Test Cases Generation for Nondeterministic Duration Systems

Lotfi Majdoub[1] and Riadh Robbana[2]

[1] LIP2 Laboratory, Tunisia

[2] Tunisia Polytechnic School, Tunisia

Abstract. In this paper, we are interested in testing duration systems. Duration systems are an extension of real-time systems for which delays that separate events depend on the accumulated times spent by the computation at some particular locations of the system.
We present a test generation method for nondeterministic duration systems that uses the approximation method. This method extends a model using an over approximation, the approximate model, containing the digitization timed words of all the real computations of the duration system. Then, we propose an algorithm that generates a set of test cases presented with a tree by considering a discrete time.

1 Introduction

Duration systems are an extension of real-time systems for which in addition to constraints on delays separating certain events that must be satisfied, constraints on accumulated times spent by computation must also be satisfied.

Timed automata constitute a powerful formalism widely adopted for modeling real-time systems [2]. Duration Variables Timed Graphs with Inputs Outputs (DVTG-IOs) are an extension of timed automata [3], which are used as a formalism to describe duration systems. DVTG-IOs are supplied with a finite set of continuous real variables that can be stopped in some locations and resumed in other locations. These variables are called *duration variables*.

Testing is an important validation activity, particularly for real-time systems, which aims to check whether an implementation, referred to as an Implementation Under Test (IUT), conforms to its specification. Testing process is difficult, expensive and time-consuming. A promising approach to improve testing consists in automatically generating test cases from formal models of specification. Using tools to automatically generate test cases may reduce the cost of the testing process.

For testing real-time systems, most works borrow several techniques from the real-time verification field due to similarities that exist between model-based testing and system verification (e.g., symbolic techniques, region graph and its variations, model checking techniques, etc.). Those techniques are used particularly to reduce the infinite state space to a finite (or at least countable) state space. Then they adapt the existing untimed test case generation algorithm. We cite as example [6][8][9].

Majdoub L. and Robbana R. (2009).
Test Cases Generation for Nondeterministic Duration Systems.
In *Proceedings of the 7th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*, pages 13-22
DOI: 10.5220/0002196500130022
Copyright © SciTePress

It is well known that the verification of real-time systems is possible due to the decidability of reachability problem for real-time systems [1]. However, it has been shown that the reachability problem is undecidable for timed graphs extended with one duration variable [5] and, consequently, it is not possible to use classical verification techniques to generate test cases for DVTG-IO.

We give in this paper a method for testing duration systems. We consider a nondeterministic duration system in the sense that the current state of the system is not known precisely and the tester has the choice between several actions to experiment. We describe the specification as well as the implementation under test with DVTG-IO. In order to reduce the infinite state space to a finite state space, we use the approximation method that extends a given DVTG-IO specification to anohter called the approximate model that contains the initial test cases as well as their discretisations. An algorithm of generating a set of test cases is given. We present this set of test cases by a tree.

This paper is organized as follows: In the next section, we present the duration variables timed graphs with inputs outputs. In section 3, we define the nondeterminism concept. Section 4 shows the approximation method. Our testing method, is introduced in section 5. Concluding remarks are presented in section 6.

2 Duration Variables Timed Graphs with Inputs Outputs

A DVTG-IO is described by a finite set of locations and a transition relation between these locations. In addition, the system has a finite set of duration variables that are constant slope continuous variables, each of them changing continuously with a rate in {0,1} at each location of the system. Transitions between locations are conditioned by arithmetical constraints on the values of the duration variables. When a transition is taken, a subset of duration variables should be reset and an action should be executed. This action can be either an input action, an output action or an unobservable action [15].

