Timed diagnostics and test case minimization for real-time systems

Ismail Berrada, Richard Castanet, Rachida Dssouli, Abdeslam En-Nouaary, Patrick Felix, Ferhat Khendek, Aziz Salah

Timed Diagnostics and Test Case Minimization for Real-Time Systems*

Ismaïl Berrada [1], Richard Castanet [1], Rachida Dssouli [2], Abdeslam En-Nouaary [2], Patrick Félix [1], Ferhat Khendek [2], and Aziz Salah [3]

[1] LaBRI - CNRS - UMR 5800 Université Bordeaux 1
33405 Talence cedex, France
{berrada,castanet,felix}@labri.fr

[2] Electrical and Computer Engineering, Concordia University
1455 de Maisonneuve W., Montreal
Quebec H3G 1M8, Canada
{dssouli,ennouaar,khendek}@ece.concordia.ca

[3] Département d'Informatique, Université du Québec à Montréal
201, avenue du Président-Kennedy, Montreal
Quebec H2X 3Y7, Canada
aziz.salah@uqam.ca

Abstract. Real-Time systems (RTS for short) are those systems whose behavior is time dependent.Reliability and safety are of paramount importance in designing and building RTS because a failure of a RTS put the public and/or the environment at risk. For the purpose of effective error reporting and testing, this paper considers the trace inclusion problem for RTS: given a path ρ (resp. ρ') of length n of a timed automaton A (resp. B), find whether the set of timed traces of ρ of length n are included in the set of timed traces of ρ' of length n such that ρ is known (the different constraints and clock updates of ρ are given) and ρ' is unknown (only the set of traces is given and the different constraints and clock updates of ρ' are unknown). The solution proposed to this problem is based on the identification of the timed diagnostics of bounds. These latter give a finite representation of the trace space corresponding to a path. Their number varies between 1 and $2 \times (n+1)$. The the problem of trace inclusion is then reduced to the inclusion of timed diagnostics of bounds. By the way, the paper shows how to use these results to minimize the number of test cases considered when testing RTS.

Keywords: Timed Input Output Automata, Trace Inclusion, Analyze, Black-Box Testing,Timed Diagnostic, Conformance.

* This research has been supported by the French RNTL project Avérroes and the Marie Curie RTN TAROT (MCRTN 505121).

1 Introduction

Nowadays, real-time systems (RTS for short) span various domains. We encounter them in our daily life such as telephone systems and video movies as well as in hospitals for patient monitoring and in airport for controlling air traffic. All these systems are time sensitive. That is, their behavior does not only depend on the logical result of the computation but also on the time at which the inputs are entered and the time at which the results are produced. It is well-known to RTS research community that the misbehavior of a RTS is generally due to the violation of the time constraints that govern the behavior of the system. Such malfunctioning may have catastrophic consequences on both human lives and environment. Therefore, it is very necessary to make sure that the implementation of a RTS is error-free before its deployment.

Two formal techniques are usually used to ensure error-free real-time systems, namely verification and testing. Verification deals with the specification of the system under consideration and aims at making sure that some functional and timing requirements are satisfied. However, testing deals with the implementation of the system being considered (usually referred to as Implementation Under Test or IUT for short) and checks its conformance to the specification of the system. To do so, three steps are required. First of all, test cases should be generated. Then, those test cases are executed against IUT and the reactions of the latter are logged. Finally, the verdict is concluded by analyzing the reactions of the IUT: if for each test case, the outputs of the IUT match the expected ones (i.e. those derived from the specification) then the IUT is said conform to its specification; otherwise the IUT is said faulty and the diagnosis process is started to locate and fix the fault.

In this paper, we study the following problem:

Trace inclusion problem. Consider a path ρ (resp. ρ') of length $n \in \mathbb{N}$ of a timed automaton A (resp. B). How to show that $TTrace(\rho) \subseteq TTrace(\rho')$ such that:

- ρ is known: the different constraints and clock updates of ρ are given.
- ρ' is unknown: only the set $TTrace(\rho')$ is given (the different constraints and clock updates of ρ' are unknown).

with $TTrace(\rho)$ are the timed traces of ρ of length n [1].

Our motivations for studying this problem are:

1. *Testing.* The testing research community distinguishes between three main testing strategies: black-box testing, white-box testing, and grey-box testing. Those testing strategies differ from each other on the way the test cases

[1] A formal definition of $TTrace()$ is given in section 3.

are generated. In the case of black-box testing of RTS, the code of IUT is unknown and only its timed traces are given. Black-box testing consists then of deriving test cases based solely on the specification of the IUT. The use of so called conformance relations give formal characterizations of conditions under which an IUT can be considered as conformant to its specification. Checking a conformance relation can be reduced, in general, to the trace inclusion problem between the implementation and the specification. By studying this problem, the paper gives necessary and sufficient conditions to check a conformance relation based on trace inclusion.

2. *Verification.* When checking a system against a property, a simple yes/no answer is not often satisfactory. Diagnostics are any kind of supplementary information (for instance, states, executions,...) which helps the user to understand why verification fails or succeeds. Diagnostics are important for the following reasons [19]:

   (a) Without them no confidence in the system's model can be gained. For instance, in case the property is not satisfied by the model, it might be that it is not the system which is wrong, but the modeling.

   (b) Even if the model is correct, the fault cannot be easily located without any guidance.

   In the case of RTS, there is a need for timed diagnostics, containing information both about the discrete state changes of the system, as well as the exact time delay between two discrete transitions.

The main contribution of this paper is the proposition of a solution to the trace inclusion problem. The proposed solution is based on the identification of the timed diagnostics of bounds corresponding to a path. These latter consider only the behaviors of the RTS on the constraint bounds. Their number varies between 1 and $2 \times (n+1)$, where n is the length of the path. The proof of the existence of those diagnostics 1) considers the constraint polyhedron corresponding to the set of constraints that each timed trace of the path has to satisfy and 2) uses some graph transformations that preserve the positivity of the graph cycles. By the way, the paper shows that the problem of trace inclusion can be then reduced to the inclusion of timed diagnostics of bounds. This result has a great influence on the test cases to considered while testing RTS. In fact, the paper provides a method to reduce the number of test cases used by using timed diagnostics of bounds.

The rest of this paper is structured as follows. Section 2 introduces the theoretical background of the paper. Section 3 presents the model of timed automata and its corresponding notations. Section 4 corresponds to the core of this paper and shows how to generate timed diagnostics from a path. Section 5, based on the result of section 4, outlines a method for minimizing the number of test cases considered while testing RTS. Section 6 presents the related work. Finally, we conclude and draw some perspectives in section 7.

