The Influence of Time on Timed Systems Testing

Sébastien Salva Hacène Fouchal

LERI-*RESYCOM*
Fax : (33) 3 26 91 33 97
Université de Reims Champagne-Ardenne,
Moulin de la Housse, BP 1039,
51687 Reims Cedex 2, France
e-mail:{Sébastien.Salva, Hacene.Fouchal, Simon.Bloch}@univ-reims.fr

Abstract

This paper deals with test quality of systems described either by timed automata or by region graphs (Alur and Dill) which are models usually used in timed systems testing. The aim of this work is to propose some quality parameters for the whole system and to measure them in order to know, a priori, the test cost and the test coverage (time and effort) before starting the test process.

Key-words : Testing, Timed Automata, Region Graph, Test quality, Real-Time systems.

1 Introduction

The verification of distributed systems is a very important step in the system design cycle. This step helps designers to validate their products in order to be implemented and used in the industry.

Presently, systems become more and more complex. This complexity is sometimes due to the addiction of timing constraints in system specifications. Then timed automata have been introduced in the 90's for taking into account these constraints. The timed automata model [AD94] has been widely studied in various cases: verification [MY96, SY96, AD92], system description [DY95, BGK$^+$96], tools [DOTY95, BL]...

This study is based on conformance testing as used in the protocol engineering area. The aim of conformance testing is to check if the implementation of a system (which is a black box) behaves as the is described in the specification of the system. Usually, we extract from the specification "test sequences" (which are pertinent sequences of actions to be executed on the system) and should be executed on the implementation. A test sequence is composed of a preamble (sequence of actions allowing to reach an action), of the action to test, and of a postamble (sequence allowing to come back to the initial state). The analysis of the verdict of this execution inform us about the conformance of the implementation to the specification.

This paper investigates especially the measurements of some criteria on the system before starting tests. The obtained measures should be able to help designers to evaluate, a priori, test costs: time, efforts and fault coverage. This evaluation is called test quality (testability). Criteria presented here are based on a specific testing methodology [PF99a] but they can be adapted to other methods since they are only based on the timed automata model.

This paper is structured as follows: In section 2, we present the main studies on testability and the timed automata model. In section ??, we present basic definitions of timed automata and a timed testing methodology. In section ??, we present the testability of untimed systems based on the work of [KGD96]. In section 3, we propose factors of test quality for timed automata. Finally, in section 5, we conclude and we give some ideas about future extensions of this work.

2 Related work

In this section, we give some ideas about studies on systems' testability and we draw briefly the main definitions about timed automata.

2.1 Testability

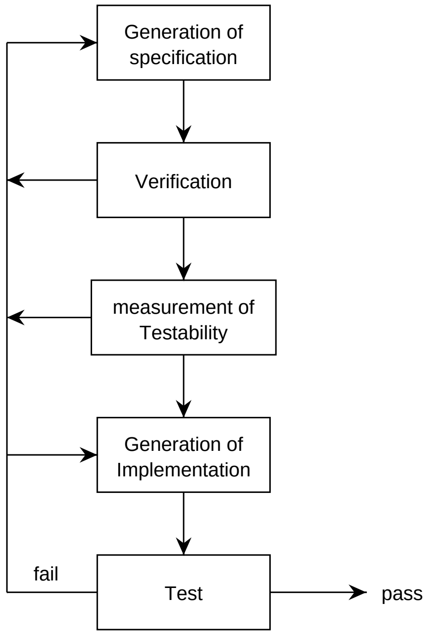
Generation of specification
Verification
measurement of Testability
Generation of Implementation
fail
Test
pass

Figure 1: Testability in life cycle

The figure 1 summarizes the software life cycle which takes into account testability measurements. The testability evaluation before the testing step allows the designer to decide if the test can be applied, or if the specification must be modified in order to obtain a more significant test result. The testability can also help the designer to choose one specification among others describing the same behavior.

In our knowledge, the main studies about testability on telecommunication protocols can be found in [Fre91, YPKP95, KDCK94]. All of these papers deal with the IOFSM model: it is a Finite State Machine where each transition is labeled with an input and an output symbol.

All these references agree with the following definition about testability: "A system is testable if it is observable and controllable". In [Fre91], these two concepts are defined with:

- the observability, which allows to determine the internal state of a system under observation. Practically, we say that a specification is observable if, for each input symbol and for each state, there exists one or several output symbols.
- the controllability, which allows to propagate specific values towards observable outputs. Practically, a specification is said controllable if, for each output symbol, it exists at least an input symbol which forces the system to give this output symbol.

In [KGD96], four factors (defined bellow), depending on observability and on controllability, are presented to evaluate several degrees of testability. Indeed, each factor gives a value, included between 0 and 1, and the more these values are close to 1, the more the system is testable. Consequently, by calculating these degrees, it is possible to evaluate the testing cost and to evaluate if all errors may be discovered. Afterwards, the designer may evaluate if this quality is sufficient for the test.

For a system A, we summarize these factors here:

- The *Controllability degree*, $C(A)$, evaluates the effort needed to reach each state in the specification. This effort depends on the cardinal of the set of preambles and on the length of these preambles, chosen for the test.
- The *Fuzziness degree*, $F(A)$, evaluates the effort, depending on the number of states which can not be distinguished. Indeed, only some methods, as the HSI method [LPB93], accept systems with non distinguishable states, but they demand more efforts to create test sequences.
- The *State Characterization degree*, $W(A)$, evaluates the effort, depending on the length of the input sequences needed to distinguish the system states.
- The *Abstraction degree*, $AB(A)$, compares the specification 's states number to the implementation 's states number. The implementation is less abstract, then it contains more states. And this difference affects the testability.

All these factors are joined in a vector, called the *Testability vector*, TV of a system A, and

$$TV(A) =< C(A), F(A), W(A), AB(A) >$$

.

2.2 Timed system model

2.2.1 Timed Input Output automata (TIOA)

Timed automata [AD94] are graphs representing timed systems during their executions. To represent time in timed systems, a set of clocks is associated to the automaton. Each clock is represented by a real value (dense time representation) and grows strictly monotonically. All clocks are set to 0 in the initial state. Clocks can be reset on any transition.

Definition 2.1 (Clock constraint [AD94]) *Let C_A be a finite set of clocks, and $x_i \in C_A$. A clock constraint δ over x_i is a boolean expression of the form $\delta = x_i \leq c \mid c \geq x_i \mid \neg\delta \mid \delta_1 \wedge \delta_2$ where $c \in \mathbb{Q}$.*

The set of clock constraints over C_A is denoted $\Phi(C_A)$.

Timed Input Output automata are extended timed automata where symbols are divided into input symbols and output symbols.

Definition 2.2 (Timed Input Output automata) *A TIOA A is defined as a tuple $< \Sigma_A, S_A, s^0_A, C_A, E_A >$, where:*

- *Σ_A is a finite alphabet,*
- *S_A is a finite set of states,*
- *s^0_A is the initial state,*
- *C_A is a finite set of clocks,*
- *$E_A \subseteq L_A \times (\{?, !\} \times \Sigma_A) \times 2^{C_A} \times \Phi(C_A)$ is the finite set of transitions.*
  An input symbol begins with "?" and an output one begins with "!".
  A tuple $< l, l', a, \lambda, G >$ represents a transition from state l to state l', labeled with the symbol a. The subset $\lambda \subset C_A$ gives the clocks to be reset within this transition, and G is a clock constraint over C_A.

A transition $S_i \xrightarrow{?x} S_j$, labeled by the input symbol "?x", models an input action induced by the external environment of the implementation. A transition $S_i \xrightarrow{!x} S_j$, labeled by the output symbol "!x", represents an output action executed by the implementation.

An example of TIOA is illustrated in figure 2, which models a part of the MAP-GSM protocol with addiction of time, represented by two clock x and y. If we consider the transition $TMP3 \xrightarrow{!O6} DE$, the clock constraint $x \geq 2$ should be satisfied in order to the system sends "!O6".

Dense time representation is modelled in TIOA with systems of inequations, added on any transitions. Testing temporal properties of an implementation with only these ones

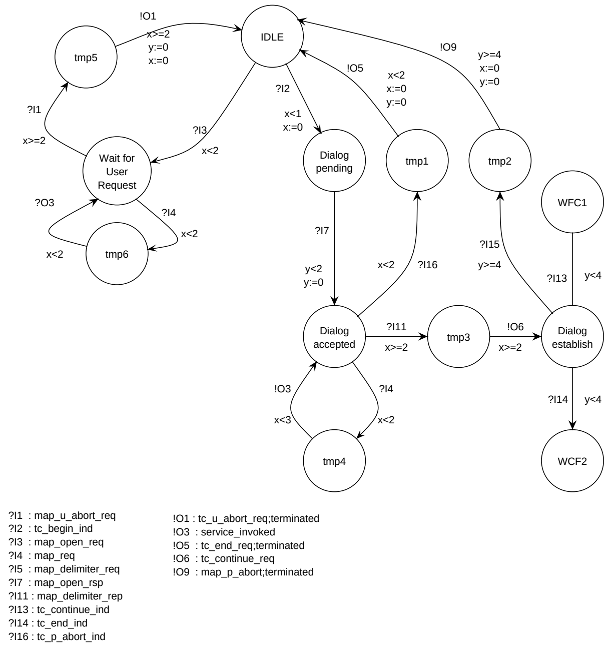

Figure 2: A part of the MAP-GSM protocol with addiction of two clocks

seems difficult since the tester can only deals with clock values, during test execution. So, most of testing methodologies, using TIOA model [LC97, PLC98] translate these constraints into time intervals. That is, for each transition $t \in E_A$, and for each clock $x \in C_A$, the minimal clock value of x allowing to reach t and the maximal clock value of x allowing to execute t, are computed. Therefore, we obtain $card(C_A)$ time intervals satisfying together the execution of t. Note, the complexity to compute these time intervals is lower than computing the corresponding region graph, but interaction between clocks is prohibited. Figure 3 represents the translated automaton of Figure 2 with time intervals.

