Interrupt Timed Automata

Beatrice Bérard[1,⋆] and Serge Haddad[2,⋆]

[1] Université Pierre & Marie Curie, LIP6/MoVe, CNRS UMR 7606, Paris, France
Beatrice.Berard@lip6.fr
[2] Ecole Normale Supérieure de Cachan, LSV, CNRS UMR 8643, Cachan, France
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 in 2-EXPSPACE and in PSPACE when the number of clocks is fixed, with a procedure based on a generalized class graph. Furthermore we 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). There exist languages accepted by an ITA_- but neither by timed automata nor by controlled real-time automata (CRTA), another extension of timed automata. However we conjecture that CRTA is not contained in ITA. So, we 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 the states. When a 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 [14], 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

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

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 [9], integration graphs [11], o-minimal hybrid systems [12,13], 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. 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 [10,8,7], we prove that for ITA, it belongs to 2-EXPSPACE. The procedure significantly extends the classical region construction of [1] by associating with each state a family of orderings over linear expressions. Furthermore, 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 for both classes: PSPACE (resp. NP) for ITA (resp. ITA$_-$).

We also investigate 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 [9]. In CRTA, clocks are also organized into a partition (according to colours) and may have different rates, but all active clocks in a given state have identical rate. We prove that there exist timed languages accepted by ITA (and also ITA$_-$) but not by a CRTA (resp. not by a TA). We conjecture that the classes ITA and CRTA are incomparable, which leads us to define a combination of the two models, 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.

Outline. In section 2, we define ITA and study its expressive power. Section 3 is devoted to the decidability of the reachability problem and section 4 extends the results for the models combining ITA and CRTA.

2 Interrupt Timed Automata

2.1 Definitions and Examples

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.

Definition 1 (Interrupt Timed Automaton). *An* interrupt timed automaton *is a tuple $\mathcal{A} = (\Sigma, Q, q_0, F, X, \lambda, \Delta)$, 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, $X = \{x_1, \ldots, x_n\}$ 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.

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.*

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 $v' = v[u]$.*

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 $(a_1, t_1) \ldots (a_p, t_p) \in (\Sigma \times \mathbb{R}_{\geq 0})^*$, 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 $t_1, \ldots, t_n$ 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

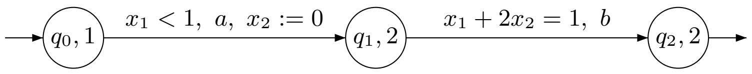


Fig. 1. An ITA $\mathcal{A}_1$ with two interrupt levels

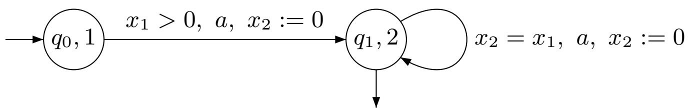


Fig. 2. An ITA $\mathcal{A}_2$ for L_2

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.

We end this paragraph with two examples of ITA. In the figures, the level of a state is indicated beside its name. 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 $L_2 = \{(a, \tau)(a, 2\tau) \dots (a, n\tau) \mid n \in \mathbb{N}, \tau > 0\}$.

2.2 Expressive Power of ITA

We now compare the expressive power of ITA with classical Timed Automata (TA) and Controlled Real-Time Automata (CRTA) [9].

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 3. *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.

We denote by TL (resp. CRTL) the family of timed languages accepted by TA (resp. CRTA), with TL strictly contained in CRTL.

Proposition 1

1. *There exists a language in ITL which is not in TL.*
2. *There exists a language in ITL whichis 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=$\{(a, \tau)(a, 2\tau) \ldots (a, n\tau) \mid n \in \mathbb{N}, \tau > 0\}$ defined above, accepted by the ITA $\mathcal{A}_2$ in Fig. 2. This language cannot be accepted by a CRTA (see [9]).

Note that we do not yet know of a language accepted by an automaton in TA (or CRTA) but not by an automaton in ITA. However, we conjecture that these classes are incomparable.

3 Reachability Is Decidable in ITA

3.1 General Case

Similarly to the decision algorithm for reachability in TA (and in CRTA), the procedure for 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 3 we show how to build the class graph which 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 4 (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, $\texttt{norm}(C,k)$, is defined by:*

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

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

Proof.
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 $R = (q, \{\preceq_k\}_{1 \le k \le \lambda(q)})$ where q is a state and $\preceq_k$ is a total preorder over E_k.

The class R describes the set of valuations:

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

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 2.

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

Discrete step. Let $R = (q, \{\preceq_k\}_{1 \le k \le \lambda(q)})$ and $R' = (q', \{\preceq'_k\}_{1 \le k \le \lambda(q')})$ 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 $\varphi = \bigwedge_{1 \le j \le J'} C_j \le 0 \wedge \bigwedge_{J'+1 \le j \le J} \neg(C_j \le 0)$. 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 $j \le J'$ 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 $k \le \lambda(q')$ and $g', h' \in E_k$.

