Fundamenta Informaticae 143 (2016) 1–25
DOI 10.3233/FI-2016-1301
IOS Press

Interrupt Timed Automata with Auxiliary Clocks and Parameters[*]

Beatrice Bérard[†]
Sorbonne Université
UPMC-Paris 6, CNRS UMR 7606
Paris, France
Beatrice.Berard@lip6.fr

Serge Haddad
ENS Cachan, LSV, CNRS, INRIA
Cachan, France
haddad@lsv.ens-cachan.fr

Aleksandra Jovanović
Department of Computer Science
University of Oxford
Oxford, UK
Aleksandra.Jovanovic@cs.ox.ac.uk

Didier Lime
École Centrale de Nantes, IRCCyN, CNRS
Nantes, France
Didier.Lime@ec-nantes.fr

Abstract. Interrupt Timed Automata (ITA) are an expressive timed model, introduced to take into account interruptions according to levels. Due to this feature, this formalism is incomparable with Timed Automata. However several decidability results related to reachability and model checking have been obtained. We add auxiliary clocks to ITA, thereby extending its expressive power while preserving decidability of reachability. Moreover, we define a parametrized version of ITA, with polynomials of parameters appearing in guards and updates. While parametric reasoning is particularly relevant for timed models, it very often leads to undecidability results. We prove that various reachability problems, including *robust reachability*, are decidable for this model, and we give complexity upper bounds for a fixed or variable number of clocks, levels and parameters.

1. Introduction

Timed and hybrid models. In order to model timed systems, the expressive model of Hybrid Automata (HA) has been proposed [1]. Since its expressive power leads to the undecidability of most verification problems, several semi-decision procedures have been designed for HA as well as subclasses with

[*]This work has been supported by projects ImpRo ANR-2010-BLAN-0317 and PACS ANR-14-CE28-0002
[†]Address for correspondence: Sorbonne Université, UPMC-Paris 6, CNRS UMR 7606, Paris, France

decidability results like Timed Automata (TA) [2]. The model of interrupt timed automata (ITA) [3, 4] was proposed as a subclass of hybrid automata, incomparable with the class of timed automata, where task interruptions are taken into account. Hence ITA are particularly suited for the modelling of scheduling with preemption.

Parametric verification. Getting a complete knowledge of a system is often impossible, especially when integrating quantitative constraints. Moreover, even if these constraints are known, when the execution of the system slightly deviates from the expected behaviour, due to implementation choices, previously established properties may not hold anymore. Additionally, considering a wide range of values for constants allows for a more flexible and robust design.

Introducing parameters instead of concrete values is an elegant way of addressing these three issues. Parametrization however makes verification more difficult. Besides, it raises new problems like parameter synthesis, *i.e.,* finding the set (or a subset) of values for which some property holds.

Parameters for timed models. Parametric reasoning is particularly relevant for quantitative features related to timing requirements, like network delays, time-outs, response times or clock drifts.

Pioneering work on parametric real time reasoning was presented in [5] for the now classical model of timed automata, with parameter expressions replacing the constants to be compared with clock values. There a decision procedure was designed for discrete-time models where at most one clock is compared to parameters (and other clocks are compared to constants) and undecidablity was proved for three clocks with parametrized constraints. Since then, many studies have been devoted to the parametric verification of timed models. Considering dense-time models, one clock compared to parameters leads to undecidability [6]. In [7], it is established that even removing equality constraints does not bring back decidability. Relaxing completeness requirement or guaranteed termination, several methods and tools have been developed for parametrized verification or parameter synthesis in timed automata [8, 9, 10, 11], time Petri nets [12, 13], and hybrid automata [14, 15]. Another research direction consists in defining subclasses of parametric timed models for which some problems become decidable. In [16, 17], L/U automata are defined by restricting the parameters to lower bounds (or in an exclusive way, upper bounds) in clock comparisons and reachability is proved decidable for this model. In [11], an alternative restriction is proposed: parameters values lie in a finite set of integers which leads to the standard complexity (PSPACE-completeness) of reachability for timed automata. In [18] the problem left open by [5] is solved: in the discrete-time framework the reachability problem with two clocks against parameters is decidable. Timed temporal parametrized logics have also been studied (see for instance [19] for decidability results with integer parameters). Summarizing, these subclasses are quite restricted either by the number of clocks or by the kind of available comparisons. It then remains a challenging issue to define expressive parametric timed models where reachability problems are decidable.

Contributions. Our contributions are twofold. First we define a more expressive version of ITA, including auxiliary clocks. We prove that this new model is strictly more expressive than the former one but retains decidability for the reachability problem. With respect to complexity issues, we provide upper bounds: 2EXPTIME in the general case, PSPACE when the number of levels is fixed and PTIME when the number of clocks is fixed. We also give a PSPACE matching lower bound when the number of levels is fixed.

Our second contribution is to enrich ITA with parameters in the spirit above. A PITA is a parametric version of ITA where polynomial parameter expressions can be combined with clock values both as additive and multiplicative coefficients. Considering only additive parametrization, we reduce reachability to the same problem in basic ITA. This reduction entails complexity upper bounds of respectively 2EXPTIME, PSPACE when the number of levels is fixed and PTIME when the number of clocks and parameters is fixed. The multiplicative setting is much more expressive and also very useful in practice, for instance to model clock drifts. We prove that reachability in parametric ITA is decidable as well as its robust variant, an important property for implementation issues. To the best of our knowledge, this is the first time such a result has been obtained for a model including a multiplicative parametrization. Furthermore, we establish upper bounds for the computational complexity: 2EXPSPACE and PSPACE when the number of levels is fixed. Our technique combines the construction of symbolic class automata from the unparametrized case and the first order theory of real numbers.

Outline. The model of Interrupt Timed Automata with auxiliary clocks is defined in Section 2, with reachability analysis in Section 3. The parametric ITA model is introduced in Section 4. The reachability analysis is split into two sections: the additive case is handled in Section 5 while the results for the multiplicative case are given in Section 6. We conclude and give some perspectives for this work in Section 7.

2. Interrupt timed automata

2.1. Notations

The sets of natural, rational and real numbers are denoted respectively by $\mathbb{N}$, $\mathbb{Q}$ and $\mathbb{R}$. We write $|Z|$ for the cardinality of a finite set Z. Given an alphabet Σ, we denote by Σ^* the set of finite words over Σ, with ε the empty word. A timed word over Σ is a finite sequence of the form $(a_1, t_1) \dots (a_n, t_n)$ where $a_i \in \Sigma$ for all $i \in \{1, \dots, n\}$ and $(t_i)_{1 \le i \le n}$ is a sequence of real numbers such that $t_i \le t_{i+1}$ for all $i \le n-1$. For a timed word $w = (a_1, t_1) \dots (a_n, t_n)$, we define $Untime(w) = a_1 \dots a_n$ as its projection on Σ^*. A timed language is a set L of timed words, with $Untime(L) = \{Untime(w) \mid w \in L\}$.