2 Background

Through-out this paper, we write $\mathbb{R}$, $\mathbb{R}^{\geq 0}$, $\mathbb{N}$ for the sets of reals, nonnegative reals and naturals, respectively. $+\infty$ (resp. $-\infty$) is the positive infinity (resp. negative) such that: $t \in \mathbb{R}$, $-\infty \leq t \leq +\infty$, $t + (+\infty) = (+\infty) + t = +\infty$ and $t + (-\infty) = (-\infty) + t = -\infty$. $\overline{\mathbb{R}}$ is the set $\mathbb{R} \cup \{+\infty, -\infty\}$. For a set P, 2^P is the powerset of P and for a given order on P, $min(P)$ is the smaller element of P. Logical and and or or are written $\wedge$ and $\vee$, respectively.

2.1 Timed event and timed sequence

Let Σ be a finite set of symbols. As usual, Σ^* will denote the set of finite sequences and $\epsilon \in \Sigma^*$ the empty sequence. τ will denotes an action not in Σ and Σ_τ will denote the set $\Sigma \cup \{\tau\}$. Let σ be a sequence. Then $|\sigma|$ will denote the length of σ (i.e. number of elements), σ_i will denote the ith element of σ (i ranges from 0 to $|\sigma| - 1$). For $X \subseteq \Sigma$, $\sigma_{|X}$ will denote the sequence obtained by erasing from σ all symbols not in X (projection on X).

A timed event over Σ is a pair $u = (a, d)$ such that $a \in \Sigma$ and $d \in \mathbb{R}^{\geq 0}$. If a is interpreted to denote an event occurrence then t is interpreted as the timestamp of occurrence of a. $\mathbf{event}(u)$ will denote the untimed event a associated to u and $\mathbf{time}(u)$ the real d. A timed sequence $\sigma = (a_1, d_1)...(a_n, d_n)$ over Σ is a member of $(\Sigma \times \mathbb{R}^{\geq 0})^*$ such that the sequence of timestamps is monotonically increasing. For example, $\sigma = (a_1, 3).(a_2, 5)$ is a timed sequence, however $\sigma' = (a_1, 3).(a_2, 2)$ is not. The set of timed sequences over Σ is noted $TS(\Sigma)$.

2.2 Valuations and Polyhedra

Valuations. Let V be a finite set of variables ranged over $\mathbb{R}^{\geq 0}$. A valuation ν over V is a function $\nu : V \mapsto \mathbb{R}^{\geq 0}$ that assigns to each variable a real value. $\mathcal{V}(V)$ will denote the set of valuations over V. Let $X \subseteq V$, $d \in \mathbb{R}$ and $\nu \in \mathcal{V}(V)$. Then $\nu[X := 0]$ is the valuation defined by $\nu[X := 0](x) = \nu(x)$ if $x \notin X$ and $\nu[X := 0](x) = 0$ otherwise. Intuitively, $\nu[X := 0]$ assigns to each variable in X the value 0 and leaves the rest of variables unchanged. $\nu + d$ is the valuation such that for all $x \in V$, $(\nu + d)(x) = \nu(x) + d$. Intuitively, $\nu + d$ is obtained from ν by advancing all variables by d.

Polyhedra. An *atomic constraint* on V is an expression of the form $x \bowtie n$ and $x - y \bowtie m$ where $x, y \in V$ and $n, m \in \mathbb{N}$. The set of formulas that are finite conjunctions of atomic constraints (resp. constraints of the form $x \bowtie n$) will be denoted by $\Phi(V)$ (resp. $\Phi_I(V)$). Elements of $\Phi(V)$ are called *polyhedra*. We write **false** for $\emptyset$ and **true** (resp. **zero**) for $\bigwedge_{\forall x \in V} x \geq 0$ (resp. $\bigwedge_{\forall x \in V} x \geq 0 \wedge \bigwedge_{\forall x \in V} x \leq 0$). Let $\nu \in \mathcal{V}(V)$ and $Z \in \Phi(V)$. Then ν satisfies Z, noted $\nu \in Z$, if ν satisfies all constraints of Z. Z is bounded iff there is $d \in \mathbb{N}$ such that for all $\nu \in Z$, $\nu + d \notin Z$. Z is equivalent to Z', noted $Z \sim Z'$, iff for all $(\nu, \nu') \in Z \times Z'$, we have $\nu \in Z'$ and $\nu' \in Z$. Intuitively, Z and Z' represent the same portion of space.

2.3 Graphs and paths

Graphs. A directed labeled graph (DLG for short) G is a triple (V, E, w), where

- V is a finite set of elements $\{v_1, v_2, \cdots, v_k\}$ called vertexes,
- E is the set of couples of distinct elements of the cartesien product $V \times V$ called edges ($E = \{(v_i, v_j) | v_i, v_j \in V \wedge v_i \neq v_j\}$),
- $w_G : E \mapsto \overline{\mathbb{R}}$ is a function that assigns to each edge a weight.

The couple $(v_i, v_j) \in E$, noted $v_i \rightarrow v_j$, represents the edge of source v_i and target v_j. Note that G is a complete graph.

Paths. Let G be a DLG. A finite/infinite path p is a sequence of edges $e_1.e_2...e_n(...)$ (e_i is an edge). A path of length n is a path of n edges. The weight of p, noted $w(p)$, is the sum of the weights of $(e_i)_{i \in [1,n]}$: $w(p) = \sum_{i \in [1,n]} w(e_i)$. Let $e = v_i \rightarrow v_j$ be an edge. Then, $path(e)$ denote the set of paths of source v_i and target v_j. A cycle with root v_i is path from v_i to itself. An elementary cycle (e-cycle for short) is a cycle does not visit a vertex twice, except from the root vertex. Let $G = (V, E, w)$ be a DLG. G is said:

- nonnegative if the weight of each cycle of G is nonnegative. Formally, for all cycle c, $w(c) \geq 0$.
- minimal if the weight of each edge e is less then or equal to the weight of each path of $path(e)$. Formally, for all $e \in E$, for all $p \in path(e)$, $w(e) \leq w(p)$.

Next, we will use the term graph to denote a DLG.

3 Timed Automata

A clock is a variable that allows to record the passage of time. It is ranged over $\mathbb{R}^{\geq 0}$, and the only assignment allowed is clock reset of the form $x := 0$.

Timed automata [1]. A timed automaton (TA) A over Σ is a tuple $A = (L, l_0, \Sigma, C, I, \rightarrow)$ such that:

- L is a finite set of locations,
- l_0 is the initial location,
- Σ is an alphabet of actions,
- C is a finite set of clocks,
- $I : L \mapsto \Phi_I(C)$ is a mapping that assigns invariants to locations, and
- $\rightarrow \subseteq L \times \Phi(C) \times \Sigma_\tau \times 2^C \times L$ is the set of edges, where an edge contains a source, a label, a guard, a set of clocks to be reset with this edge, and a target.

