RAPPORT DE RECHERCHE

L R I

DESIGNING TIMED TEST CASES FROM REGION GRAPHS

PETITJEAN E

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud-LRI

03/2003

Rapport de Recherche N° 1354

CNRS – Université de Paris Sud
Centre d'Orsay
LABORATOIRE DE RECHERCHE EN INFORMATIQUE
Bâtiment 650
91405 ORSAY Cedex (France)

Designing timed test cases from region graphs.

Eric Petitjean

LRI, CNRS UMR 8623
University of Paris XI, 91450 ORSAY CEDEX, France
email : eric.petitjean@lri.fr

Abstract

Timed automata are a widely studied model for timed systems and is commonly used for test and verification. Formal methods are often based on their associated translation as region graphs, with the addition of a new symbol - δ - to represent the elapse of time. In this paper, we study the semantics of this symbol and exhibit its three possible roles as input, output and internal action. We describe then the consequences of this ambiguous nature on the design and the execution of timed test cases generated from a region graph. We also study its impact on methods based on timed test purposes, and define new clock zones for each possible verdict.

Keywords : Timed automata, region graph, conformance testing, controllability, test design, test purpose, test execution.

1 Introduction

Testing is widely recognized as a crucial step in the life cycle of computer systems, and it alone weighs approximately half the development cost of any software. With the increasing presence of time evolving systems, such as multimedia or communication protocols, the automatic generation of test sequences for these systems has become in the last decade an important stake for developers. Since their introduction by Alur and Dill in [AD90, AD94], timed automata are one of the most studied models for real-time systems, both for verification and testing. They form, along with their classical translation as region graphs, are the underlying model of several existing verification tools (like KRONOS [DOTY95], UPPAAL [LPY97] and HYTECH [HHWT97]) and automatic test generation algorithms [EnDKE98, JSD97, FP01].

This translation as a region graph usually introduces a new symbol, δ, in order to model the elapse of time and allows to consider any other translation as being instantaneous. This symbol has already be identified as an ambiguous one, being neither really an input nor an output symbol. Both functions has been associated uniformally to it in timed test generation techniques, but it has always raised problems when test sequences were to be executed. In this paper, we study the possible occurences of δ in a region graph and we identify the three different roles it can play, namely input, ouput and internal action.

This constatation, and mainly the identification of its occurences as an internal action, implies that almost every region graph is nondeterministic and that test sequences as single branches are no more relevant, at least not with simple associated verdicts. We therefore propose a new shape for the elementary components of timed test cases, named timed test elements, and we introduce the multiple timed output element to take into account all possible moments of reception of an output action sent by the implementation under test from a given state. Then, we describe the execution algorithm of this new element and we give a first outline of a global test execution algorithm in an uncontrollable context.

Finally, we study the impact of the variations in the design of test cases on test generation methods based on timed test purposes, and we propose new definitions for clock zones labeled with the verdicts pass, fail, and inconclusive depending on their coherence with the specification and the test purpose.

2 Inherent uncontrollability of region graphs

2.1 Alur and Dill's timed automata

Since they have been proposed as a model in [AD90], timed automata have been widely used to model finite-state real-time systems. The global structure of such automata is the follwing one : each automaton has a finite set of states and a finite set of clocks which are real-valued variables ; all clocks proceed at the same rate and measure the amount of time that has elapsed since they were started or reset ; each transition of the system might reset some of the clocks, and has an associated enabling condition, which is a constraint on the clock values, and which the clock values must satisfy to allow the firing of the transition.

Definition 1 Clock constraints and clock guard
A clock constraint *over a set C of clocks is a boolean expression of the form $x \sim c$ where $x \in C$, $\sim \in \{<, \leq, =, \geq, >\}$, and $c \in \mathbf{N}$*
A clock guard *over C is a conjunction of clock constraints over C.*

Definition 2 Timed Automata
A timed automaton A is defined as a tuple $(\Sigma_A, L_A, l_A^0, C_A, E_A)$, where :

- *Σ_A is a finite alphabet,*
- *L_A is a finite set of states,*
- *$l_A^0 \in S$ is the initial state,*
- *C_A is a finite set of clocks,*
- *$E_A \subseteq L_A \times L_A \times \Sigma_A \times 2^{C_A} \times \Phi(C_A)$ is the set of transitions.*

An edge (l, l', a, λ, G) represents a transition from state l to state l' on input or output symbol a. The subset $\lambda \subseteq C_A$ allows some clocks to be reset with this transition, and G is a clock guard over C_A. $\Phi(C_A)$ is the set of clock guards over C_A

A classical refinement of these timed automata [Kon94] consists in the usual dinstinction between input and output actions. It divides the alphabet into two subsets : the first one contains the input symbols, beginning with a ?, and the second one the output symbols, beginning with a !. Since our study deals exlusively with these timed input/output automata, no confusion should be possible and we will in all this paper use the terms 'timed automaton' for 'timed input/output automaton'.

