Béatrice Bérard
Serge Haddad

Interrupt Timed Automata: A Step Further

Research Report LSV-09-01

January 2009

Laboratoire Spécification et Vérification

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Interrupt Timed Automata: A Step Further *

B. Bérard[1]**, S. Haddad[2]**

[1] Université Pierre & Marie Curie, LIP6/MoVe, CNRS UMR 7606, Paris, France
E-mail: Beatrice.Berard@lip6.fr

[2] Ecole Normale Supérieure de Cachan, LSV, CNRS UMR 8643, Cachan, France
E-mail: Serge.Haddad@lsv.ens-cachan.fr

Abstract. In this work, we introduce the class of Interrupt Timed Automata (ITA), which are well suited to the description of multi-task systems with interruptions in a single processor environment. This model is a subclass of hybrid automata. While reachability is undecidable for hybrid automata we show that in ITA the reachability problem is decidable. More precisely we show that the untimed language of an ITA is effectively regular via a generalized class graph. We establish that the reachability problem for ITA is in 2-NEXPTIME and in NP when the number of clocks is fixed. To prove the results on reachability, we first consider a subclass ITA$_-$ which still describes usual interrupt systems and for which the reachability problem is in NEXPTIME and in NP when the number of clocks is fixed (without any class graph). Then we prove that any ITA can be reduced to an equivalent ITA$_-$ w.r.t. the language equivalence. Additionally we show that the languages of ITA are neither closed under complementation nor under intersection. We also compare the expressive power of TA and ITA and prove that the corresponding families of languages are incomparable. The result also holds for languages accepted by controlled real-time automata (CRTA), another extension of timed automata. So, we finally combine ITA with CRTA in a model which encompasses both classes and show that the reachability problem is still decidable.

Keywords: hybrid automata, timed automata, multi-task systems, interruptions, decidability of reachability

1 Introduction

Context. The model of timed automata (TA), introduced in [1], has proved very successful due to the decidability of the emptiness test. A timed automaton consists of a finite automaton equipped with real valued variables, called clocks, which evolve synchronously with time, during the sojourn in states. When a

* This report extends the paper entitled "Interrupt Timed Automata" to appear in FOSSACS'09. Main new results are the extension of the model with policies (and the associated algorithms), propositions 5 and 6, incomparability of TL and ITL and non closure of ITL under complementation and intersection.

** Work partly supported by project DOTS (ANR-06-SETI-003)

discrete transition occurs, clocks can be tested by guards, which compare their values with constants, and reset. The decidability result was obtained through the construction of a finite partition of the state space into regions, leading to a finite graph which is time-abstract bisimilar to the original transition system, thus preserving reachability.

Hybrid automata have subsequently been proposed as an extension of timed automata [15], with the aim to increase the expressive power of the model. In this model, clocks are replaced by variables which evolve according to a differential equation. Furthermore, guards consist of more general constraints on the variables and resets are extended into (possibly non deterministic) updates. However, since reachability is undecidable for this model, many classes have been defined, between timed and hybrid automata, to obtain the decidability of this problem. Examples of such classes are multi-rate or rectangular automata [2], some systems with piece-wise constant derivatives [3], controlled real-time automata [10], integration graphs [12], o-minimal hybrid systems [13, 14], some updatable timed automata [6] or polygonal hybrid systems [4].

Contribution. In this paper, we define a subclass of hybrid automata, called Interrupt Timed Automata (ITA), well suited to the description of multi-task systems with interruptions in a single processor environement. In an ITA, the finite set of control states is organized according to *interrupt levels*, ranging from 1 to n, with exactly one active clock for a given level. The clocks from lower levels are suspended and those from higher levels are not yet defined. On the transitions, guards are linear constraints using only clocks from the current level or the levels below and the relevant clocks can be updated by linear expressions, using clocks from lower levels. For a transition increasing the level, the newly relevant clocks are reset. At last, every state has state has a policy (lazy, urgent or delayed) that rules the sojourn time. This model is rather expressive since it combines variables with rate 1 or 0 (usually called stopwatches) and linear expressions for guards or updates.

While the reachability problem is well known to be undecidable for automata with stopwatches [11, 8, 7], we prove that it is decidable for ITA.

More precisely, we first show that the untimed language of an ITA is effectively regular. The corresponding procedure significantly extends the classical region construction of [1] by associating with each state a family of orderings over linear expressions. One deduces from this construction a decision algorithm for reachability in 2-EXPSPACE.

In order to investigate the reachability problem, we define a slight restriction of the model, leading to a subclass ITA$_-$ for which reachability can be decided in NEXPTIME. Furthermore when the number of clocks is fixed, the complexity is (greatly) reduced to NP. Then we prove that for any ITA one can build an equivalent ITA$_-$ w.r.t. language equivalence whose size is at most doubly exponential w.r.t. the size of the ITA and polynomial when the number of clocks are fixed. Combining this reduction with the decision procedure for reachability of ITA$_-$ leads to a an algorithm for reachability of ITA in 2-NEXPTIME and

in NP when the number of clocks is fixed. This should be compared to TA with 3 clocks for which reachability is PSPACE complete [9].

We also study the expressive power of the class ITA, in comparison with the original model of timed automata and also with the more general controlled real-time automata (CRTA) proposed in [10]. In CRTA, clocks and states are coloured and a rate is associated with every state. During the visit of a state, all clocks coloured by the colour of the state evolve with the state rate while the others do not evolve. We prove that the corresponding families of languages ITL and TL, as well as ITL and CRTL, are incomparable. Additionally we show that ITL is neither closed under complementation nor under intersection.

So we finally define a combination of the two models (ITA and CRTA), the CRTA part describing a basic task at an implicit additional level 0. For this extended model denoted by ITA^+ (with ITA^+_- as a subclass), we show that reachability is still decidable with the same complexity and which is in PSPACE when the number of clocks is fixed.

Outline. Interrupt Timed Automata are defined in section 2 and section 3 is devoted to the proof of effective regularity for untimed languages from ITA. In section 4, we study the decidability and the complexity of the reachability problem. Section 5 investigates the expressive power of ITA and section 6 extends the decidability results for a model combining ITA and CRTA.

2 Interrupt Timed Automata

The sets of natural numbers, rational numbers and real numbers are denoted respectively by $\mathbb{N}$, $\mathbb{Q}$ and $\mathbb{R}$, with $\mathbb{Q}_{\geq 0}$ (resp. $\mathbb{R}_{\geq 0}$) for the set of non negative rational (resp. real) numbers.

Let X be a set of clocks. A linear expression over X is a term of the form $\sum_{x\in X} a_x x + b$ where b and the a_xs are in $\mathbb{Q}$. We denote by $\mathcal{C}^+(X)$ the set of constraints obtained by conjunctions of atomic propositions of the form $C \bowtie 0$, where C is a linear expression and $\bowtie$ is in $\{<, \leq, \geq, >\}$. The subset of $\mathcal{C}^+(X)$ where linear expressions are restricted to the form $x + b$, for $x \in X$ and $b \in \mathbb{Q}$ is denoted by $\mathcal{C}(X)$. An update over X is a conjunction of the form $\bigwedge_{x\in X} x := C_x$ where C_x is a linear expression. We denote by $\mathcal{U}^+(X)$ the set of updates over X and by $\mathcal{U}(X)$ the subset of $\mathcal{U}^+(X)$ where for each clock x, the linear expression C_x is either x (value unchanged) or 0 (clock reset).

A clock valuation is a mapping $v : X \mapsto \mathbb{R}$ and we denote by $\mathbf{0}$ the valuation assigning the value 0 to all clocks. The set of all clock valuations is $\mathbb{R}^X$ and we write $v \models \varphi$ when valuation v satisfies the clock constraint φ. For an element d of $\mathbb{R}_{\geq 0}$, the valuation $v+d$ is defined by $(v+d)(x) = v(x)+d$, for each clock x in X. For a linear expression $C = \sum_{x\in X} a_x x + b$, the real number $v[C]$ is defined by $\sum_{x\in X} a_x v(x) + b$. For an update u defined by $\bigwedge_{x\in X} x := C_x$, the valuation $v[u]$ is defined by $v[u](x) = v[C_x]$ for x in X. The linear expression $C[u]$ is obtained by substituting in C every x by C_x.

The model of ITA is based on the principle of multi-task systems with interruptions, in a single processor environment. We consider a set of tasks with

different priority levels, where a higher level task represents an interruption for a lower level task. At a given level, exactly one clock is active with rate 1, while the clocks for tasks of lower levels are suspended, and the clocks for tasks of higher levels are not yet activated. Moreover every state has a time policy which indicates whether time may or must elapse in a state.