The labels in Σ represent the observable interactions of a system; the special label $\tau \notin \Sigma$ represents an unobservable, internal action. A transition $t = (l, Z, a, r, l') \in \rightarrow$ is noted by $l \xrightarrow{Z,a,r} l'$. $\mathcal{TA}(\Sigma)$ will denote the set of all TAs over Σ.

Semantics. The semantics of a TA A is defined by associating a labeled transition system (LTS) $S(A) = (S, s_0, \Gamma, \rightarrow_A)$. A state of $S(A)$ is a pair $(l, \nu) \in Q$ such that l is a location of A and ν is valuation for C such that ν satisfies the invariant $I(l)$. The initial state s_0 of $S(A)$ is (l_0, ν) where $\nu \in \mathbf{zero}$. Labels Γ are included in $\Sigma_\tau \cup \{\epsilon(d) \mid d \in \mathbb{R}\}$ such that $\{\epsilon(d) \mid d \in \mathbb{R}\}$ corresponds to the elapse of time (Waiting d units of time is noted $\epsilon(d)$). There are two types of transitions in $S(A)$:

- *State change due to elapse of time*: for a state (l, ν) and $d \in \mathbb{R}^{\geq 0}$ $(l, \nu) \xrightarrow{\epsilon(d)}_A (l, \nu + d)$ if for all $0 \leq d' \leq d$, $\nu + d' \in I(l)$ (a timed transition).
- *State change due to a location-edge*: for a state (l, ν) and an edge (l, Z, a, r, l'), $(l, \nu) \xrightarrow{a}_A (l', \nu[r := 0])$ if $\nu \in Z$ and $\nu[r := 0] \in I(l')$ (a discrete transition).

Remark 1. We assume that TA we are working with have non Zenon behavior, i.e. the automata should not enforce infinity many events in a finite interval of time. This implies the absence of computation with infinitely many actions with finite cumulative delay. We also assume the requirement of progress, i.e. systems are supposed to execute forever.

Runs. Let $A = (L, l_0, \Sigma, C, I, \rightarrow)$ be a TA and $\sigma \in TS(\Sigma_\tau)$ be a timed sequence such that $|\sigma| = n$. A run r of A over σ, denoted by $(\overline{l}, \overline{\nu})$, is a finite sequence of the form:

$$r : (l_0, \nu_0) \xrightarrow{\sigma_1} (l_1, \nu_1) \ ... \ (l_{n-1}, \nu_{n-1}) \xrightarrow{\sigma_n} (l_n, \nu_n)$$

with $l_i \in L$, and $\nu_i \in \mathcal{V}(C)$, for all $i \in [0, n]$, satisfying the following requirements:

1. Initiation: for all $x \in C$, $\nu_0(x) = 0$.
2. Consecution: for all $i \in [1, n]$, there is an edge $t_i = (l_{i-1}, Z_i, event(\sigma_i), r_i, l_i)$ of A, such that:
   - $\nu_{i-1} + (time(\sigma_i) - time(\sigma_{i-1})) \in Z_i$.
   - ν_i equals to $(\nu_{i-1} + (time(\sigma_i) - time(\sigma_{i-1})))[r_i := 0]$.
   - $\nu_{i-1} + d \in I(s_{i-1})$ holds for all $0 \leq d \leq time(\sigma_i) - time(\sigma_{i-1})$.

Intuitively, at the initial state l_0, the values of clocks are defined to be zero. When the transition t_{i+1} from state l_i to l_{i+1} occurs, we use the value $\nu_i + time(\sigma_{i+1}) - time(\sigma_i)$ to check the clock constraints, however, at time $time(\sigma_{i+1})$, the value of clocks reseted in t_{i+1} are defined to be 0. By convention, $time(\sigma_0)$ is equal to 0.

Example 1. Consider the TA A of the Fig.1 and the timed sequence $(a, 2)(b, 3.7)$. The run corresponding to this sequence is given below. A clock interpretation is represented by listing the values $[x, y]$.

$$(l_1, [0, 0]) \xrightarrow{(a,2)} (l_2, [2, 0]) \xrightarrow{(b,3.7)} (l_3, [3.7, 1.7]).$$

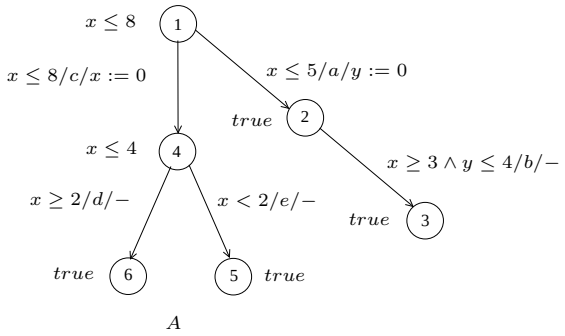

Fig. 1. Timed automata.

The set of timed sequences of A, noted $Run(A)$, is defined by:

$$Run(A) = \{\sigma \mid A \text{ has a run over } \sigma \in TS(\Sigma_\tau)\}.$$

The set of timed traces of A, noted $TTrace(A)$, is defined by:

$$TTrace(A) = \{\sigma \mid \exists \sigma' \in Run(A),\ \sigma = \sigma'_{|\Sigma}\}.$$

Finally, for a path ρ of A, we use $TTrace(\rho)$ to denote the set of timed traces of length n of the automaton A_ρ induced by ρ [2].

4 Timed diagnostic of a path

Let A be a TA. A timed diagnostic of A is an element of $TTrace(A)$. In this part, we provide an approach to extract timed diagnostics from a given path ρ. As we will see in the next section, timed diagnostics can be used to improve the confidence in the system's model or to test a RTS.

The idea behind our approach is as follows: to the path ρ, we can associate a constraint polyhedron Z_ρ which define the set of constraints to be satisfied by each element of $TTrace(\rho)$. The polyhedron Z_ρ can be then represented as a graph G_ρ. This graph has nonnegative cycles iff the polyhedron Z_ρ is not empty. By defining some transformations on the graph G_ρ that preserve its positivity, we identify some timed diagnostics of ρ called the *timed diagnostics of bounds*. These latter give a finite representation of the trace space of ρ.