It is noticeable that the initial model has indeed proven so popular that multiple other extensions or derived models has been developed, among which grid automata [JSD97], timed automata with periodic clock constraints [CG00], updatable timed automata [BDFP00], and extended timed input/output automata [Lau99].

2.2 Region graphs

In [AD94], Alur and Dill develop an algorithm of translation of a timed automaton into a labeled transition system whose transitions are free of clock constraints. The underlying idea is to solve the problem raised by the infinity of values any clock can take by grouping them in classes, named regions. These regions are granted two essential properties : the notion of immediate time succesor is a function and in one given region, every clock constraint of the automaton is either uniformally satisfied or unsatisfied. Originally, the alphabet of the region graph was identical to the automaton's one and time could pass while a transition was crossed. A classical variation of Alur and Dill's region graphs consists in the addition of a new symbol, δ, to represent the elapse of time. Thus, any transition labeled with another symbol is considered to be taken instantaneously, i.e. the clocks which are not reset have the same value before and after the transition is fired. Our study will concern exclusively this variation, which is one of the most used where formal methods are to be applied to region graphs, especially in the fields of test and verification.

Definition 3 Clock valuation
A clock valuation over a set of clocks C is a map v that assigns to each clock $x \in C$ a value in $\mathbf{R}^+$ (set of nonnegative reals). The set of clock valuation is noted $V(C)$.
A clock valuation v satisfies a clock guard G, denoted $v \models G$, if and only if G evaluates to

true under v.
For $d \in \mathbf{R}^+$, $v+d$ denotes the clock valuation which assigns a value $v(x)+d$ to each clock x.
For $X \subseteq C$, $[X \mapsto d]v$ denotes the clock valuation for C which assigns d to each $x \in X$, and agrees with v over the rest of the clocks.

Definition 4 Clock region
Let $A = (\Sigma_A, L_A, l_A^0, C_A, E_A)$ be a timed automaton.
$\forall x_i \in C_A$, let $c_x = max\{c \mid ((x \leq c) \vee (c \leq x))$ is a constraint over $x_i\}$
The equivalence relation $\sim$ is defined over the set $V(C_A)$; $v \sim v'$ iff :
$\forall x \in C_A, (\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor) \vee ((v(x) \geq c_x) \wedge (v'(x) \geq c_x))$
$\forall x, y \in C_A \mid ((v(x) \leq c_x) \wedge (v(y) \leq c_y)), (\{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\}))$
$\forall x \in C_A \mid v(x) \leq c_x, (\{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0)$
A clock region for A is an equivalence class of clock valuations induced by $\sim$. Let $[v]$ denote the clock region to which v belongs.

Definition 5 Region graph
Let $A = (\Sigma_A, L_A, l_A^0, C_A, E_A)$ be a timed automaton. The (classical) region graph of A is the Büchi automaton $RA = (\Sigma_{RA}, S_{RA}, s_{RA}^0, E_{RA})$ where:

- $\Sigma_{RA} = \Sigma_A$, *where δ represents the elapse of time*
- $S_{RA} \subseteq \{\langle s, [v]\rangle \mid s \in S_A \wedge v \in V(C_A)$
- $s_{RA}^0 = \langle l_A^0, [v_0]\rangle$ *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 $q' = \langle s', [v']\rangle$ on action a, iff either*
  - $a \neq \delta$, E_A *contains a transition (s, s', a, λ, G) and there is $d \in \mathbf{R}^+$ such that $(v+d) \models G$ and $v' = [\lambda \mapsto 0](v+d)$,*
  - $a = \delta$, $s = s'$ *and there exists $d \in \mathbf{R}^+$ such that $v' = v + d$.*

Since these definitions lead often to a partition of the n-dimensional space of clock valuations (where n is the number of clocks) with more elements than necessary, some algorithms [ACH+92, YL93] have been developed to minimize the obtained region graph and merge such clock regions as those whose separation did not express any change in the behaviour of the specified system. These unions of clock regions are called clock zones, and correspond geometrically speaking to n-dimensional polyhedrons.