Given two sets F, G with F finite, we denote by $\mathcal{L}in(F, G)$ the set of linear expressions $\sum_{f \in F} a_f f + b$ where the a_f's and b belong to G. We also denote by $\mathcal{D}\mathit{iff}(F)$ the set of expressions $f - f'$ with $f, f' \in F$.

Clock constraints. Let X be a finite set of clocks and let Y, Z be disjoint subsets of X. We denote by $\mathcal{C}(Y, Z)$ the set of constraints obtained by conjunctions of atomic propositions of the form $C \bowtie 0$, where C is an expression in $\bigcup_{y \in Y} \mathcal{L}in(Z \cup \{y\}, \mathbb{Q}) \cup \mathcal{D}\mathit{iff}(Y)$ and $\bowtie \in \{>, \ge, =, \le, <\}$. Such a constraint either compares with zero a linear expression of clocks in $Y \cup Z$ including at most one clock of Y, or compares two clocks of Y. We also set $\mathcal{C}(X) = \bigcup_{Y, Z \subseteq X} \mathcal{C}(Y, Z)$.

Updates. An *update* over X is a conjunction of assignments of the form $\wedge_{y \in Y} \, y := C_y$, where $Y \subseteq X$ and $C_y \in \mathcal{L}in(X, \mathbb{Q})$. The set of updates is written $\mathcal{U}(X)$. For an expression C and an update u, the expression $C[u]$ is obtained by "applying" u to C, *i.e.*, simultaneously substituting each x by C_x in C, if $x := C_x$ is the update for x in u. For instance, for clocks $X = \{x_1, x_2\}$, expression $C = 2x_2 - 2x_1 + 3$

and the update u defined by $x_1 := 1 \wedge x_2 := 3x_1 + 2$, applying u to C yields the expression $C[u] = 2(3x_1 + 2) - 2(1) + 3 = 6x_1 + 5$.

Valuations. A *clock valuation* is a mapping $v : X \mapsto \mathbb{R}$, with $\mathbf{0}$ the valuation where all clocks have value 0. For a valuation v and an expression $C \in \mathcal{L}in(X, \mathbb{Q})$, we note $v(C) \in \mathbb{R}$ the result of evaluating C w.r.t. v. Given an update u and a valuation v, the valuation $v[u]$ is defined by $v[u](x) = v(x)$ if x is unchanged by u and $v[u](x) = v(C_x)$ if $x := C_x$ is the update for x in u. For instance, let $X = \{x_1, x_2, x_3\}$ be a set of three clocks. For valuation $v = (2, 1.5, 3)$ and update u defined by $x_1 := 1 \wedge x_3 := x_3 - x_1$, applying u to v yields the valuation $v[u] = (1, 1.5, 1)$.

2.2. Interrupt Timed Automata

Definitions. The behaviour of an ITA can be viewed as the one of an operating system with interrupt levels. With each level are associated a set of states and a set of clocks partitioned into a *main* clock and *auxiliary* clocks. In a state of a given level, exactly one clock of this level is active (rate 1), while the other clocks at lower or equal levels are suspended (rate 0), and the clocks at higher levels are not yet activated and thus contain value 0. The enabling conditions on transitions, called *guards*, are constraints over clocks of the current level or main clocks of lower levels (with some restrictions that will be explained after the definition). Transitions can *update* the clock values. If the transition decreases (resp. increases) the level, then each clock which is relevant after (resp. before) the transition can (1) be left unchanged, (2) be updated with a linear expression of main clocks of strictly lower levels or (3) be updated with another clock at the same level (with some restrictions). Roughly speaking, the restrictions are introduced to forbid at some level any (direct or indirect) influence of the auxiliary clocks at lower levels on the behaviour of the ITA.

Definition 2.1. An *interrupt timed automaton* (ITA) is a tuple $\mathcal{A} = \langle \Sigma, n, Q, q_0, Q_f, \lambda, X, \mathtt{act}, \Delta \rangle$, where:

- Σ is a finite alphabet;
- n is the number of levels;
- Q is a finite set of states, q_0 is the initial state and Q_f is a subset of Q of final states. The mapping $\lambda : Q \to \{1, \ldots, n\}$ associates with each state its level. We denote by $Q_i = \lambda^{-1}(i)$ the set of states at level i;
- $X = \biguplus_{i=1}^{n} X_i$ is the set of clocks partitioned according to the levels and $X_i = \{x_i\} \uplus Y_i$ includes a *main* clock x_i and a set of *auxiliary* clocks Y_i. The set of main clocks of levels less than k is denoted by $X^{\mathtt{m}}_{<k} = \{x_i \mid i < k\}$;
- $\mathtt{act} : Q \to X$ with $q \in Q_i \Rightarrow \mathtt{act}(q) \in X_i$ associates with a state its *active* clock;
- $\Delta \subseteq Q \times \mathcal{C}(X) \times (\Sigma \cup \{\varepsilon\}) \times \mathcal{U}(X) \times Q$ is a finite set of transitions. Let $q \xrightarrow{\varphi, a, u} q'$ be a transition in Δ with $k = \lambda(q)$ and $k' = \lambda(q')$. The guard φ is a constraint in $\mathcal{C}(X_k, X^{\mathtt{m}}_{<k})$.
  - if $k \leq k'$ then the update u is of the form

$$\bigwedge_{z \in \bigcup_{i \leq k} X_i} z := C_z$$

- if $k > k'$ then the update u is of the form

$$\bigwedge_{z \in \bigcup_{i \leq k'} X_i} z := C_z \ \wedge \bigwedge_{z \in \bigcup_{k' < i \leq k} X_i} z := 0$$

where, when $z \in X_i$,

- either $C_z = z$, meaning that z is unchanged;
- or $C_z = \sum_{j<i} a_j x_j + b$, *i.e.*, z is updated by an expression over main clocks of lower levels;
- or $C_z = z' \in X_i$ if either $z \in Y_i$ or $i = k = k'$. While an auxiliary clock can be updated by another clock of the same level, for the main clock this is only possible when the level of both states of the transition is the level of the clock.

