic Graphs) and no longer trees.

[5] In order not to overload the figure we draw only one dashed arrow for several edges leading to fail.

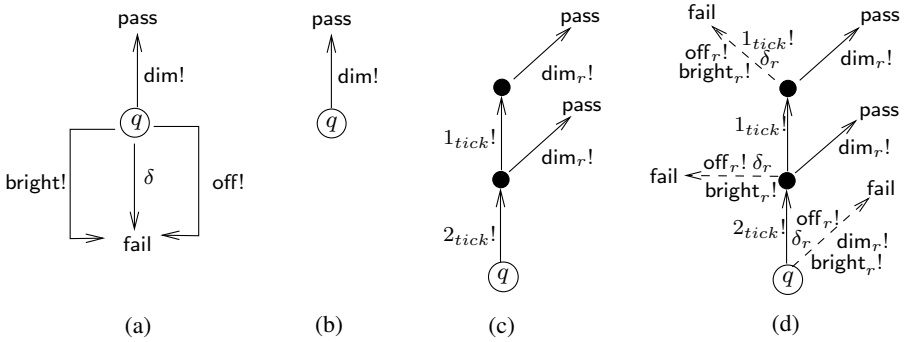


FIG. 7 – The different steps for refining the edges emanating from an output-node: (a) the original edges emanating from the output-node q, (b) removing all the high-level edges leading to fail, (c) replacing the edge labelled with dim with its digital-timed refinement, (d) adding the low-level edges leading to fail.

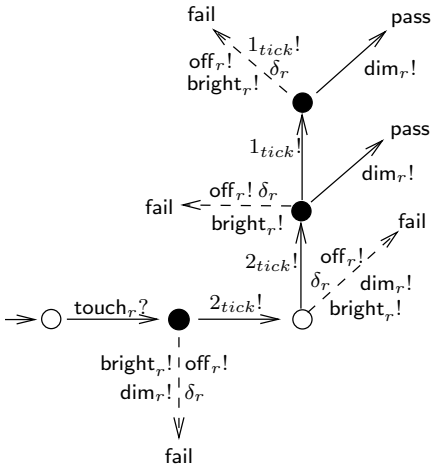


Fig. 8 – A possible digital-timed refinement of the test case of Figure 2.

7 Estimation of Memory Saving

Consider an IOTS $S = (Q, \mathsf{In}, \mathsf{Out}, T, q_0)$ and the corresponding set of refinement rules. Let n be the size of the set of nodes Q and m the size of T. We assume that the length of the analog-timed refinement of each edge of $Spec$ is between l_{min} and l_{max}. Let $S_r = \mathsf{analog}_{\mathsf{ref}}(S)$. We call Q_r the set of nodes of S_r and T_r its set of transitions. Let n_r be the size of Q_r and m_r the size of T_r. Then, it is not difficult to check the following.

Proposition 2 *We have:*

1. $l_{min} * m \leq m_r \leq l_{max} * m$;
2. $n + (l_{min} - 1) * m \leq n_r \leq n + (l_{max} - 1) * m$.

Thus, the memory saving at the specification level is linear in the number of transitions of the low-level specification S and the length of the analog-timed refinements.

Now, we consider an untimed test T generated by the untimed test generation algorithm of (Tretmans, 1999). Let $T_r = \mathsf{digital}_{\mathsf{ref}}(T)$. Also, let p and p_r be the number of edges composing T and T_r, respectively. We assume that $\Delta = \Delta'$.

Proposition 3 *We have:* $l_{min} * p \leq p_r \leq (2 * \Delta + 1) * l_{max} * p$.

8 Conclusion

In this work, we succeeded to make the link between: (I) The framework for untimed testing based on the model of IOTS and the (untimed) conformance relation ioco (Tretmans, 1999) and; (II) Our framework for real-time testing based on the model of TIOTS and the timed conformance relation tioco (Krichen and Tripakis, 2004). We have proposed a new method for generating digital-clock tests based on the so-called action refinement techniques.

Possible future directions of this work are: (1) To extend the method from TIOTS to general timed automata specifications. (2) To be able to handle more general, aperiodic digital-clock models, as in (Krichen and Tripakis, 2004, 2005). (3) To consider more general action refinement rules (e.g., an action is refined into a tree rather than a "linear" trace).

References

Krichen, M. and S. Tripakis (2004). Black-box conformance testing for real-time systems. In *11th International SPIN Workshop on Model Checking of Software (SPIN'04)*, Volume 2989 of *LNCS*. Springer. 1, 3, 12

Krichen, M. and S. Tripakis (2005). An expressive and implementable formal framework for testing real-time systems. In *The 17th IFIP Intl. Conf. on Testing of Communicating Systems (TestCom'05)*, Volume 3502 of *LNCS*. Springer. 1, 12

Tretmans, J. (1999). Testing concurrent systems: A formal approach. In *CONCUR'99*, Volume 1664 of *LNCS*. Springer. 2, 10, 12

van der Bijl, M., A. Rensink, and J. Tretmans (2005). Action refinement in conformance testing. In F. Khendek and R. Dssouli (Eds.), *TestCom*, Volume 3502 of *Lecture Notes in Computer Science*, pp. 81–96. Springer. 2