For the rest of this paper, $\rho = t_1 \cdots t_n$ will denote a path of A such that $t_i = (l_{i-1}, Z_i, a_i, r_i, l_i)$, for all $i \in [1, n]$. $V = \{v_1, v_2, \cdots, v_n\}$ will denote a set of variables ranged over $\mathbb{R}^{\geq 0}$, $V_0 = V \cup \{v_0\}$ the set V extended with a fictive variable v_0 equals to 0 all time. We will confound elements of $\Phi(V)$ with elements of $\Phi(V_0)$ and a valuation over V with a valuation over V_0.

[2] A_ρ has the same states and transitions as ρ.

4.1 Graph-theoretic formulation

Constraint polyhedron. Let $\sigma = (a_1, d_1) \cdots (a_n, d_n) \in TTrace(\rho)$. According to the definition of $TTrace(\rho)$, the difference instants $(d_i)_{i\in[1,n]}$ satisfy a set of constraints related to the transitions of ρ as shown in next example.

Example 2. Consider the path ρ of the automaton of Fig.1, defined by:

$$l_1 \xrightarrow{x\leq 5/a/y:=0} l_2 \xrightarrow{x\geq 3 \wedge y\leq 4/b/-} l_3.$$

such that $I(l_1) = x \leq 8$ and $I(l_1) = I(l_2) = true$. Let v_i, $i \in [1, 2]$, be the instant of firing, according to a global clock, of the transition of source l_{i-1} and target l_i. Then, the location l_2 is reachable if only if the system:

$$S = \begin{cases} 0 \leq v_1 \leq v_2 \\ v_1 \leq 8 \\ v_1 \leq 5 \\ v_2 \geq 3 \\ v_2 - v_1 \leq 4 \end{cases}$$

has a solution. In other words, the polyhedron Z_ρ defined by:

$$Z_\rho \; = \; 0 \leq v_1 \wedge v_1 \leq v_2 \wedge v_1 \leq 5 \wedge v_2 \geq 3 \wedge v_2 - v_1 \leq 4$$

is not empty. By consequence, $\sigma = (a, d_1).(b, d_2) \in TTrace(\rho)$ iff the valuation ν defined by $\nu(v_1) = d_1$, $\nu(v_2) = d_2$ is in Z_ρ. □

In a more general case, we can associate to ρ, a constraint polyhedron Z_ρ over variables $V_0 = \{v_0, v_1, \cdots, v_n\}$ such that: $\sigma \in TTrace(\rho)$ iff the valuation $\nu \in \mathcal{V}(V_0)$ defined by $\nu(v_i) = time(\sigma_i)$ is in Z_ρ, for all $\forall i \in [0, n]$. The construction of Z_ρ is illustrated in the Fig. 2.
Let $last_i^x$ be the index of the transition where the clock x has been reset most recently before i ($i \in [0, n]$). Recall that $\bowtie \in \{\leq, \geq\}$, $V_0 = \{v_0, \cdots, v_n\}$ and all clocks are reseted at the initial location l_0. For all $i \in [0, n]$, v_i represents the instant of firing t_i according to a global clock. By convention, we assume the existence of a transition t_0 where all clocks are rested at instant $v_0 = 0$. Initially, Z_ρ is equal to *true* (line 1) and the suite $(v_i)_{i\in[0,n]}$ is monotonically increasing (line 3). At step $i \in [1, n]$ of the algorithm, if t_i has term over x of the form $x \bowtie k$ (line 5) then, the constraint $v_i - v_j \bowtie k$ is added to Z_ρ, where x is last rest in j (line 6). If t_i has term of the form $x - y \bowtie k$ (line 7) then, the constraint $v_p - v_q \bowtie k$ is added to Z_ρ, where $last_i^x = q$ et $last_i^y = p$ (line 8). In fact, the time elapsed since the last reset of x (resp. y) in the transition t_q (resp. t_p) is equal to $v_i - v_q$ (resp. $v_i - v_p$). Thus, $x - y = (v_i - v_q) - (v_i - v_p) = v_p - v_q$. Finally, the same approach is applied to the invariant of a state (lines 9-10).

Input : $\rho = t_1 \cdots t_n$, $t_i = (l_{i-1}, Z_i, a_i, r_i, l_i)$, for all $i \in [1, n]$.
Output : $Z_\rho \in \Phi(V_0)$
Begin
1. $Z_\rho := true$
2. **For** $i := 1$ **to** n **do**
3. $Z_\rho := Z_\rho \wedge v_{i-1} \leq v_i$
4. **For** $x \in C$ **do**
5. **If** the guard of t_i has a term of the form $x \bowtie k$ **then**
6. $Z_\rho := Z_\rho \wedge v_i - v_j \bowtie k$, where $j = last_i^x$.
7. **If** the guard of t_i has a term of the form $x - y \bowtie k$ **then**
8. $Z_\rho := Z_\rho \wedge v_p - v_q \bowtie k$, where $q = last_i^x$ and $p = last_i^y$.
9. **If** the invariant of l_{i-1} has a term of the form $x \bowtie k$ **then**
10. $Z_\rho := Z_\rho \wedge v_i - v_j \bowtie k$, where $j = last_i^x$.

End

Fig. 2. Constraint polyhedron.

Convention. Without losing generality and for simplicity reasons, we assume that Z_ρ can be written syntaxely as:

$$Z_\rho = \bigwedge_{v_i, v_j \in V_0, v_i \neq v_j} (v_i - v_j \leq l_{ij}), \quad l_{ij} \in \overline{\mathbb{R}}.$$

In fact, a constraint of the form $v_i \leq c$ can be written as $v_i - v_0 \leq c$ (v_0 is equal to 0) and $v_i \leq c \wedge v_i \leq c'$ can be written as $v_i - v_0 \leq min(c, c')$. Furthermore, if v_i does not have upper bound in Z_ρ, then we can add the constraint $v_i - v_0 \leq +\infty$. These remarks hold for a constraint of the form $v_i - v_j \leq c$.

Constraint graph. Let $Z_\rho = \bigwedge_{v_i, v_j \in V_0, v_i \neq v_j} (v_i - v_j \leq l_{ij})$ be the constraint polyhedron associated to ρ. The constraint graph $G_\rho = (V_0, E, w)$ associated to Z_ρ is the graph defined by:

$$w(v_j \rightarrow v_i) = l_{ij} \wedge v_j \rightarrow v_i \in E \Longleftrightarrow v_i - v_j \leq l_{ij} \text{ is a term of } Z_\rho.$$

Theorem 1. *$Z_\rho \not\sim$ **false** iff G_ρ is nonnegative.* □

Intuitively, the set $TTrace(\rho)$ is not empty iff the constraint graph G_ρ does not contain negative cycles. The proof of the theorem can be found in [8]. We will said that Z_ρ is in its canonical form if the graph G_ρ is minimal.