The interest of distinguishing main clocks from auxiliary clocks lies in the possibility of referencing a main clock at higher levels whereas auxiliary clocks are only used at the current level. Referencing multiple clocks of a same level prevents the design of a saturation procedure for reachability, as defined later on. In addition, in order to avoid indirect references in updates, we forbid that a main clock at a level lower than the current one may be updated by an auxiliary clock.

Example 2.2. In Figure 1, a part of an ITA is represented with all states at level 2, main clocks x_1 and x_2 at levels 1 and 2 respectively and auxiliary clock y_1 at level 1. The upper transition from q_1 to q_2 (drawn as a dashed arrow), is forbidden since it updates the main clock x_1 of level 1 with an auxiliary clock of the same level. If we would allow such a transition, the firing of the next transition would depend on an indirect relation between x_2 and y_1. We will later show on this example when the construction of the class automaton would fail. □

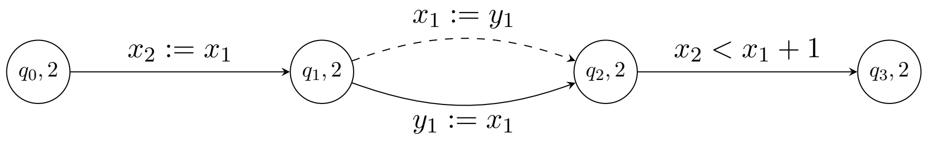


Figure 1. Restrictions on clock updates in ITA

The semantics of an ITA is described by a transition system, where a configuration (q, v) consists of a state q of the ITA and a clock valuation v.

Definition 2.3. The semantics of an ITA $\mathcal{A}$ is defined by the (timed) transition system $\mathcal{T}_{\mathcal{A}} = (S, s_0, \to)$. The set of configurations is $S = \{(q, v) \mid q \in Q,\ v \in \mathbb{R}^X\}$, with initial configuration $s_0 = (q_0, \mathbf{0})$. The relation $\to$ on S consists of two types of steps:

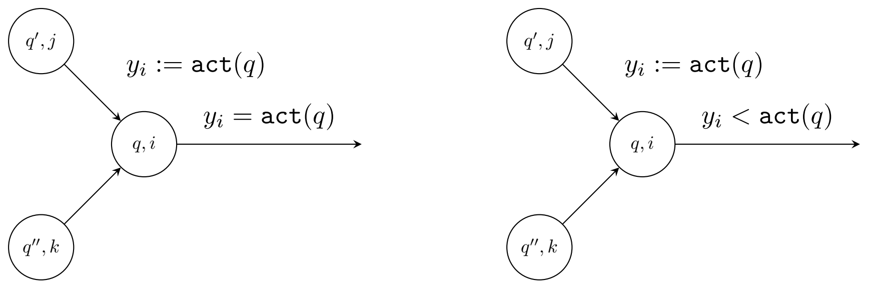
Time steps: Only the active clock in a state can evolve, all other clocks are suspended. For a state q, a time step of duration d is defined by $(q, v) \xrightarrow{d} (q, v')$ with $v'(\texttt{act}(q)) = v(\texttt{act}(q)) + d$ and $v'(x) = v(x)$ for any other clock x. We write $v' = v +_q d$.

Discrete steps: A discrete step $(q, v) \xrightarrow{e} (q', v')$ can occur for some transition $e = q \xrightarrow{\varphi,a,u} q'$ in Δ such that $v \models \varphi$ and $v' = v[u]$.

A *run* of $\mathcal{A}$ is a finite path in the transition system $\mathcal{T}_{\mathcal{A}}$, which can be written as an alternating sequence of (possibly null) time and discrete steps. A state $q \in Q$ is *reachable* from q_0 if there is a path in $\mathcal{T}_{\mathcal{A}}$ from $(q_0, \mathbf{0})$ to (q, v), for some valuation v. A run with label $d_1 a_1 d_2 a_2 \dots d_n a_n$ is *accepting* if it starts in $(q_0, \mathbf{0})$ and ends in (q, v), for some $q \in Q_f$ and some valuation v. For such a run, the timed word $w = (a_1, d_1)(a_2, d_1 + d_2) \dots (a_n, d_1 + \dots + d_n)$ (where pairs with ε actions are removed) is said to be *accepted* by $\mathcal{A}$. The timed language of $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of timed words accepted by $\mathcal{A}$. The untimed language of $\mathcal{A}$ is $Untime(\mathcal{L}(\mathcal{A}))$.

We now show several properties of this model related to the presence of auxiliary clocks.

Example 2.4. (Simulation of timing policies)
The earlier definition of ITA from [4] is a restriction of Definition 2.1 without auxiliary clocks but where a policy, which can be either urgent, delayed or lazy, is associated with each state. In a lazy state time may elapse, in an urgent state time may not elapse and in a delayed state time must elapse. We show in Figure 2 how to model timing policies with a dedicated auxiliary clock per level, say y_i. When entering a state q of level i from a state q' of level $j \geq i$, the auxiliary clock y_i is updated with the value of the active clock of q. By definition, when entering a state q of level i from a state q'' of level $k < i$, y_i and the active clock of q are null. Thus checking whether time has elapsed in q is equivalent to check whether $\mathtt{act}(q) > y_i$ (Figure 2(b)), both values must be equal otherwise (Figure 2(a)). When q is a lazy state there is nothing to check. $\square$



(a) For urgent state q, with $k < i \leq j$ (b) For delayed state q, with $k < i \leq j$

Figure 2. Simulating timing policies

Example 2.5. (About expressiveness)
Consider the ITA $\mathcal{A}_1$ of Figure 3 with a single level and single final state q_2. The main clock x is active in all states and y is an auxiliary clock. Its untimed language is $(ab)^+$. In the accepted timed words, there is an occurrence of a at each time unit and the successive occurrences of b come each time closer to the next occurrence of a than previously. More formally, its timed language $L = \mathcal{L}(\mathcal{A}_1)$ is defined by:

$$\begin{aligned} L = \big\{(a, t_1)(b, t_2) \quad \dots \quad & (a, t_{2p+1})(b, t_{2p+2}) \mid p \in \mathbb{N}, \\ & \forall 0 \leq i \leq p,\ t_{2i+1} = i + 1 \text{ and } i + 1 < t_{2i+2} < i + 2, \\ & \forall 1 \leq i \leq p,\ t_{2i+2} - t_{2i+1} < t_{2i} - t_{2i-1}\big\} \end{aligned}$$

It has been shown in [4] that this timed language cannot be accepted by an ITA without auxiliary clocks, which yields the next proposition. □

Proposition 2.6. There exists a timed language of an ITA with a single level and one auxiliary clock that cannot be accepted by an ITA without auxiliary clocks.