Definition 1 (Interrupt Timed Automaton). *An* interrupt timed automaton *is a tuple* $\mathcal{A} = (\Sigma, Q, q_0, pol, F, X, \lambda, \Delta)$*, where:*

- Σ *is a finite alphabet,*
- Q *is a finite set of states,* q_0 *is the initial state,* $F \subseteq Q$ *is the set of final states,*
- $pol : Q \mapsto \{L, U, D\}$ *is the timing policy of states where* L *stands for* lazy*,* U *for* urgent *and* D *for* delayed*,*
- $X = \{x_1, \ldots, x_n\}$ *consists of* n *interrupt clocks,*
- *the mapping* $\lambda : Q \mapsto \{1, \ldots, n\}$ *associates with each state its level*
- *and* $\Delta \subseteq Q \times [\mathcal{C}^+(X) \times (\Sigma \cup \{\varepsilon\}) \times \mathcal{U}^+(X)] \times Q$ *is the set of transitions. We call* $x_{\lambda(q)}$ *the active clock in state* q*. Let* $q \xrightarrow{\varphi,a,u} q'$ *in* Δ *be a transition with* $k = \lambda(q)$ *and* $k' = \lambda(q')$*. The guard* φ *contains only clocks from levels less than or equal to* k*: it is a conjunction of constraints of the form* $\sum_{j=1}^{k} a_j x_j + b \bowtie 0$*. The update* u *is of the form* $\wedge_{i=1}^{n} x_i := C_i$ *with:*
  - *if* $k' < k$*, i.e. the transition decreases the level, then* C_i *is of the form* $\sum_{j=1}^{i-1} a_j x_j + b$ *or* $C_i = x_i$ *for* $1 \leq i \leq k'$ *and* $C_i = x_i$ *otherwise;*
  - *if* $k' \geq k$ *then* C_i *is of the form* $\sum_{j=1}^{i-1} a_j x_j + b$ *or* $C_i = x_i$ *for* $1 \leq i \leq k$*,* $C_i = 0$ *if* $k < i \leq k'$ *and* $C_i = x_i$ *if* $i > k'$*.*

Thus, clocks from levels higher than the target state are ignored, and when newly relevant clocks appear upon increasing the level, they are reset. By default, the policy of a state is the lazy policy. Note that in a classical timed automaton, these policies can be enforced with clock constraints.

Definition 2 (Semantics of an ITA). *The semantics of an ITA* $\mathcal{A}$ *is defined by the transition system* $\mathcal{T}_\mathcal{A} = (S, s_0, \rightarrow)$*. The set* S *of configurations is* $\{(q, v) \mid q \in Q,\ v \in \mathbb{R}^X\}$*, with initial configuration* $(q_0, \mathbf{0})$*. An accepting configuration of* $\mathcal{T}_\mathcal{A}$ *is a pair* (q, v) *with* q *in* F*. The relation* $\rightarrow$ *on* S *consists of two types of steps:*

Time steps: *Only the active clock in a state can evolve, all other clocks are suspended. For a state* q *with active clock* $x_{\lambda(q)}$*, a time step of duration* d *is defined by* $(q, v) \xrightarrow{d} (q, v')$ *with* $v'(x_{\lambda(q)}) = v(x_{\lambda(q)}) + d$ *and* $v'(x) = v(x)$ *for any other clock* x*. When* $pol(q) = U$*, time steps from* q *are forbidden.*

Discrete steps: *A discrete step* $(q, v) \xrightarrow{a} (q', v')$ *occurs if there exists a transition* $q \xrightarrow{\varphi,a,u} q'$ *in* Δ *such that* $v \models \varphi$ *and* $v' = v[u]$*. When* $pol(q) = D$ *every discrete step from* q *must be immediately preceded by a non null time step.*

Remarks. Observe that in state q the only relevant clocks are $\{x_k\}_{k\leq\lambda(q)}$ since any other clock will be reset before being tested for the first time in the future. We have not stated this feature more explicitely in the definition for the sake of simplicity.

Concerning updates, if we allow a slight generalization, substituting $x_i := \sum_{j=1}^{i-1} a_j x_j + b$ by $x_i := \sum_{j=1}^{i} a_j x_j + b$, it is easy to simulate a two-counter machine with a three clocks-ITA, thus implying undecidability of reachability for the model.

A timed word is a finite sequence $(a_1, t_1)\dots(a_p, t_p) \in (\Sigma \times \mathbb{R}_{\geq 0})^*$, where the t_i's form a non decreasing sequence. A timed language is a set of timed words. For a timed language L, the corresponding untimed language, written $Untime(L)$, is the projection of L on Σ^*. For an ITA $\mathcal{A}$, a run is a path in $\mathcal{T}_\mathcal{A}$ from the initial to an accepting configuration such that time steps alternate with discrete steps: $(q_0, v_0) \xrightarrow{d_1} (q_0, v_0') \xrightarrow{a_1} (q_1, v_1) \cdots \xrightarrow{d_n} (q_{n-1}, v_{n-1}') \xrightarrow{a_n} (q_n, v_n)$, with $v_0 = \mathbf{0}$. The sequence $t_1, \dots, t_n$ of absolute dates associated with this run is $t_i = \sum_{j=1}^{i} d_j$ and a timed word accepted by $\mathcal{A}$ is obtained by removing from the sequence $(a_1, t_1)\dots(a_n, t_n)$ the pairs such that $a_i = \varepsilon$. We denote by $\mathcal{L}(\mathcal{A})$ the set of timed words accepted by $\mathcal{A}$. ITL denotes the family of timed languages accepted by an ITA.

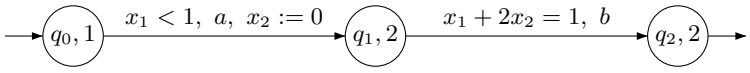

Fig. 1. An ITA $\mathcal{A}_1$ with two interrupt levels

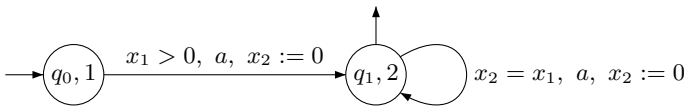

Fig. 2. An ITA $\mathcal{A}_2$ for L_2

We end this paragraph with two examples of ITA (used later in section 5). In the figures, the level of a state is indicated beside its name and all states are lazy. For the automaton $\mathcal{A}_1$ in Fig. 1, state q_0 is the initial state with level 1. States q_1 and q_2 are on level 2, and q_2 is the final state. There are two interrupt clocks x_1 and x_2. Entering state q_1 at time $1 - \tau$ for some τ, clock x_1 is suspended and state q_2 is reached at time $1 - \tau + t$ with $1 - \tau + 2t = 1$. The language accepted by $\mathcal{A}_1$ is thus $L_1 = \{(a, 1-\tau)(b, 1-\tau/2) \mid 0 < \tau \leq 1\}$. The ITA in Fig. 2 also has two levels and two interrupt clocks x_1 and x_2. It accepts $L_2 = \{(a, \tau)(a, 2\tau)\dots(a, n\tau) \mid n \in \mathbb{N}, \tau > 0\}$.

3 Untiming ITL

3.1 Construction

Similarly to TA (and to CRTA), the proof of the effective regularity of the langugage of an ITA $\mathcal{A}$ is based on the construction of a (finite) class graph which is time abstract bisimilar to the transition system $\mathcal{T}_{\mathcal{A}}$. However the construction of classes is much more involved than in the case of TA. More precisely, it depends on the expressions occurring in the guards and updates of the automaton (while in TA it depends only on the maximal constant occurring in the guards). We associate with each state q a set of expressions $Exp(q)$ with the following meaning. The values of clocks giving the same ordering of these expressions correspond to a class. In order to define $Exp(q)$, we first build a family of sets $\{E_i\}_{1\leq i\leq n}$. Then $Exp(q) = \bigcup_{i\leq\lambda(q)} E_i$. Finally in proposition 2 we show how to build the class graph which proves the regularity of languages and additionally decides the reachability problem.

We first introduce an operation, called *normalization*, on expressions relative to some level. As explained in the construction below, this operation will be used to order the respective values of expressions at a given level.

Definition 3 (Normalization). *Let $C = \sum_{i\leq k} a_i x_i + b$ be an expression over $X_k = \{x_j \mid j \leq k\}$, the k-*normalization *of C, $\mathtt{norm}(C,k)$, is defined by:*

- *if $a_k \neq 0$ then $\mathtt{norm}(C,k) = x_k + (1/a_k)(\sum_{i<k} a_i x_i + b)$;*
- *else $\mathtt{norm}(C,k) = C$.*

Since guards are linear expressions with rational constants, we can assume that in a guard $C \bowtie 0$ occurring in a transition outgoing from a state q with level k, the expression C is either $x_k + \sum_{i<k} a_i x_i + b$ (by k-normalizing the expression and if necessary changing the comparison operator) or $\sum_{i<k} a_i x_i + b$.

Construction of $\{E_k\}_{k\leq n}$. The construction proceeds top down from level n to level 1 after initializing $E_k = \{x_k, 0\}$ for all k. As we shall see below, when handling the level k, we add new terms to $\{E_i\}_{1\leq i\leq k}$.

- At level k, first for every expression $\alpha x_k + \sum_{i<k} a_i x_i + b$ (with $\alpha \in \{0,1\}$) occurring in a guard of an edge leaving a state of level k, we add $-\sum_{i<k} a_i x_i - b$ to E_k.
- Then we iterate the following procedure until no new term is added to any E_i for $1 \leq i \leq k$.
  1. Let $q \xrightarrow{\varphi,a,u} q'$ with $\lambda(q') \geq k$ and $\lambda(q) \geq k$. Let $C \in E_k$, then we add $C[u]$ to E_k.
  2. Let $q \xrightarrow{\varphi,a,u} q'$ with $\lambda(q') \geq k$ and $\lambda(q) < k$. Let $C, C' \in E_k$, then we compute $C'' = norm(C[u] - C'[u], \lambda(q))$. Let us write C'' as $\alpha x_{\lambda(q)} + \sum_{i<\lambda(q)} a_i x_i + b$ with $\alpha \in \{0,1\}$. Then we add $-\sum_{i<\lambda(q)} a_i x_i - b$ to $E_{\lambda(q)}$.

