Test Case Minimization for Real-Time Systems Using Timed Bound Traces*

Ismaïl Berrada[1], Richard Castanet[1], Patrick Félix[1], and Aziz Salah[2]

[1] LaBRI - CNRS - UMR 5800 Université Bordeaux 1,
33405 Talence cedex, France
{berrada, castanet, felix}@labri.fr
[2] 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 an RTS puts 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 A is known but not B. We assume that the traces of ρ' are only defined by a decision procedure.

The proposed solution is based on the identification of a set of timed bound traces. The latter gives a finite representation of the trace space of a path. The number of these timed bounds varies between 1 and $2 \times (n+1)$. The trace inclusion problem is then reduced to the inclusion of timed bound traces. The paper shows also how these results can be used to reduce the number of test cases for an RTS.

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

1 Introduction

Nowadays, real-time systems (RTS for short) span various domains of our daily life such as telephone systems, patient monitoring systems, and air traffic control. All these systems are time sensitive because their behavior does not only depend on the logical result of the computation but also on the time at which the inputs and outputs are observed. It is well-known to RTS research community that the misbehavior of an RTS is generally due to the violation of time

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

constraints. Such malfunctioning may have catastrophic consequences on both human lives and the environment. Therefore, it is very necessary to make sure that the implementation of an RTS is error-free before its deployment.

Two formal techniques, namely verification and testing, are usually used to detect errors in RTS systems. Verification aims at checking that a specification or a model of the system respects some functional and timing requirements. However, testing deals with the implementation of the system, usually referred to as Implementation Under Test or IUT for short, and checks its conformance to the specification of the system in three steps. First of all, test cases are generated according to some coverage criteria. Then, those test cases are executed against the IUT and its reactions are logged. Finally, the verdict is concluded by analyzing the reactions of the IUT: if the behavior of the IUT during test cases doesn't conform to its specification, the IUT is said faulty.

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

Our motivation for studying this problem is 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 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 the necessary and sufficient conditions to check a conformance relation based on trace inclusion. These conditions can be then used to reduce the number of test cases considered for testing an RTS.

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 bound traces of a path. The latter considers 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 traces 1) considers the constraint polyhedron corresponding to the set of constraints that each timed

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

trace of the path has to satisfy and 2) uses some graph transformations that preserve the positivity of the graph cycles.

As a second contribution, the paper proposes an approach to reduce the number of test cases considered while testing RTS. The proposed approach is based on the use of the simulation graph introduced by Tripakis [19]. The fact that the trace inclusion problem can be solved by the inclusion of timed bound traces, provides a method to reduce the number of test cases.

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 traces 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 smallest element of P. Logical "and" and "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 denote an action not in Σ and Σ_τ the set $\Sigma \cup \{\tau\}$. Let σ be a sequence and $X \subseteq \Sigma$. Then, $\sigma_{|X}$ is 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 d is interpreted as the timestamp of the occurrence of a. 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 $TS(\Sigma)$. Note that, when $X \subseteq \Sigma$, the projection of a timed sequence σ over X is obtained by erasing from σ all symbols such that the associated event is not in X.

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 all 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 a 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.

c-Closure [19]. Let $c \in \mathbb{N}$. Two valuations ν and ν' over V are called c-equivalent if:

- for any $x \in V$, either $\nu(x) = \nu'(x)$ or $(\nu(x) > c$ and $\nu'(x) > c)$.
- for any pair $(x, y) \in V^2$, either $\nu(x) - \nu(y) = \nu'(x) - \nu'(y)$ or $(|\nu(x) - \nu(y)| > c$ and $|\nu'(x) - \nu'(y)| > c)$.

Polyhedra. An atomic constraint over V is an expression of the form $x \bowtie n$ or $x - y \bowtie m$ where $(x, y) \in V^2$, $\bowtie \in \{\leq, \geq\}$ and $(n, m) \in \mathbb{N}^2$. The set of formulas that are finite conjunctions of atomic constraints (resp. of 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 **true** for $\bigwedge_{\forall x \in V} x \geq 0$ and **zero** for $\bigwedge_{\forall x \in V}(x \leq 0 \wedge x \geq 0)$. 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 $\nu \in Z$, $\nu + d \notin Z$.

Given a polyhedron Z, the c-closure of Z, noted $close(Z, c)$, is the greatest polyhedron Z' such that $Z \subseteq Z'$, and for all $\nu' \in Z'$ there exists $\nu \in Z$ such that ν and ν' are c-equivalent.