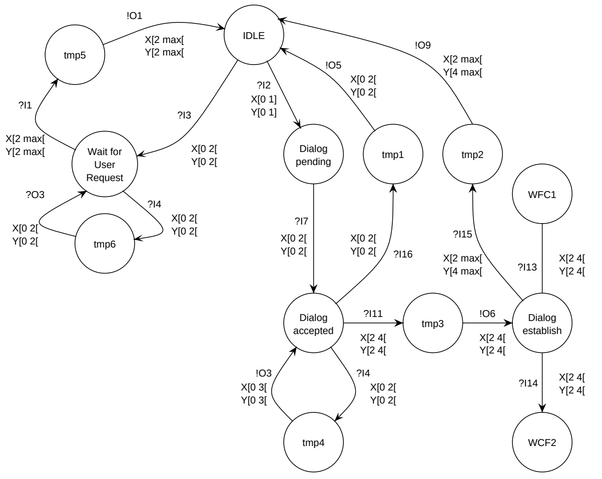

Figure 3: Clock constraints modelled with time intervals

2.2.2 Region Graph

Definition 2.3 (Clock valuation [AD94]) *A clock valuation over a set of clocks C_A is a mapping v that assigns to each clock $x \in C_A$ a value in $\mathbb{R}^+$, called clock value. We denote the set of clock valuation by $V(C_A)$.*

A clock valuation v satisfies a clock constraint G, denoted $v \models G$, if and only if G is evaluated to true under v.
For $d \in \mathbb{R}^+$, $v + d$ denotes the clock valuation which assigns a value $v(x) + d$ to each clock x. For $X \subseteq C_A$, $[X \to d]v$ denotes the clock valuation for C_A which assigns d to each $x \in X$, and agrees with v over the rest of the clocks.

So, if n is the number of clocks of A, a clock valuation is a n-tuple of clock values. With this previous definition, we can say the future behavior of a timed system is determined by its states and by the clock valuations over C_A. This motivates a new representation of timed systems, by combining these states and these clock valuations. An equivalence relation is defined in [AD94] in order to gather clock values which have the same integral parts.

For any $t \in \mathbb{R}^+$, $fract(t)$ denotes the fractional part of t and $\lfloor t \rfloor$ denotes the integral part of t.

Definition 2.4 (Clock Region [AD94]) *Let $A =< \Sigma_A, L_A, l_A^0, C_A, E_A >$ be a timed automaton. For each $x \in C_A$, let c_x be the largest integer c such that $(x \leq c)$ or $(c \leq x)$ is a subformula of some clock constraints appearing in E_A.*

The equivalence relation $\sim$ is defined over the set of all clock interpretations for C_A; $v \sim v'$ iff all the following conditions hold:

- *For all $x \in c_A$, either $\lfloor v(x) \rfloor$ and $\lfloor v'(x) \rfloor$ are the same, or both $v(x)$ and $v'(x)$ are greater than c_x.*
- *For all $x, y \in C_A$ with $v(x) \leq c_x$ and $v(y) \leq c_y$, $fract(v(x)) \leq fract(v(y))$ iff $fract(v'(x)) \leq fract(v'(y))$.*
- *For all $x \in C_A$ with $v(x) \leq c_x$, $fract(v(x)) = 0$ iff $fract(v'(x)) = 0$.*

A clock region for A is an equivalence class of clock valuations induced by $\sim$.

We will use $[v]$ the clock region to which v belongs. Clock regions can be illustrated by polyhedrons containing several vertices. For a clock region R, we also say a clock valuation $c \in R$ iff c satisfies all the inequations of R.

Definition 2.5 (Time Successor) *A clock region R' is a time-successor of a clock region R iff for each $v \in R$, there exists a positive $t \in \mathbb{R}$ such that $v + t \in R'$*

Region graphs are equivalent representations of timed automata where timed constraints are moved into states. A region graph state is a tuple containing one state of the TIOA and one clock region. This new representation allows to distinguish each time interval during which an action may be executed. An algorithm to transform TIOA into region graphs is given in [AD94].

Definition 2.6 (Region Graph) *Let $A = (\Sigma_A, S_A, s_A^0, C_A, E_A)$ be a timed input output automaton. A region graph of A is an automaton $RA = (\Sigma_{RA}, S_{RA}, s_{RA}^0, E_{RA})$ where:*

- *$\Sigma_{RA} = \Sigma_A \cup \delta$,where δ represents the time elapsing,*
- *$S_{RA} \subseteq \{\langle s, [v] \rangle \mid s \in S_A \wedge v \in V(C_A)\}$*
- *$s_{RA}^0 = \langle s_A^0, [v_0] \rangle$ where $v_0(x) = 0$ for all $x \in C_A$*
- *R_A has a transition, $q \xrightarrow{a}_{RA} q'$, from state $q(\langle s, [v] \rangle)$ to state $q'(\langle s', [v'] \rangle)$ with the symbol a, iff either*
  - *$a \neq \delta$ and there is a transition $(s, s', a, \lambda, G) \in E_A$ and $d \in \mathbf{R}^+$ such that $(v+d) \models G$ and $v' = [\lambda \mapsto 0](v + d)$,*
  - *$a = \delta$, $s \neq s'$ and there exists $d \in \mathbf{R}^+$ such that $v' = v + d$.*

Region graphs can be minimized using the algorithm, described in [YL93], which generates the portion of the minimized system that is reachable in polynomial time. Consequently, all clock regions, in which the same actions can be executed, are gathered into one clock region. So, the number of states of a region graph is strongly reduced.

For example, we obtain from the TIOA, illustrated in figure 2, the following minimized clock regions:

Clock Region	Inequations
r_1	$0 \leq x = y < 1$
r_2	$x \leq 2$, $y \leq 2$ and $x \leq y \leq x + 1$
r_3	$y \leq x < 2$
r_4	$y \leq x \leq y + 2$ and $2 \leq x < 3$
r_5	$y \leq x \leq y + 2$ and $3 \leq x < 4$
r_6	$y \leq x \leq y + 2$ and $x \geq 4$

They also can be illustrated with figure 4.

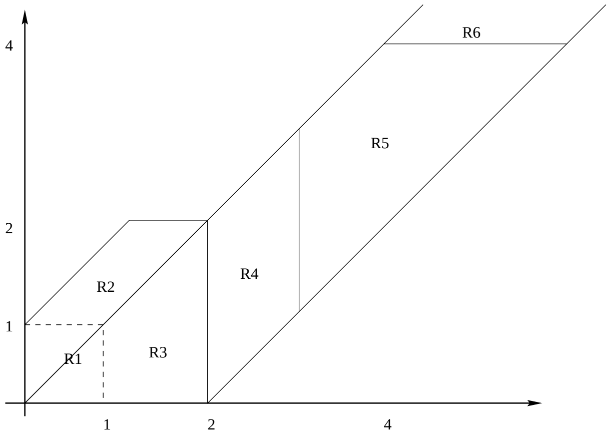

Figure 4: Clock regions of A

Finally, the figure 5 describe the minimized region graph generated from the previous TIOA (figure 2).

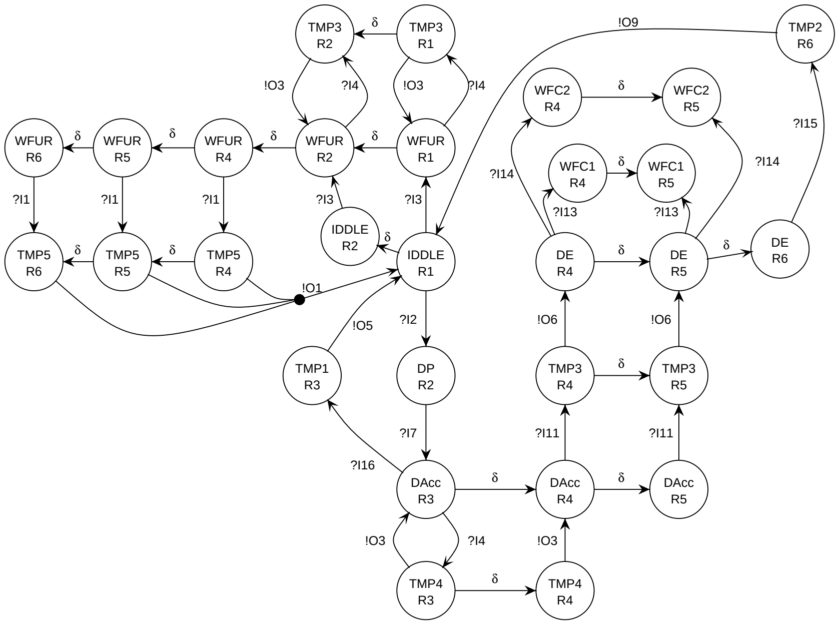

Figure 5: The region graph obtained from A

2.3 Timed testing methodologies

We present briefly the main and common steps of the timed system test methodologies which aims to check the temporal and behavior properties of timed implementation, by means of a conformance relation, describing an equivalence relation between this one and the specification.

The first step consists in generating sequences of actions, called test sequences by gathering information on specification behavior in order to characterize it. These sequences can be generated automatically or can be given by the designer.