In this article, and particularly in section 4 where test execution is described, we will consider the region graphs to be minimal.

2.3 δ : an ambiguous symbol

Even if the δ symbol is useful since it grants the formal region graph model an increased readability and allows at once to identify the clock zones satisfying a given time constraint in the time automaton, it nonetheless becomes very confusing when an attempt to automatically generate test cases is carried out. The main question is to decide whether it should be treated as an input or an output symbol. Both choices have already been taken in different works on the subject, which tends to prove that both show some pertinence, but unhappily they obviously exclude one another.

Actually, δ is neither an input or an ouput symbol, or more precisely it is both, and assumes another additional function, depending on his place in the graph and on the outgoing transitions of the states it joins.

Let us consider first the case when the timed automaton which the region graph has been translated from is controllable (or, as it is also defined, deterministic w.r.t. test). In this case, if a state has an outgoing transition t labeled by an output symbol, it can bear no other transition whose clock constraints are satisfied when t clock constraints are. This ensures that the system described in the specification will never have to chose between an input or an output, or between two output actions.

2.3.1 Input

When a transition labeled with the δ symbol links two states which both have an outgoing transition labeled with the same input symbol, it only means that the global set of clock valuations in which this symbol can be submitted to the implementation has been divided in several clock zones because of other clock contraints expressed somewhere else in the specification. Since we consider here the controllable case, we are ensured that these two transitions are the only ones for the two considered states.

From a behavioral point of view, the user, or in our context the tester, is able here to decide in which state it will submit the input symbol, and consequently decide whether or not it will execute the δ transition to let time evolve until the next clock zone. δ can, and must, be in this case considered as a decision from the tester, the implementation's only role being to wait for the input symbol to be submitted. It has therefore, as far as test is concerned, the same characteristics as an input symbol.

For example, in the region subgraph shown in figure 1, a same input symbol $?A$ can be submitted to the system from state 0 in both clock zones z and z' . In this situation, the tester can choose to submit $?A$ directly from zone z or to wait until the clock valuation belongs to z' (i.e. execute the δ transition) to perform the very same action. This choice provides, by itself, some granularity in the testing of the whole period when it is possible to submit $?A$.

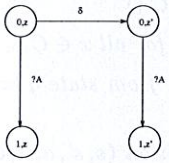


Figure 1: δ as an input symbol

2.3.2 Internal action

When a transition labeled with the δ symbol links two states which both have an outgoing transition labeled with the same output symbol, it only means that the global set of clock valuations in which this symbol can be sent by the implementation has been divided in several clock zones because of other clock contraints expressed somewhere else in the specification.

In this case, in comparaison with the previous one (section 2.3.1), the repective roles of the implementation and the tester are swapped : the tester waits for the reception of the output symbol, and the implementation chooses autonomously the moment of the sumbmission. Since we are dealing here with conformance testing, we consider the implementation as a black box, and we have no more observation power than control power over this choice which is, from an external point of view, perfectly arbitrary. To sum it up, the firing of the δ transition, in this case, is the consequence of a unilateral decision from the implementation, and is not expressed, at least immediately, by any element in the control flow : it has all the characteristics of an internal action.

For example, in the region subgraph shown in figure 2, a same output symbol $!B$ can be sent by the system from state 0 in both clock zones z and z'. In this case, the tester's only function is to wait for the reception of this symbol and to consider it to be valid (or conform) in both zones z and z'.

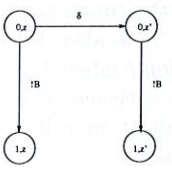


Figure 2: δ as an internal action

2.3.3 Output

This last case is probably the one most open to discussion, since it could have been included in the previous one (section 2.3.2), considering δ as an internal action and proceeds indeed from the same type of transition in the original timed automaton. We will try and explain why we have chosen to make it a distinct one.