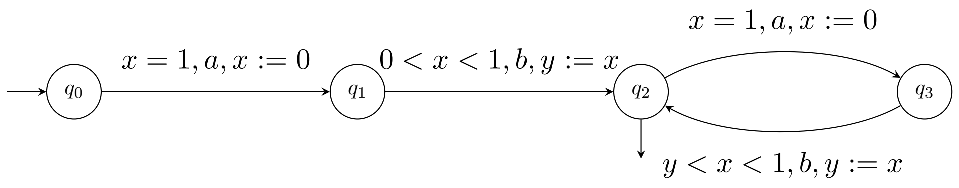


Figure 3. ITA $\mathcal{A}_1$ with an auxiliary clock

Adding auxiliary clocks also has an impact on the complexity of decision problems for ITA. In [4], it is shown that the state reachability problem is in PTIME for a fixed number of levels without auxiliary clocks. The next proposition establishes a lower bound for this problem in ITA with a single level.

Proposition 2.7. The state reachability problem for ITA with a single level is PSPACE-hard.

Proof:
We proceed by reducing the planification problem to our reachability problem. The planification problem is defined by n propositional variables $p_1, \dots, p_n$ and a set R of m rules. Each rule $r \in R$ is defined by a guard $\bigwedge_{j=1}^{k} \ell_j$, with literals $\ell_j \in \{p_1, \neg p_1, \dots, p_n, \neg p_n\}$, and an update $\bigwedge_{j=1}^{h} p_{\alpha_j} := b_j$ with $b_j \in \{\mathbf{false}, \mathbf{true}\}$. Initially all propositions are false and the planification problem consists in deciding whether there exists a sequence of rules $r_1 \dots r_k$ applicable from the initial state and leading to the state where all propositions are true. This problem is PSPACE-complete [20].
The corresponding ITA has n auxiliary clocks $y_1, \dots, y_n$ and two states q_0, which is the initial state, and q_1, which is the final state, both with active clock x_1. Each rule yields a transition looping around q_0 and an additional transition from q_0 to q_1 "checking" that the goal has been reached. This reduction is illustrated in Figure 4. □

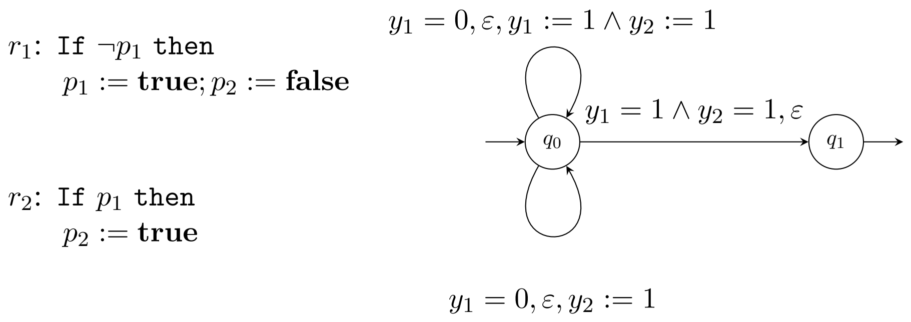

r_1: If $\neg p_1$ then
$p_1 := \mathbf{true}; p_2 := \mathbf{false}$
r_2: If p_1 then
$p_2 := \mathbf{true}$
$y_1 = 0, \varepsilon, y_1 := 1 \wedge y_2 := 1$
$y_1 = 1 \wedge y_2 = 1, \varepsilon$
q_0
q_1
$y_1 = 0, \varepsilon, y_2 := 1$

Figure 4. Illustrating the reduction for PSPACE-hardness

3. Reachability analysis of ITA

We prove in this section that the untimed language of an ITA is a regular language for which a finite automaton can effectively be built. Similarly to previous cases, the proof is based on the construction of a (finite) class graph which is time abstract bisimilar to the transition system $\mathcal{T}_{\mathcal{A}}$. This result also holds for infinite words with standard Büchi conditions. As a consequence, we obtain decidability of the reachability problem, as well as decidability for plain CTL^* model-checking.

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). Given an ITA $\mathcal{A}$ with n levels, we associate with each level k of $\mathcal{A}$ a set of expressions E_k. In addition to X_k, expressions in E_k only use main clocks of lower levels (that are frozen) as their values will be needed to be compared with the active clock of level k, to detect if a transition can be fired at this level. A class will consist of a state q and for all $k \leq \lambda(q)$ a total preorder on E_k. Finally we show in Theorem 3.6 how to build the class graph which proves the regularity of the untimed language. This immediately yields a reachability procedure given in Theorem 3.7.

3.1. Construction of $\{E_k\}_{k\leq n}$

We first recall the *normalization* operation [4], on expressions relative to some level. As explained below, this operation will be used to order expression values at a given level.

Definition 3.1. (Normalization)
Let $k \leq n$ and $C = \sum_{i\leq k} a_i x_i + b$ be an expression over clocks in $X^{\mathtt{m}}_{<k+1}$, the *k-normalization* of C, denoted by $\mathtt{norm}(C,k)$, is defined by:

$$\mathtt{norm}(C,k) = \begin{cases} x_k + (1/a_k)(\sum_{i<k} a_i x_i + b) \text{ if } a_k \neq 0 \\ C \text{ otherwise.} \end{cases}$$

Observe that k-normalization only operates on expressions where no clocks of levels higher than k can be used. Let $C \bowtie 0$ be a guard occurring in a transition outgoing from a state q with level k and $C = a_k z + \sum_{i<k} a_i x_i + b$ with $z \in X_k$ (in the saturation procedure we do not consider guards of the form $z - z'$ with z, z' in X_k). By rescaling the expression and if necessary changing the comparison operator we may assume that C is written as $\alpha z + \sum_{i<k} a_i x_i + b$, with $\alpha \in \{0,1\}$.

The construction of $\{E_k\}_{k\leq n}$ must be adapted to handle auxiliary clocks. It proceeds top down from level n to level 1 after initialization $E_k = X_k \cup \{0\}$ for all k. When level k is handled, new terms are added to E_i for $1 \leq i \leq k$. These expressions are those needed to compute a (pre)order on the expressions in E_k.

1. At level k, first for each expression $\alpha z + \sum_{i<k} a_i x_i + b$ (with $\alpha \in \{0,1\}$ and $z \in X_k$) 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.