In order to check the conformity of the implementation, each test sequence is executed by means of a specific test architecture [PF99b, ?], which consists in:

- An implementation under test (I.U.T), composed by two points of control and observation (PCO), one for submission of the input events and another one for the observation of the output events.
- The tester, which executes a specific algorithm able to submit and to control the implementation test. In [], the tester does not contain clocks but can interrogate the I.U.T ones to obtain clock values. In contrary, in [PF99b], the tester uses its own clocks synchronized with the I.U.T ones. As the I.U.T is seen as a black box, the only way to check behavior properties, with the tester, is to apply test sequences to the I.U.T, and by observing its reactions (output actions).

3 Timed Systems Testability

Testability of untimed systems, allows to evaluate the quality of test induced by the observability and the controllability of theses ones. For timed systems, all the previous degrees can be always used to evaluate these two notions.

However, it is necessary to introduce several new degrees, depending only on timing constraints to evaluate the influence of time on timed systems testing. Timed systems can be represented with many models, but usually, TIOA and region graph models are at the heart of any verification or testing technique (for example see [DY95, NSY92, LC97]). In this field, we will focus on theses models, in order to study their temporal properties, and we will show that some of these ones prevent the coverage of all of the parts of the system, which implies a lost of error detection, and that the other ones improve the testing cost.

For each temporal property, reducing timed systems testability, we define a new degree, giving a value included between 0 and 1 : so, the more each degree is close to 1, the more the timed system is testable, and in contrary, if each degree is close to 0, the test will not prove the conformance of the whole implementation. Consequently, the computation of these ones, which can be done before the test and before the generation of the test suite, will allow to evaluate if an implementation may be correctly tested in taking into account timing constraints.

In all this section, we will use a system described with a TIOA $A = (\Sigma_A, S_A, s_A^0, C_A, E_A)$ and its region graph $RA = (\Sigma_{RA}, S_{RA}, s_{RA}^0, E_{RA})$.

3.1 Preliminaries

Let δ representing the elapse of time in a system A and $RA = (\Sigma_{RA}, S_{RA}, s_{RA}^0, E_{RA})$ the region graph of A. So, we define these sets:

- I_A is the set of time intervals computed from A (section 2.2.1).
- $Q_A = \{\text{sequence } \sigma \mid \forall \text{ transition } t \in \sigma, \sigma \text{ contains only one time } t\}$ is the set of cover sequences of A. This set contains the sequences of A without taking into account the property of connectivity of A. To build this set, the cover tree of the graph can be construct. This tree must begin with the initial state. The sequences of Q_A are all the sequences of the tree. We also denote Q_{RA} the set of cover sequences of RA.
- $R_{RA} = \{[v] \mid \exists(s, [v]) \in S_{RA}\}$, is the set of clock regions of the system RA
- $U_R = \{(s, R) \mid (s, R) \in S_{RA}, \exists a \neq \delta, \exists(s_2, R') \in S_{RA}, (s, R) \xrightarrow{a} (s_2, R')\}$, is the set of state which may be executed in the region R, and where an action may be performed. This set allows to know the number of references to the clock region R in the region graph.

- $AT = \{(s, a, \lambda, s') \in E_{RA} \mid a \neq \delta\}$, is the set of transitions where each action is not an time elapsing. $AT \subseteq E_{RA}$
- $IT = \{(s, a, \lambda, s') \in AT \mid a$ is an input symbol$\}$ is the set of input actions.

3.2 TIOA Testability

By studying timing constraints of TIOA, modelled by time intervals (see section 2.2.1), we propose to define four degrees which are the *TIOA Time Shape Degree* (S_{tioa}), *TIOA Time Reachability Degree* (R_{tioa})), *TIOA Controllability Degree* (C_{tioa}), *TIOA Time Independency Degree* (I_{tioa}).

The first two degrees allows to evaluate the test coverage of the system behavior, and thus, to evaluate the part of the system which may not to be tested. The last ones measure a part of the efforts needed for the test execution, depending, in this case, on the number of clocks of the system, and the time needed to execute all of its actions. Because we don't consider the testing methodology and the test sequences, we cannot evaluate with precision the cost of the test execution, but we evaluate a factor which influences it.

Then we gather the previous degrees into a vector, called the *Timed testability Vector* of TIOA, such as: $TTV_{tioa}(A) =< S_{tioa}, R_{tioa}, C_{tioa}, I_{tioa} >$

3.2.1 TIOA Time Shape Degree

Timed systems may be specified with some clock constraints, as $x_i \geq c, x_i \in C_A, c \in \mathbb{R}^+$, which can generate an infinite time interval for x_i. Such an interval I cannot be used for testing, without being bounded since the testing time must be finite. Indeed, for transitions labeled with an input symbol $?A$, the choice of clock values, used for sending $?A$ to the I.U.T, seems difficult on account of the infinity number of clock values in I. And for transitions labeled with an output symbol, the tester, which must receive the symbol for any clock value of I could wait indefinitely.

Consequently, all the time intervals of A must be bounded, before testing. However, we test only a part of the behavior of A and this implies that some potentials errors, executable in infinite intervals, could not be detected. So, the system can be only partially tested, and fault detection is reduced.

So, the number of infinite time intervals influences the coverage of all parts of the system. Hence, we propose a new degree, denoted the *TIOA Time Shape Degree* (S_t) in order to measure this evaluation. Let $INF_A = \{I \in I_A \mid I = [a, \infty], a \in \mathbb{R}^+\}$, be the set of infinite time intervals of A. For the system A, $S_t(A)$ is calculated as:

$$S_t(A) = 1 - \frac{card(INF_A)}{card(I_A)} \text{ and } 0 \leq S_t(A) \leq 1$$

Consequently, the greater $S_t(A)$ is, the more the system A will be covered by the test. If $S_t(A)$ equals to 1, the system A does not contains infinite time intervals, and A is the most testable. In contrary, if $S_t(A)$ equals to 0, all of the time intervals are infinite, and only a little part of the behavior of A will be tested. The detection of faults is not sufficient.

Consider the TIOA, illustrated in figure 3, and its specifics time intervals. By measuring the TIOA Time Shape Degree, we obtain four transitions with infinite time intervals. And as $Card(I_A) = 34$, we obtain $S_{tioa} = (A) = \frac{13}{17}$. Consequently, during the test execution, the system will be only partially tested, and the execution of "?I1", "!O1", "?15" and "!O9" may generate non detectable errors.

3.2.2 TIOA Time Reachability Degree

During the test, the behavior of the I.U.T is difficult to be controlled, i.e, it is difficult to force execution of sequences of actions, at specific clock values. Indeed, an output action $S \xrightarrow{!A} S'$ can be executed at any clock values of the time intervals set I, satisfying the execution of this action.

But, if we want to test an action "A", each implementation's clock x_i must satisfy at least one clock value of the time interval $[T_{i1} T_{i2}] \in I$. We will say the clocks must enter in their corresponding time intervals, i.e. these ones must be reachable. If one or more time intervals cannot be reached, then the action won't be able to be tested. So, the system can be only partially tested, and fault coverage is reduced.

Let $\xrightarrow[\lfloor T'_{11}T'_{12}\rfloor\ldots\lfloor T'_{n1}T'_{n2}\rfloor]{A}\xrightarrow[\lfloor T_{11}T_{12}\rfloor\ldots\lfloor T_{n1}T_{n2}\rfloor]{B}$ be two consecutive transitions with, for each clock x_i the timing constraints represented by the time interval $[T_{i1}T_{i2}]$. We consider that the set I of the times intervals $[T_{11}T_{12}], ..., [T_{i1}T_{i2}], ..., [T_{n1}T_{n2}]$ can be reachable by the clocks if the following assumptions are satisfied :

Case 1: If "A" is an output action, for each clock $x_i \in C_A$, the time interval $[T_{i1}T_{i2}]$ must contain clock values which do not belong to $[T'_{i1}T'_{i2}]$, thus $T_{i2} > T'_{i2}$. The execution of "A" is uncontrollable, so this hypothesis must be verified to always reach I.

Case 2: For each clock $x_i \in C_A$ and $x_j \in C_A$ with $i \neq j$, the elapse of time between $[T'_{i1}T'_{i2}]$ and $[T_{i1}T_{i2}]$ must be the same as the elapse of time between $[T'_{j1}T'_{j2}]$ and $[T_{j1}T_{j2}]$, since the clocks grows with the same manner and strictly monotonically. So, $T_{i1} - T'_{i1} = T_{i2} - T'_{i2} = T_{j1} - T'_{j1} = T_{j2} - T'_{j2}$.

Case 3: For each clocks $x_i \in C_A$ and $x_j \in C_A$ with $i \neq j$, if x_j is reset with the execution of "A", the elapse of time, needed to reach $[T_{i1}T_{i2}]$ from $[T'_{i1}T'_{i2}]$ must be between $[T_{j1}T_{j2}]$, and this can be verified with: $T_{i1} \in [T_{j1} + T'_{i1}T_{j2} + T'_{i1}]$ and $T_{i2} \in [T_{j1} + T'_{i2}T_{j2} + T'_{i2}]$

Proof:

For the case 1, suppose that $T_{i2} \leq T'_{i2}$. $[T'_{i1}T'_{i2}]$ is composed by the following intervals $[T'_{i1}T_{i1}] + [T_{i1}T_{i2}] + [T_{i2}T'_{i2}]$. If the output action "A" is executed by the system at any clock value of $[T_{i2}T'_{i2}]$ ("A" is uncontrollable), then the time interval $[T_{i1}T_{i2}]$ cannot be reached by the clock. Consequently the assumption $T_{i2} > T'_{i2}$ must be satisfy.