2.1 Formal Definition

We consider X a finite set of duration variables. A guard on X is a boolean combination of constraints of the form $x \prec c$ where $x \in X, c \in N, \prec \in \{<, \leq, >, \geq\}$. Let $\Gamma(X)$ be the set of guards on X. A DVTG-IO describing duration systems is a tuple $S = (Q, q_0, E, X, Act, \gamma, \alpha, \delta, \partial)$ where Q is a finite set of locations, q_0 is the initial location, $E \subseteq Q \times Q$ is a finite set of transitions between locations, $Act = Act_{In} \cup Act_{Out} \cup \{\xi\}$ is a finite set of input actions (denoted by $?a$), output actions (denoted by $!a$) and unobservable action ξ, $\gamma : E \longrightarrow \Gamma(X)$ associates to each transition a guard which should be satisfied by the duration variables whenever the transition is taken, $\alpha : E \longrightarrow 2^X$ gives for each transition the set of duration variables that should be reset when the transition is taken, $\delta : E \longrightarrow Act$ gives for each transition the action that should be executed when the transition is taken, $\partial : Q \times X \longrightarrow \{0, 1\}$ associates with each location q and each duration variable x the rate at which x changes continuously in q.

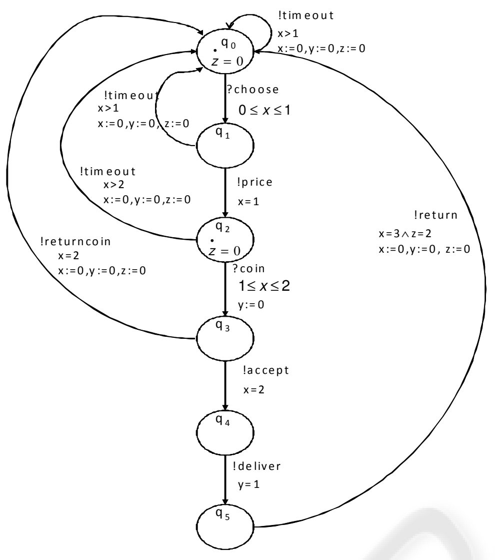

Fig. 1. DVTG-IO of a vending machine.

Example. To illustrate DVTG-IO, we show on Figure 1 the specification of a vending machine. This machine delivers a beverage after receiving a coin from the user. It is composed of locations $\{q_0, q_1, q_2, q_3, q_4, q_5\}$ where q_0 is the initial location, transitions between locations and is supplied with a set of input actions $\{?choose, ?coin\}$, output actions $\{!price, !accept, !deliver, !returncoin, !return, !timeout\}$ and three duration variables x, y, z. Duration variables x and y are clocks used to make constraints on the time execution of the vending machine, z is a duration variable, it is stopped ($\dot{z}= 0$) in locations q_0, q_2 and it is used to make constraint on the time spent by the system.

2.2 State Graph

The semantic of DVTG-IO is defined in terms of a state graph over states of the form $s = (q, \nu)$ where $q \in Q$ and $\nu : X \longrightarrow \mathbb{R}^+$ is a valuation function that assigns a real value to each duration variable. Let St_S be the set of states of S. We notice that St_S is an infinite set due to the value of duration variables taken on $\mathbb{R}^+$. A state (q, ν) is called *integer state* if $\nu : X \longrightarrow \mathbb{N}$. We denote by $N(St_S)$ the set of integer states of S.

Given a valuation ν and a guard g, we denote by $\nu \models g$ the fact that valuation of g under the valuation ν is true.

We define two families of transition between states : discrete transition $(q, \nu) \overset{a}{\rightsquigarrow} (q', \nu')$ where $(q, q') \in E,\ \delta(q, q') = a,\ \nu \models \gamma(q, q')$ is true and $\nu'(x) = \nu(x)\ \forall x \in X \backslash \alpha(q, q')$, $\nu'(x) = 0\ \forall x \in \alpha(q, q'),$ that corresponds to moves between locations using transition in E, timed transition $(q, \nu) \overset{t}{\rightsquigarrow} (q, \nu')$ such that $t \in \mathbb{R}^+$ and $\nu'(x) = \nu(x) + \partial(q, x) * t\ \ \forall x \in X$, that corresponds to transitions due to time progress at some location q.

The state graph associated with S is $(St_S, \rightsquigarrow)$ where $\rightsquigarrow$ denotes the union of all discrete and timed transitions.