Operations on Polyhedra. We define the operations $Z[X := 0]$ and $Z^{\uparrow}$ of forward clock reset and forward time elapse of a polyhedron Z, respectively, as follows ($X \subseteq V$):

$$Z[X := 0] = \{\nu[X := 0] \mid \nu \in Z\} \qquad Z^{\uparrow} = \{\nu + d \mid \nu \in Z, d \in \mathbb{R}^{\geq 0}\}$$

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. An edge has 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; the special label $\tau \notin \Sigma$ represents an unobservable, internal action. A transition $t = (l, Z, a, r, l') \in \rightarrow$ is noted by $l \xrightarrow{Z,a,r} l'$. $\mathcal{TA}(\Sigma)$ denotes 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 couple $(l, \nu) \in S$ such that l is a location of A and ν is valuation over C such that ν satisfies the invariant $I(l)$. The initial state s_0 of $S(A)$ is (l_0, ν) where $\nu \in$ **zero**. Labels of 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).

Runs. Let $A = (L, l_0, \Sigma, C, I, \rightarrow) \in \mathcal{TA}(\Sigma)$ and $\sigma = (a_1, d_1)...(a_n, d_n) \in TS(\Sigma_\tau)$. A run r of A over σ, denoted by $(\overline{l}, \overline{\nu})$, is a finite sequence of the form:

$$r : (l_0, \nu_0) \xrightarrow{(a_1,d_1)} (l_1, \nu_1) \ ... \ (l_{n-1}, \nu_{n-1}) \xrightarrow{(a_n,d_n)} (l_n, \nu_n)$$

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 $t_i = (l_{i-1}, Z_i, a_i, r_i, l_i)$ of A, such that:
   - $\nu_{i-1} + (d_i - d_{i-1}) \in Z_i$.
   - ν_i equals to $(\nu_{i-1} + (d_i - d_{i-1}))[r_i := 0]$.
   - $\nu_{i-1} + d \in I(l_{i-1})$ holds for all $0 \leq d \leq d_i - d_{i-1}$.

Intuitively, at the initial location 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 $\nu_i + (d_{i+1} - d_i)$ to check the clock constraints, however, at time d_{i+1}, the value of clocks that are reset in t_{i+1} is defined to be 0. By convention, d_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 $[x, y]$.

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

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

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

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

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

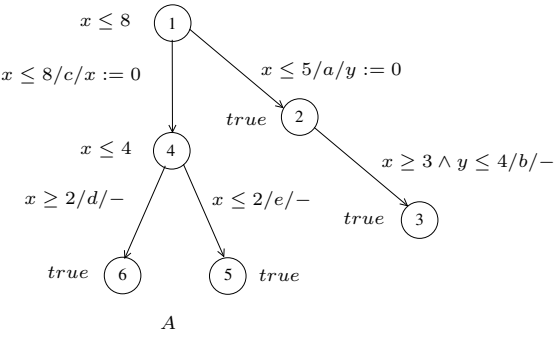

Fig. 1. Timed automata

Finally, for a path ρ of A of length n (i.e. a suite of n transitions 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 Bound Traces of a Path

The goal of this section is to provide an approach to extract timed traces from a given path. As we will see in the next section, these traces can be used to test RTS.

The idea behind our approach is as follows: to a path ρ, we can associate a constraint polyhedron Z_ρ defining the set of constraints to be satisfied by each trace of $TTrace(\rho)$. From this polyhedron, we identify some timed traces of ρ called the *timed bound traces* (TBT). These latter give a finite representation of the trace space of ρ. The proof of the existence of TBT is based on some transformations on the constraint graph associated to a polyhedron, and can be found in Annex B.

For the rest of this paper, $\rho = t_1 \cdots t_n$ will denote a path of a TA $A = (L, l_0, \Sigma, C, I, \rightarrow)$ such that $t_i = (l_{i-1}, Z_i, a_i, r_i, l_i)$, for all $i \in [1, n]$. $V = \{v_1, v_2, ..., v_n\}$ will denote a set of variables ranged over $\mathbb{R}^{\geq 0}$, and $V_0 = V \cup \{v_0\}$ the set V extended with a fictive variable v_0 which is always equals to 0. We will confound elements of $\Phi(V)$ with elements of $\Phi(V_0)$ and a valuation over V with a valuation over V_0.