Proposition 1. *The construction procedure of* $\{E_k\}_{k\leq n}$ *terminates and the size of every* E_k *is bounded by* $B^{2^{n(n-k+1)}+1}$ *where* B *is the maximum between* 2 *and the number of edges of the ITA.*

Proof. Given some k, we prove the termination of the stage relative to k. Observe that the second step only adds new expressions to $E_{k'}$ for $k' < k$. Thus the two steps can be ordered. Let us prove the termination of the first step of the saturation procedure. We denote $E_k^0 \equiv E_k$ at the beginning of this stage and $E_k^i \equiv E_k$ after the insertion of the i^{th} item in it. With each added item $C[u]$ can be associated its *father* C. Thus we can view E_k as an increasing forest with finite degree (due to the finitess of the edges). Assume that this step does not terminate. Then we have an infinite forest and by König lemma, it has an infinite branch $C_0, C_1, \ldots$ where $C_{i+1} = C_i[u_i]$ for some update u_i such that $C_{i+1} \neq C_i$. Observe that the number of updates that change the variable x_k is either 0 or 1 since once x_k disappears it cannot appear again. We split the branch into two parts before and after this update or we still consider the whole branch if there is no such update. In these (sub)branches, we conclude with the same reasonning that there is at most one update that change the variable x_{k-1}. Iterating this process, we conclude that the number of updates is at most $2^k - 1$ and the length of the branch is at most 2^k. Thus the final size of E_k is at most $E_k^0 \times B^{2^k}$ since the width of the forest is bounded by B.

In the second step, we add at most $B \times (|E_k| \times (|E_k| - 1))/2$ to E_i for every $i < k$. This concludes the proof of termination.

We now prove by a painful backward induction that as soon as $n \geq 2$, $|E_k| \leq B^{2^{n(n-k+1)}+1}$. We define $p_k \equiv |E_k|$.

Basis case $k = n$
$p_n \leq p_n^0 \times B^{2^n}$ where p_n^0 is the number of guards of the outgoing edges from states of level n. Thus:
$p_n \leq B \times B^{2^n} = B^{2^n+1} = B^{2^{n(n-n+1)}+1}$
which is the claimed bound.

Inductive case
Assume that the bound holds for $k < j \leq n$. Due to the second step of the procedure, we have:
$p_k^0 \leq B + B \times ((p_{k+1} \times (p_{k+1} - 1))/2 + \cdots + (p_n \times (p_n - 1))/2)$
$p_k^0 \leq B + B \times (B^{2^{n(n-k)+1}+2} + \cdots + B^{2^{n+1}+2})$
$p_k^0 \leq B \times (n - k + 1) \times B^{2^{n(n-k)+1}+2}$
$p_k^0 \leq B \times B^n \times B^{2^{n(n-k)+1}+2}$ *(here we use* $B \geq 2$*)*
$p_k^0 \leq B^{2^{n(n-k)+1}+n+3}$
$p_k \leq B^{2^{n(n-k)+1}+2^k+n+3}$

Let us consider the term $\delta = 2^{n(n-k+1)} + 1 - 2^{n(n-k)+1} - 2^k - n - 3$
$\delta \geq (2^{n-1} - 1)2^{n(n-k)+1} - (2^k + n + 2)$
$\delta \geq (2^{n-1} - 1)2^{n(n-k)+1} - (2^{n-1} + 2^n)$
$\delta \geq (2^{n-1} - 1)2^{n(n-k)+1} - 2^{n+1} \geq 0$

Thus: $p_k \leq B^{2^{n(n-k)+1}+2^k+n+3} \leq B^{2^{n(n-k+1)}+1}$

which is the claimed bound.

Proposition 2. *The untimed language of an ITA is effectively regular.*

Proof.

First, we assume that the policy of every state is lazy. At the end of the proof, we explain how to adapt the construction for states with urgent or delayed policies.

Class definition. Let $\mathcal{A}$ be an ITA, the decision algorithm is based on the construction of a (finite) class graph which is time abstract bisimilar to the transition system $\mathcal{T}_{\mathcal{A}}$. A class is a syntactical representation of a subset of reachable configurations. More precisely, it is defined as a pair $R = (q, \{\preceq_k\}_{1 \leq k \leq \lambda(q)})$ where q is a state and $\preceq_k$ is a total preorder over E_k.

The class R describes the set of valuations:

$$[\![R]\!] = \{(q, v) \mid \forall k \leq \lambda(q)\ \forall (g, h) \in E_k,\ g[v] \leq h[v] \text{ iff } g \preceq_k h\}$$

Observe that the number of classes is bounded by:

$$|Q| \cdot 3^{B^{2^{(n^2)}+1}}$$

where n is the number of clocks of $\mathcal{A}$ and B is defined in proposition 1.

As usual, there are two kinds of transitions in the graph, corresponding to discrete steps and time steps.

Discrete step. Let $R = (q, \{\preceq_k\}_{1 \leq k \leq \lambda(q)})$ and $R' = (q', \{\preceq'_k\}_{1 \leq k \leq \lambda(q')})$ be two classes. There is a transition $R \xrightarrow{e} R'$ for a transition $e : q \xrightarrow{\varphi, a, u} q'$ if there is some $(q, v) \in [\![R]\!]$ and $(q', v') \in [\![R']\!]$ such that $(q, v) \xrightarrow{e} (q', v')$. In this case, for all $(q, v) \in [\![R]\!]$ there is a $(q', v') \in [\![R']\!]$ such that $(q, v) \xrightarrow{e} (q', v')$. This can be decided as follows.

Firability condition. Write $\varphi = \bigwedge_{1 \leq j \leq J'} C_j \leq 0 \wedge \bigwedge_{J'+1 \leq j \leq J} \neg(C_j \leq 0)$. By definition of an ITA, for every j, $C_j = \alpha x_{\lambda(q)} + \sum_{i < \lambda(q)} a_i x_i + b$ (with $\alpha \in \{0, 1\}$). By construction $C'_j = -\sum_{i < \lambda(q)} a_i x_i - b \in E_{\lambda(q)}$. If $j \leq J'$ then we require that $\alpha x_{\lambda(q)} \preceq_k C'_j$. If $j > J'$ then we require that $\neg(\alpha x_{\lambda(q)} \preceq_k C'_j)$.

Successor definition. R' is defined as follows. Let $k \leq \lambda(q')$ and $g', h' \in E_k$.

1. Either $k \leq \lambda(q)$, by construction, $g'[u], h'[u] \in E_k$ then $g' \preceq'_k h'$ iff $g'[u] \preceq_k h'[u]$.
2. Or $k > \lambda(q)$, let $D = g'[u] - h'[u] = \sum_{i \leq \lambda(q)} c_i x_i + d$, and $C = norm(D, \lambda(q))$, and write $C = \alpha x_{\lambda(q)} + \sum_{i < \lambda(q)} a_i x_i + b$ (with $\alpha \in \{0, 1\}$). By construction $C' = -\sum_{i < \lambda(q)} a_i x_i - b \in E_{\lambda(q)}$.
   When $c_{\lambda(q)} \geq 0$ then $g' \preceq'_k h'$ iff $C' \preceq_{\lambda(q)} \alpha x_{\lambda(q)}$.
   When $c_{\lambda(q)} < 0$ then $g' \preceq'_k h'$ iff $\alpha x_{\lambda(q)} \preceq_{\lambda(q)} C'$.

By definition of $[\![\ ]\!]$,

- $\forall (q, v) \in [\![R]\!]$, if there exists $(q, v) \xrightarrow{e} (q', v')$ then the firability condition is fulfilled and (q', v') belongs to $[\![R']\!]$.
- If the firability condition is fulfilled then $\forall (q, v) \in [\![R]\!]$ there exists $(q', v') \in [\![R']\!]$ such that $(q, v) \xrightarrow{e} (q', v')$.

Time step. Let $R = (q, \{\preceq_k\}_{1 \leq k \leq \lambda(q)})$.
There is a transition $R \xrightarrow{succ} Post(R)$ for $Post(R) = (q, \{\preceq'_k\}_{1 \leq k \leq \lambda(q)})$, the time successor of R, which is defined as follows.

For every $k' < \lambda(q)$ $\preceq'_k = \preceq_k$. Let $\sim = \preceq_{\lambda(q)} \cap \preceq^{-1}_{\lambda(q)}$ be the equivalence relation induced by the preorder. On equivalence classes, this (total) preorder becomes a (total) order. Let V be the equivalence class containing $x_{\lambda(q)}$.

1. Either $V = \{x_{\lambda(q)}\}$ and it is the greatest equivalence class. Then $\preceq'_{\lambda(q)} = \preceq_{\lambda(q)}$ (thus $Post(R) = R$).
2. Either $V = \{x_{\lambda(q)}\}$ and it is not the greatest equivalence class. Let V' be the next equivalence class. Then $\preceq'_{\lambda(q)}$ is obtained by merging V and V', and preserving $\preceq_{\lambda(q)}$ elsewhere.
3. Either V is not a singleton. Then we split V into $V \setminus \{x_{\lambda(q)}\}$ and $\{x_{\lambda(q)}\}$ and "extend" $\preceq_{\lambda(q)}$ by $V \setminus \{x_{\lambda(q)}\} \preceq'_{\lambda(q)} \{x_{\lambda(q)}\}$.

By definition of $[\![\,]\!]$, $\forall (q, v) \in [\![R]\!]$, there exists $d > 0$ such that $(q, v+d) \in Post(R)$ and $\forall 0 \leq d' \leq d,\ (q, v + d') \in R \cup Post(R)$.

The initial state of this graph is defined by the class R_0 with $[\![R_0]\!]$ containing $(q_0, \mathbf{0})$ which can be straightforwardly determined. The final states are all $R = (q, \{\preceq_k\}_{1 \leq k \leq \lambda(q)})$ with $q \in F$.