2.3 Computation Sequences and Timed Words

A Computation sequence of a DVTG-IO is defined as a finite sequence of configurations. A configuration is a pair (s, τ) where s is a state and τ is a time value. Let C_S be the set of configurations of S. Intuitively, a computation sequence is a finite path in the state graph of an extension of S by an observation clock that records the global elapsed time since the beginning of the computation. Formally, if we extend each transition relation from states to configurations, then a computation sequence of S is $\sigma = (s_0, 0) \rightsquigarrow (s_1, \tau_1) \rightsquigarrow ... \rightsquigarrow (s_n, \tau_n)$. Let $CS(S)$ be the set of computations sequences of S.

A timed word is a finite sequence of timed actions. A timed action is a pair $a\tau$ where $a \in Act$ and $\tau \in \mathbb{R}^+$, meaning that action a takes place when the observation clock is equal to τ. A timed action $a\tau$ is called integer timed action if $\tau \in \mathbb{N}$. A timed word is a sequence $\omega = a_1\tau_1a_2\tau_2...a_n\tau_n$ where a_i is an action and τ_i is a value of the observation clock. We notice that $\tau_i \leq \tau_{i+1}$. Let $L(S)$ be the set of timed words of S.

A sequence $\omega = a_1\tau_1a_2\tau_2...a_n\tau_n$ is considered a timed word of $L(S)$ if and only if there exists a computation sequence $\sigma = (s_0, \tau_0) \rightsquigarrow (s_1, \tau_1) \rightsquigarrow ... \rightsquigarrow (s_n, \tau_n) \in CS(S)$ such that $a_i = \delta(q_{i-1}, q_i)$ for $i = 1, .., n$ and $s_i = (q_i, \nu_i)$. For simplicity, we may write $(s_0, \tau_0) \overset{\omega}{\rightsquigarrow} (s_n, \tau_n)$.

Let $\omega = a_1\tau_1a_2\tau_2...a_n\tau_n$ be a timed word and $a \in Act,\ \tau \in \mathbb{R}^+$ such that $\tau_n \leq \tau$ then we denote by $\omega.a\tau$ the timed word obtained by adding $a\tau$ to ω and we have $\omega.a\tau = a_1\tau_1a_2\tau_2...a_n\tau_na\tau$.

3 Nondeterministic Duration Variables Timed Graphs

In this section, we present the concept of nondeterminism used to characterize DVTG-IO describing the specification of duration systems. Our study concerns three forms of nondeterminism.

External Choice. A state in the state graph of a DVTG-IO has an external choice if, from this state, the tester can choose between several input actions to experiment on the implementation under test. A DVTG-IO S has an external choice iff $\exists s \in St_S,\ \exists a, b \in Act_{In},\ a \neq b$ such that $s \overset{a}{\rightsquigarrow}$ and $s \overset{b}{\rightsquigarrow}$.

Observable Nondeterminism. The observable nondeterminism means that the output action generated by the implementation under test is not unique. A DVTG-IO S is observable nondeterministic iff $\exists s \in St_S,\ \exists a, b \in Act_{Out},\ a \neq b$ such that $s \overset{a}{\rightsquigarrow}$ and $s \overset{b}{\rightsquigarrow}$.

Nondeterminism. A DVTG-IO is said nondeterministic if the current state is not necessarily determined by observing their execution. A DVTG-IO S is nondeterministic iff $\exists s, t, u \in St_S,\ \exists a \in Act,\ t \neq u$ such that $s \overset{a}{\rightsquigarrow} t$ and $s \overset{a}{\rightsquigarrow} u$.

4 Approximation

The approximation method is used in the verification of duration systems [13]. We adapt this method to test duration systems.

4.1 Digitization

We present the notion of digitization [7], which is suitable for the systems in which we are interested.

Let $\tau \in \mathbb{R}^+$. For every $\epsilon \in [0, 1[$ called *digitization quantum*, we define the digitization of τ, $[\tau]_\epsilon$ =if $\tau \leq (\lfloor\tau\rfloor + \epsilon)$ then $\lfloor\tau\rfloor$ else $\lceil\tau\rceil$.