4.1 Constraint Polyhedron

Let $\sigma = (a_1, d_1) \dots (a_n, d_n) \in TTrace(\rho)$. According to the definition of $TTrace(\rho)$, the different instants $(d_i)_{i \in [1,n]}$ satisfy a set of constraints related to the transitions of ρ. So, we can associate to ρ a constraint polyhedron Z_ρ

[2] A_ρ has the same states and transitions as 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 $\nu(v_i) = d_i$, for all $\forall i \in [0, n]$, is in Z_ρ.

In order to define the constraint polyhedron, we need some additional notations. For a clock $x \in C$ and $i \in [1, n]$, $\rho_i = t_1...t_i$ will denote the path composed of the first i transitions of ρ. $last_i^x$ will denote the index of the transition where the clock x has been reset most recently before i. Recall that all clocks are reset at the initial location l_0, and thus $last_i^x = 0$ if x was not reset in ρ_i. Z_i will denote the constraint polyhedron associated to ρ_i. The construction of Z_i is done by induction: for $i \in [1, n]$,

1. $Z_0 = true$
2. Z_i is obtained from Z_{i-1} as follows:
   - $Z_i := Z_{i-1} \wedge v_{i-1} \leq v_i$
   - If the guard of t_i has a term of the form $x \bowtie k$ then $Z_i := Z_i \wedge v_i - v_j \bowtie k$, where $j = last_i^x$.
   - If the guard of t_i has a term of the form $x - y \bowtie k$ then $Z_i := Z_i \wedge v_p - v_q \bowtie k$, where $q = last_i^x$ and $p = last_i^y$.
   - If the invariant of l_{i-1} has a term of the form $x \bowtie k$ then $Z_i := Z_i \wedge v_i - v_j \bowtie k$, where $j = last_i^x$.

Next, Z_n will be noted Z_ρ.

Proposition 1. *$\sigma = (a_1, d_1) \cdots (a_n, d_n) \in TTrace(\rho)$ iff there is a valuation $\nu \in Z_\rho$ such that $\nu(v_i) = d_i$, for all $\forall i \in [0, n]$.* □

Proof. See Annex A. □

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

$$(l_1, x \leq 8) \xrightarrow{x \leq 5/a/y:=0} (l_2, true) \xrightarrow{x \geq 3 \wedge y \leq 4/b/-} (l_3, true).$$

Recall that v_0 is equal to zero all time. Then,

$$Z_0 = true, \quad Z_1 = v_0 \leq v_1 \wedge v_1 - v_0 \leq 5 \wedge v_1 - v_0 \leq 8$$

$$Z_\rho = Z_2 = v_0 \leq v_1 \wedge v_1 - v_0 \leq 5 \wedge v_1 - v_0 \leq 8 \wedge v_1 \leq v_2 \wedge v_2 - v_0 \geq 3 \wedge v_2 - v_1 \leq 4$$

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. □

Convention. Without losing the generality and for simplicity reasons, we assume that Z_ρ can be written (syntactically) 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 a 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 $v_i - v_j \leq c$.

Definition 1. *$Z_\rho = \bigwedge_{v_i,v_j \in V_0, v_i \neq v_j}(v_i - v_j \leq l_{ij}) \neq \emptyset$ is said in its canonical form if for all $i \in [0,n]$, $j \in [0,n]$, there exists a valuation $\nu \in Z_\rho$ such that : $\nu(v_i) - \nu(v_j) = l_{ij}$* □

Definition 2. *The canonical form of Z_ρ, noted $cf(Z_\rho)$, is the greatest canonical polyhedron included in Z_ρ.* □

Note that $cf(Z_\rho)$ and Z_ρ represent the same space portion and $cf(Z_\rho) = Z_\rho$ if Z_ρ is in its canonical form.

4.2 Main Results

Theorem 1. *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 ($Z_\rho \neq \emptyset$). 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. See Annex B. □

Computation of $\nu_k^M(Z_\rho)$ and $\nu_k^m(Z_\rho)$. Theorem 1 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: $cf(Z_\rho) = (v_2 - v_0 \leq 5) \wedge (v_0 - v_2 \leq 0) \wedge (v_1 - v_0 \leq 9) \wedge (v_0 - v_1 \leq -3) \wedge (v_1 - v_2 \leq 4) \wedge (v_2 - v_1 \leq 2)$. Then,

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

$$\nu_0^m(Z_\rho) = \begin{pmatrix} 3 \\ 0 \end{pmatrix} \quad \nu_1^m(Z_\rho) = \begin{pmatrix} 9 \\ 5 \end{pmatrix} \quad \nu_2^m(Z_\rho) = \begin{pmatrix} 3 \\ 5 \end{pmatrix} \qquad \square$$

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 $\sigma^{mk} \in TTrace(\rho)$ (according to proposition 1).