Until now, we have showed that we can associate to ρ a constraint graph G_ρ such that $TTrace(\rho)$ is not empty iff G_ρ does not contain negative cycles. In order to proof the main theorem of subsection 4.3, we need to define some transformation on G_ρ. Subsection 4.2 introduces three transformations that keep the positivity and/or the minimality of the transformed graph. To save space we omitted the proof of lemmas in this subsection, but they are based on the comparison of the weights of e-cycles, and can be found in [3].

4.2 Graph transformation

Transformation $m()$. Let $m()$ the function that associate to each graph $G = (V, E, w_G)$ the graph $G' = (V, E, w_{G'})$ such that: for each edge $v_p \to v_q \in E$,

$$w_{G'}(v_p \to v_q) = min(\{w_G(p) \,|\, p \in path(v_p \to v_q)\}).$$

Intuitively, the weight of $e = v_p \to v_q$, in G', is equal to the minimal weight, in G, of all path of source v_p and target v_q. This weight is either reached by a path, i.e. there exist $p \in path(e)$ such that $w_{G'}(e) = w_G(p)$, or $w_{G'}(e) = -\infty$ when $\{w_G(p) \,|\, p \in path(e)\}$ is not bounded. Note that, G' is a minimal graph. This transformation preserves the positivity of cycles as established in the next lemma.

Lemma 1. *G is a nonnegative graph iff $m(G)$ is not.* □

Transformation $R_{i\to *}()$. Let $R_{i\to *}()$ the function that associate to each graph $G = (V, E, w_G)$ the graph $G' = (V, E, w_{G'})$ such that: for each edge $v_p \to v_q \in E$,

$$w_{G'}(v_p \to v_q) = \begin{cases} -w_G(v_q \to v_p) & if\ v_q = v_i \\ w_G(v_p \to v_q) & otherwise \end{cases}$$

Intuitively, if $v_p \to v_q$ is not an incoming edge of the vertex v_i then, this edge keeps the same weight in G and G'. Otherwise, the weight of $v_p \to v_q$ is replaced, in G', by the opposite weight of the outgoing edge $v_i \to v_q$ of v_i. The next lemma establishes some properties of this transformation related to the minimality and the positivity of the transformed graph.

Lemma 2. *Let G be a nonnegative graph and $i \in [1, n]$. Consider the graph $G' = m(R_{i\to *}(G))$. Then,*

1. *$R_{i\to *}(G)$ is a nonnegative graph.*
2. *If G is minimal then, for all edge $v_p \to v_q \in E$:*

$$w_{G'}(v_p \to v_q) = \begin{cases} w_G(v_p \to v_q) & if\ v_p = v_i \\ -w_G(v_q \to v_p) & if\ v_q = v_i \\ -w_G(v_i \to v_p) + w_G(v_i \to v_q) & otherwise \end{cases}$$

□

Intuitively, the transformation $R_{i\to *}()$ preserves the positivity of cycles. When G is minimal and nonnegative, the second point of the lemma gives a method to compute the minimal graph associated to $R_{i\to *}(G)$ using the weights of G.

Transformation $R_{*\to i}()$. This transformation is similar to $R_{i\to *}()$. For a graph $G = (V, E, w_G)$, the transformed graph $G' = (V, E, w_{G'})$ is defined by: for each edge $v_p \to v_q \in E$,

$$w_{G'}(v_p \to v_q) = \begin{cases} -w_G(v_q \to v_p) & if\ v_p = v_i \\ w_G(v_p \to v_q) & otherwise \end{cases}$$

Intuitively, the only difference between G and G' is in the weights of outgoing edges of vertex v_i: for all $v_i \to v_q \in E$, $w_{G'}(v_i \to v_q)$ is equal to the opposite weight of $w_G(v_q \to v_i)$. The next lemma reports properties similar to those of $R_{i\to *}(G)$.

Lemma 3. *Let G be a nonnegative graph and $i \in [1, n]$. Consider the graph $G' = m(R_{*\to i}(G))$. Then,*

1. *$R_{*\to i}(G)$ is a nonnegative graph.*
2. *If G is minimal then, for all edge $v_p \to v_q \in E$:*

$$w_{G'}(v_p \to v_q) = \begin{cases} w_G(v_p \to v_q) & if\ v_q = v_i \\ -w_G(v_q \to v_p) & if\ v_p = v_i \\ w_G(v_p \to v_i) - w_G(v_q \to v_i) & otherwise \end{cases}$$

□

4.3 Main result

Until now, we have defined the constraint polyhedron Z_ρ (resp. graph G_ρ) associated to ρ and we have defined some transformations over positive graphs. These results allow us to introduce the next theorem. Before that, let $m(G_\rho) = (V_0, E, w_m)$ the minimal graph of G_ρ. The canonical form of Z_ρ, noted $cf(Z_\rho)$, is the polyhedron defined by:

$$cf(Z) = \bigwedge_{\forall v_i, v_j \in V_0,\, v_i \neq v_j} (v_i - v_j \leq l_{ij}) \text{ such that } v_j \to v_i \in E,\ w_m(v_j \to v_i) = l_{ij}.$$

Note that $cf(Z_\rho)$ and Z_ρ represent the same space portion.

Theorem 2. *Let ρ be a path and $cf(Z_\rho) = \bigwedge_{v_i, v_j \in V_0,\, v_i \neq v_j} (v_i - v_j \leq l_{ij})$ be the canonical form of its constraint polyhedron. Assume that Z_ρ is bounded and not empty ($Z_\rho \not\sim false$). Then, for all $k \in [0, n]$:*

1. *There is a valuation $\nu_k^M(Z_\rho)$ of Z_ρ such that: for all $i \in [0, n]$, $i \neq k$, $\nu_k^M(Z_\rho)(v_i) - \nu_k^M(Z_\rho)(v_k) = l_{ik}$.*
2. *There is a valuation $\nu_k^m(Z_\rho)$ of Z_ρ such that: for all $i \in [0, n]$, $i \neq k$, $\nu_k^m(Z_\rho)(v_k) - \nu_k^m(Z_\rho)(v_i) = l_{ki}$.* □

Intuitively, if Z_ρ is bounded and nonempty, then for each variable $v_k \in V_0$, there is a valuation $\nu_k^M(Z_\rho)$ (resp. $\nu_k^m(Z_\rho)$) which reaches the bounds $(l_{ik})_{k\neq i, i\in[0,n]}$ (resp. $(l_{ki})_{k\neq i, i\in[0,n]}$) of $cf(Z_\rho)$ constraints, where v_k is a right (resp. left) member. We have assumed that Z_ρ is bounded to ensure the existence of $\nu_k^M(Z)$. The valuations $\nu_k^m(Z)$ exist even Z_ρ is not bounded because variables of V_0 are ranged over $\mathbb{R}^{\geq 0}$.