Given $\epsilon \in [0, 1[$, the digitization of a timed word $\omega = a_1\tau_1a_2\tau_2...a_n\tau_n$ is $[\omega]_\epsilon = a_1\ [\tau_1]_\epsilon\ a_2\ [\tau_2]_\epsilon...a_n\ [\tau_n]_\epsilon$

Therefore, it is not difficult to see that: $[\omega.at]_\epsilon = [\omega]_\epsilon.a[t]_\epsilon$

Moreover, it is easier to relate digitizations of a computation sequence and its timed word. If σ is a computation sequence and ω is its corresponding timed word then for $\epsilon \in [0, 1[$, $[\omega]_\epsilon$ is the corresponding timed word of $[\sigma]_\epsilon$.

We denote by $Digit(L(S))$ the set of all the digitizations of all the real timed word of S. We notice that $Digit(L(S))$ is finite.

The digitization is used to reduce the infinite set of states to a finite set of states. A question that one may ask is whether $Digit(L(S)) \subseteq L(S)$ or not.

Example. In the following example, we will see that we can have a DVTG-IO with only one duration variable for which there exists a real timed word such that all their digitizations are not timed words of the system.

Let $\omega = ?choose\ 0.5\ !price\ 1\ ?coin\ 1.5\ !accept\ 2\ !deliver\ 2.5\ !return\ 3$ be the real timed word of the DVTG given in the figure 1.

There are two digitizations of ω :

$[\omega]_\epsilon = ?choose\ 0\ !price\ 1\ ?coin\ 1\ !accept\ 2\ !deliver\ 2\ !return\ 3\ ;\ 0.5 < \epsilon < 1$

$[\omega]_\epsilon = ?choose\ 1\ !price\ 1\ ?coin\ 2\ !accept\ 2\ !deliver\ 3\ !return\ 3\ ;\ 0 \leq \epsilon \leq 0.5$

It is easier to verify on the DVTG-IO given in figure 1 that ω is a timed word, however their two digitizations $[\omega]_\epsilon$ for $0.5 < \epsilon < 1$ and $0.5 < \epsilon < 1$ are not timed words of the considered DVTG-IO.

4.2 Approximate Model

As we have seen in the previous example, some timed words of a DVTG-IO do not have any digitizations in S. The idea given in [13] consists of over approximating the model S by an approximate model S' such that $Digit(L(S)) \subseteq L(S')$.

Definition 1. The function $\beta : X \times E \longrightarrow \mathbb{N}$ calculates for each variable $x \in X$ and each transition $e = (q, q')$ the maximum of restarts of x from the last reset of x until the location q in each way.

A restart of a variable x is the change of its rate from 0 to 1. After a reset of a variable x, if the rate of a variable x in the current location is 1, then the access to this location is considered as a restart of x. For example, if in the first location x has a rate equal to 1 then the access to the initial state is considered as restart. That is why, for the clocks, the function β is equal to 1 for each transition.

Definition 2. The approximate model $S' = App(S)$ is obtained from S by transforming each guard of a transition e of the form $u \prec x \prec w$ by the guard:

– if $u - \beta(x, e) \geq -1$ then $u - \beta(x, e) + 1 \leq x \leq w + \beta(x, e) - 1$

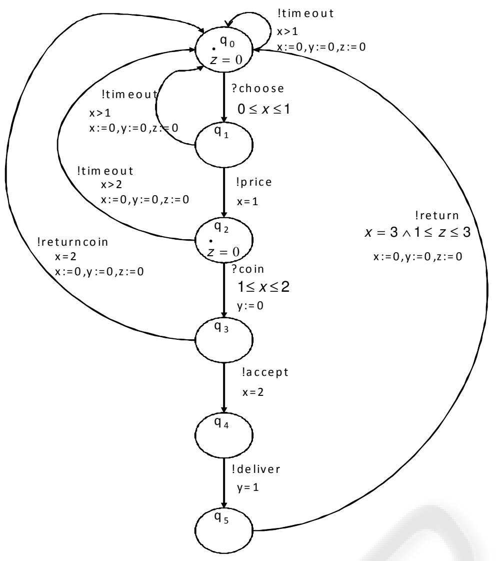

Fig. 2. The approximate model.