2. Then the following procedure is iterated until no new term is added to any E_i for $1 \leq i \leq k$.

(a) Let $q \xrightarrow{\varphi,a,u} q'$ with $\lambda(q) \geq k$ and $\lambda(q') \geq k$. For any $C \in E_k$, we add $C[u]$ to E_k. Observe that due to our restrictions on updates $C[u]$ is still either of the form $z \in X_k$ or of the form $\sum_{j<k} a_j x_j + b$.

(b) Let $q \xrightarrow{\varphi,a,u} q'$ with $\lambda(q) < k$ and $\lambda(q') \geq k$. Let C and C' be two different expressions in E_k. We compute $C'' = \mathtt{norm}(C[u] - C'[u], \lambda(q))$, choosing an arbitrary order between C and C' in order to avoid redundancy. 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 $E_{\lambda(q)}$.

Note that in step 2.(b), while C and C' belong to E_k, due to the update u, only clocks of levels less than or equal to $\lambda(q)$ occur in $C[u]$ and $C'[u]$. Hence the normalization is well defined.

Lemma 3.2. For an ITA $\mathcal{A}$, let H be the number of constraints in the guards, U the number of updates in the transitions (we assume $U \geq 2$) and $M = \max\{|X_k| + 1 \mid 1 \leq k \leq n\}$. The construction procedure of $\{E_k\}_{k \leq n}$ terminates and the size of every E_k is bounded by $(H+M)^{2^{n-k}} \times U^{2^{n(n-k+1)}}$.

Proof:
Given some k, we prove the termination of the stage relative to k. Observe that step 2.(b) of the iteration only adds new expressions to E_h for $h < k$. Thus steps 2.(a) and 2.(b) can be ordered. Let us prove the termination of step 2.(a). We define E_k^0 as the set E_k at the beginning of this stage and E_k^i as this set after 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 finiteness of the edges) and finitely many roots. 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 updates of the form $x := x'$ do not modify the current set E_k^i since x' is one of the roots of the forest. Moreover, the number of updates that change the variables $x \in X_k$ is either 0 or 1 since once x disappears it cannot appear again (due to the use of clocks from strictly lower levels only). 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 reasoning that there is at most one update that change the variables $x \in X_{k-1}$. 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.

The final size of E_k is thus at most $|E_k^0| \times U^{2^k}$ since the width of the forest is bounded by U. In step 2.(b), we add at most $U \times (|E_k| \times (|E_k| - 1))/2$ expressions to E_i for every $i < k$. This concludes the proof of termination.

We now prove by a backward induction that as soon as $n \geq 2$, $|E_k| \leq (H+M)^{2^{n-k}} \times U^{2^{n(n-k+1)}}$. The doubly exponential size of E_n (proved above) is propagated downwards by the saturation procedure. We define $p_k = |E_k|$.

Basis case $k = n$. We have $p_n \leq p_n^0 \times U^{2^n}$ where p_n^0 is bounded by $H+M$, hence $p_n \leq (H+M) \times U^{2^n}$ which is the claimed bound.

Inductive case. Assume that the bound holds for $k < j \leq n$. After step 1., E_k contains $H+M$ expressions and, due to the expressions added by all executions of step 2.(b) of the procedure at strictly higher levels, we have after this step:

$$\begin{array}{rcl} p_k^0 & \leq & (H+M)+U\times[(p_{k+1}\times(p_{k+1}-1))/2+\cdots+(p_n\times(p_n-1))/2] \\ p_k^0 & \leq & (H+M)+U\times\left[(H+M)^{2^{n-k}}U^{2^{n(n-k)+1}}+\cdots+(H+M)^2U^{2^{n+1}}\right] \\ & & \text{(replacing all terms } x(x-1)/2 \text{ by } x^2/2) \\ p_k^0 & \leq & (n-k+1)\times(H+M)^{2^{n-k}}U^{2^{n(n-k)+1}} \quad \text{(replacing all terms by the largest)} \\ p_k^0 & \leq & (H+M)^{2^{n-k}}\times U^{2^{n(n-k+1)+n}} \quad \text{(here we use } U\geq 2 \text{ and } n\geq 2). \end{array}$$

Taking into account step 2.(a) of the procedure for level k, we have:

$$p_k \leq (H+M)^{2^{n-k}}\times U^{2^{n(n-k)+1}+2^k+n}.$$

Let us consider the term $\delta = 2^{n(n-k+1)} - 2^{n(n-k)+1} - 2^k - n = 2^{n(n-k)+1}(2^{n-1}-1) - 2^k - n$. We have $\delta \geq 2^{n+1} - 2^n \geq 0$, which yields the claimed bound. □

In order to analyze the space requirements triggered by the saturation procedure, we establish the following lemma bounding the number of bits used for integers involved in the rational constants of expressions in all E_k.

Lemma 3.3. Let $\mathcal{A}$ be an ITA, and let b_0 be the maximal number of bits for integers occurring in $\mathcal{A}$. If b is the number of bits of an integer constant, occurring in an expression of some E_k, then $b \leq ((n+1)!)^2 9^n b_0$.

Proof:
Without loss of generality we assume that $b_0 \geq 2$. Since it only induces a polynomial blow up, we also assume that there is a single denominator, denoted by s, for the rationals occurring in updates.
Let b_k be the number of bits of an integer occurring in some expression before operations of level $n-k$ are performed. We establish a relation between b_k and b_{k+1}. At level $n-k$, step 1 involves a normalization on guards. Thus a numerator is multiplied by a denominator to produce the new integers leading to a number of bits $2b_k$. For an expression that was already present in E_{n-k}, its coefficients are modified in order to get a common denominator by taking the product of the original denominators. After this transformation the maximal number of bits is bounded by $(n-k+1)b_k$.
Let $C = \sum_{i\leq n-k} a_i x_i + b$ be an expression built after step 2.(a). Examining the successive updates, the coefficient a_i can be expressed as $\sum_{d\in\mathcal{D}}\prod_{j\in d} c_{d,j}$ where $\mathcal{D}$ is the set of subsets of $\{i,\ldots,n-k\}$ containing i and $c_{d,j}$ are either coefficients of the updates or coefficients of an expression built before this step. The same reasoning applies to b. Before summing the products over $d\in\mathcal{D}$, the integers are transformed in order to get the same denominator by multiplying every denominator (and corresponding numerator) by s^i with $0\leq i\leq n-k$. So the maximal absolute value of the numerator of such a coefficient is bounded by $2^{n-k}(2^{(n-k+1)b_k})^{n-k+1}2^{(n-k)b_0} \leq (2^{2b_k+1})^{(n-k+1)^2}$ which implies a maximal number of bits equal to $(n-k+1)^2(2b_k+1)$ for the numerators of the a_i's and b. The maximal absolute value of the denominator of such a coefficient is less than $(2^{(n-k+1)b_k})^{n-k+1}2^{(n-k)b_0}$ which implies a maximal number of bits bounded by $(n-k+1)^2(2b_k)$ for the denominators of the a_i's and b.
At step 2.(b), the difference $C[u]-C'[u]$ requires to compute the lcm of two denominators (bounded by their product). So the difference operation leads to a bound $(n-k+1)^2(4b_k+2)$ for the numerators of its coefficients and $(n-k+1)^2(4b_k)$ for the denominators.