When a transition labeled with the δ symbol leads from a state have an outgoing transition labeled with the output symbol $!B$ to a state where no such transition exists, it means that the clock guard enabling the emission of $!B$ has seen its evaluation evolve from true to false with the change of clock zone. Since clock guards are always conjunctions of clock constraints, once the δ transition crossed, the output symbol $!B$ cannot be emitted any more (at least not in the context of the same transition of the initial timed automaton). This additional information is the reason for our distinguishing between this case and the previous one. The δ transition stays as uncontrollable as before, but reaching its arrival clock zone implies the non-reception of the output symbol. Even if to consider it an ouput action is a rather abusive interpretation of the term, since it traduces more an absence of ouput action, the *delta* symbol, in this case, has nonetheless much more to say in a test context than it had in the previous one.

For example, in the region subgraph shown in figure 2, the output symbol $!B$ can be sent by the system from state 0 in clock zone z, but not in z'. In this case, if the reception of $!B$ is compulsory for the conformance to the specification, the state $(0, z')$ must be associated with a *fail* verdict.

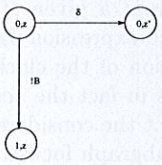


Figure 3: δ as an output symbol

2.3.4 Uncontrollable case

If the initial timed automaton is uncontrollable, the δ transitions of its associated region graph may, and almost automatically do, assume more than one role among the three described in section 2.3. It obviously adds to the multiplicity of the functions represented by this symbol, and compels us to consider all possible cases whenever we have to include such transitions in test sequences, both in generation and execution.

3 Implications on test cases design

3.1 Timed elements

In [FP01], the authors define the timed test elements as the atomic components of timed test cases, and identify their three possible nature : timed input elements, timed output elements, and time elapsing elements. We recall here briefly the corresponding definitions.

Definition 6 *Timed input element*
A timed input element, or TIE, is a tuple $(s_i, z_i, ?a, s_f, z_f)$ *where* $s_i, s_f \in \Sigma_A$, $?a$ *is an input action, and* z_i, z_f *are clock zones.*

Definition 7 *Timed output element*
A timed output element, or TOE, is a tuple $(s_i, z_i, !b, s_f, z_f)$ *where* $s_i, s_f \in \Sigma_A$, $!b$ *is an output action, and* z_i, z_f *are clock zones.*

Definition 8 *Time elapsing element*
A time elapsing element, or TEE, is a tuple $(s_i, z_i, \delta, s_i, z_f)$ *where* $s_i \in \Sigma_A$, *and* z_i, z_f *are clock zones.*

According to these definitions, timed test elements are no more than transitions in the minimal region graph. We will now exhibit the consequences of the ambigous nature of δ on each of these elements.

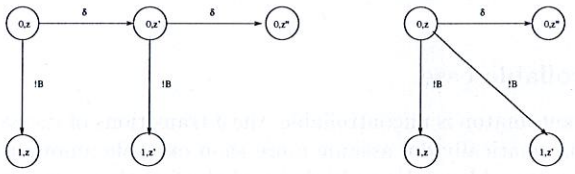
3.2 Timed output element

To take into account the uncontrollability of the translation in the region graph of a transition labeled with an output symbol (see section 2.3.2), any test case must, whenever it contains a TOE, contain also all the corresponding following TOE - w.r.t. time, not control flow, and the last TEE. Practically, we obtain from this point no more a test case as a single branch, but as a test tree taking into account the whole set of clock values in which the output symbol can be submitted, starting from the current clock zone, as well as the moment from when this submission becomes invalid. Practically, we obtain a multiple timed output element, which we define as :

Definition 9 *Multiple timed output element*
A multiple timed output elements is a set $MTOE = (s_i, z_i, !b, s_f, z) \cup (s_i, z_i, \delta, s_i, z_e)$ *where* z_e *is the first time successor of* z_i *such as the state* (s_i, z_e) *has no outgoing transition labeled by* $!B$*, and there exists for each clock zone* z *a clock zone* z' *and a path* $(s_i, z_i) \xrightarrow{\delta*} (s_i, z') \xrightarrow{!b} (s_f, z)$ *in the minimal region graph.*

A multiple timed output element is no more a transition of the minimal region graph, but one of its subtrees containing, from a given state (s, i, z_i), all paths from this state whose corresponding word matches the expression $\delta*!b$, along with a last path only labeled by some δ's which models the expiration of the clock guard enabling the submission of the output symbol. This structure models in fact the actual observable global behaviour of the specification from the state (s_i, z_i) w.r.t the considered output symbol $!b$.