Given a state q such that $pol(q) = U$, for every class $R = (q, \{\preceq_k\}_{1 \leq k \leq \lambda(q)})$ we delete the time steps outgoing from R. The case of a state q such that $pol(q) = D$, is a little bit more involved. First we partition classes between *time open* classes, where for every every configuration of the class there exists a small amount of time elapse that let the new configuration in the same class, and *time closed* classes. The partition is performed w.r.t. the equivalence class V of $x_{\lambda(q)}$ for the relation $\sim$ (see above in the proof). The class R is time open iff $V = \{x_{\lambda(q)}\}$. Then we replace every time closed class R by two copies R^- and R^+. A time edge entering R is redirected towards R^+ while a discrete edge entering R is redirected towards R^-. A time step $R \xrightarrow{succ} R'$ is replaced by two transitions $R^- \xrightarrow{succ} R'$ and $R^+ \xrightarrow{succ} R'$, while a discrete step $R \xrightarrow{e} R'$ is replaced by the transition $R^+ \xrightarrow{e} R'$.

Since there is at most one time edge outgoing from a class, the number of edges of the new graph is at most twice the number of edges in the original graph.

Proposition 3. *The reachability problem for ITA is decidable and belongs to* 2-EXPSPACE *and* PSPACE *when the number of clocks is fixed.*

Proof. The reachability problem is then solved by a non deterministic search of a path in this graph (without building it) leading to a 2-EXPSPACE complexity. When the number of clocks is fixed the length of this path is at most exponential w.r.t. the size of the problem leading to a PSPACE procedure.

In the next section, we improve both complexity bounds of the above proposition

3.2 Example

We illustrate this construction of a class automaton for the automaton $\mathcal{A}_1$ from section 2 (see figure 3, where dashed lines indicate time steps).

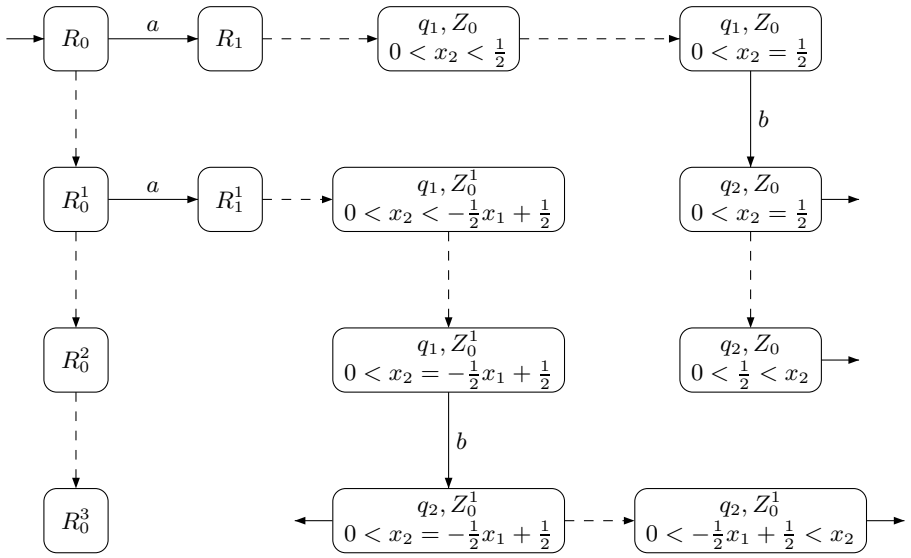

Fig. 3. The class automaton for $\mathcal{A}_1$

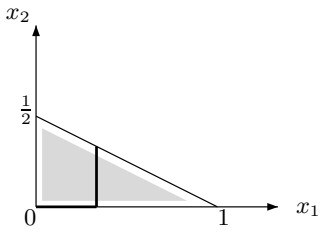

Fig. 4. A possible trajectory in $\mathcal{A}_1$

In this case, we obtain $E_1 = \{x_1, 0, 1\}$ and $E_2 = \{x_2, 0, -\frac{1}{2}x_1 + \frac{1}{2}\}$. In state q_0, the only relevant clock is x_1 and the initial class is $R_0 = (q_0, Z_0)$ with $Z_0 : x_1 = 0 < 1$. Its time successor is $R_0^1 = (q_0, Z_0^1)$ with $Z_0^1 : 0 < x_1 < 1$. Transition a leading to q_1 can be taken from both classes, but not from the next time successors $R_0^2 = (q_0, 0 < x_1 = 1)$ and $R_0^3 = (q_0, 0 < 1 < x_1)$.

Transition a switches from R_0 to $R_1 = (q_1, Z_0, x_2 = 0 < \frac{1}{2})$, because $x_1 = 0$, and from R_0^1 to $R_1^1 = (q_1, Z_0^1, x_2 = 0 < -\frac{1}{2}x_1 + \frac{1}{2})$. Transition b is fired from those time successors for which $x_2 = -\frac{1}{2}x_1 + \frac{1}{2}$.

A geometric view is given in figure 4, with a possible trajectory: first the value of x_1 increases from 0 in state q_0 (horizontal line) and, after transition a occurs, its value is frozen in state q_1 while x_2 increases (vertical line) until reaching the line $x_2 = -\frac{1}{2}x_1 + \frac{1}{2}$. The light gray zone is $(0 < x_1 < 1,\ 0 < x_2 < -\frac{1}{2}x_1 + \frac{1}{2})$, associated with q_1.

4 Complexity of the reachability problem

4.1 A simpler model

In practice, the clock associated with some level measures the time spent in this level or more generally the time spent by some tasks at this level. Thus when going to a higher level, this clock is "frozen" until returning to this level. The following restriction of the ITA model takes this feature into account.

Definition 4. *The subclass ITA$_-$ of ITA is defined by the following restriction on updates. For a transition $q \xrightarrow{\varphi,a,u} q'$ of an automaton $\mathcal{A}$ in ITA$_-$ (with $k = \lambda(q)$ and $k' = \lambda(q')$), the update u is of the form $\wedge_{i=1}^{n} x_i := C_i$ with:*

- *if $k' < k$, $u = \wedge_{i=1}^{n} x_i := x_i$ i.e. no updates;*
- *if $k' \geq k$ then C_k is of the form $\sum_{j=1}^{k-1} a_j x_j + b$ or $C_k = x_k$, $C_i = 0$ if $k < i \leq k'$ and $C_i = x_i$ otherwise.*

Observe that the automata of figures 1 and 2 belong to ITA$_-$. So the expressiveness results of proposition 8 still hold for ITA$_-$.

It turns out that the reachability problem for ITA$_-$ can be solved more efficiently.

In the sequel, the level of a transition is the level of its source state. We also say that a transition is lazy (resp. urgent, delayed) if the policy of its source state is lazy (resp. urgent, delayed).

Proposition 4. *The reachability problem for ITA$_-$ belongs to* NEXPTIME *and to* NP *when the number of clocks is fixed.*

Proof. Let $\mathcal{A} = (\Sigma, Q, q_0, F, pol, X, \lambda, \Delta)$ be an ITA$_-$. Let $E = |\Delta|$ be the number of transitions and for a given run let m_k be the number of transitions of level k.

Assume that there is a run ρ from (q_0, v_0) to some configuration (q_f, v_f). We build a run ρ' from (q_0, v_0) to (q_f, v_f) which fulfills:

- $m'_1 \leq (E+2)^3$
- $\forall k\ m'_{k+1} \leq (E+2)^3(m'_k + 1)$

Thus $\sum_{k=1}^{n} m'_k = O(E^{3n})$.

We iteratively modify the run ρ by considering the transitions of level k from 1 to n. For the basis case $k = 1$, we consider in the run ρ the subsequence $(e_1, \cdots, e_p)$ of **transitions in Δ of level 1 which update** x_1. Observe that if $e_i = e_j$ for some $i < j$, we can remove the subrun between these two transitions, because x_1 is the only relevant clock before the firing of e_i (or e_j). Thus we obtain a run with at most E such transitions.

Now we consider a subsequence $(e'_1, \cdots, e'_r)$ of **lazy or delayed transitions of level 1 which let unchanged** x_1 occurring between two of the previous transitions (or before the first or after the last). Observe that if $e'_i = e'_j$ for some $i < j$, we can replace the subrun between these two transitions by a time step corresponding to the difference of values of x_1. Indeed, since there is no update, the clock value after the second transition is greater than or equal to the value after the first transition. Furthermore the time step is possible since the source state has a lazy or delayed policy.

At last consider a subsequence $(e''_1, \cdots, e''_r)$ of **urgent transitions which let unchanged** x_1 occurring between two of the previous transitions (or before the first or after the last). Observe that if $e''_i = e''_j$ for some $i < j$, we can remove the subrun between these two transitions, because time has not elapsed at level 1 so the value of x_1 is unchanged.

Thus we obtain a run with at most $E+E(E+1)+E((E+1)^2+1) \leq (E+2)^3$, transitions of level 1.

Assume that the bound holds at levels less than $k+1$ and consider the subrun between two consecutive transitions of level less than $k+1$ (or before the first or after the last). By definition of ITA$_-$, the values of clocks $x_1, \ldots, x_k$ are unchanged during this subrun. Thus for two occurrences of the same transition of level $k+1$, the update of x_{k+1} is the same. So we can apply the same reasoning as for the basis case, thus leading to the claimed bound.

The decision procedure is as follows. It non deterministically guesses a path in the ITA$_-$ whose length is less than or equal to the bound. In order to check that this path yields a run, it builds a linear program whose variables are $\{x_i^j\}$, where x_i^j is the value of clock x_i after the jth step, and $\{d_j\}$ where d_j is the amount of time elapsed during the jth step, when j corresponds to a time step. The equations and inequations are deduced from the guards and updates of discrete transitions in the path and the delay of the time steps. The size of this linear program is exponential w.r.t. the size of the ITA$_-$. As a linear program can be solved in polynomial time [16], we obtain a procedure in NEXPTIME. If the number of clocks is fixed the number of variables is now polynomial w.r.t. the size of the problem.