- otherwise $0 \le x \le w + \beta(x, e) - 1$
  where $u, w \in \mathbb{N},\ x \in X,\ and\ \prec \in \{<, \le\}$

Example. If we apply the approximation method on the DVTG-IO of figure 1, we obtain the approximate model of figure 2, that consists of replacing the guard $x = 3 \wedge z = 2$ associated to the transition (q_5, q_0) by the guard $x = 3 \wedge 1 \le z \le 3$, because we have $\beta(z, (q_5, q_0)) = 2$.

It is easier to verify that the digitizations of the previous example belong to the approximate model.

Proposition 1 demonstrates that for every timed word of the specification model, its digitizations belong to the approximate model.

Proposition 1 [11]. $\forall\ \omega \in L(S)$ we have $[\omega]_\epsilon \in L(S')$ for each $\epsilon \in [0, 1[$ ■

5 Testing Method

Here, we introduce our testing method for nondeterministic duration systems that is based on the approximation method.

5.1 Test Generation with the Approximate Model

First, let us introduce the notion of an observation which is a sequence of controllable (inputs) and observable (outputs) actions that are either executed or produced by the IUT followed with its occurrence time. Formally, we describe an observation by a timed word $\omega = a_1\tau_1 a_2\tau_2 ... a_n\tau_n$ where $a_i \in Act$ and $\tau_i \in \mathbb{R}^+$.

Our result is based on a reduction of the infinite state graph associated with $S' = App(S)$ to the countable state graph $(N(St_{S'}), \overset{1}{\rightsquigarrow} \cup \overset{a}{\rightsquigarrow})$, where the space of states is the set of integer states. Transitions between states are either discrete transition $(q, \nu) \overset{a}{\rightsquigarrow} (q', \nu')$ labeled with action in Act, or timed transition $(q, \nu) \overset{1}{\rightsquigarrow} (q, \nu')$ labeled with a constant delay of time equal to 1. Notice that ν and $\nu' \in [X \rightarrow \mathbb{N}]$. Clearly, the digitizations of all timed words $Digit(L(S))$ are included in $(N(St_{S'}), \overset{1}{\rightsquigarrow} \cup \overset{a}{\rightsquigarrow})$.

5.2 The Test Tree

We use the countable state graph $(N(St_{S'}), \overset{1}{\rightsquigarrow} \cup \overset{a}{\rightsquigarrow})$ to generate a finite set of test cases. This set of test cases is represented by a tree called *test tree* .

The test tree is composed by nodes that are sets of integer configurations and transitions between those nodes. A node in the test tree is a finite set of integer configurations (s, τ) such that $s \in N(St_{S'})$, $\tau \in \mathbb{N}$ and represents the possible current integer configurations of the IUT. The root is the initial configuration of $(N(St_{S'}), \overset{1}{\rightsquigarrow} \cup \overset{a}{\rightsquigarrow})$ that is $(s_0,\ \tau_0)$. We remember that we consider a nondeterministic duration system and that the current system configuration are not known precisely. The transition between one node and its successor is labelled with a timed action $a\tau$ such that $a \in Act$ and $\tau \in \mathbb{N}$. A path from the root to one leaf of the tree represents a digitization of a timed word.

Example. An example of test tree is given in figure 3. That is the test tree constructed from the approximate model of figure 2. Each path of the test tree from the root to a leaf corresponds to an integer computation sequence of the approximate model.

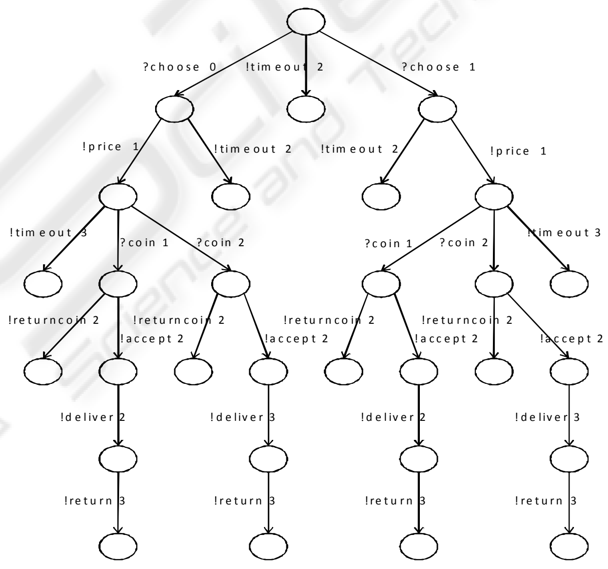

Fig. 3. The test tree.