The final step 2.(b) consists in multiplying a numerator and a denominator of some coefficients leading to a bound $(n-k+1)^2(8b_k+2) \leq (n-k+1)^2(9b_k)$ for b_{k+1}, which yields the desired bound. $\square$

3.2. Construction of the Class Automaton

In order to analyze the size of the class automaton defined below, we recall an adaptation of a classical result about partitions of n-dimensional Euclidean spaces.

Definition 3.4. Let $\{H_k\}_{1\leq k\leq m}$ be a family of hyperplanes of $\mathbb{R}^n$. A *region* defined by this family is a connected component of $\mathbb{R}^n \setminus \bigcup_{1\leq k\leq m} H_k$. An *extended region* defined by this family is a connected component of $\bigcap_{k\in I} H_k \setminus \bigcup_{k\notin I} H_k$ where $I \subseteq \{1,\ldots,m\}$ with the convention that $\bigcap_{k\in\emptyset} H_k = \mathbb{R}^n$.

Proposition 3.5.
1. [21] The number of regions defined by the family $\{H_k\}_{1\leq k\leq m}$ is at most $\sum_{i=0}^{n}\binom{m}{i}$.
2. [4] The number of extended regions defined by the family $\{H_k\}_{1\leq k\leq m}$ is at most:
$\sum_{p=0}^{n}\binom{m}{p}\sum_{i=0}^{n-p}\binom{m-p}{i} \leq e^2 m^n$.

Theorem 3.6. The untimed language of an ITA is regular.

Proof:
Starting from an ITA $\mathcal{A}$, and handling auxiliary clocks, we build a finite automaton which is time abstract bisimilar to the transition system $\mathcal{T}_{\mathcal{A}}$ and thus accepts $Untime(\mathcal{L}(\mathcal{A}))$.

Class definition. A state of the automaton, called class, is a syntactical representation of a subset of reachable configurations. 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, for $1 \leq k \leq \lambda(q)$. The class R describes the set of configurations:

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

The initial state is the class R_0 such that $[\![R_0]\!]$ contains $(q_0, \mathbf{0})$ and can be straightforwardly determined. The final states are all classes $R = \big(q, \{\preceq_k\}_{1\leq k\leq \lambda(q)}\big)$ with $q \in Q_f$.

Observe that fixing a state, the set of configurations $[\![R]\!]$ of a non empty class R is exactly an extended region associated with the hyperplanes defined by the comparison of two expressions of some E_k. From Lemma 3.2, adding all $|E_k|$'s gives an upper bound of $(H+M)^{2^n} \times U^{2^{n^2}}$ for the total number of expressions of any level. Hence an upper bound of the number of hyperplanes is obtained by squaring this number, yielding $(H+M)^{2^{(n+1)}} \times U^{2^{(n^2+1)}}$. Using Point 2. of Proposition 3.5 with this value for m, the number of semantically different classes for a given state is bounded by:

$$e^2 m^n = e^2 (H+M)^{K2^{n+1}} \times U^{K2^{n^2+1}} \tag{1}$$

where $K = \sum_{k=1}^{n} |X_k| \leq nM$ is the total number of clocks. Since semantical equality between classes can be tested in polynomial time w.r.t. their size [22], we implicitly consider in the sequel of the proof classes modulo the semantical equivalence.

There are two kinds of transitions, corresponding to discrete steps and abstract 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 $(q, v) \xrightarrow{e} (q', v')$. We first show how the firability condition can be decided by examination of R and then how R' is defined when the condition is satisfied.

firability condition. For a transition e like above at level $\ell = \lambda(q)$, write $\varphi = \bigwedge_{j\in J} C_j \bowtie_j 0$. Since we assumed rescaled guards, for every j, $C_j = \alpha z + \sum_{i<k} a_i x_i + b$ (with $\alpha \in \{0, 1\}$ and z in X_ℓ) or $C_j = z - z'$ with $z, z' \in X_\ell$. In the first case $C'_j = -\sum_{i<\ell} a_i x_i - b$ and z belong to E_ℓ and in the second case $z, z' \in E_\ell$ both by construction. For each $j \in J$, we define a condition depending on $\bowtie_j$. For instance, in the first case if the constraint in φ is $C_j \leq 0$, we check that $\alpha z \preceq_\ell C'_j$, or if the constraint in φ is $C_j > 0$ we check that $\alpha z \not\preceq_\ell C'_j \wedge C'_j \preceq_\ell \alpha z$. The second case is handled similarly.

Successor definition. To define class R' we take into account both the current preorders and the update of the transition that induces the "real" expressions to be compared. Let $k \leq \lambda(q')$ and $g, h \in E_k$.

1. Either $k \leq \ell$, then by construction, $g[u], h[u] \in E_k$ then $g \preceq'_k h$ iff $g[u] \preceq_k h[u]$.

2. Or $k > \ell$, let $D = g[u] - h[u]$. Due to our restrictions on updates for $i \leq \ell$, $x_i[u]$ can only be equal to x_i or $\sum_{j<i} \alpha_j x_j + \beta$. Thus D can be written as $\sum_{i\leq\ell} c_i x_i + d$. We set $C = \mathtt{norm}(D, \ell)$ and write $C = \alpha x_\ell + \sum_{i<\ell} a_i x_i + b$ (with $\alpha \in \{0, 1\}$). By construction, $C' = -\sum_{i<\ell} a_i x_i - b \in E_\ell$.
   When $c_\ell \geq 0$ then $g \preceq'_k h$ iff $\alpha x_\ell \preceq_\ell C'$.
   When $c_\ell < 0$ then $g \preceq'_k h$ iff $C' \preceq_\ell \alpha x_\ell$.

By definition of $[\![\cdot]\!]$, we obtain:

- For any $(q, v) \in[\![R]\!]$, 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 for each $(q, v) \in[\![R]\!]$ there exists $(q', v') \in [\![R']\!]$ such that $(q, v) \xrightarrow{e} (q', v')$.