4.2 From ITA to ITA$_-$

In this subsection we prove that ITA and ITA$_-$ are equivalent w.r.t. the associated languages and we obtain a better upper bound for the reachability problem of ITA.

Proposition 5. *Given an ITA $\mathcal{A}$, we build an automaton $\mathcal{A}'$ in ITA$_-$ accepting the same language and with the same clocks such that its number of edges (resp. states) is doubly exponential w.r.t. the number of edges (resp. states) in $\mathcal{A}$ and polynomial when the number of clocks is fixed.*

Proof. First we associate with every pair of levels $i \geq j$, a set of expressions $F_{i,j}$ inductively defined by:

- $F_{i,i} = \{x_i\}$
- $\forall i > j\ F_{i,j} = F_{i-1,j} \cup \{e[\{x_k \leftarrow e_k\}_{k<j}] \mid e$ is the expression of an update of an edge of level $i \wedge \forall k\ e_k \in F_{i,k}\}$

As in the proof of proposition 1, one establishes that the size of every $F_{i,j}$ is at most doubly exponential w.r.t. the number of clocks and polynomial w.r.t. the size of ITA when the number of clocks is fixed.

$\mathcal{A}'$ is then defined as follows.

- The set of states is $Q' = \{(q^+, e_1, \ldots, e_{i-1}) \mid q \in Q \wedge \lambda(q) = i \wedge \forall j\ e_j \in F_{i,j}\}$ $\cup \{(q^-, e_1, \ldots, e_i), \mid q \in Q \wedge \lambda(q) = i \wedge \forall j\ e_j \in F_{n,j}\}$, with $pol(q^+, e_1, \ldots, e_{i-1}) = pol(q)$ and $pol(q^-, e_1, \ldots, e_i) = U$. Moreover $\lambda(q^+, e_1, \ldots, e_{i-1}) = \lambda(q^-, e_1, \ldots, e_i) = \lambda(q)$.
- Let $q \xrightarrow{\varphi,a,u} q'$ with $i' = \lambda(q') \geq \lambda(q) = i$, and $u = \wedge_{j=1}^{i} x_j := C_j$. Then for every $(q^+, e_1, \ldots, e_{i-1})$ there is a transition $(q^+, e_1, \ldots, e_{i-1}) \xrightarrow{\varphi',a,u'}$ $(q'^+, e'_1, \ldots, e'_{i'-1})$ with $\varphi' = \varphi(\{x_j \leftarrow e_j\}_{j<i})$, $u' = x_i := C_i[\{x_j \leftarrow e_j\}_{j<i}]$, $\forall j < i\ e'_j = C_j[\{x_k \leftarrow e_k\}_{k<i}]$ and $\forall i \leq j < i'\ e'_j = x_j$
- Let $q \xrightarrow{\varphi,a,u} q'$ with $i' = \lambda(q') < \lambda(q) = i$ and $u = \wedge_{j=1}^{i} x_j := C_j$. Then for every $(q^+, e_1, \ldots, e_{i-1})$ there is a transition $(q^+, e_1, \ldots, e_{i-1}) \xrightarrow{\varphi',a,u'}$ $(q'^-, e'_1, \ldots, e'_{i'})$ with $\varphi' = \varphi(\{x_j \leftarrow e_j\}_{j<i})$, $u' = \emptyset$, $\forall j \leq i'\ e'_j = C_j[\{x_k \leftarrow e_k\}_{k<i}]$
- For every $(q^-, e_1, \ldots, e_i)$, there is a transition $(q^-, e_1, \ldots, e_i) \xrightarrow{true,\varepsilon,x_i:=e_i}$ $(q^+, e_1, \ldots, e_{i-1})$.

In words, the automaton $\mathcal{A}'$ memorises in a state at level i for every clock $x_{i'}$ at a lower level an expression depending on $x_1, \ldots, x_{i'}$ that provides its current value when evaluated by the value of every clock $x_{i''}$ at the last time the automaton has reached a level upper than i''.

Thus given a transition, one modifies the guard according to these expressions. The modification of the update consists to only apply the update at the current level and to takes into account the other updates in the expressions labelling the destination state. When the transition increases the level, the expression associated with a new "frozen" clock (x_j for $i \leq j < i'$) is the clock itself. The urgent states $(q^-, -)$ are introduced for handling the case of a transition that decreases the level. In this case, one reaches such a state that memorizes also the expression of the clock at the current level. From this state a single transition must be (immediately) taken whose effect is to perform the update corresponding to the memorised expression.

The initial state of $\mathcal{A}'$ is $(q_0^+, x_1, \ldots, x_{i-i})$ with $\lambda(q_0) = i$. The final states of $\mathcal{A}'$ are the states $(q^+, -)$ for $q \in F$. It is routine to check that the languages of the two automata are identical.

Proposition 6. *The reachability problem for ITA belongs to* 2-NEXPTIME *and* NP *when the number of clocks is fixed.*

Proof. Given an ITA $\mathcal{A}$, we build the ITA$_-$ $\mathcal{A}'$ of proposition 5. Then we apply on $\mathcal{A}'$ the reachability procedure of proposition 4. In this procedure, we consider paths of length $O(E'^{3n}) = O(\left(E^{2^{p(n)}}\right)^{3n}) = O(E^{3n2^{p(n)}})$ with p some polynomial. This establishes the claimed upper bounds.

5 Expressive power of ITA

In this section, we compare the expressive power of ITA with classical Timed Automata (TA) and also with Controlled Real-Time Automata (CRTA) [10]. We denote by TL (resp. CRTL) the family of timed languages accepted by TA (resp. CRTA), with TL strictly contained in CRTL, and we prove that:

Proposition 7. *The families TL and ITL are incomparable. The families CRTL and ITL are incomparable.*

5.1 ITL is not contained in TL, nor in CRTL

Recall that a Timed Automaton is a tuple $\mathcal{A} = (\Sigma, Q, q_0, F, X, \Delta)$, where Σ is a finite alphabet, Q is a finite set of states, q_0 is the initial state, $F \subseteq Q$ is the set of final states, X is a set of clocks and $\Delta \subseteq Q \times [\mathcal{C}(X) \times (\Sigma \cup \{\varepsilon\}) \times \mathcal{U}(X)] \times Q$ is the set of transitions.
Since all clocks evolve with rate 1, the only difference from ITA in the definition of semantics concerns a time step of duration d, which is defined by $(q, v) \xrightarrow{d} (q, v + d)$.

CRTA extend TA with the following features: the clocks and the states are partionned according to colors belonging to a set Ω and with every state is associated a rational velocity. When time elapses in a state, the set of active clocks (i.e. with the color of the state) evolve with rate equal to the velocity of the state while other clocks remain unchanged. For sake of simplicity, we now propose a slightly simplified version of CRTA.

Definition 5. *A CRTA $\mathcal{A} = (\Sigma, Q, q_0, F, X, up, low, vel, \lambda, \Delta)$ on a finite set Ω of colors is defined by:*
- Σ is the alphabet of actions,
- Q is a set of states, $q_0 \in Q$ is the initial state, F is the set of final states,
- X is a set of clocks, up and low are mappings which associate with each clock respectively an upper and a lower bound, $vel : Q \mapsto \mathbb{Q}$ is the velocity mapping,
- $\lambda : X \cup Q \mapsto \Omega$ is the coloring mapping and

- Δ is the set of transitions. A transition in Δ has guards in $\mathcal{C}(X)$ with constants in $\mathbb{Q}$ and updates in $\mathcal{U}(X)$ (i.e. only reset). The lower and upper bound mappings satisfy $low(x) \leq 0 \leq up(x)$ for each clock $x \in X$, and $low(x) \leq b \leq up(x)$ for each constant such that $x \bowtie b$ is a constraint in $\mathcal{A}$.

The original semantics of CRTA is rather involved in order to obtain decidability of the reachability problem. It ensures that entering a state q in which clock x is active, the following conditions on the clock bounds hold : if $vel(q) > 0$ then $x \geq low(x)$ and if $vel(q) < 0$ then $x \leq up(x)$. Instead (and equivalently) we add a syntactical restriction which ensures this behaviour. For instance, if a transition with guard φ and reset u enters state q with $vel(q) < 0$ and if x is the only clock such that $\omega(x) = \omega(q)$, then we replace this transition by two other transitions: the first one has guard $\varphi \wedge x > up(x)$ and adds $x := 0$ to the reset condition u, the other has guard $\varphi \wedge x \leq up(x)$ and reset u. In the general case where k clocks have color $\omega(q)$, this leads to 2^k transitions. With this syntactical condition, again the only difference from ITA concerns a time step of duration d, defined by $(q, v) \xrightarrow{d} (q, v')$, with $v'(x) = v(x) + vel(q)d$ if $\omega(x) = \omega(q)$ and $v'(x) = v(x)$ otherwise.

The next proposition show that ITA cannot be reduced to TA or CRTA. Observe also that the language of the first assertion is very simple: all words have length 2.

Proposition 8.
1. There exists a language in ITL whose words have bounded length which is not in TL.
2. There exists a language in ITL which is not in CRTL.