For the case 2, let $T'_{i1} \leq x \leq T'_{i2}$ and $T'_{j1} \leq y \leq T'_{j2}$ the clock values of the clock x_i and x_j during which "A" is executed. The clocks of C_A grows in the same manner and strictly monotonically and none clock is reset, so let $d \in \mathbb{R}^+$ such as $T_{i1} \leq x' = x + d \leq T_{i2}$ and $T_{j1} \leq y' = y + d \leq T_{j2}$. x' and y' are the clock values of $[T_{i1}T_{i2}]$ and $[T_{j1}T_{j2}]$ reached by the clocks after the execution of "A". $T_{i1} \leq x + d \leq T_{i2}$ is equivalent to $T_{i1} - d \leq x \leq T_{i2} - d$, thus $T_{i1} - d = T'_{i1}$ and $T_{i2} - d = T'_{i2}$. In the same way, we easily obtain $T_{j1} - d = T'_{j1}$ and $T_{j2} - d = T'_{j2}$. Consequently, $T_{i1} - T'_{i1} = T_{i2} - T'_{i2} = T_{j1} - T'_{j1} = T_{j2} - T'_{j2}$.

For the case 3, let x be the clock value of x_i after the execution of "A". For the clock x_j, this value obviously equals to 0. Consequently, the elapse of time d, needed to reach the set of time intervals I, is such as $T_{j1} \leq d \leq T_{j2}$. Let $x' = x + d$ the clock value reached by x_i in $[T_{i1}T_{i2}]$. We obtain $T'_{i1} + d \leq x' = x + d \leq T'_{i2} + d$, and $d = T_{i1} - T'_{i1} = T_{i2} - T'_{i2}$. So, we obtain $T_{j1} \leq T_{i1} - T'_{i1} \leq T_{j2}$, and finally $T_{i1} \in [T_{j1} + T'_{i1}T_{j2} + T'_{i1}]$. In the same way, we conclude that $T_{i2} \in [T_{j1} + T'_{i2}T_{j2} + T'_{i2}]$.

■

So, for a transition t, consecutive of t', such as $\xrightarrow[\lfloor T'_{11}T'_{12}\rfloor\ldots\lfloor T'_{n1}T'_{n2}\rfloor]{\{?!\}\times\Sigma_A}\xrightarrow[\lfloor T_{11}T_{12}\rfloor\ldots\lfloor T_{n1}T_{n2}\rfloor]{\{?!\}\times\Sigma_A}$, let N_t be the number of time intervals satisfying all of the previous cases. We define a first factor ρ measuring the difficulty to reach $I_t = \{[T_{11}T_{12}], ..., [T_{n1}T_{n2}]\}$ from the time intervals $I_{t'}$, obtained after having executed the action of t'. Note that if t is an outgoing transition of the initial state, then $I_{t'} = \{0, ..., 0\}$. ρ is evaluated as:
$\rho(I_t) = \frac{N_t}{card(C_A)}$, and $0 \leq \rho(I_t) \leq 1$

Now, let SEQ_t be the set of sequences of Q_A, allowing to reach t. Each sequence $\sigma \in SEQ_t$ does not contain redundant transition. The difficulty to reach I_t with σ is denoted $Reach(I_t, \sigma)$ and is evaluated as:
$Reach(I_t, \sigma) = \prod_{t \in \sigma} \rho(I_t)$

Then, we can finally define the time reachability of the set of time intervals I_t = with the *Time Intervals Reachability Degree* of I_t, denoted $RT(I_t)$ and evaluated as:

$$RT(I_t) = \frac{\sum_{\sigma_i \in SEQ_t} Reach(I_t, \sigma_i)}{Card(SEQ_t)} \text{ and } 0 \leq RT(I_t) \leq 1$$

If, for a transition $t \in E_A$, $RT(I_t) = 1$, each clock $x_i \in C_A$ can reach the corresponding time interval $[T_{i1}T_{i2}]$, and we say I_t is completely time reachable. In contrary, if $RT(I_t) = 0$, the transition t is not testable, hence only a part of the system will be tested. A modification of the timing constraints should be necessary.

Finally, for the whole system, we define the *TIOA Time Reachability Degree* of the TIOA A, denoted $R_{tioa}(A)$ which aims to measure the average reachability of time intervals sets I_t of A, with $t \in E_A$. It can be evaluated as:

$$R_{tioa}(A) = \frac{\sum_{\xrightarrow[I_t]{\{!?\}\times\Sigma_A}\in E_A} RT(I_t)}{Card(E_A)}, \text{ and } 0 < R_{tioa}(A) \leq 1$$

Consequently, the greater $R_{tioa}(A)$ is, the more testable the system is. If $R_{tioa}(A) = 1$, all time intervals are reachable by the clocks for at least for one clock value. If $R_{tioa}(A)$ is close to 0, most of the time intervals are not reachable by the clocks, thus only a little part of the system (the outgoing transitions of the initial state) can be checked during the test execution. We can note this factor can also influence the choice of the preambles, indeed, for reaching an action of the system, all actions of the preamble p must be executed, thus all time intervals of p must be reachable.

The following results show the reachability of each time intervals of the TIOA, illustrated in figure 3 .

Transition	Time Intervals	$card(SEQ)$	RT_{rg}
$I \xrightarrow{?I2} DP$	X[0 1] Y[0 1]	0	1
$I \xrightarrow{?I3} WFUR$	X[0 2] Y[0 2]	0	1
$WFUR \xrightarrow{?I4} TMP3$	X[0 2] Y[0 2]	1	1
$TMP3 \xrightarrow{?I4} WFUR$	X[0 2] Y[0 2]	1	1
$WFUR \xrightarrow{?I1} TMP5$	X[2 10] Y[2 10]	2	1
$TMP5 \xrightarrow{!O1} I$	X[2 10] Y[2 10]	2	1
$DP \xrightarrow{?I7} DAcc$	X[0 2] Y[0 2]	1	1
$DAcc \xrightarrow{?I4} TMP4$	X[0 2] Y[0 2]	1	1
$TMP4 \xrightarrow{!O3} DAcc$	X[0 3] Y[0 3]	1	1
$DAcc \xrightarrow{?I16} TMP1$	X[0 2] Y[0 2]	2	$\frac{1}{2}$
$TMP1 \xrightarrow{!O3} I$	X[0 2] Y[0 2]	2	$\frac{1}{2}$
$DAcc \xrightarrow{?I11} TMP3$	X[2 4] Y[2 4]	2	1
$TMP3 \xrightarrow{!O6} DE$	X[2 4] Y[2 4]	2	1
$DE \xrightarrow{?I15} TMP2$	X[4 10] Y[4 10]	2	1
$TMP2 \xrightarrow{!O9} I$	X[4 10] Y[4 10]	2	1
$DE \xrightarrow{?I13} WFC1$	X[2 4] Y[2 4]	2	1
$DE \xrightarrow{?I14} WFC2$	X[2 4] Y[2 4]	2	1

Finally, the Time intervals Reachability Degree becomes $R_{tioa}(A) = \frac{16}{17}$. So, most of the time intervals are reachable during test execution, but the system will be partially tested on account of the consecutive transitions $TMP4 \xrightarrow{!O3} DAcc \xrightarrow{?I16} TMP1$. Indeed, this case corresponds to the third case of unreachability. So, the execution of actions "?I16" and "!O5" may not to be always checked, during the testing process.

3.2.3 TIOA Controllability Degree

The Controllability Degree, first defined in [KGD96], aims to evaluate the difficulty to reach each action of the implementation, during the test. The set of sequences, allowing to reach these actions, is called the set of preambles, denoted P. Each preamble $p \in P$ constitutes a bone of the test sequences and P guarantees the access of each action. Consequently, P affects the testing cost. Indeed, the longer the execution of preambles is, the longer the execution of the test will be. So, a degree is necessary to evaluate the testing cost, depending on P. P is not unique, but we consider that $P \subseteq Q_A$. In this case, the maximum length of a preamble p is $n-1$, with n the number of states of A, and no action of p is redundant.

With untimed systems, the length of preambles is sufficient to evaluate the Controllability Degree. Indeed, the longer the sequences of actions, applied to an I.U.T is, the more difficult, the action to reach, will be.

With timed systems, execution of preambles needs time, so the length of preambles is not sufficient. To evaluate the difficulty to reach each action of a timed system, we must take into account the time consuming induced by the execution of each preamble $p \in P$. As this time is not constant, we consider the three following cases :

- After executing an action a_1 of p, it may be necessary to wait before executing the next action a_2. Indeed, the clock of the system must satisfy all of the time intervals of a_2 to execute it, so an elapse of time could be needed.
- Let $\xrightarrow{?A}$ be a transition labeled by an input action, and $[T_{11}T_{12}], ..., [T_{n1}T_{n2}]$ be the time intervals of the clocks $X_1, ..., X_n$. During the test execution, the tester can send "?A" for any clock values satisfying these time intervals. For evaluating the execution cost of p, we will consider two executions: a first one, during which each input action is sent immediately (none elapse of time), and a second one, during which each input action is executed the latest possible. In this last case, an elapse of time is necessary before sending each input action, and this one equals to $min\{[T_{12} - X_1], ..., [T_{n2} - X_n]\}$ which represents the maximal interval of time during which the action can be executed.
- Let $\xrightarrow{!A}$ be a transition labeled by an output action, and $[T_{11}T_{12}], ..., [T_{n1}T_{n2}]$ be the time intervals of the clocks $X_1, ..., X_n$. During the test execution, the tester should receive the symbol "!A" at any clock values satisfying the time intervals. As we cannot determine when the action is executed, we will consider "!A" is received the latest possible. In this case, we must also consider the existence of an elapse of time before the reception of "!A" and this one also equals to $min\{[T_{12} - X_1], ..., [T_{n2} - X_n]\}$.