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

For every $i < \ell$, we define $\preceq'_i = \preceq_i$. Let $\sim$ be the equivalence relation $\preceq_\ell \cap \preceq_\ell^{-1}$ induced by the preorder. On equivalence classes, this (total) preorder becomes a (total) order. Let V be the equivalence class containing $\mathtt{act}(q)$.

1. Either $V = \{\mathtt{act}(q)\}$ and it is the greatest equivalence class. Then $\preceq'_\ell = \preceq_\ell$ (thus $Post(R) = R$).

2. Either $V = \{\mathtt{act}(q)\}$ and it is not the greatest equivalence class. Let V' be the next equivalence class. Then $\preceq'_\ell$ is obtained by merging V and V', and preserving $\preceq_\ell$ elsewhere.

3. Either V is not a singleton. Then we split V into $V \setminus \{\mathtt{act}(q)\}$ and $\{\mathtt{act}(q)\}$ and "extend" $\preceq_\ell$ by $V \setminus \{\mathtt{act}(q)\} \preceq'_\ell \{\mathtt{act}(q)\}$.

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

From the properties above, this finite automaton accepts $Untime(\mathcal{L}(\mathcal{A}))$. □

Observe that if in Example 2.2, we add the forbidden transition then the saturation procedure will produce the expression $y_1 - x_1 - 1$ in E_1 corresponding to the actual test for firing the rightmost transition. Unfortunately, the preorder of level 1 is just able to compare x_1 to y_1 but not to $y_1 + 1$.

Theorem 3.7. The reachability problem for Interrupt Timed Automata is decidable and belongs to 2EXPTIME. It is in PTIME when the number of clocks is fixed and PSPACE-complete when the number of levels is fixed.

Proof:
The reachability problem is solved by building the class graph and applying a standard reachability algorithm. The number of expressions in the E_k's is doubly exponential w.r.t. the size of the model (see Lemma 3.2). The size of an expression is exponential w.r.t. the size of the model (see Lemma 3.3). So the size of a class representation is also doubly exponential in the size of the model. The size of the graph, bounded by the number of semantically different classes, is only polynomial w.r.t. the size of a class due to Point 2. of Proposition 3.5. This leads to a 2EXPTIME complexity. Observe that no complexity gain can be obtained by a nondeterministic search without building the graph.
Again using these lemmas and Point 2. of Proposition 3.5, when the number of clocks is fixed the size of the graph is at most polynomial in the size of the problem, leading to a PTIME procedure.
On the other hand, when the number of levels is fixed, the size of a class representation is polynomial while the number of classes is exponential (see K in Equation (1)). Thus a nondeterministic search can be performed without building the graph, which yields a complexity in PSPACE. The PSPACE hardness is a consequence of Proposition 2.7. □

Remarks. This result should be compared with the similar one for TA. The reachability problem for TA is PSPACE-complete and thus less costly to solve than for ITA. Fixing the number of levels in ITA yields the same complexity. Moreover, fixing the number of clocks does not reduce the complexity for TA (when this number is greater than or equal to 3) while this problem belongs now to PTIME for ITA. Summarizing, the main source of complexity for ITA is the number of levels and clocks, while in TA it is the binary encoding of the constants [23].

4. Parametric interrupt timed automata

Parametric ITA are similar to ITA but they include polynomials of parameters from a set P, in guards and updates. Given two sets F, G, we denote by $\mathcal{P}ol(F, G)$, the set of polynomials with variables in F and coefficients in G and by $\mathcal{F}rac(F, G)$, the set of rational functions with variables in F and coefficients in G (i.e. quotients of polynomials). Observe that $\mathcal{L}in(F, G)$ can be seen as the subset of polynomials with degree at most one.

Definition 4.1. A *parametric interrupt timed automaton* (PITA) is a tuple $\mathcal{A} = \langle P, \Sigma, n, Q, q_0, Q_f, \lambda, X, \mathtt{act}, \Delta \rangle$, where:

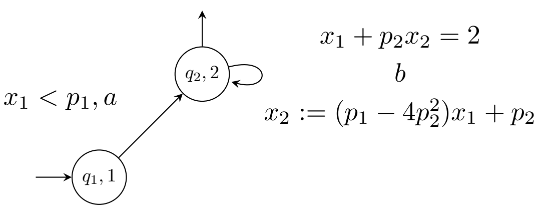
- P is a finite set of parameters,
- all other elements are defined as for ITA except that expressions appearing in guards or updates belong to $\mathcal{L}in(X, \mathcal{P}ol(P, \mathbb{Q}))$: in such an expression $\sum_{z \in Z} a_z z + b$, the a_z's and b are polynomials over P with coefficients in $\mathbb{Q}$.

So an ITA is a PITA with $P = \emptyset$. When all expressions occurring in guards and updates are in $\mathcal{L}in(X \cup P, \mathbb{Q})$ (which can be seen as a subset of $\mathcal{L}in(X, \mathcal{P}ol(P, \mathbb{Q}))$), the PITA is said to be *additively parametrized*. In contrast, in the general case, it is called *multiplicatively parametrized*.

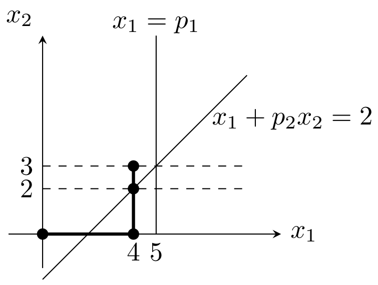
As in the unparametrized case, updates operate on expressions. For instance, for clocks in $X = \{x_1, x_2\}$, parameters in $P = \{p_1, p_2, p_3\}$, expression $C = p_2x_2 - 2x_1 + 3p_1$ and the update u defined by $x_1 := 1 \wedge x_2 := p_3x_1 + p_2$, applying u to C yields the expression $C[u] = p_2p_3x_1 + p_2^2 + 3p_1 - 2$. Note that the use of multiplicative parameters for clocks may result in polynomial coefficients when updates are applied. Here a *clock valuation* is a mapping $v : X \mapsto \mathcal{P}ol(P, \mathbb{R})$. For a valuation v and an expression $C \in \mathcal{L}in(X, \mathcal{P}ol(P, \mathbb{Q}))$, $v(C) \in \mathcal{P}ol(P, \mathbb{R})$ is obtained by evaluating C w.r.t. v. Given an update u and a valuation v, the valuation $v[u]$ is defined by $v[u](x) = v(C_x)$ for x in X if $x := C_x$ is the update for x in u and $v[u](x) = v(x)$ otherwise. For instance, let $X = \{x_1, x_2, x_3\}$ be a set of three clocks. For valuation $v = (2p_2, 1.5, 3p_1^2)$ and update u defined by $x_1 := 1 \wedge x_3 := p_1x_3 - x_1$, applying u to v yields the valuation $v[u] = (1, 1.5, 3p_1^3 - 2p_2)$.