In figure 4, we compare a region subgraph focused on an output transition and its corresponding MTOE represented as a tree.



(a) A region subgraph (b) MTOE

Figure 4: Specification and observable behaviour

It is noticeable that such a simplification could in no way be performed in the region graph directly since it supresses states which can be reached by different paths than the MTOE root, even considering only the control flow.

3.3 Timed input element and time elapsing element

In section 3.2, we have already taken into account two out of the three possible roles that the δ symbol can assume : internal action and output symbol. The only role left is the input action one, which is actually the only one controllable and consequently testable by itself, and which will be treated the same way as initial input symbols.

This class of elements can, and must, be treated individually in the test generation, and their definition remains unchanged, although we will from now restrict the notion of time elapsing element to the occurences of the δ symbol in the minimal region graph where it assumes at least the function of an input symbol.

```
begin
do
  read clock valuation v
while (v ∉ z_e and !b is not received)
if !B is received
  read clock valuation v
  current state = (s_f, [v])
  continue
else
  current state = (s_i, z_e)
  continue
endif
end
```

Figure 5: Execution algorithm of a MTOE

4 Test execution

4.1 Test architecture

A specific architecture for timed testing has been proposed in [PF99]. In this architecture, the tester is divided in two parts : the clock part, which contains the clocks appearing in the specification (modeled as a timed input/output automaton), and the behaviour part, whose function is to communicate with the implementation through the PCOs, i.e. to send inputs to the IUT and receive outputs from it. These two parts communicate with one another in the following way : the control part can, at any moment, ask the clock part for the value of any clock and it receives immediately the answer.

This architecture keeps the implementation as a black box, but prevents the tester from detecting some faults at the exact place where they occur, mainly when clock resets are concerned.

4.2 Controllable case

4.2.1 Time elapsing element

This element is undoubtedly the easiest one to execute : it consists only in waiting for the clock values to evolve from some clock zone to another one. The corresponding algorithm is obvious.

4.2.2 Multiple output timed element

The execution algorithm translates the parallel execution of the tester and the MTOE. From state $(0, z)$, the tester is idle and waits for one of the two following events to occur : reception of the output symbol $!b$ or entrance in the clock zone z_e. These events concerns two different kinds of observation : the first one is received through the PCO associated with the symbol (or action) $!b$ while the second one is detected thanks to a repeated check of the values of clocks in the clock part of the tester. It is given in figure exete.

4.2.3 Timed input element

The division of the whole set of clocks in which the input symbol can be submitted is not only compulsory to keep coherent with the specification, it also provides a useful granularity for testing the infinite clock zones. Several methods have been proposed to treat and/or increase this granularity : test of a single random clock value in each zone, test of predeterminate values uniformally dispersed on a grid with a step equal to $1/k$ where $k \in \mathbf{N}$ [EnDKE98], test of the extreme reachable clock values in each visited zone (value of arrival in the state from wich the input symbol is to be submitted and furthest value in the same zone reachable by

single elapse of time) [FPS00]. Each of these methods lead to their own associated execution algorithm and lead to an individual test coverage (which is always, mathematically speaking, null, since we are in an infinite space).

4.3 Uncontrollable case

If the initial timed automaton modeling the system is already uncontrollable, we may have, from any state, several outgoing transitions executable simultaneously, one of which at least is labeled by an output symbol. In the process of test execution, any output transition thus coupled with a timed element in our test case must be taken into account, and its execution must lead to a final state associated with the inconclusive verdict. In fact, our considerations rejoin here with those already explored for the nondeterminist cases. And, as it is classically assumed, we consider that a sufficiently large number of executions will cover exhaustively all the possible paths in our test cases.

In order to reduce the number of these executions, some tactical considerations may be taken into account, which may prove also sound for the controllable case, when computing the preambles for our test cases. Since the additional uncontrollability in the translation in a region graph is exclusively due to the transitions labeled with output symbols, it would seem reasonable to choose preambles with as few TOE as possible, in order to achieve an optimal probability to actually reach the expected state at the expected time.

5 Derivation of timed test purposes

Apart from the methods consisting in testing the conformance of the implementation of a global system with its formal specification w.r.t. a specific conformance relation, a large amount of effort has been dedicated to the derivation of test sequences from test purposes. These latter works, focusing on some explicitly expressed characteristics of the system rather than seeing it as a whole, lead to a generally lesser cost and are therefore more welcome in the industrial world.