After studying these assumptions, we propose the following algorithm to compute the minimal time execution E_{min} and the maximal time execution E_{max}.

Algorithm Computation of E_{min}

```
$V = \{X_1, ..., X_N\}$ is a set of clocks initialized to 0

FOR each $t \in p$ with the time intervals
$\{[T_{11}T_{12}]...[T_{11}T_{12}]\}$

    % All time intervals not yet reached
    IF $X_1 \notin [T_{11}T_{12}]$ OR ... OR $X_N \notin [T_{N1}T_{N2}]$
    THEN { $d = MAX\{T_{11} - X_1, ..., T_{N1} - X_1\}$
           $V = \{X_1 + d, ..., X_N + d\}$
           $E_{min} = E_{min} + d$
    IF $t$ is labeled with an output symbol
    THEN { $d = MIN\{[T_{12} - X_1], ..., [T_{n2} - X_n]\}$
           $E_{min} = E_{min} + d$
           $V = \{X_1 + d, ..., X_N + d\}$
           Reset the clocks of
    IF $t$ is labeled with an input symbol
    THEN Reset the clocks of $\lambda$
ENDFOR
```

Algorithm Computation of E_{max}

```
$V = \{X_1, ..., X_N\}$ is a set of clocks initialized to 0

FOR each $t \in p$ with the time intervals
$[T_{11}T_{12}]...[T_{11}T_{12}]$

    % All time intervals not yet reached
    IF $X_1 \notin [T_{11}T_{12}]$ OR ... OR $X_N \notin [T_{N1}T_{N2}]$
    THEN $\begin{cases} d = MAX\{T_{11} - X_1, ..., T_{N1} - X_1\} \\ V = \{X_1 + d, ..., X_N + d\} \\ E_{min} = E_{min} + d \end{cases}$
    $d = MIN\{[T_{12} - X_1], ..., [T_{n2} - X_n]\}$
    $E_{min} = E_{min} + d$
    $V = \{X_1 + d, ..., X_N + d\}$
    Reset the clocks of $\lambda$
ENDFOR
```

Then, to evaluate the cost to reach a transition t with the preamble p, we define the *TIOA Controllability Degree*, denoted $CT_{tioa}(p,t)$ and calculated with:

$$CT_{tioa}(p,t) = 1 - \frac{E_{min}(p) + E_{max}(p)}{max_{p_i \in Q_A}\{E_{min}(p_i) + E_{max}(p_i)\}}$$

$$\text{and } 0 \leq CT_{tioa}(p,t) \leq 1$$

This degree evaluates the difficulty to reach a transition t with p by comparing the two execution of p with a preamble $p_i \in Q_A$ which have the greatest execution time. If $CT_{tioa}(p,t)$ equals to 0, p is the preamble of Q_A which demands the longest time to reach t. In contrary, if $CT_{tioa}(p,t)$ equals to 1, p is executing without consuming time.

Now, we can measure the difficulty to reach all of the actions of the system A, by calculating the average of CT_{tioa} of each preamble $p_{\xrightarrow{t}}$, allowing to reach a transition t. Therefore, the *Timed System Controllability Degree*, denoted $C_{tioa}(A)$ is evaluated with:

$$C_{tioa}(A) = \frac{\sum\limits_{t \in E_A} CT_{tioa}(p_{\xrightarrow{t}}, t)}{card(E_A)} \text{ and } 0 \leq C_{tioa}(A) \leq 1$$

Consequently, the longer the time, needed to execute the set of preamble is, the less testable the system A is, and the greatest test costs will be. If $C_{tioa}(A)$ equals to 0, execution of the preamble of P demands the longest time in comparison with all the preambles of Q_A that we can choose. In this case, test costs are the greatest. So, the set of preambles P should be modified. If $C_{tioa}(A)$ equals to 1, the preambles of P does not need time to be executed. A such situation can be obtained if no preamble is necessary to reach actions, or if time intervals length equals to 0.

The measurement of the TIOA Controllability Degree of the example of figure 3 gives the following results. We choose the set of preambles denoted in the following tabular. The sequence of Q_A which is executed with the longest time is $?I2?I7?I11!O6?I15!O9?I3?I1$ with $E_{min} = 12$ and $E_{max} = 21$.

Preamble	Nb of Transition Reached	E_{min}	E_{max}	CT_{rg}
ϵ	2	0	0	1
$?I3$	2	0	2	$\frac{31}{33}$
$?I3?I4$	1	0	2	$\frac{31}{33}$
$?I3?I1$	1	2	10	$\frac{21}{33}$
$?I2$	1	0	1	$\frac{32}{33}$
$?I2?I7$	3	0	2	$\frac{31}{33}$
$?I2?I7?I16$	1	0	3	$\frac{30}{33}$
$?I2?I7?I4$	1	0	3	$\frac{30}{33}$
$?I2?I7??I4!O3I11$	1	3	5	$\frac{26}{33}$
$?I2?I7?I11!O6$	3	4	5	$\frac{24}{44}$
$?I2?I7?I11!O6?I15$	1	4	11	$\frac{18}{33}$

As $Card(E_A) = 17$, we obtain $CT(A) = \frac{41}{51}$. This study shows the set P, we have chosen, allows to reach actions of the system rapidly, in comparison with all its sequences, because most of preambles are executed with a low cost. However, the preamble $?I2?I7?I4!O3?I11$ should be replaced by $?2?I7?I11$ which demands less efforts, thus which is more useful for testing.

3.2.4 TIOA Time Independency Degree

Dense time notion is modelled in a TIOA A with one set of clocks C_A, and this one can be composed by any number of clocks, and it is rarely minimal. Indeed, most of specifications, are written in a high-level abstraction with timeouts, and later compiled into timed automata, having a clock number proportional to these timeouts, but these ones are rarely active in the same time. However, this non minimal number of clocks also influences the test costs.

Indeed, the complexity of constructing time intervals depends essentially on the cardinal of C_A, since the number of time intervals per transitions equals to C_A. The complexity to translate TIOA to region graph is also essentially based on $Card(C_A)$ (see [AD94]). Costs of tests execution are also influenced by this number, since the tester must check if either each clocks of the I.U.T (test architecture described in []) or each of its own clocks (test architecture described in [PF99b]) satisfies the specified time intervals, after executing an output action.

As the number of clocks may be infinite, the influence of these ones, on the test, seems difficult to evaluate. So, we consider that a system, specified with a number of clocks upper to the number of transitions, must be reduced.

Now, we can define the *TIOA Time Independency Degree* of a TIOA A, denoted $I_{tioa}(A)$, which aims to measure the influence of the number of clocks on the test costs. This one is evaluated as:

$$I_{tioa}(A) = 1 - \frac{Card(C_A) - 1}{Card(E_A) - 1} \text{ and } 0 \leq I_{tioa}(A) \leq 1$$

Consequently, the more $I_{tioa}(A)$ is close to 1, the more the test costs are reduced and the system is testable. $IfI_{tioa}(A)$ is equals to 1, only one clock is used to specify the system, so the test costs depends essentially on the controllability of A. If $I_{tioa}(A)$ equals to 0, the number of clocks equals to the number of transitions, and the system should be reduced. In [DY96], a method is proposed in order to reduce the number of clocks in a extended timed automaton.

If we apply this degree on the TIOA of the Figure 3, we obtain $I_{tioa} = \frac{15}{16}$. We can conclude that the testing costs are little influenced by the number of clocks of the system since it contains only two clocks.

3.3 Region graph Testability

With region graphs, timing constraints are represented with clock regions (see section 2.2.2), and after studying them, we propose also to define four factors to evaluate the testability of region graph model. These factors are the same as TIOA ones but the properties influencing region graphs test are different: the Timed Shape degree ($S_{rg}(A)$), the Timed Reachability degree ($R_{rg}(A)$), the Timed Controllability degree ($C_{rg}(A)$) and the Timed Independency degree ($I_{rg}(A)$). The first two degrees influence also the coverage of the system during the test, and the last ones the cost of the test execution.

In the same way, we gather them in a vector, called the *Timed Testability Vector* of region graphs, such as:

$$TTV_{rg}(A) =< S_{rg}(RA), R_{rg}(rA), C_{rg}(rA), D_{rg}(RA) >$$

3.3.1 Region Graph Shape degree

Region graphs may also contain clock regions which can contains infinite clock valuations. Indeed, some clock constraints, like $x_i \geq c, x_i \in C_A, c \in \mathbb{R}^+$ can imply such clock regions. We define such a clock region R_i which have infinite values as an infinite clock region:

Definition 3.1 (Infinite Clock Region) *Let $[v]$ be a clock region. $[v]$ is an infinite clock region iff: $\forall d \in \mathbb{R}^+, \exists v \in [v], \exists x_i \in C_A, v(x_i) + d \in [v]$*

The test of an action, in such a infinite clock region R_i is impossible, indeed, as we have seen in section **??** the testing time must be finite, thus such a clock region must be bounded, that is all clocks, given infinite clock values, must be bounded. In this case, we still test only a part of the region graph RA because its behavior can be only checked in a smaller clock region $R'_i \subset R_i$, bounded for each infinite clock. So, the system can only be partially tested, and fault coverage is reduced , on account of these clock regions.