Proof. To prove the first point, consider the ITA $\mathcal{A}_1$ in Fig. 1. Suppose, by contradiction, that L_1 is accepted by some timed automaton $\mathcal{B}$ in TA (possibly with ε-transitions) and let d be the granularity of $\mathcal{B}$, *i.e.* the gcd of all rational constants appearing in the constraints of $\mathcal{B}$ (thus each such constant can be written k/d for some integer k). Then the word $w = (a, 1 - 1/d)(b, 1 - 1/2d)$ is accepted by $\mathcal{B}$ through a finite path. Consider now the automaton $\mathcal{B}'$ in TA, consisting of this single path (where states may have been renamed). We have $w \in \mathcal{L}(\mathcal{B}') \subseteq \mathcal{L}(\mathcal{B}) = L$ and $\mathcal{B}'$ contains no cycle. Using the result in [5], we can build a timed automaton $\mathcal{B}''$ without ε-transition and with same granularity d such that $\mathcal{L}(\mathcal{B}'') = \mathcal{L}(\mathcal{B}')$, so that $w \in \mathcal{L}(\mathcal{B}'')$. The accepting path for w in $\mathcal{B}''$ contains two transitions : $p_0 \xrightarrow{\varphi_1, a, r_1} p_1 \xrightarrow{\varphi_2, b, r_2} p_2$. After firing the a-transition, all clock values are $1 - 1/d$ or 0, thus all clock values are $1 - 1/2d$ or $1/2d$ when the b-transition is fired. Let $x \bowtie c$ be an atomic proposition appearing in φ_2. Since the granularity of $\mathcal{B}''$ is d, the $\bowtie$ operator cannot be $=$ otherwise the constraint would be $x = 1/2d$ or $x = 1 - 1/2d$. If the constraint is $x < c$, $x \leq c$, $x > c$, or $x \geq c$, the path will also accept some word $(a, 1 - 1/d)(b, t)$ for some $t \neq 1 - 1/2d$. This is also the case if the constraint φ_2 is true. We thus obtain a contradiction with $\mathcal{L}(\mathcal{B}'') \subseteq L$, which ends the proof.

To prove the second point, consider the language:

$$L_2 = \{(a,\tau)(a,2\tau)\dots(a,n\tau) \mid n \in \mathbb{N}, \tau > 0\}$$

defined above, accepted by the ITA $\mathcal{A}_2$ in Fig. 2. This language cannot be accepted by a CRTA (see [10]).

5.2 TL is not contained in ITL

We now prove that there exists a language in TL that does not belong to ITL.

The next lemma is based on a counting argument and does not depend on the semantics of ITA$_-$ except for the policy.

Lemma 1. *Let $\mathcal{A}$ be an ITA$_-$ with m transitions and n clocks, then in a sequence $(e_1,\dots,e_l)$ of transitions of $\mathcal{A}$ where $l > (m+2n)^{3n}$, there exist $i < j$ with $e_i = e_j$ such that the level of any transition e_k with $i \le k \le j$ is greater than or equal to the level of e_i, say p, and:*

- *either e_i updates x_p,*
- *either no e_k with $i \le k \le j$ updates x_p and e_i is delayed or lazy.*
- *or no e_k with $i \le k \le j$ updates x_p and e_i and e_j occur at the same instant.*

Proof. Assume that the conclusions of the lemma are not satisfied, we claim that $l \le (m+2n)^{3n}$.

First we prove that the number of transitions of level h that occur between two occurrences of transitions of strictly lower level is less than or equal to $(m+2)^3$. Indeed there can be no more than m occurrences of transitions that update x_h. Then between two such transitions (or before the first or after the last) there can be no more than m lazy or delayed transitions of level h that do not update x_h. Finally between any kind of previous transitions (or before the first or after the last), there can be no more than m urgent transitions that do not update x_h, since they all occur at the same instant.

Summing up, there can be no more than $m + m(m+1) + m((m+1)^2+1) \le (m+2)^3$ transitions of level h that occur between two occurrence of transitions of strictly lower level.

Now we prove by induction that the number of transitions at level less than or equal to h is at most $(m+2h)^{3h}$. This is true for $h = 1$ by the previous proof. Assume the formula valid for h, then grouping the transitions of level $h+1$ between the occurrences of transition of lower level (or before the first or after the last), we obtain that the number of transitions at levels less than or equal to $h+1$ is at most:
$(m+2h)^{3h} + ((m+2h)^{3h}+1)(m+2)^3 \le (m+2h)^{3h+3} + 2(m+2h)^{3h}$
$\le (m+2h+2)^{3h+3}$

We now establish a pumping lemma for ITL. In words, for every language, there exists a constant B such that in any word (with time elapsing before any occurrence of a letter) of the language with B *consecutive tagged* positions, there exist two tagged positions *possibly equal* such that:

– either a new word of the language is obtained by erasing the subword between the two positions without any time shift of the remaining suffix. In this case the two positions must be different.
– or a new word of the language is obtained by duplicating the (possibly empty) subword between the two positions and shifting by a *non null* duration at least equal to the duration of the subword, the time occurrence of the suffix starting at the beginning of the duplicating subword.

Notation. Given a timed word $w = (a_1, \tau_1) \dots (a_l, \tau_l)$ and $i \leq j$, we define $w_{[i,j]}$ as the word $(a_i, \tau_i) \dots (a_j, \tau_j)$. Let d be duration, then $w + d$ denotes the word where all dates in w have been increased by d.

Lemma 2 (Pumping lemma). *Let L be a language of ITL. Then there exists $B \in \mathbb{N}$ $(B > 0)$ such that for every timed word $w = (a_1, \tau_1) \dots (a_l, \tau_l) \in L$ with $\forall 1 \leq i < l,\ 0 < \tau_i < \tau_{i+1}$ and every i such that $1 \leq i < i + B \leq l$ there exists $i \leq i' \leq j' \leq i + B$ such that:*

– *either $i' < j'$ and $w_{[1,i'-1]} w_{[j',l]}$ belongs to L,*
– *or there exists $d > 0$ such that $d \geq \tau_{j'} - \tau_{i'}$ and $w_{[1,j'-1]}(w_{[i',l]} + d)$ belongs to L.*

Proof. Due to proposition 5, we may assume that L is accepted by some automaton $\mathcal{A}$ in ITA$_-$. We choose $B = (m + 2n)^{3n}$, where m is its number of transitions and n its number of clocks.

Now take a word $w = (a_1, \tau_1) \dots (a_l, \tau_l) \in L$ and i fulfilling the hypothesis. Let $\sigma \in \mathcal{A}$ be an accepting sequence for w with a minimal number of transitions. Let $\sigma' = e_1, \dots, e_h$ be the subsequence of σ corresponding to $(a_i, \tau_i) \dots (a_{i+B}, \tau_{i+B})$. Observe that $h > B$. By lemma 1, there exist $k < k'$ with $e_k = e_{k'}$ such that the level of any transition $e_{k''}$ with $k \leq k'' \leq k'$ is greater than or equal to the level of e_k, say p, and:

1. either e_k updates x_p,
2. either no $e_{k''}$ with $k \leq k'' \leq k'$ updates x_p and e_k is delayed or lazy.
3. or no $e_{k''}$ with $k \leq k'' \leq k'$ updates x_p and e_k and $e_{k'}$ occur at the same instant.

Case 1. We observe that the values of clocks $x_1, \dots, x_p$ after the transitions e_k and $e_{k'}$ are identical. We now distinguish two subcases.

Either e_k and $e_{k'}$ occur at the same time instant say τ. By miminality of σ, at least a transition between e_k and $e_{k'}$ must be labelled by a letter (and not by ε). Otherwise we could delete the sequence of transitions and still accept w. If we now delete the subsequence $e_k, \dots, e_{k'-1}$, the resulting sequence accepts a word w' corresponding to the first conclusion.

Or e_k occurs at a time instant τ and $e_{k'}$ occurs at $\tau + d$ with $d > 0$. Then we can repeat the timed subsequence $e_k, \dots, e_{k'-1}$ before the occurrence of the subsequence $e_k, \dots, e_l$ and the subsequence starting with the duplication is shifted by d. This yields the second conclusion.

Case 2. We observe that the value of the clocks $x_1, \ldots, x_{p-1}$ after the transitions e_k and $e_{k'}$ are identical and that the value of the clock x_p is increased by the time elapsed between the occurrences of e_k and $e_{k'}$, say d. If $d = 0$, the same reasoning as previously done leads to the first conclusion.

Otherwise, since e_k is lazy or delayed we can substitute to the subsequence $e_k, \ldots, e_{k'-1}$ the time delay $d > 0$ equal to its duration and obtain from σ a new accepting sequence σ''. The minimality of σ implies that a letter occurs in the subsequence $e_k, \ldots, e_{k'-1}$. So σ'' accepts a word that corresponds to the first conclusion.

Case 3. Since the timed subsequence $e_k \ldots e_{k'-1}$ has zero duration we can delete it. The same reasonning as previously done leads to the first conclusion.

More elaborated versions of the pumping lemma could be established. But this one is enough for our purposes.

Let L_3 be the language defined by

$$L_3 = \{(a, \tau_1)(b, \tau_2) \ldots (a, \tau_{2l+1})(b, \tau_{2l+2}) \mid l \in \mathbb{N} \wedge$$

$$\forall 0 \leq i \leq l \; \tau_{2i+1} = i+1 \wedge i+1 < \tau_{2i+2} < i+2 \wedge \forall 1 \leq i \leq l \; \tau_{2i+2} - (i+1) < \tau_{2i} - i$$

In words, the a's occur at time units, there is exactly an occurrence of b between two occurrences of a and the successive occurrences of b come closer to the occurrence of the preceding a. This language is in TL as can be checked on the TA of figure 5 (first proposed in [1]).