Definition 3. *The timed sequences $(\sigma^{Mk})_{k\in[0,n]}$ and $(\sigma^{mk})_{k\in[0,n]}$ are called the timed bound traces (TBT) associated to ρ.* $\square$

Timed bound traces give a finite representation of the trace space of a path. According to Theorem 1, the number of TBT is $2 \times (n + 1)$ (n is the length of the path). However, this number varies between 1 and $2 \times (n + 1)$ and depends on the number of clock resets used in the path. Thus, a path without clock resets has at most 2 TBT. 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] allows to compute this form and to test if a polyhedron is empty. Its complexity is $O(n^3)$.

4.3 Trace Inclusion

Consider a path ρ (resp. ρ') of length $n \in \mathbb{N}$ of a timed automaton A (resp. B). 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 (resp. $Z_{\rho'}$) is bounded and not empty.

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

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 bound traces of $TTrace(\rho)$ are also timed traces of $TTrace(\rho')$.

Proof. See Annex C. $\square$

5 Application: Testing

A test case (test for short) is an experience performed on the IUT by 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 they 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 tester 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 in advance.

5.1 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 symbolic state (i.e. a location l of A and a polyhedron Z), and $t = (l, Z', a, r, l')$ be a transition of A. Then,

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

Intuitively, $post_c()$ contains all states (and their c-closure) that can be reached from states 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 a 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 $t = (l, Z_1, a, r, l')$ 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 in [19] 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 is 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.

5.2 Digital-Clock Test Derivation

Our goal here 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. The reader can found in [3] a complete algorithm to derive tests for digital-clock/analog-clock testers.

For generating tests, our approach uses the simulation graph. In fact, as we have said, $S(A, c)$ gives a finite representation of the reachable state space; each

path of $S(A, c)$ 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 applied, in general, to derive a set of paths from the graph $S(A, c)$. Let $ATS_M(A)$ be a set of paths derived from $S(A, c)$ with a method M, and with respect to a given coverage criterion (states, transitions,...). Element of $ATS_M(A)$ can not be used directly to test a given implementation of A because they are abstract.

Each path $\rho \in ATS_M(A)$ defines 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 $2 \times (n + 1)$ test cases corresponding to the timed bound traces of ρ. Here, we assume that the time is bounded for each $\rho \in ATS_M(A)$ because testing is a finite experience. When a path $\rho \in ATS_M(A)$ is not bounded, we can choose a natural constant MAX to limit the time of observations.

Thus, the approach that we introduce derives abstract paths form the simulation graph; for each abstract path derived, between 1 and $2 \times (n+1)$ test cases are generated corresponding to TBT of this path. These latter are then decorated by the different verdicts. Our approach does not suffer from the explosion problem, since we use only tests that meet the timed bound traces.

6 Related Work

Regarding works in analyzing RTS, [2] have studied the problem of timestamp generation. The solution proposed consists in computing one timed trace corresponding to the minimal accumulate delay run. The approach of Tripakis for generating timed diagnostics presented in [19, 20] was based on 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 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 trace corresponding to a state. In [16], the authors propose several algorithms to compute the minimal timed diagnostic that reach a given state.

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 proposed 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]. The authors propose then a heuristic to compact chains of ticks, but this heuristic does not give always minimal tests and it is not trivial.

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. The solution that we have proposed was based on the use of timed bound traces and does not suffer from these problems.

7 Discussion

In this paper, we have studied the trace inclusion problem of RTS. Our solution was based on the identification of the timed bound traces (TBT) corresponding to a given path. The trace inclusion problem is then reduced to the inclusion of TBT. As an application, the paper showed how to use these results to reduce the number of test cases for an RTS. The idea behind our approach was the use of the simulation graph to derive abstract paths and the generation of a finite set of test cases from each abstract path corresponding to the timed bound traces.

To our knowledge, the identification of TBT, and the solution proposed for trace inclusion problem are new results. Furthermore, our approach for generating tests, does not suffer from an exorbitant number of test cases because we consider only test cases corresponding to TBT.

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 is required. This model is quite restrictive, and the generalization will benefit many RTS. Especially, the determinism assumption may be broken by the on-the-fly determinization techniques. Of course, for the class of event-recording automata (ERA), the determinism assumption is not a limitation since this class of timed automata can be determinized.

Finally, timed bounds traces can be used to report counterexamples during timing verification: once the verification tool determines the sequence of transitions that leads to a violation of a safety property, the timed bound traces provide greater diagnostic feedback. In this case, the TBT are called timed bound diagnostics.