1. Either $k \le \lambda(q)$, 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 $D = g'[u] - h'[u] = \sum_{i \le \lambda(q)} c_i x_i + d$, 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 $c_{\lambda(q)} \ge 0$ 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 $R = (q, \{\preceq_k\}_{1 \leq k \leq \lambda(q)})$.
The time successor $Post(R) = (q, \{\preceq'_k\}_{1 \leq k \leq \lambda(q)})$ of R 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 $\forall 0 \leq d' \leq d,\ (q, v + d') \in R \cup Post(R)$.

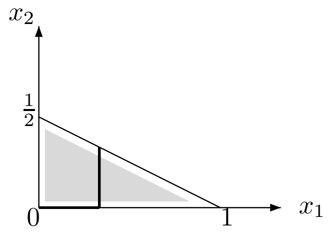
The initial state of this graph is defined by the class R_0 with $[\![R_0]\!]$ containing $(q_0, \mathbf{0})$ which can be straightforwardly determined. The reachability problem is then solved by a non deterministic search of a path in this graph (without building it) leading to the complexity stated in the proposition. 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.

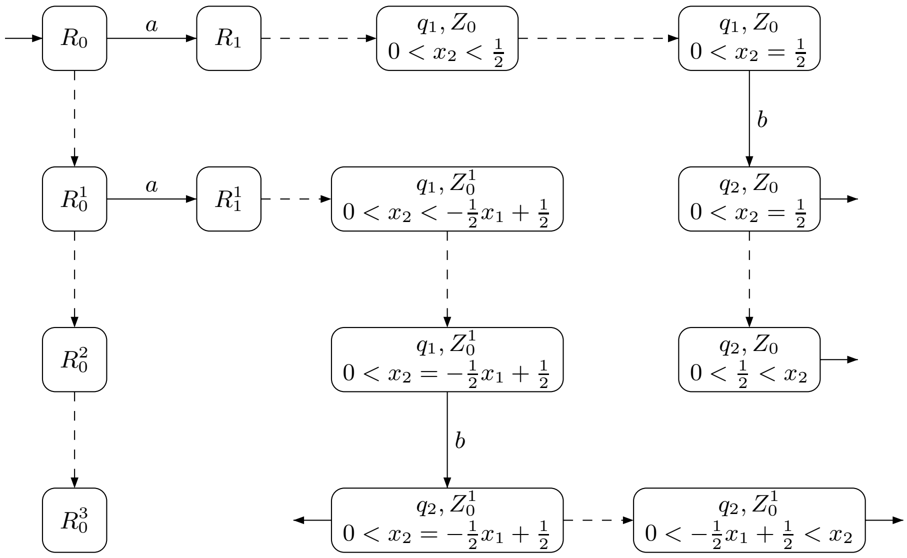
Example. We illustrate this construction of a class automaton for the automaton $\mathcal{A}_1$ from section 2 (see figure 3, where dashed lines indicate time successors).

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 below, 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.





R_0
a
R_1
q_1, Z_0 $0 < x_2 < \frac{1}{2}$
q_1, Z_0 $0 < x_2 = \frac{1}{2}$
b
R_0^1
a
R_1^1
q_1, Z_0^1 $0 < x_2 < -\frac{1}{2}x_1 + \frac{1}{2}$
q_2, Z_0 $0 < x_2 = \frac{1}{2}$
R_0^2
q_1, Z_0^1 $0 < x_2 = -\frac{1}{2}x_1 + \frac{1}{2}$
q_2, Z_0 $0 < \frac{1}{2} < x_2$
b
R_0^3
q_2, Z_0^1 $0 < x_2 = -\frac{1}{2}x_1 + \frac{1}{2}$
q_2, Z_0^1 $0 < -\frac{1}{2}x_1 + \frac{1}{2} < x_2$

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

3.2 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 5. *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 1 still hold for ITA$_-$.

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

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

Proof. Let $\mathcal{A} = (\Sigma, Q, q_0, F, X, \lambda, \Delta)$ be an ITA$_-$. In the sequel, the level of a transition is the level of its source state. Let $E = |\Delta|$ be the number of transitions and given a fixed run, let m_k be the number of occurrences 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:

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

Thus $\sum_{k=1}^{n} m'_k = O(E^{2n})$.

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 transitions of level 1 occurring between two of these 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. Thus we obtain a run with at most $(E+1)^2$ transitions of level 1 (including at most $E(E+1)$ transitions without update).