Let $I_{R_i} = \{x_j \in C_A \mid \forall M \in \mathbb{R}^+, \exists v \in R_i \mid v(x_j) + M \in R_i\}$, be the set of infinite clocks of the system in the region R_i. $Card(I_{R_i})$ affect the testability. Indeed, the greater $Card(I_{R_i})$ is, the less completely tested a system will be in R_i. So, we propose a first factor, measuring this evaluation. We call it the *Time Shape Degree of a Region R_i*, denoted ST, and we evaluate it with

$$ST(R_i) = 1 - \frac{card(I_{R_i})}{card(C_A)}, \text{ and } 0 \leq ST(R_i) \leq 1$$

If $ST(R_i)$ equals 1, we say R_i is completely shaped, and all actions, which can be executed in R_i, will be completely tested in R_i. If $ST(R_i)$ equals 0, the system is the most partially tested in R_i.

To measure this evaluation on the complete system A, we define a new degree, called the *Region Graph Time Shape Degree*, denoted $S_{rg}(RA)$ and we evaluate it, for the system RA with:

$$S_{rg}(RA) = \frac{\sum\limits_{R_i \in R_{RA}} ST(R_i) * card(U_{R_i})}{card(AT)} \text{ and } 0 \leq S_{rg}(RA) \leq 1$$

Therefore, the greater $S_{rg}(RA)$ is, the more testable the system will be. If $S_{rg}(A)$ equals 1, all the clock regions are completely bounded, and $\forall R_i \in R_{RA}, I_{R_i} = \emptyset$. In this case, the system may be theoretically completely tested, and we say the system is completely timed shaped. On the other hand, if $S_{rg}(A)$ equals 0, RA have a unique infinite clock region R_i, with $I_{R_i} = C_A$. In this case, the system is the less testable.

If we consider the Region graph, illustrated in figure 5, and its six regions (figure 4), we evaluate:

Clock Region	Nb Clocks Unbounded	$card(U_{R_i})$	ST_{rg}
R_1	0	4	1
R_2	0	4	1
R_3	0	4	1
R_4	0	7	1
R_5	0	6	1
R_6	2	4	0

and $Card(AT) = 29$. Finally, we obtain $S_{rg} = (RA) = \frac{25}{29}$. This factor means the system RA is not completely timed shaped, on account of the clock region R_6 which contains infinite tuple of values.

3.3.2 Clock Region Reachability Degree

The translation of TIOA into region graphs generates clock regions which are the representation of timing constraints of TIOA. All clock regions of a sequence of transitions are time successor, that is, for each clock valuation of the initial clock region, the clocks can reach a valuation of a next clock region. In spite of this new property, we will show that clock regions are not always reachable on account of the uncontrollable behavior of the system, that is on account of the output actions.

Let $\rho(R_i, (S, R_j) \xrightarrow{a} (S', R_i))$ be the factor which evaluates the difficulty to reach the region R_i with the transition $(S, R_j) \xrightarrow{a} (S', R_i)$, knowing that the system's clocks are in R_j. The evaluation of this factor depends on three cases:

- If R_i is the first region reached by the system's clocks, (without executing any action), R_i is always reachable. In this case, the difficulty to reach R_i is denoted $\rho(R_i, \epsilon)$, and $\rho(R_i, \epsilon)$ equals to 1.
- If the action is an input transition $(S, R_j) \xrightarrow{?A} (S', R_i)$, then, the difficulty to reach R_i also equals to 1. Indeed, the tester define when the I.U.T executes the action, by sending "?A" at a clock valuation of R_j. The tester control the execution of the action. So, R_i is always reachable, and $\rho(R_i, (S, R_j) \xrightarrow{?A} (S', R_i))$ equals to 1.
- If the action is an output action, $(S, R_j) \xrightarrow{!A} (S', R_i)$, the difficulty to reach R_i depends only on the behavior of the I.U.T. If this output action may be performed in several clock regions $R_j, ..., R_l$, and allows to reach $R_i, ..., R_k$, we obtain the graph of the figure 6. This case comes from the times intervals which are cut into several clock regions during the region graph translation process.

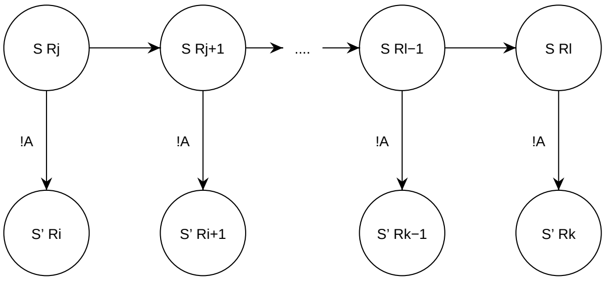

Figure 6: An example of reachability with an output action

In this case, the difficulty to reach R_i is obviously different from 1, because others clock regions can also be reached, instead of R_i. We consider the I.U.T 's clocks can reach $R_i, ..., R_k$, with the same rate. Therefore, the difficulty to reach R_i, with $(S, R_j) \xrightarrow{!A} (S', R_i)$ is measured with:
$\rho(R_i, (S, R_j) \xrightarrow{!A} (S', R_i)) = \frac{1}{Card(\{S_i | S \xrightarrow{\delta} S_i\})+1}$

- Finally, if the action represents an elapse of time $(S, R_j) \xrightarrow{\delta} (S', R_i)$, the difficulty to reach R_i depends on the number of output action which can also be executed from (S, R_j). Indeed, either the system reaches R_i with δ or it executes any output action. Moreover, note that we cannot have two outgoing transitions labeled with δ since clocks grow simultaneously. In this case, the difficulty to reach R_i with $(S, R_j) \xrightarrow{\delta} (S', R_i)$ becomes:
$\rho(R_i, (S, R_j) \xrightarrow{\delta} (S', R_i)) = \frac{1}{Card(\{S_i | S \xrightarrow{! \times \Sigma_{RA}} S_i\})+1}$.

In most of the cases, it is necessary to execute a sequence of actions to reach a clock region of RA. So, let $SEQ_{R_i} \in Q_{RA}$ be the set of sequences allowing to reach R_i. The reachability of R_i with a sequence $\sigma \in SEQ_{R_i}$, denoted $Reach(R_i, \sigma)$ is evaluated with:
$Reach(R_i, \sigma) = \prod_{((s,r) \xrightarrow{a} (s',r')) \in \sigma} \rho(r', ((s, r) \xrightarrow{a} (s', r')))$.

Finally, we measure the reachabilty of R_i, in taking into account all the sequences σ_i of SEQ_{R_i}. Thus, the *Timed Reachability Degree of a Region R_i*, denoted $RT(R_i)$ is obtained with:

$$RT(R_i) = \frac{\sum\limits_{\sigma_i \in SEQ_{R_i}} Reach(R_i, \sigma_i)}{Card(SEQ_{R_i})} \text{ and } 0 < RT(R_i) \leq 1$$

So, if $RT(R_i)$ equals 1, the tester may always reach R_i with any action of SEQ_{R_i}. On the other hand, if $RT(R_i)$ is close to 0, R_i can be rarely reached, because of most of the actions in SEQ_{R_i} can also reach others clock regions.

Now, we can define the Timed Reachability of a system, which evaluates the reachability of the clock regions of the system. Consequently, the *Timed System Reachability Degree* , denoted $R_{rg}(A)$ is evaluated with:

$$R_{rg}(A) = \frac{\sum\limits_{R_i \in R_{RA}} RT(R_i) * Card(U_{R_i})}{Card(AT)} \text{ and } 0 < R_{rg}(A) \leq 1$$

So, the greater $R_{rg}(A)$ is, the more testable the system is. If $R_{rg}(A) = 1$, we say the system is completely timed reachable. All the clock regions of RA are completely reachable, and actions can be tested in all time intervals. On the contrary, if $R_{rg}(A)$ is close to 0, several actions will not be tested, because most of the clock regions will not be reached by the I.U.T 's clocks.

The following results show the reachability of each clock region of the region graph, illustrated in figure 5 .

Clock Region	$card(SEQ_{R_i})$	RT_{rg}	$Card(U_{R_i})$
R_1	32	$\frac{41}{64}$	4
R_2	14	$\frac{11}{14}$	4
R_3	4	$\frac{7}{8}$	2
R_4	28	$\frac{35}{56}$	7
R_5	54	$\frac{29}{54}$	6
R_6	36	$\frac{23}{36}$	4

The Clock Region Reachability Degree obtained is $R_t(A) \simeq \frac{267}{400}$. Consequently, we can conclude that this system is not completely timed reachable, since no clock region cannot be always reached with sequences of actions of the system. Indeed, the action "!O3" allows to reach the clock regions R_1, R_2 from the state $(TMP3, R_1)$, and R_3, R_4 from the state $(TMP4, R_3)$. In this case, we must take care of the choice of the preambles for testing. Each action of the system can be tested, if no preamble contains the transitions $(TMP3, R_1) \xrightarrow{!O3} (WFUR, R_1)$, $(TMP4, R_3) \xrightarrow{!O3} (DAcc, R_3)$. Indeed, without theses preambles, all clock regions will be reached, during the test.

3.3.3 Region Graph Controllability Degree

The evaluation of the Controllability degree, for the region graph model, is roughly the same as the TIOA Controllability Degree one. Indeed, test cases, generated from region graphs, still contains preambles whose contains actions which are executed in time intervals, modelled by clock regions.