The generation of test sequences from test purposes is often performed through a synchronous product of the test purpose and the specification. Since the test purposes are but parts of the global specification, they do not cover all possible paths from any given state, and the execution of the associated test sequences lead to the three usual verdicts - pass, inconclusive and fail - depending on the result being conform to both test purpose and specification, only the specification, or none of them. Introducing in this problematics the time element, the authors of [SPF01] define a timed synchronous product and associate these verdicts not only to behavioral state but also to set of clock values computed from the test zones appearing in the minimal region graphs of the test purpose and of the specification.

The timed test purposes are classically defined as acyclic timed automata, whose states are also states of the timed automaton of the specification, but whose clocks are not necesarily the same. However, in order to remain coherent with the notion of conformance testing, all clock guards in the test purpose must be more restrictive than the corresponding ones in the specification.

If the assignation of verdicts to sets of clock values do not have to be modified when the considered transition is labeld by an input symbol or by δ as an input symbol, we have already observed in section 3 than every output symbol lead, in the test cases, to a tree modeling all possible moments for the execution of the transition. In this case, the sets pass, fail, and inconclusive must have their definition adapted. For a given initial state s_i and an output symbol $!b$, let $MTOE_s = (s_i, z_i, !b, s_f, z_s) \cup (s_i, z_i, \delta, s_i, z_{e_s})$ and $MTOE_p = (s_i, z_i, !b, s_f, z_p) \cup (s_i, z_i, \delta, s_i, z_{e_p})$ be the respective multiple timed output elements of the specification and of the timed test purpose. We define then the zones as follows :

- $Z_{PASS} = \bigcup z_p \cap \bigcup z_s$
- $Z_{INCONCL} = \bigcup z_s \setminus \bigcup z_p$
- $Z_{FAIL} = \mathbb{R}^n \setminus \bigcup z_s$

These definitions are sound since the clock guards in the test purpose are stronger than in the specification, and therefore $\bigcup z_p \subset \bigcup z_s$. We give in figure 6 a simple example of what these zones could be.

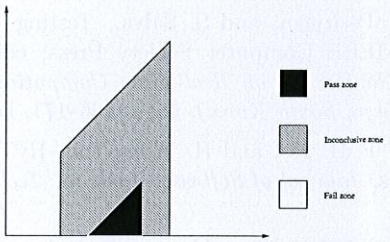


Figure 6: Zones and associated verdicts

6 Conclusion and perspectives

We have presented in this paper a theoretical basis for designing and executing timed test cases generated from a specification modeled as a region graph. The δ symbol, which is at the very center of our study, has been proven to assume different roles in this graph, namely input, output, and internal action. We have from this observation defined a new shape for test cases through the introduction of the multiple timed output event which takes into account not only one output transition in the region graphs, but all the possible paths leading from the same head state with the same output symbol, the δ symbol not being in this case considered as part of the control flow. We have then proposed an execution algorithm for this timed test element, along with an outline of a global execution algorithm when the timed automaton is uncontrollable. Finally, we have described the consequences of the ambiguous nature of δ on methods based on timed test purposes, especially when associating verdicts with sets of clock values comes into consideration.

As a future work, we intend to study the impact of this study on other test sequences generation methods, such as state characterization (is it still feasible ?) and methods based on the *ioco* and *conf* conformance relations. Whenever the adaptation is possible, we will try and develop new generation algorithms coherent with the test elements we have defined.

References

[ACH+92] R. Alur, C. Courcoubetis, N. Halbwachs, D. Dill, and H. Wong-Toi. Minimization of timed transition systems. In R. Cleaveland, editor, *Proceedings CONCUR 92,* Stony Brook, NY, USA, volume 630 of *Lecture Notes in Computer Science*, pages 340–354. Springer-Verlag, 1992.

[AD90] R. Alur and D. Dill. Automata for modeling real-time systems. In M. Paterson, editor, *Proceedings* 17^{th} *ICALP,* Warwick, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer-Verlag, July 1990.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[BDFP00] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Expressiveness of Updatable Timed Automata. *Lecture Notes in Computer Science*, 1893:232–242, August 2000.