Proof. To proof the theorem, it is equivalent to show that, for all $k \in [0, n]$, the polyhedra:

$$Z_k^M = \bigwedge_{v_i V_0,\, v_i \neq v_k} (v_i - v_k \leq l_{ik} \wedge v_k - v_i \leq -l_{ik}) \wedge \bigwedge_{v_i, v_j \in V_0,\, v_i \neq v_j \neq v_k} (v_i - v_j \leq l_{ij})$$

and

$$Z_k^m = \bigwedge_{v_i V_0,\, v_i \neq v_k} (v_k - v_i \leq l_{ki} \wedge v_i - v_k \leq -l_{ki}) \wedge \bigwedge_{v_i, v_j \in V_0,\, v_i \neq v_j \neq v_k} (v_i - v_j \leq l_{ij})$$

are not empty sets ($Z_k^M \not\sim false$ and $Z_k^m \not\sim false$). In fact, let G_ρ be the constraint graph of $cf(Z_\rho)$ and $k \in [0, n]$. $cf(Z_\rho)$ is canonical then G_ρ is minimal. $Z_\rho \not\sim false$ implies that G_ρ is a nonnegative graph (theorem 1). Now, one can notice that the constraint graph $G(Z_k^M)$ (resp. $G(Z_k^m)$) associated to Z_k^M (resp. Z_k^m) is nothing else than the graph obtained from G_ρ by the transformation $R_{k\to *}()$ (resp.$R_{*\to k}()$) defined above: $G(Z_k^M) = R_{k\to *}(G_\rho)$ et $G(Z_k^m) = R_{*\to k}(G_\rho)$. So, according to the first point of the lemma 2 (resp. lemma 3), we deduce that $G(Z_k^M)$ (resp. $G(Z_k^m)$) is a nonnegative graph and by consequence, $Z_k^M =\not\sim false$ (resp. $Z_k^m =\not\sim false$). Furthermore, the second point of lemma 2 (resp, lemma 3) gives a method to compute the canonical form of Z_k^M (resp. Z_k^m). □

Computation of $\nu_k^M(Z_\rho)$ and $\nu_k^m(Z_\rho)$. Theorem 2 establishes the existence of valuations $(\nu_k^M(Z_\rho))_{k\in[0,n]}$ and $\nu_k^m(Z_\rho)_{k\in[0,n]}$, and their unicity. Having in mind that $v_0 = 0$, a direct application of this theorem gives: for all $k \in [0, n]$,

1. $\nu_k^M(Z_\rho)$ is the valuation defined by:
   - If $k = 0$ then $\nu_k^M(Z_\rho)(v_i) = l_{i0}$.
   - Else $\nu_k^M(Z_\rho)(v_i) = -l_{0k} + l_{ik}$ and $\nu_k^M(Z_\rho)(v_k) = -l_{0k}$

   for all $i \in [1, n]$, $i \neq k$.
2. $\nu_k^m(Z_\rho)$ is the valuation defined by:
   - If $k = 0$ then $\nu_k^m(Z_\rho)(v_i) = -l_{0i}$.
   - Else $\nu_k^m(Z_\rho)(v_i) = l_{k0} - l_{ki}$ and $\nu_k^m(Z_\rho)(v_k) = l_{k0}$

   for all $i \in [1, n]$, $i \neq k$.

Example 3. Let $Z_\rho = 0 \leq v_1 \wedge v_1 \leq v_2 \wedge v_1 \leq 5 \wedge v_2 \geq 3 \wedge v_2 - v_1 \leq 4$ be the constraint polyhedron of the example 2. Z_ρ is bounded. Its canonical form is defined by: $cl(Z_\rho) = (v_2 - v_0 \leq 5) \wedge (v_0 - v_2 \leq 0) \wedge (v_1 - v_0 \leq 9) \wedge (v_0 - v_1 \leq -3) \wedge (v_1 - v_2 \leq 4) \wedge (v_2 - v_1 \leq 2)$. Then,

$$\nu_0^M(Z_\rho) = \begin{pmatrix} 9 \\ 5 \end{pmatrix} \begin{matrix} v_1 \\ v_2 \end{matrix} \quad \nu_1^M(Z_\rho) = \begin{pmatrix} 3 \\ 5 \end{pmatrix} \quad \nu_2^M(Z_\rho) = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$
$$\nu_0^m(Z_\rho) = \begin{pmatrix} 3 \\ 0 \end{pmatrix} \quad \nu_1^m(Z_\rho) = \begin{pmatrix} 9 \\ 5 \end{pmatrix} \quad \nu_2^m(Z_\rho) = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

□

Now, consider the two suites of timed sequences $(\sigma^{Mk})_{k\in[0,n]}$ and $(\sigma^{mk})_{k\in[0,n]}$ defined by:

- $\sigma^{Mk} = (a_1, \nu_k^M(Z_\rho)(v_1)) \cdots (a_n, \nu_k^M(Z_\rho)(v_n))$.
- $\sigma^{mk} = (a_1, \nu_k^m(Z_\rho)(v_1)) \cdots (a_n, \nu_k^m(Z_\rho)(v_n))$.

Note that, for all $k \in [0, n]$, $\sigma^{Mk} \in TTrace(\rho)$ and $\sigma^{mk} \in TTrace(\rho)$.

Definition 1. *The timed sequences $(\sigma^{Mk})_{k\in[0,n]}$ and $(\sigma^{mk})_{k\in[0,n]}$ are called the timed diagnostics of bounds associated to ρ. σ^{Mk} (resp. σ^{mk}) is called a timed diagnostic of minimal bounds (resp. maximal).* □

The number of timed diagnostics of bounds varies between 1 et $2 \times (n+1)$ (n is the length of the path). The complexity of computing σ^{Mk} or σ^{mk} from $cf(Z_\rho)$ is $O(n)$. The computation of the canonical form of a polyhedron depends on the data structures used. The algorithm given in [8] allow to compute this form and to test if a polyhedron is empty. Its complexity is $O(n^3)$.

5 Application: error reporting and testing

5.1 Error reporting

Timed diagnostics of bounds can be used to report counterexamples during timing verification: once the verification tool determines the sequence of transitions that leads to a violation of the safety property, the timed diagnostics of bounds provide greater diagnostic feedback.

5.2 Testing

Trace inclusion. To show that $TTrace(\rho) \subseteq TTrace(\rho)$ is equivalent to show that $cf(Z_\rho) \subseteq cf(Z_{\rho'})$. We consider here the case where Z_ρ is known and only the set $TTrace(\rho')$ is known. We assume that Z_ρ is bounded and not empty.