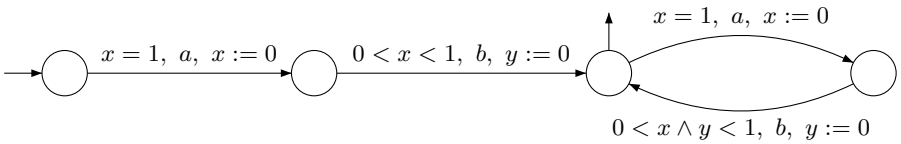

Fig. 5. A timed automaton for L_3

Lemma 3. *The language L_3 does not belong to ITL.*

Proof. Assume the contrary.
Pick some word $w = (a, \tau_1)(b, \tau_2) \ldots (a, \tau_{2B+1})(b, \tau_{2B+2}) \in L_3$ with B being the bound given by the pumping lemma. We tag the first B positions and apply the pumping lemma.

In the first case, the erased subword cannot contain an a since in the new word there would be a time unit without an a. So the erased subword is necessarily a b (but not the last one), thus yielding a new word with two consecutive as, contrary to the definition of L_3.

In the second case, we establish a contradiction that depends on the length of the "duplicated" subword:

– If its length is null, there is a non null shift of a suffix containing some a and there is a time unit without an a before the first a in the suffix.
– If its length is equal to 1, there is a repetition of an a or a b.
– If its length is greater than or equal to 2, there is an occurrence of ab (or ba) which is duplicated later with the same time difference.

Note that the feature preventing L_3 to be in ITL lies in the decreasing delays between the a's and their immediately following b. A language in ITL can record k different constant delays, using $k+1$ clocks. For instance on the alphabet $\Sigma = \{a_1, \ldots, a_k\}$, the language $M_k = \{(a_1, \tau_1) \ldots (a_k, \tau_k)(a_1, \tau_1 + 1) \ldots (a_k, \tau_k + 1) \ldots (a_1, \tau_1 + n) \ldots (a_k, \tau_k + n) \mid n \geq 1, \tau_2 \leq \cdots < \tau_k \leq \tau_1 + 1\}$ is accepted by an ITA$_-$ with $k+1$ clocks. The figure below illustrates the case where $k = 3$, with all states lazy.

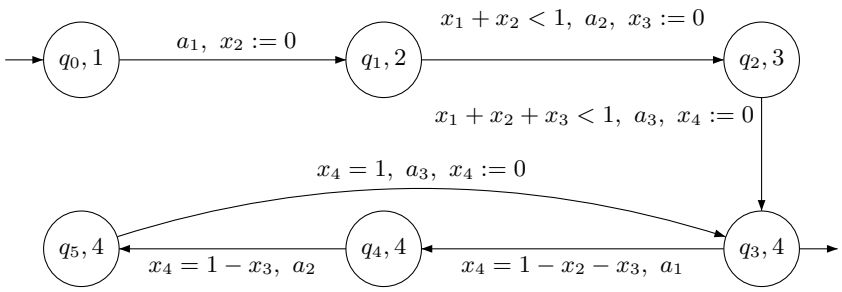

Fig. 6. An interrupt timed automaton for M_3

5.3 Closure under complementation and intersection

Proposition 9. *ITL is not closed under complementation.*

Proof. We prove that L_3^c is an ITL. A timed word belongs to L_3^c iff one of the following assertions hold:

– An a occurs not at a time unit.
– An a is missing at some time unit that precedes some letter of the word.
– A b occurs at a time unit.
– There is no b in an interval $[i, i+1]$ with an a at time $i \in \mathbb{N}$.
– There are two bs in an interval $[i, i+1]$ with an a at time $i \in \mathbb{N}$.
– There is an occurrence of $abab$ such that the time difference between the two first occurrences is smaller than or equal to the time difference between the two last occurrences.

Since ITL is trivially closed under union, it is enough to prove that each assertion from the set above can be expressed by an ITA. The five first assertions are straightforwardly modelled by an ITA with a single clock (and ε-transitions) and we present in figure 7 an ITA with two clocks corresponding to the last one.

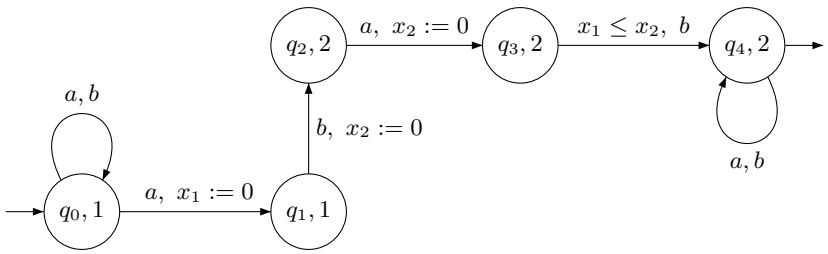

Fig. 7. An ITA for the language defined by assertion 6

Proposition 10. *ITL is not closed under intersection.*

Proof. L_3 is the intersection of L'_3 and L''_3 defined by:

- The words of L'_3 are $(a,1)(b,\tau_1)\dots(a,n)(b,\tau_n)$ with $\forall 1 \leq i \leq n\ i < \tau_i < i+1$.
- The words of L''_3 are $(a,\tau'_1)(b,\tau_1)\dots(a,\tau'_n)(b,\tau_n)$ with $\forall 1 \leq i \leq n-1\ \tau_{i+1} - \tau_i < 1$.

Both languages are accepted by one-clock ITA (which are also one-clock TA). In case of L'_3, (1) the clock is reset at every occurrence of an a; (2) an a must occur when the clock is 1 and (3) a single b must occur when the clock is in $(0,1)$. In case of L''_3, (1) the clock is reset at every occurrence of a b (2) a b must occur when the clock is less than 1 except for the first b and (3) a single a must occur before every occurrence of a b.

6 Combining ITA and CRTA

We finally define an extended class denoted by ITA$^+$, including a set of clocks at an implicit additional level 0, corresponding to a basic task described as in a CRTA.

Definition 6 (ITA$^+$). *An* extended interrupt timed automaton *is a tuple* $\mathcal{A} = (Q, q_0, F, pol, X \uplus Y, \Sigma, \Omega, \lambda, up, low, vel, \Delta)$*, where:*

- Q *is a finite set of states,* q_0 *is the initial state and* $F \subseteq Q$ *is the set of final states.*
- $pol : Q \mapsto \{L, U, D\}$ *is the timing policy of states.*
- $X = \{x_1, \dots, x_n\}$ *consists of* n *interrupt clocks and* Y *is a set of basic clocks,*
- Σ *is a finite alphabet,*
- Ω *is a set of colours, the mapping* $\lambda : Q \uplus Y \mapsto \{1, \dots, n\} \uplus \Omega$ *associates with each state its level or its colour, with* $x_{\lambda(q)}$ *the active clock in state* q *for* $\lambda(q) \in \mathbb{N}$ *and* $\lambda(y) \in \Omega$ *for* $y \in Y$*, For every state* $q \in \lambda^{-1}(\Omega)$*, one has* $pol(q) = L$*.*

– *up and low are mappings from* Y *to* $\mathbb{Q}$ *with the same constraints as CRTA (see definition 5), and* $vel : Q \mapsto \mathbb{Q}$ *is the clock rate with* $\lambda(q) \notin \Omega \Rightarrow vel(q) = 1$
– $\Delta \subseteq Q \times [\mathcal{C}^+(X \cup Y) \times (\Sigma \cup \{\varepsilon\}) \times \mathcal{U}^+(X \cup Y)] \times Q$ *is the set of transitions. Let* $q \xrightarrow{\varphi,a,u} q'$ *in* Δ *be a transition.*
  1. *The guard* φ *is of the form* $\varphi_1 \wedge \varphi_2$ *with the following conditions. If* $\lambda(q) \in \mathbb{N}$, φ_1 *is an ITA guard on* X *and otherwise* $\varphi_1 = true$. *Constraint* φ_2 *is a CRTA guard on* Y *(also possibly equals to true).*
  2. *The update* u *is of the form* $u_1 \wedge u_2$ *fullfilling the following conditions. Assignments from* u_1 *update the clocks in* X *with the constraints of ITA when* $\lambda(q)$ *and* $\lambda(q')$ *belong to* $\mathbb{N}$. *Otherwise it is a global reset of clocks in* X. *Assignments from* u_2 *update clocks from* Y, *like in CRTA.*

Any ITA can be viewed as an ITA$^+$ with Y empty and $\lambda(Q) \subseteq \{1, \ldots, n\}$, and any CRTA can be viewed as an ITA$^+$ with X empty and $\lambda(Q) \subseteq \Omega$. Class ITA$^+$ combines both models in the following sense. When the current state q is such that $\lambda(q) \in \Omega$, the ITA part is inactive. Otherwise, it behaves as an ITA but with additional constraints about clocks of the CRTA involved by the extended guards and updates. The semantics of ITA$^+$ is defined as usual but now takes into account the velocity of CRTA clocks.