The main difference comes from the calculus of the minimal and the maximal time execution of a preamble p, $E_{min}(p)$ and $E_{max}(p)$. Indeed, the elapse of time needed to reach a next clock region after having executed an action, and the elapse of time needed to execute an action the latest possible, must be calculated as follow:

Elapse of time needed to reach the next clock region

Consider the sequence $(s_1, R_1) \xrightarrow{A} (s_2, R_2) \xrightarrow{\delta} (s_3, R_3) \xrightarrow{B} (s_4, R_4)$. We need to calculate the elapse of time, represented by δ, needed to reach R_3 from any clock valuation $v_{init} \in R_2$, reached after executing the action "A", that is the minimal value d such as $v_{init} + d \in R_3$.

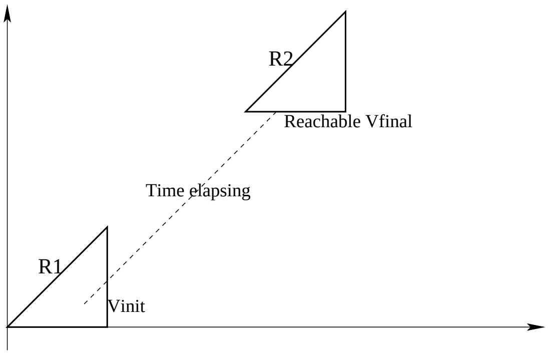

Figure 7: Reaching another clock region in v_{final} from v_{init}

Let $v_{init} = (v_1, ..., v_n)$. As, the system clocks grow with the same manner, strictly monotonically, it is easy to prove they take the values of the equation $\begin{cases} x_1 - x_2 = v_1 - v_2 \\ ... \\ x_{n-1} - x_n = v_{n-1} - v_n \end{cases}$

So, the first clock valuation v_{final} reached by the clocks in R_3 is unique and is obtained by resolving the system of inequations

$$\Delta = \begin{cases} \text{Inequations of } R_3 \\ x_1 - x_2 = v_1 - v_2 \\ ... \\ x_{n-1} - x_n = v_{n-1} - v_n \end{cases}$$

Finally, the elapse of time δ equals to the difference between v_{init} and the minimal solution v_{final} of Δ. An example, illustrating this elapse of time is given in 7.

Elapse of Time needed to execute an action the latest possible in a clock region

Actions of region graphs can be executed between the first clock valuation v_{init} of a clock region R, reached by the clocks, and the last one v_{final}. For the calculus of E_{min}, we consider that the output actions are executed the latest possible, that is at v_{final}. And for the calculus of E_{max}, all of the actions must be executed at this one. So, another elapse of time, which is not represented in region graphs, is needed to reach v_{final}. The figure 8 illustrates this elapse of time.

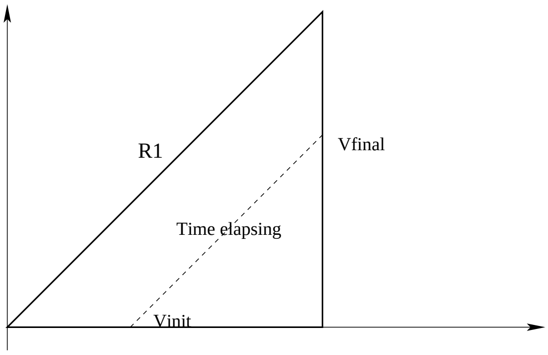

Figure 8: Reaching the last clock valuation v_{final} of a clock region from v_{init}

The time, needed to reach v_{final} can be computed as the previous way. The clocks also take the values of the equation $\begin{cases} x_1 - x_2 = v_1 - v_2 \\ ... \\ x_{n-1} - x_n = v_{n-1} - v_n \end{cases}$

And v_{final} is the maximal solution of the system of inequations

$$\Delta = \begin{cases} \text{Inequations of } R \\ x_1 - x_2 = v_1 - v_2 \\ ... \\ x_{n-1} - x_n = v_{n-1} - v_n \end{cases}$$

Finally, the elapse of time equals to the difference $v_{final} - v_{init}$.

$E_{min}(p)$ and $E_{max}(p)$ can be still computed with the algorithms, described in section 3.2.3, by replacing the calculus of elapse of time by the previous ones.

The difficulty to reach a transition t with the preamble $p \in P$ is still evaluated with the *Region Graph Controllability Degree*, denoted $CT_{rg}(p, t)$ and calculated with:

$$CT_{rg}(p, t) = 1 - \frac{E_{min}(p) + E_{max}(p)}{max_{p_i \in Q_RA}\{E_{min}(p_i) + E_{max}(p_i)\}}$$

$$\text{and } 0 \leq CT_{rg}(p, t) \leq 1$$

The measurement of the cost of the time execution of the set of preambles P, can be evaluated with the *Region Graph Controllability Degree*, denoted $C_{rg}(RA)$ which is calculated as:

$$C_{rg}(RA) = \frac{\sum_{t \in AT} CT_{rg}(p_{\xrightarrow{t}}, t)}{card(AT)} \text{ and } 0 \leq C_{rg}(RA) \leq 1$$

Consequently, the longer the time, needed to execute the set of preamble is, the less testable the system A is, and the greatest test costs will be. In the same way, if $C_{rg}(A)$ equals to 0, execution of the preamble of P demands the longest time in comparison with all the preambles of Q_{RA}, and test costs are the greatest. If $C_{rg}(RA)$ equals to 1, the preambles of P does not need time to be executed. This situation is obtained if all of the clock regions of RA can be represented with the point $(0, ..., 0)$.

Now, we can evaluate the Timed Controllability Degree on the example of figure 5. We choose the set of preambles, denoted in the following tabular. The sequence of Q_{RA} which is executed with the longest time is $?I2?I7?I11!O6?I15!O9?I3?I1$ with $E_{min} = 12$ and $E_{max} = 21$. We obtain the following results:

Preamble	Nb of transitions reached	E_{min}	E_{max}	CT_{rg}
$?I3$	5	0	2	$\frac{31}{33}$
$?I3?I4$	2	0	2	$\frac{31}{33}$
$?I3?I1$	3	2	10	$\frac{21}{33}$
$?I3?I1!O1$	3	2	10	$\frac{21}{33}$
$?I2$	1	0	1	$\frac{32}{33}$
$?I2?I7$	4	0	2	$\frac{31}{33}$
$?I2?I7?I16$	1	0	3	$\frac{30}{33}$
$?I2?I7?I4$	2	0	3	$\frac{30}{33}$
$?I2?I7?I11$	2	2	5	$\frac{26}{33}$
$?I2?I7?I11!O6$	5	4	5	$\frac{24}{33}$
$?I2?I7?I11!O6?I15$	1	4	11	$\frac{18}{33}$

As $Card(AT) = 29$, we obtain $CT(RA) = \frac{219}{319}$. This study shows the set P, we have chosen, allows to reach actions of the system rapidly, in comparison with all its sequences, because most of preambles are executed with a low cost. However, the preamble $?I3?I1!O1$ should be replaced by ϵ which is a preamble which demands no efforts, thus which is more useful for testing.

3.3.4 Timed System Independency Degree

For timed systems, the methodology, described in [PF00], aims to test each action $(S_i, R_i) \xrightarrow{?A} (S_j, R_j) \in E_{RA}$, at clock valuations equals to the vertices of R_i. Thus, let ϑ_{R_i} be the set of vertices of the clock region R_i. The greater the number of vertices is, the longer the test of an input action, executed in R_i will be. Indeed, let p be the preamble allowing to reach an input action to test. p will be applied $card(\vartheta_{R_i})$ times to the I.U.T, in order to reach each vertex of R_i.

So, $card(\vartheta_{R_i})$ affect also the test cost of A. To evaluate this difficulty, we suggest a new degree, called the *Timed Independency Degree of a Region* R_i, denoted $IT(R_i)$ which measures the difficulty to test any action in R_i, due to the number of its vertices. Let N be $max(card(\vartheta_{R_i}) \mid R_i \in R_{RA}$. We obtain the following:

$$IT(R_i) = \frac{N - card(\vartheta_{R_i})}{N-1} \text{ and } 0 \leq IT(R_i) \leq 1$$

So, if $IT(R_i)$ equals to 0, R_i is the clock region of R_{RA} which have the more of vertices. The test of an action in R_i will be the longest. On the other hand, if $IT(R_i)$ equals to 1, R_i contains only one vertex, thus only one clock valuation. Thus, the test cost is minimal with R_i

For the complete system A, the greater the number of clock regions vertices is, the greater the test cost of A will be. So, we define the *Timed System Independency Degree*, denoted $I_t(A)$, to measure this evaluation:

$$I_t(A) = \frac{\sum_{R_i \in R_{RA}} IT(R_i) * Card(U_{R_i})}{card(AT)}$$

$$\text{and } 0 \leq I_t(A) \leq 1$$

Thus, the more independent on time the system is, the less effort the test demands, and the more testable A is. If $I_t(A)$ equals 0, all the clock regions of the system are built with N vertices. In this case, the test will be the longest, and the test cost will be the greatest. On the other hand, if $I_t(A)$ equals 1, only one clock valuation by clock region is used for testing. Thus an action is checked only one time in a such clock region. The test will be rapidly executed.

Now, we can apply this factor to the system described in figure 2. Before, we will bound the infinite regions with $x = 2$ and $y = 2$. Then, we obtain:

Clock Region	Number of vertices	U_{R_i}	IT
R_1	3	3	$\frac{1}{3}$
R_2	2	3	$\frac{2}{3}$
R_3	3	1	$\frac{1}{3}$
R_5	4	2	0
R_6	4	2	0

and $Card(AT) = 11$. Consequently, we obtain $I_t(A) = \frac{10}{33}$. We can conclude that the test cost is important with this system, on account of the number of vertices of the clock regions.