5.3 Algorithm of Generating Test Tree

We adapt the untimed test generation algorithm of [15].

```
Input : $N(G_{S'}) = (N(St_{S'}), \overset{1}{\rightsquigarrow} \cup \overset{a}{\rightsquigarrow})$
Output : Test Tree T
$T = \{(\mathrm{s}_0,0)\}$ the one-node tree
while $True$
  for each leaf $C$ of $T$ distinct from $pass$
    $Out(C)$
    $In(C)$
    if $Out(C) \cup In(C) = \emptyset$ then
      $C = pass$
    else
      do randomly $\{1;2\}$
        case 1 :
          for each $a\tau \in Out(C) \cup In(C)$
            $C' = C\ after\ a\tau$
            append edge $C \overset{a\tau}{\rightsquigarrow} C'$ to $T$
        case 2 :
          $C = pass$
End while
```

$Out(C)$ (resp. $In(C)$) is the set of all timed output actions (resp. the set of all timed input actions) that can occur when the system is at configuration of C. $C\ after\ a\tau$ is the set of all configurations that can be reached from C after the execution of the timed word $a\tau$. Notice that $Out(C), In(C)$ and $C\ after\ a\tau$ are finite sets. They are calculated in $N(G_{S'}) = (N(St_{S'}), \overset{1}{\rightsquigarrow} \cup \overset{a}{\rightsquigarrow})$.

The generating test tree algorithm operates as follows : initially the test tree contains one node that is the initial configuration of S' : $(s_0, 0)$. For every leaf C of the tree, the algorithm calculates the set of integer timed actions ($In(C)$ and $Out(C)$) that can be taken when the system is in C. For each timed action $a\tau$ the algorithm claculates $C' = C\ after\ a\tau$, the set of configurations obtained when $a\tau$ is executed, and appends the edge $C \overset{a\tau}{\rightsquigarrow} C'$ to the tree. The algorithm can stop a path of the tree by appending the node *pass* in the leaf.

Proposition 2 [12]. Let $\omega \in L(S)$ be a timed word and $[\omega]_\epsilon \in L(S')$ its digitization for $\epsilon \in [0,1[$ if $\exists\ a \in Act$ and $\exists \tau' \in \mathbb{N}$ such that $[\omega]_\epsilon.a\tau' \in L(S')$ then $\forall \tau \in]\tau' - 1 + \epsilon,\ \tau' + \epsilon\ ]$ we have $\omega.a\tau \in L(S)$■

The test generation from the approximate specification model can give to the tester the action and the integer time value of its execution on the IUT in discrete time. The above proposition shows that if the tester executes the action (input or output) within a real-time interval, defined by the proposition, then the conformance of the observation recorded on the IUT is preserved according to the approximate model.

Proposition 3. Let $\omega = a_1\tau_1 a_2\tau_2 ... a_n\tau_n$ be a timed word that corresponds to a path from the root to a leaf in T_S.

then $\exists\ \omega' \in L(S)$ such that $[\omega']_\epsilon = \omega$

Proof.

We proceed by a recursif proof on the size of ω.

Let $\omega_i = a_1\tau_1a_2\tau_2...a_i\tau_i$ with $i \leq n$ be the timed word obtained in the level i of the test tree, we have $\omega_n = \omega$

For $i = 0, \ \omega_0 = \emptyset$ the proposition is true because $(\omega'_0 = \emptyset)$ $\omega'_0 \in L(S)$ and we have $[\omega'_0]_\epsilon = \omega_0$

For $i < n$ we suppose that the proposition is true for i and we try to demonstrate for $i + 1$

$\exists\ \omega'_i \in L(S)$ such that $[\omega'_i]_\epsilon = \omega_i$

Given $a\tau \in Out(S'\ after\ \omega_i) \cup In(S'\ after\ \omega_i)$

We have $\omega_i.a\tau \in L(S')$

From the proposition 2, $\forall \tau' \in]\ \tau - 1 +\ \epsilon,\ \ \tau +\ \epsilon\ ]$ we have $\omega'_i.a\tau' \in L(S)$

So $[\omega'_i.a\tau']_\epsilon = \omega_i.a\tau$ ■

A path in the test tree is a discrete timed word obtained from the coutable state graph $(N(St_{S'}), \overset{1}{\leadsto} \cup \overset{a}{\leadsto})$ associated to the approximate model $S' = App(S)$. In proposition 3, we demonstrate that a path in the test tree corresponds to a digistization of a timed word belonging to the initial model S.