Definition 7 (Semantics of ITA$^+$). *The semantics of an automaton* $\mathcal{A}$ *in ITA*$^+$ *is defined by the transition system* $\mathcal{T}_\mathcal{A} = (S, s_0, \rightarrow)$. *The set* S *of configurations is* $\{(q, v) \mid q \in Q,\ v \in \mathbb{R}^{X \cup Y}\}$, *with initial configuration* $(q_0, \mathbf{0})$. *An accepting configuration of* $\mathcal{T}_\mathcal{A}$ *is a pair* (q, v) *with* q *in* F. *The relation* $\rightarrow$ *on* S *consists of time steps and discrete steps, the definition of the latter being the same as before:*

Time steps: *Only the active clocks in a state can evolve, all other clocks are suspended. For a state* q *with* $\lambda(q) \in \mathbb{N}$ *(the active clock is* $x_{\lambda(q)}$*), a time step of duration* d *is defined by* $(q, v) \xrightarrow{d} (q, v')$ *with* $v'(x_{\lambda(q)}) = v(x_{\lambda(q)}) + d$ *and* $v'(x) = v(x)$ *for any other clock* x. *For a state* q *with* $\lambda(q) \in \Omega$ *(the active clocks are* $Y' = Y \cap \lambda^{-1}(\lambda(q))$*), a time step of duration* d *is defined by* $(q, v) \xrightarrow{d} (q, v')$ *with* $v'(y) = v(y) + vel(q)d$ *for* $y \in Y'$ *and* $v'(x) = v(x)$ *for any other clock* x. *When* $pol(q) = U$, *time steps from* q *are forbidden.*

Discrete steps: *A discrete step* $(q, v) \xrightarrow{a} (q', v')$ *occurs if there exists a transition* $q \xrightarrow{\varphi,a,u} q'$ *in* Δ *such that* $v \models \varphi$ *and* $v' = v[u]$. *When* $pol(q) = D$ *every discrete step from* q *must be immediately preceded by a non null time step.*

In order to illustrate the interest of the combined models, an example of a (simple) login procedure is described in figure 8 as a TA with interruptions at a single level.

First it immediately displays a prompt and arms a time-out of 1 t.u. handled by clock y (transition $init \xrightarrow{p} wait$). Then either the user answers correctly within this delay (transition $wait \xrightarrow{ok} log$) or he/she answers incorrectly or let time elapse, both cases with transition $wait \xrightarrow{er} init$, and the system prompts

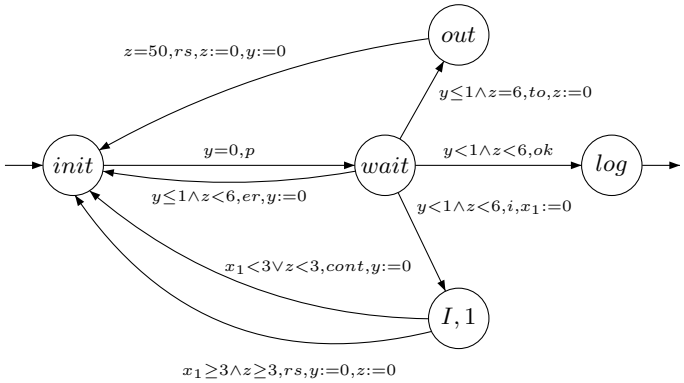

Fig. 8. An automaton for login in ITA$^+$

again. The whole process is controlled by a global time-out of 6 t.u. (transition $wait \xrightarrow{to} out$) followed by a long suspension (50 t.u.) before reinitializing the process (transition $out \xrightarrow{rs} init$). Both delays are handled by clock z. At any time during the process (in fact in state *wait*), a system interrupt may occur (transition $wait \xrightarrow{i} I$). If the time spent (measured by clock x_1) during the interrupt is less than 3 t.u. or the time already spent by the user is less than 3, the login process resumes (transition $I \xrightarrow{cont} init$). Otherwise the login process is reinitialized allowing again the 6 t.u. (transition $I \xrightarrow{rs} init$). In both cases, the prompt will be displayed again. Since invariants are irrelevant for the reachability problem we did not include them in the models. Of course, in this example state *wait* should have invariant $y \leq 1 \wedge z \leq 6$ and state *out* should have invariant $z \leq 50$.

We extend the decidability and complexity results of the previous models when combining them with CRTA. Class ITA$^+_-$ is obtained in a similar way by combining ITA$_-$ with CRTA.

Proposition 11.
1. The reachability problem for ITA$^+_-$ belongs to NEXPTIME *and is* PSPACE*-complete when the number of interrupt clocks is fixed.*
2. The reachability problem for ITA$^+$ is decidable and belongs to 2-NEXPTIME *and is* PSPACE*-complete when the number of interrupt clocks is fixed.*

Proof.
Case of ITA$^+_-$. We first consider the reachability problem for two states q_i and q_f such that $\lambda(q_i) \in \Omega \wedge \lambda(q_f) \in \Omega$. The procedure consists in performing a non deterministic search along an elementary path where the vertices are graph classes of the CRTA. Let (q, Z) be the current class, the procedure chooses non deterministically the next class (q', Z') and checks that there exists a configuration of (q, Z) and an execution only through states q'' with $\lambda(q'') \in \mathbb{N}$ that leads to a configuration of (q', Z'). This is solved as previously by non deterministically choosing an execution path, building a linear program related to the path

(of exponential size) and solving it. Let us prove that such a path can be chosen such that its length is bounded by.

Let $\mathcal{A} = (Q, q_0, F, pol, X \uplus Y, \Sigma, \Omega, \lambda, up, low, vel, \Delta)$ be an ITA$^+_-$. Let $E = |\Delta|$ be the number of transitions, p be the number of CRTA clocks and for a given run let m_k be the number of transitions of level k.

Assume that there is a run π from $(q, v) \in (q, Z)$ to some configuration $(q', v') \in (q', Z')$ such that all intermediate states q'' are such that $\lambda(q'') \in \mathbb{N}$. We say that a transition e of π usefully resets a clock $y \in Y$ if it is the first transition of π that resets y. Observe that there are at most p useful resetting transitions and that between two such successive transitions (or before the first one or after the last one) the value of the clocks of Y are unchanged when transitions are fired.

We consider a subrun ρ between two such successive transitions (or before the first one or after the last one) from (q_1, v_1) to (q_2, v_2).

Exactly as in proposition 4, we build a subrun ρ' from (q_1, v_1) to (q_2, v_2) which fulfills:

- $m'_1 \leq (E+2)^3$
- $\forall k\ m'_{k+1} \leq (E+2)^3(m'_k + 1)$

Thus $\sum_{k=1}^{n} m'_k = O(E^{3n})$. Concatenating the subruns, the useful resetting transitions and the initial transition, one obtains a run π' from (q, v) to (q', v') of length in $O(pE^{3n})$.

The key point ensuring correctness of the procedure is that the existence of a solution depends only on the starting class (q, Z) and not on the configuration inside this class. This is due to the separation of guards and updates between the two kinds of clocks on the transitions.

When state q_i (resp. q_f) is not at the basis level, the procedure adds an initial (resp. final) guess also checked by a linear program. When the number of clocks is fixed the dominant factor is the path search in the class graph and PSPACE hardness follows from the result in TA.

Case of ITA$^+$. We transform the ITA part of the automaton in ITA$_-$ *via* the procedure of proposition 5 and apply the procedure for ITA$^+_-$.

7 Conclusion

We have proposed a subclass of hybrid automata, called ITA. An ITA describes a set of tasks, executing at interrupt levels, with exactly one active clock at each level. We prove that the reachability problem is decidable in this class, with a procedure in 2-NEXPTIME. We also consider restrictions on this class, that make the complexity of decision lower (in NEXPTIME). We show that these results still hold for a combination of ITA with the class CRTA. When the number of clocks is fixed, the complexity bound is the same as the one of TA for the combined model (PSPACE complete) and even better in case of ITA (in NP). The corresponding families of languages ITL and CRTL are incomparable.

Acknowledgement

We wish to thank the anonymous referees of Fossacs'09 for their comments and questions which led us to improve and extend this study.

References

1. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science* 126:183–235, 1994.
2. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science* 138:3–34, 1995.
3. E. Asarin, O. Maler and A. Pnueli. Reachability Analysis of Dynamical Systems having Piecewise-Constant Derivatives. *Theoretical Computer Science* 138:35–66, 1995.
4. E. Asarin, G. Schneider and S. Yovine. Algorithmic Analysis of Polygonal Hybrid Systems, Part I: Reachability. *Theoretical Computer Science* 379(1-2):231–265, 2007.
5. B. Bérard, V. Diekert, P. Gastin and A. Petit. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae* 36:145–182, 1998.
6. P. Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design* 24(3):281–320, 2004.
7. T. Brihaye, V. Bruyère and J.-F. Raskin. On Model-Checking Timed Automata with Stopwatch Observers. *Information and Computatiion* 2004(3):408–433, 2006.
8. F. Cassez and K. G. Larsen. The impressive power of stopwatches. *Proceedings of CONCUR'00*, number 1877 in LNCS, pages 138–152, 2000.
9. C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1(4):385-415, 1992.
10. F. Demichelis and W. Zielonka. Controlled timed automata. *Proceedings of CONCUR'98*, number 1466 in LNCS, pages 455–469, 1998.
11. T.A. Henzinger, P.W. Kopke, A. Puri and P. Varaiya. What's decidable about hybrid automata ? *Journal of Computer and System Sciences* 57:94–124,1998.
12. Y. Kesten, A. Pnueli, J. Sifakis and S. Yovine. Decidable Integration Graphs. *Information and Computation* 150(2):209–243, 1999.
13. G. Lafferriere, G. J. Pappas and S. Yovine. A new class of decidable hybrid systems. *Proceedings of HSCC'99*, number 1569 in LNCS, pages 137–151, 1999.
14. G. Lafferriere, G. J. Pappas and S. Yovine. Symbolic reachability computations for families of linear vector fields. *Journal of Symbolic Computation*, 32(3):231–253, 2001.
15. O. Maler, Z. Manna and A. Pnueli. From Timed to Hybrid Systems. *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, number 600 in LNCS, pages 447–484, 1992.
16. C. Roos, T. Terlaky, and J.-Ph. Vial. *Theory and Algorithms for Linear Optimization. An Interior Point Approach*. Wiley-Interscience, John Wiley & Sons Ltd, West Sussex, England, 1997.