Corollary 1. *$TTrace(\rho) \subseteq TTrace(\rho')$ if and only if $\sigma^{Mk} \in TTrace(\rho')$ and $\sigma^{mk} \in TTrace(\rho')$, for all $k \in [0, n]$.* □

Intuitively, the corollary gives the necessary and sufficient conditions to show that $TTrace(\rho) \subseteq TTrace(\rho')$. In fact, it is sufficient to show that timed diagnostics of bounds of $TTrace(\rho)$ are also timed diagnostics of $TTrace(\rho')$.

Proof. The proof is a consequence of $TTrace(\rho) \subseteq TTrace(\rho')$ iff $cf(Z_\rho) \subseteq cf(Z_{\rho'})$. As $(\nu_k^M(Z_\rho))_{k\in[0,n]}$ and $(\nu_k^m(Z_\rho))_{k\in[0,n]}$ reach all bounds of $cf(Z_\rho)$, then if $\nu_k^M(Z_\rho) \in Z_{\rho'}$ and $\nu_k^m(Z_\rho) \in cf(Z_{\rho'})$, we can deduce that bounds of $cf(Z_\rho)$ are less than bounds of $cf(Z_{\rho'})$. The density and convexity properties of sets $cf(Z_\rho)$ and $cf(Z_{\rho'})$ imply that all $\nu \in cf(Z_\rho)$ is also in $cf(Z_{\rho'})$. □

Test cases and Testers. A test case (test for short) is an experience performed on the IUT by an observer (the tester). In the case of RTS, there are different types of tests, depending on the capabilities of the tester to observe and react to event. Analog-clock tests [9, 13] can measure precisely the real-time delay between observed actions. Digital-clock tests can only count how many "ticks" of a finite-granularity clock have occurred between two actions. Analog-clock testers can measure real-time precisely, but there are difficult (if not impossible) to implement for real-time IUT. Digital-clock testers have access to a periodic clock/counter and are implementable for any IUT. However, they can announce a "Pass" verdict when it is "Fail" (the reception of an event "a" after 2.7 units of time and the same reception after 2.8 units of time will look the same for a digital-clock tester). The use of a digital-clock testers does not mean the discretization of time, the specification is still dense-time but the capabilities of the tester are discrete-time. In this paper, we consider digital-clock testers. Furthermore, we will consider static tests, i.e. the response of the digital-clock tester is the same and known at advance.

Digital-clock test derivation. Our goal her is not to provide a complete method to derive digital-clock tests, but only to give the broad lines of an approach to build statically digital-clock tests. In [3] the reader can found a complete algorithm to derive tests for digital-clock/analog-clock testers.

Simulation graph [19]. Tripakis defines a number of different abstractions for timed automata and study the properties they preserve. These abstractions are based on the simulation graph, which is built by forward reachability and preserves all linear properties. In the simulation graph, the passage of time is hidden and only the discrete-state changes can be observed. Let A be a TA, $S = (l, Z)$ be a zone (i.e. a location l of A and a polyhedron Z), and $t = (l, Z', a, r, l')$ be a transition of A

$$\mathbf{post}_c(S, t) = \{(l', close(((Z \cap Z')[r := 0])^\uparrow, c))\}$$

Intuitively, $post()$ contains all states (and their c-closure) that can be reached from state in $S.$, by taking transition t and letting some time pass. Given the initial location l_0 of A, the simulation graph $S(A, c)$ (c is a natural constant greater than the closure of A) is generated using an depth-first search starting from $S_0 = (l_0, \mathbf{zero}^\uparrow)$ and generating for each vertex $S = (l, Z)$ in the stack, the successors $S' = post_c(S, t)$, for each transition $t = (l, g, z, r, l')$ of source l in A. The exploration of the branch leading to S_i is stopped if: either $S_i = \emptyset$

or there is a previously generated vertex $S_i \subset S'$. Otherwise, S_i is added to the set of vertexes and $S \xrightarrow{a} S_i$ to the set of edges of the simulation graph. It has been shown that $S(A,c)$ is finite and there is a run of A from l_0 to l_f if in the simulation graph there in a vertex $S = (l_f, -)$. Moreover, for each path $S_0 = (l_0, Z_0) \xrightarrow{a_1} S_1 = (l_1, Z_1) ... \xrightarrow{a_n} S_n = (l_n, Z_n)$ in the simulation graph, there is a run $r = (\overline{l}, \overline{\nu})$ of A such that $\nu_i \in Z_i$, for all $i \in [0, n]$, and vice versa.

Test derivation. The idea of our approach to generate tests is to use the simulation graph. As we have said, $S(A,c)$ gives a finite representation of the reachable state space; each path of it has a run of A and each run of A is inscribed in path of $S(A,c)$. Classical methods for untimed systems can be used, in general, to derive a set of path from $S(A,c)$. Let $ATS(A)$ a set of paths derived from $S(A,c)$ with respect to a given coverage criterion (states, transitions,...). Element $ATS(A)$ can not be used directly to test a given implementation of A because they are abstract. Each path $\rho \in ATS(A)$ define a set of timed traces $TTrace(\rho)$. Corollary 1 has a great influence on test cases considered for the path ρ. In fact, according to this corollary, the number of distinct tests required for the trace inclusion is between 1 and $2 \times n$ test cases corresponding to the timed diagnostics of bounds of ρ.

Thus, compared to other works, our approach does not suffer from the explosion problem, since we use only tests that meet the timed diagnostics of bounds.

6 Related Work

Regarding works in analyzing RTS, [2] have studied the problem of timestamp generation: given a path of the automaton, we wish to check if there is a corresponding execution, and if so, generate a possible sequence of time values at which the individual transitions are traversed. The solution proposed consists in computing one timed diagnostic corresponding to the minimal accumulate delay run. The approach of Tripakis for generating timed diagnostics presented in [19, 20] was based in a symbolic analysis. The solution proposed uses the simulation graph to generate abstract paths. For each abstract path, the authors chose randomly the instant of firing the transitions. In [14], the authors show that the existence of timed diagnostics associated to a symbolic path, but do not provide a method to compute them. In [10], the authors propose to use the verification tool Uppaal to generate the optimal timed diagnostic corresponding to a path. In [16], the authors propose several algorithms to compute the minimal timed diagnostic that reach a given state.

To our knowledge, the identification of the timed diagnostics of bounds is a new result, and the problem of trace inclusion has not been studied before.