4 Discussion and concluding remarks

In this paper, we have introduced some factors for evaluating test quality of timed system, modelled with TIOA or region graphs. We have shown how they can be measured and their influence on the testing process.

4.1 Summery

We have show that, in many cases, the timed implementation may be only partially tested, on account either of the infinity of the time intervals or clock regions of the specification or of the unrechability of some of its timed parts. So, the testability of timed systems does not depend only on its observation and its control.

So, that motivates a new definition of testability for timed systems which is:

Definition 4.1 *A timed system is testable iff it is observable, controllable, and iff it can be completely tested in all of its time parts: that is iff all time intervals or clock regions can be always reached by the implementation 's clocks, and if each one can be completely covered during the system functioning.*

The measurement of timed systems testability, before the test sequences generation, is obtained with the *Timed Testability vector*, TTV_{tioa} or TTV_{rg}, and with the *Testability vector* $TV(A)$, defined in [KGD96]. A unique value of testability can be evaluated with a level-headed average of all these degrees.

A system A is the most testable if we obtain the following vectors $TTV_{tioa}(A) =< 1,1,1,1 >$ and $TV(A) =< 1,1,1,1 >$, or $TTV_{rg}(RA) =< 1,1,1,1 >$ and $TV(RA) =< 1,1,1,1 >$. But the best TTV can be rarely obtained, since the performance is not always compatible with the test coverage. In fact, the best testability of timed systems depends mainly on criteria's designer.

For example, if we want to test a software or a protocol, the more rapidly as possible, even if it will be only partially tested in time, the vector $< C_{tioa}, I_{tioa} >$ or $< C_{rg}, I_{rg} >$ is sufficient to measure the test costs. Indeed, C_{tioa} or C_{rg} evaluates the effort needed to execute the preambles of A or RA, and I_{tioa} or I_{rg}, the efforts due to the addiction of time.

On the other hand, if we want to test completely the system independently of test cost, we can only consider the vector $< S_{tioa}, R_{tioa} >$, or $< S_{rg}, R_{rg} >$. These tuple evaluate whether the system is tested in all of its timing parts and if these ones will be completely covered, during the test.

Other degrees can be defined or some of the previous ones can be modified to be more expressive and adapted to a specific testing methodology. For example, for methodologies, [PF99a, ENDKE98], using a characterization of each state for generating test sequences, a degree could be defined to evaluate the cost of this characterization, in taking into account behavior but also temporal properties of the specification. The controllability degree could be also redefined to consider the test execution, described in methodologies. This allows to measure, with precision, the costs of the test. An exemple of metrics, depending on the methodology described in [PF99a], is given in [SSb00].

4.2 Complexity

The calculation of these degrees can be long, especially with real time systems, because their complexity is proportional to their number of clock regions or time intervals and their number of actions.

- For the Timed Shape degree, the complexity of S_{tioa} is proportional to K, with K the number of actions. The complexity of $S_{rg}$is proportional to N, with N the number of clock regions.
- For the Timed Reachability Degree, the complexity of R_{tioa} is proportional to $K * card(Q_A) + M * K$, with M the number of states. Q_A is obtained by constructing the cover tree of A , and then enumerating the partial sequences in the tree. Since each action appear once in the tree, the complexity is proportional to $M * K$. Moreover, the complexity of the calculus of the reachability of one clock region is obtained by scanning, in the worst, all sequences of Q_A. For region graphs, we consider the number of clock regions, otherwise the timed reachability is constructed in the same manner, so the complexity of R_{rg} is proportional to $N * card(Q_{RA}) + M * K$.
- For the Timed Controllability Degree, the complexity of C_{tioa} is proportional to $card(Q_A) * 2K + M * K$. For constructing this degree, two executions are considered, and the length of a preamble is, in the worst case, equals to $K - 1$. Then, it is necessary to scan each preamble, and in the worst case, the set of preambles $P = Q_A$. The complexity to construct Q_A is still equals to $M * K$. For region graphs, the complexity is equivalent to the previous one, and is proportional to $card(Q_{RA}) * 2K + M * K$.
- For the Timed Independency Degree, the complexity is proportional to K for TIOA or to N for region graphs.

5 Conclusion

In this paper, we have introduced some factors for evaluating test quality of timed systems. These factors depends heavily on system specification structure. These factors consider the behavior part as well as the timing part of a system. We have shown how they can be measured on any timed system and their influence on the testing process.

For the near future, we intend to include the computation of these factors in a testing tool which has been undertaken in our laboratory since one year. We have also began working on a testing generation technique based on state characterization based on a region graph as defined by Alur and Dill [AD94]. This last step will help us in the design of our timed testing tool.

References

[AD92] R. Alur and D. Dill. The theory of timed automata. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings REX Workshop on Real-Time: Theory in Practice,* Mook, The Netherlands, June 1991, volume 600 of *Lecture Notes in Computer Science*, pages 45–73. Springer-Verlag, 1992.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[BGK$^+$96] J. Bengtsson, D. Griffioen, K. Kristoffersen, K.G. Larsen, F. Larsson, P. Pettersson, and W. Yi. Verification of an audio protocol with bus collision using UPPAAL. In *Proceedings of the 8th International Conference on Computer Aided Verification,* New Brunswick, New Jersey, USA, volume 1102 of *Lecture Notes in Computer Science*, pages 244–256. Springer-Verlag, August 1996.

[BL] J. Bengtsson and F. Larsson. *UPPAAL - a Tool Suite for Symbolic and Compositional Verification of Real-Time Systems (Draft).*

[DOTY95] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool Kronos. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages –. Springer-Verlag, 1995.

[DY95] C. Daws and S. Yovine. Two examples of verification of multirate timed automata with KRONOS. In *Proceedings of the 1995 IEEE Real-Time Systems Symposium, RTSS'95,* Pisa, Italy. IEEE Computer Society Press, 1995.

[DY96] C. Daws and S. Yovine. Reducing the number of clock variables of timed automata. In *Proceedings of the 1996 IEEE Real-Time Systems Symposium, RTSS'96,* Washington DC, USA. IEEE Computer Society Press, 1996.

[ENDKE98] A. En-Nouaary, R. Dssouli, F. Khendek, and A. Elqortobi. Timed test cases generation based on state characterization technique. In *19th IEEE Real Time Systems Symposium (RTSS'98)* Madrid, Spain, 1998.

[Fre91] R. S. Freedman. Testability of software components. *IEEE transactions on Software Engineering*, 17(6), june 1991.

[KDCK94] K. Karoui, R. Dssouli, O. Cherkaoui, and A. Khoumsi. Estimation de la testabilité d'un logiciel modélisé par les relations. 1994. research report #921.

[KGD96] K. Karoui, A. Ghedamsi, and R. Dssouli. A study of some influencing factors in testability and diagnostics based on fsms. 1996. research report #1048.

[LC97] Patrice Laurencot and Richard Castanet. Int egration of Time in Canonical Testers for Real-Time Systems. In *International Workshop on Object-Oriented Real-Time Dependable Systems, California.* IEEE Computer Society Press, 1997.

[LPB93] G. Luo, A. Pentrenko, and G.v Bochman. Selecting test sequences for partially-specified nondeterministic finite state machines. feb 1993. research report #864.

[MY96] O. Maler and S. Yovine. Hardware timing verification using KRONOS. In *Proceedings of the IEEE 7th Israeli Conference on Computer Systems and Software Engineering, ICCBSSE'96.* IEEE Computer Society Press, June 1996.

[NSY92] X. Nicollin, J. Sifakis, and S. Yovine. Compiling real-time specifications into extended automata. *IEEE Transactions on Software Engineering*, 18(9):794–804, September 1992.

[PF99a] Eric Petitjean and Hacène Fouchal. From Timed Automata to Testable Untimeed Automata. In *24th IFAC/IFIP International Workshop on Real-Time Programming, Schloss Dagstuhl, Germany*, 1999.

[PF99b] Eric Petitjean and Hacène Fouchal. A Realistic Architecture for Timed Systems. In *5th IEEE International Conference on Engineering of Complex Computer Systems, Las Vegas, USA*, pages 109–118, 1999.

[PF00] E. Petitjean and H. Fouchal. A fault detection on timed systems. Report: Resycom-2000-01-01, LERI-RESYCOM (Université de Reims), 2000.

[PLC98] O. Koné P. Laurencot and R. Castanet. On the Fly Test Generation for Real Time Protocols. In *International Conference on Computer Communications and Networks, Louisiane U.S.A*, 1998.

[SSb00] Hacene Fouchal Sebastien Salva and Simon bloch. Metrics for Timed Systems Testing. In *?th OPODIS International Conference on Distribued Systems, Paris*, 2000.

[SY96] J. Sifakis and S. Yovine. Compositional specification of timed systems. In *Proceedings of the 13th Annual Symp. on Theoretical Aspects of Computer Science, STACS'96*, volume 1046 of *Lecture Notes in Computer Science*, pages 347–359, Grenoble, France, 1996. Springer-Verlag.

[YL93] M. Yannakakis and D. Lee. An efficient algorithm for minimizing real-time transition systems (Extended abstract). In C. Courcoubetis, editor, *Proceedings of the 5th International Conference on Computer Aided Verification,* Elounda, Greece, volume 697 of *Lecture Notes in Computer Science*, pages 210–224. Springer-Verlag, 1993.

[YPKP95] N. Yevtushenko, A. Petrenko, K. Karoui, and S. Prokopenko. On the design for testability of communication protocols. *IWPTS'95*, 1995.