[CG00] C. Choffrut and M. Goldwurm. Timed Automata with Periodic Clock Constraints. *Journal of Automata, Language and Combinatorics*, 5(4):371–404, 2000.

[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 208–219. Springer-Verlag, 1995.

[EnDKE98] A. En-nouaary, R. Dssouli, F. Khendek, and A. Elqortobi. Timed test case generation based on state characterization techniques. In *Proceedings of the 19th IEEE Real Time Systems Symposium, RTSS'98 (Madrid, Spain)*, 1998.

[FP01] H. Fouchal and E. Petitjean. Test Case Derivation for Timed Systems. In *Proceedings of the International Conference On Principles Of DIstrbuted Systems, OPODIS'01 (Manzanillo, Mexico)*, December 2001.

[FPS00] H. Fouchal, E. Petitjean, and S. Salva. Testing Timed Systems with Timed Purposes. In IEEE Computer Society Press, editor, *Proceedings of the 7th International Conference on Real-Time Computing Systems and Applications, RTCSA'00 (Cheju, South Korea)*, pages 166–171, December 2000.

[HHWT97] T.A. Henziger, P.-H. Ho, and H. Wong-Toi. HyTech : A Model Checker for Hybrid Systems. *Journal of Software Tools for Technology Transfer*, 1(1-2):110–112, 1997.

[JSD97] F.W. Vaandrager J. Springintveld and P.R. D'Argenio. Timed testing automata. Report CS-R9712, CWI, Amsterdam, August 1997.

[Kon94] O. Kone. *Interconnexion de systèmes ouverts : Test d'Interopérabilité, Test avec contraintes de Temps Physique.* PhD thesis, Univ. of Bordeaux I, 1994.

[Lau99] P. Laurençot. *Intégration du temps dans les tests de protocoles de communication.* PhD thesis, Univ. of Bordeaux I, January 1999.

[LPY97] K.G. Larsen, P. Petterson, and W. Yi. Uppaal in a nutshell. *Journal of Software Tools for Technology Transfer*, 1(1-2):134–152, 1997.

[PF99] E. Petitjean and H. Fouchal. A Realistic Architecture for Timed Testing. In *Proceedings of the 5th IEEE International Conference of Engineering of Complex Computer Systems, ICECCS'99 (Las Vegas, Nevada)*, pages 109–118. IEEE Computer Society, October 1999.

[SPF01] S. Salva, E. Petitjean, and H. Fouchal. A Simple Approach to Testing Timed Systems. In *Proceedings of the Workshop on Formal Approaches to Testing of Software, FATES'01 ((Aalborg, Denmark)*, pages 93–107, August 2001.

[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.

RAPPORTS INTERNES AU LRI - ANNEE 2003

N°	Nom	Titre	Nbre de pages	Date parution
1345	FLANDRIN E LI H WEI B	A SUFFICIENT CONDITION FOR PANCYCLABILITY OF GRAPHS	16 PAGES	01/2003
1346	BARTH D BERTHOME P LAFOREST C VIAL S	SOME EULERIAN PARAMETERS ABOUT PERFORMANCES OF A CONVERGENCE ROUTING IN A 2D-MESH NETWORK	30 PAGES	01/2003
1347	FLANDRIN E LI H MARCZYK A WOZNIAK M	A CHVATAL-ERDOS TYPE CONDITION FOR PANCYCLABILITY	12 PAGES	01/2003
1348	AMAR D FLANDRIN E GANCARZEWICZ G WOJDA A P	BIPARTITE GRAPHS WITH EVERY MATCHING IN A CYCLE	26 PAGES	01/2003
1349	FRAIGNIAUD P GAURON P	THE CONTENT-ADDRESSABLE NETWORK D2B	26 PAGES	01/2003
1350	FAIK T SACLE J F	SOME b-CONTINUOUS CLASSES OF GRAPH	14 PAGES	01/2003
1351	FAVARON O HENNING M A	TOTAL DOMINATION IN CLAW-FREE GRAPHS WITH MINIMUM DEGREE TWO	14 PAGES	01/2003
1352	HU Z LI H	WEAK CYCLE PARTITION INVOLVING DEGREE SUM CONDITIONS	14 PAGES	02/2003
1353	JOHNEN C TIXEUIL S	ROUTE PRESERVING STABILIZATION	28 PAGES	03/2003