By considering this result and the result obtained in the proposition 2, we can use the test tree to generate a discrete test case, then we can experiment it by considering continuous time. For generating an input timed action that should be executed on the IUT, the tester chooses one integer timed action $a\tau$ from the test tree. By proposition 2, the action a can be applied within the real-time interval $]\tau - 1 + \epsilon, \tau + \epsilon]$.

6 Conclusions

We have introduced a method for testing nondeterministic duration systems. First, we used the DVTG-IO as a formalism to model specification. Second, we presented the approximation method. This method extends a given DVTG-IO to another called approximate model that contains the initial test cases as well as their digitizations. Then, we proposed an algorithm that generate a set of test cases presented in a tree by considering a discrete time, we demonstrated that test cases generated from the approximate model correspond to the digitization of timed words of the specification model. At the end, we showed how those test cases are executed on the implementation by considering a continous time.

In the future work we plan to see how we can extend our approach to test duration systems with quiescence actions and how we can use this approach to test hybrid systems.

References

1. Alur R., Courcoubetis C., and Dill D., *Model-Checking for Real-Time Systems*, 5th Symp. on logic in Computer Science, 1990.
2. Alur R.and Dill D., *A Theory of Timed Automata*, Theoretical Computer Science, 126 : 183-235, 1994.

3. Bouajjani A., Echahed R., Robbana R., *Verifying Invariance Properties of Timed Systems with Duration Variables*, Formal Techniques in Real-Time and Fault Tolerant Systems, 1994.
4. Cassez F., Larsen K.G, *The Impressive Power of Stopwatches*, In Proc. Conference on Concurrency Theory (CONCUR'00), Penssylvania, USA, 2000
5. Cerans K., *Decidability of Bisimulation Equivalence for Parallel timer Processes,* In Proc. Computer Aided Verification (CAV'92), Springer-Verlag, 1992, LNCS 663.
6. En-Nouaary A., Dssouli R., Khender F., and Elqortobi A., *Timed Test cases generation based on state characterisation technique*, In RTSS'98. IEEE, 1998.
7. Henzinger T., Manna Z., and Pnuelli A., *What good are digital clocks?*, In ICALP'92, LNCS 623, 1992.
8. Hessel A., Pettersson P., *A Test Case Generation Algorithm for Real-Time Systems*, In Proc. 4th international Conference on Quality software, pp. 268-273, 2004.
9. Krichen M, Tripakis S., *Black-Box Conformance Testing for Real-Time Systems*, SPIN'04 Workshop on Model Checking Software, 2004.
10. Majdoub L. and Robbana R., *Test Purpose of Duration Systems,* 4th MSVVEIS, pages 67-75, Cyprus, May 2006.
11. Majdoub L. and Robbana R., *Testing Duration Systems using an Approximation Method*, Depcos-RELCOMEX, pp.119-126, Szklarska Poreba, Poland, June 2007.
12. Majdoub L. and Robbana R., *Testing Duration systems,* Journal Européen des Systèmes Automatisés, Approches formelles pour la validation de systèmes temps-réel, vol 42 n°9/2008, pp. 1111-1134, November 2008.
13. Robbana R, *Verification of Duration Systems using an Approximation Approach*, Journal Computer Science and Technology, Vol 18, N$^{\circ}$2, pp. 153-162, March 2003.
14. Springintveld J., Vaandrager F., and D'Argenio P., *Testing Timed Automata,* Theoretical Computer Science, 254, 2001.
15. Tretmans J., *Testing Concurrent Systems : A Formal Approach*, CONCUR'99 , 10th Int, conference on Concurrency Theory, pages 46-65, 1999.