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 Ouverte 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, 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.

Annex A: Proof of Proposition 1

Proof. For all $i \in [0, n]$, v_i represents the instant of firing t_i according to a global clock. Thus, the suite $(v_i)_{i\in[0,n]}$ is monotonically increasing. By convention, we assume the existence of a transition t_0 where all clocks are reset at instant $v_0 = 0$. Initially, Z_0 is equal to *true*. At step $i \in [1, n]$, if t_i has a term over x of the form $x \bowtie k$, then the actual value of x corresponds to the time elapsed since the last reset of x. Thus, the value of x is exactly $v_i - v_j$ where $j = last_x^i$. The constraint $v_i - v_j \bowtie k$ is then added to Z_i. If t_i has a term of the form $x - y \bowtie k$ then, the constraint $v_p - v_q \bowtie k$ is added to Z_ρ, where $q = last_i^x$ et $p = last_i^y$. In fact, the time elapsed since the last reset of x (resp. y) in 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 location. □

Annex B: Proof of Theorem 1

In subsection 4.1, we have showed that we can associate to ρ a constraint polyhedron Z_ρ. In order to proof the main theorem of subsection 4.2, we need to define the constraint graph G_ρ associated to the polyhedron Z_ρ and some transformations on G_ρ. Before that, let us recall some graph notions.

Graph Notations

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 cartesian 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 $G = (V, E, w)$ be a DLG. A 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 defined by: $w(p) = \sum_{i\in[1,n]} w(e_i)$. Let $e = v_i \rightarrow v_j$ be an edge. Then, $path(e)$ is 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 that does not visit a vertex twice, except from the root vertex. The graph 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 than 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.

Graph-Theoretic Formulation

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 (Recall that $V_0 = \{v_0, v_1, ..., v_n\}$):

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

Proposition 2. *Z_ρ is not empty 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].

Definition 4. *Z_ρ is said in its canonical form if its constraint graph G_ρ is minimal.* □

This definition is equivalent to the definition 1 (subsection 4.1). Next, we will introduce three transformations that keep the positivity and/or the minimality of the transformed graph. To save space we omitted the proof of the next lemmas, but they are based on the comparison of the weights of e-cycles, and can be found in [3]. Let $G = (V_0, E, w_G)$ be the constraint graph of Z_ρ.

Transformation $m()$. The function $m()$ associates to $G = (V_0, E, w_G)$ the graph $G' = (V_0, E, w_{G'})$ 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 paths of source v_p and target v_q. This weight is either reached by a path, i.e. there is $p \in path(e)$ 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.

Proposition 3. *G is a nonnegative graph iff $m(G)$ is not.* □

Intuitively, the transformation $m()$ preserves the positivity of cycles.

Definition 5. *Let $m(G_\rho) = (V_0, E, w_m)$ be the minimal graph of G_ρ. The canonical form of Z_ρ, noted $cf(Z_\rho)$, is the polyhedron defined by:*

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

□

This definition is equivalent to definition 2 (subsection 4.1) and gives a method to compute the canonical form of a polyhedron.

Transformation $R_{i\to *}()$. Let $i \in [0, n]$. The function $R_{i\to *}()$ associates to $G = (V_0, E, w_G)$ the graph $G' = (V_0, E, w_{G'})$ such that: for each edge $v_p \to v_q \in E$,

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

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 1. *Let G be a nonnegative graph and $i \in [0, n]$. 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 edges $v_p \to v_q \in E$:*

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

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 $R_{i\to *}()$. The transformed graph $G' = (V_0, E, w_{G'})$ is defined by: for each edge $v_p \to v_q \in E$,

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

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 2. *Let G be a nonnegative graph and $i \in [0, n]$. 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 edges $v_p \to v_q \in E$:*

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

Proof of Theorem 1

Proof. To prove 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 ($Z_k^M \neq \emptyset$ and $Z_k^m \neq \emptyset$). 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. $Z_\rho \neq \emptyset$ implies that G_ρ is a nonnegative graph (proposition 2). 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 1 (resp. lemma 2), we deduce that $G(Z_k^M)$ (resp. $G(Z_k^m)$) is a nonnegative graph and by consequence, $Z_k^M \neq \emptyset$ (resp. $Z_k^m \neq \emptyset$). Furthermore, the second point of lemma 1 (resp, lemma 2) gives a method to compute the canonical form of Z_k^M (resp. Z_k^m). □

Annex C: Proof of Corollary 1

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 the 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 $cf(Z_{\rho'})$. □