Regarding works on testing, [13] propose a method to derive analog/digital-clock test cases. The approach proposed was based on a symbolic analysis. However, the propose method has the following problems:

1. The generation method for digital tests considers "ticks" of clocks as an observable event. As a consequence of this choice, is the presence of long chains of ticks in the test cases generated as reported in [13]: "Digital-clock test can sometimes grow large because they contain a number of "chains" of ticks". The authors propose then a heuristic to compact chains of ticks, but this heuristic does not give always minimal test and it is not trivial: "Reducing the size of test representations is a non-trivial problem in general, related to compression and algorithmic complexity theory".
2. The choice a the instants of emission is randomly. By consequence, the number of test cases generated is important [13] : "We have obtained 68, 180, 591, and 2243 tests for depth levels 5, 6 , 7 and 8, respectively"

The solution that we have proposed was based in the use of timed diagnostics of bounds and does not suffer from these problems. As we have show, the inclusion of specification traces in the implementation traces can be established by the timed diagnostics of bounds.

An extension of test theory for Mealy machines in the case of dense RTS was proposed by Springintveld et al. [18]. The authors suggested to perform a kind of discretization of the region graph model. Another work generating test sequences for a discretized deterministic timed automaton is given by En-Nouaary et al. in [7]. The authors propose to build a grid automaton from the region graph, and use a Wp method for the generation assuring a good coverage of the initial specification, but the number of generated test cases can be large. In [5], an implicit clock is used, the time is discrete and the proposed model is a temporized transition system. In [12], the authors have chosen as model temporized automata with discrete time. The model is transformed into automaton without time, but with two special events on clocks: set and expire. In [6], the system specification is based on a constraint graph. From a fault model, the authors define test criteria and generate test cases according to the test criteria. Since constraint graph is used as a model, only delays can be expressed between two successive events, and the coverage of faults cannot be complete. In [15], the generation of test cases is produced from logic formula (time is expressed by using two constructors: future and past). A unique clock is used and the temporal domain is discrete. [11] propose a generation method based on must/may traceability. The authors propose to test first, the correctness of the implementation of states and transitions. For that, they transform the specification into a FSM, and use the UIOv-method to derive test cases. [17] use symbolic analysis for event-recording automata inspired by the Uppaal model-checker.

All of these methods successfully generate timed test cases but most of them suffer from an exorbitant number of test cases.

7 Conclusion

In this paper, we have studied the trace inclusion problem of RTS. Our solution was based on the identification of the timed diagnostics of bounds corresponding

to a given path. The trace inclusion problem is then reduced to the inclusion of the timed diagnostics of bounds between.

Timed diagnostics of bounds can be used to report counterexamples during timing verification. Regading testing, our solution was based on the use of the simulation graph of Tripakis to derive abstract tests, and to generate from each abstract test, the set of test cases corresponding to the timed diagnostics of bounds. In this way, the number of tests generated still lower.

To have a complete coverage of the timed trace space of the specification while testing (according to corollary 1), the assumption of the event determinism of the specification and the implementation is required. This model is quite restrictive, and the generalization will benefit many RST. Espacially, the determinism assumption may be broken by the on-the-fly determinisation techniques.

References

1. R. Alur and D. Dill. A theory of timed automata, *Theoretical Computer Science*, 126:183-235, 1994.
2. R. Alur, R. Kurshan and M. Viswanathan. Membership problems for timed and hybrid automata. 19th IEEE Real-Time Systems Symposium, 1998.
3. Ismail Berrada. Modélisation, Analyse et Test des Systèmes Communicants à Contraintes Temporelles : Vers une Approche Ouvert du Test. *Phd thesis, Université Bordeaux 1*, Bordeaux, France, 14 December, 2005.
4. Laura Brandán and Ed Brinksma. A test generation framework for quiescent real-time systems. *Proceedings of the 4rd International Workshop on Formal Approaches to Testing of Software, FATES2004*, Linz, Austria September 21, 2004.
5. Rachel Cardell-Oliver. Conformance testing of real-time systems with timed automata specifications. *Formal Aspects of Computing*, 12(5):350-371, 2000.
6. Duncan Clarke and Insup Lee. Automatic test generation for the analysis of a real-time system: case study. In *3rd IEEE Real-Time Technology and Applications Symposium*,
7. A. En-Nouaary, R. Dssouli, F. Khenedek and A. Elqortobi. Timed test cases generation based on state characterization technique. *In 19th IEEE Real Time Systems Symposium (RTSS'98)*, Madrid, Spain, 1998.
8. Robert W. Floyd. Algorithm 97 (shortest path). *Communications of the ACM*,18(3):165-172, 1964.
9. T. Henzinger, Z. Manna and A. Pnueli. What good are digital clocks?. *ICALP'92*, LNCS 623, 1992.
10. Anders Hessel, Kim G. Larsen, Brian Nielson, Paul Pettersson and Arne Skou. Time-optimal real-time test case generation using Uppaal. *In FATES2003*, Montreal, Quebec, Canada, October, LNCS 2931, pp. 118-135, Springer.
11. T. Higashino, A. Nakata, K. Taniguchi and A. Cavalli. Generating test cases for a timed i/o automaton model. *TESTCOM99*, Budapest, Hungary, September 1999.
12. A. Koumsi, M. Akalay, R. Dssouli, A. En-Nouaary, L. Granger. An approach for testing real time protocols, *TESTCOM*, Ottawa, Canada, 2000.
13. M. Krichen and S. Tripakis. Black-box conformance testing for real-time systems. *In SPIN 2004*, Spring-Verlag Heidelberg, pp. 109-126, 2004.
14. Kim G. Larsen, Paul Pettersson and Wang Yi. Diagnostic model-checking for real-time systems. *In Proc. of WVCHS III*, number 1066 in LNCS, pp. 575-586. Springer-Verlag, October 1995.

15. Dino Mandrioli, Sandro Morasca and Angelo Morzenti. Generating test cases for real-time systems from logic specifications. *ACM Transactions on Computer Systems*, 13(4):365-398, 1995.
16. P. Niebert, S. Tripakis and S. Yovine. Minimum-time reachability for timed automata. *In Mediterranean Conference on Control and Automation*, 2000.
17. B. Neilson ans A. Skou. Automated test generation for timed automata. *TACAS'01*, LNCS 2031, Springer 2001.
18. Jan Springintveld, Frits Vaandrager and Pedro R. D'Argenio. Testing timed automata. *Theoretical Computer Science*, 252(1-2):225-257, March 2001.
19. Stavros Tripakis. The formal analysis of timed systems in practice. PhD thesis, Université Joseph Fourier de Grenoble, 1998.
20. Stavros Tripakis. Timed diagnostics for reachability properties. *In Tools and Algorithms for the Construction and Analysis of Systems, TACAS'99*, Amsterdam, Holland, 1999.