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 [15], 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 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* $\mathcal{A} = (Q, q_0, F, X \uplus Y, \Sigma, \Omega, \lambda, up, low, vel, \Delta)$*, where:*

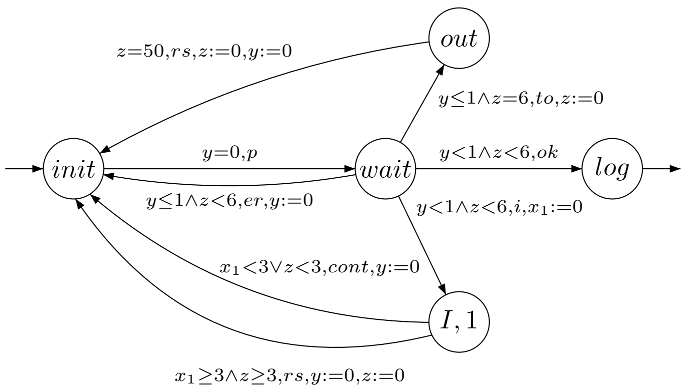
- *Q is a finite set of states, q_0 is the initial state and $F \subseteq Q$ is the set of final states.*
- *$X = \{x_1, \ldots, x_n\}$ consists of n interrupt clocks and Y is a set of basic clocks,*
- *Σ is a finite alphabet,*
- *Ω is a set of colours, the mapping $\lambda : Q \uplus Y \mapsto \{1, \ldots, n\} \uplus \Omega$ 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 $y \in Y$,*
- *up and low are mappings from Y to $\mathbb{Q}$ with the same constraints of CRTA (see definition 3), 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.*

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 $v' = v[u]$.*



In order to illustrate the interest of the combined models, an example of a (simple) login procedure is described in the figure above 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 $\mathit{init} \xrightarrow{p} \mathit{wait}$). Then either the user answers correctly within this delay (transition $\mathit{wait} \xrightarrow{ok} \mathit{log}$) or he/she answers incorrectly or let time elapse, both cases with transition $\mathit{wait} \xrightarrow{er} \mathit{init}$, and the system prompts again. The whole process is controlled by a global time-out of 6 t.u. (transition $\mathit{wait} \xrightarrow{to} \mathit{out}$) followed by a long suspension (50 t.u.) before reinitializing the process (transition $\mathit{out} \xrightarrow{rs} \mathit{init}$). Both delays are handled by clock z. At any time during the process (in fact in state *wait*), a system interrupt may occur (transition $\mathit{wait} \xrightarrow{i} I$). 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 $I \xrightarrow{cont} \mathit{init}$). Otherwise the login process is reinitialized allowing again the 6 t.u. (transition $I \xrightarrow{rs} \mathit{init}$). 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. Proofs are omitted here.

Proposition 5

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

5 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-EXPSPACE. 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 and even better in case of ITA_-. Whether the classes TA or CRTA are contained in ITA and whether ITA_- is a strict subclass of ITA are open questions.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126, 183–235 (1994)
2. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical Computer Science 138, 3–34 (1995)
3. Asarin, E., Maler, O., Pnueli, A.: Reachability Analysis of Dynamical Systems having Piecewise-Constant Derivatives. Theoretical Computer Science 138, 35–66 (1995)
4. Asarin, E., Schneider, G., Yovine, S.: Algorithmic Analysis of Polygonal Hybrid Systems, Part I: Reachability. Theoretical Computer Science 379(1-2), 231–265 (2007)
5. Bérard, B., Diekert, V., Gastin, P., Petit, A.: Characterization of the expressive power of silent transitions in timed automata. Fundamenta Informaticae 36, 145–182 (1998)
6. Bouyer, P.: Forward analysis of updatable timed automata. Formal Methods in System Design 24(3), 281–320 (2004)
7. Brihaye, T., Bruyère, V., Raskin, J.-F.: On Model-Checking Timed Automata with Stopwatch Observers. Information and Computatiion 2004(3), 408–433 (2006)
8. Cassez, F., Larsen, K.G.: The impressive power of stopwatches. In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, vol. 1877, pp. 138–152. Springer, Heidelberg (2000)
9. Demichelis, F., Zielonka, W.: Controlled timed automata. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 455–469. Springer, Heidelberg (1998)
10. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? Journal of Computer and System Sciences 57, 94–124 (1998)
11. Kesten, Y., Pnueli, A., Sifakis, J., Yovine, S.: Decidable Integration Graphs. Information and Computation 150(2), 209–243 (1999)
12. Lafferriere, G., Pappas, G.J., Yovine, S.: A new class of decidable hybrid systems. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) HSCC 1999. LNCS, vol. 1569, pp. 137–151. Springer, Heidelberg (1999)
13. Lafferriere, G., Pappas, G.J., Yovine, S.: Symbolic reachability computations for families of linear vector fields. Journal of Symbolic Computation 32(3), 231–253 (2001)
14. Maler, O., Manna, Z., Pnueli, A.: From Timed to Hybrid Systems. In: Huizing, C., de Bakker, J.W., Rozenberg, G., de Roever, W.-P. (eds.) REX 1991. LNCS, vol. 600, pp. 447–484. Springer, Heidelberg (1992)
15. Roos, C., Terlaky, T., Vial, J.-P.: Theory and Algorithms for Linear Optimization. An Interior Point Approach. Wiley-Interscience, John Wiley & Sons Ltd., West Sussex (1997)