A *parameter valuation* is a mapping $\pi : P \mapsto \mathbb{R}$. For a parameter valuation π and an expression $C \in \mathcal{L}in(X, \mathcal{P}ol(P, \mathbb{Q}))$, $\pi(C) \in \mathcal{L}in(X, \mathbb{R})$ is obtained by evaluating C w.r.t. π. If $C \in \mathcal{P}ol(P, \mathbb{Q})$, then $\pi(C) \in \mathbb{R}$. Given a parameter valuation π, a clock valuation v and an expression $C \in \mathcal{L}in(X, \mathcal{P}ol(P, \mathbb{Q}))$ we write $\pi, v \models C \bowtie 0$ when $\pi(v(C)) \bowtie 0$.

Given a parameter valuation π and a PITA $\mathcal{A}$, substituting the parameters by their value according to π yields an ITA, denoted by $\mathcal{A}(\pi)$, where the coefficients of clocks are in $\mathbb{R}$. So the semantics of $\mathcal{A}$ w.r.t. parameter valuation π is defined by the (timed) transition system $\mathcal{T}_{\mathcal{A}(\pi)}$. A state q is reachable from q_0 for valuation π if q is reachable from q_0 in $\mathcal{A}(\pi)$.



(a) A PITA $\mathcal{A}_2$ with two interrupt levels



(b) A possible run in $\mathcal{A}_2$ for π

Figure 5. An example of PITA and a possible execution

Example 4.2. A PITA $\mathcal{A}_2$ is depicted in Figure 5(a), with two interrupt levels. Every level i has only a main clock x_i. Fixing the parameter valuation π: $p_1 = 5$ and $p_2 = -1$, the run $(q_1, 0, 0) \xrightarrow{4} (q_1, 4, 0) \xrightarrow{a} (q_2, 4, 0) \xrightarrow{3} (q_1, 4, 2) \xrightarrow{b} (q_2, 4, 3)$ is obtained as follows. After staying in q_1 for 4 time units, a can be fired and the value of x_1 is then frozen in state q_2, while x_2 increases. Transition b can be taken if $x_1 + p_2x_2 = 2$, hence for $x_2 = 2$, after which x_2 is updated to $x_2 = (p_1 - 4p_2^2)4 + p_2 = 3$. A geometric view of this run w.r.t. π is given (in bold) in Figure 5(b). □

Reachability problems. We consider several reachability problems for this class. Let $\mathcal{A}$ be a PITA with initial state q_0 and q be a state of $\mathcal{A}$. The *Existential (resp. Universal) Reachability Problem* asks whether q is reachable from q_0 for some (resp. all) parameter valuation(s). *Scoped* variants of these problems are obtained by adding as input a set of parameter valuations given by a first order formula over the reals or a polyhedral constraint. The *Robust Reachability Problem* asks whether there exists a parameter valuation π and a real $\varepsilon > 0$ such that for all π' with $\|\pi - \pi'\|_\infty < \varepsilon$, q is reachable from q_0 for π' (where $\|\pi\|_\infty = max_{p \in P}|\pi(p)|$). When satisfied, this property ensures that small parameter perturbations do not modify the reachability result. It is also related to parameter synthesis where a valuation has to be enlarged to an open region with the same reachability goal.

5. Reachability analysis with additive parametrization

We start with the easier particular case of additive parametrization, *i.e.*, expressions occurring in guards and updates are linear expressions on clocks and parameters with rational coefficients. We first prove that the existential parametrized reachability problem can be reduced to the reachability problem on (non-parametrized) ITA.

Proposition 5.1. For any additively parametrized PITA $\mathcal{A}$, with set of states Q and initial state q_0, there exists a (non-parametrized) ITA $\mathcal{A}'$, with set of states Q', containing Q, and initial state q'_0 fulfilling the following equivalence. For every $q \in Q$:
there exists π such that q is reachable from q_0 in $\mathcal{A}$ for π iff q is reachable from q'_0 in $\mathcal{A}'$.

Proof:
For any additively parametrized PITA $\mathcal{A}$ with n levels, and k parameters $p_1, ..., p_k$, we build an equivalent ITA $\mathcal{A}'$ with $n+k+1$ levels and then use the complexity results of section 3. The construction is shown in Figure 6.
The ITA $\mathcal{A}'$ consists of a "prefix" (the first $k+1$ levels) connected to the original automaton $\mathcal{A}$ (with its n levels). The main clocks of levels 1 to k encode the parameters $p_1, \ldots, p_k$ of $\mathcal{A}$. In order to simplify further references, we also call these clocks $p_1, ..., p_k$. Similarly, the main clock of the first level is called p_0. None of these $k+1$ first level has any auxiliary clock. Since level numbers start at 1, each clock p_i is active in level $i+1$ in (the prefix of) $\mathcal{A}'$.
In the first level of $\mathcal{A}'$, clock p_0 is active. After some arbitrary time, a transition, with no guard, is taken to the state of the second level and clock p_0 is frozen. In the second level, clock p_1 is active and the same procedure continues: after some time a transition to the next level is taken, and clock p_1 is frozen, and so on for the first k levels. In these first k levels, any run of $\mathcal{A}'$ chooses a non-negative fixed value for the clocks $p_0, \ldots, p_{k-1}$, and hence almost for the parameters of $\mathcal{A}$. Parameters may however have negative values so level $k+1$ serves as a technicality to choose the final sign of the corresponding clocks. This is done by assigning p_{i-1} or $-p_{i-1}$ to clock p_i, between each two consecutive states, for all $i \in [1..k-1]$, in a run without any delay in any of the states of level $k+1$ (the other runs, with delays in the states of level $k+1$, overlap on those corresponding to other parameter valuations and are therefore not a problem). In the last state of level $k+1$, the frozen clocks $p_1, ..., p_k$ can therefore have any arbitrary real value assigned. The automaton finally proceeds to the initial state of $\mathcal{A}$ keeping the values of these additional clocks. Since they correspond to levels lower than any level of $\mathcal{A}$ they can be used liberally enough in the guards and updates of $\mathcal{A}$. The obtained automaton $\mathcal{A}'$ is an ITA and parameters of $\mathcal{A}$ are modeled as clocks in $\mathcal{A}'$.

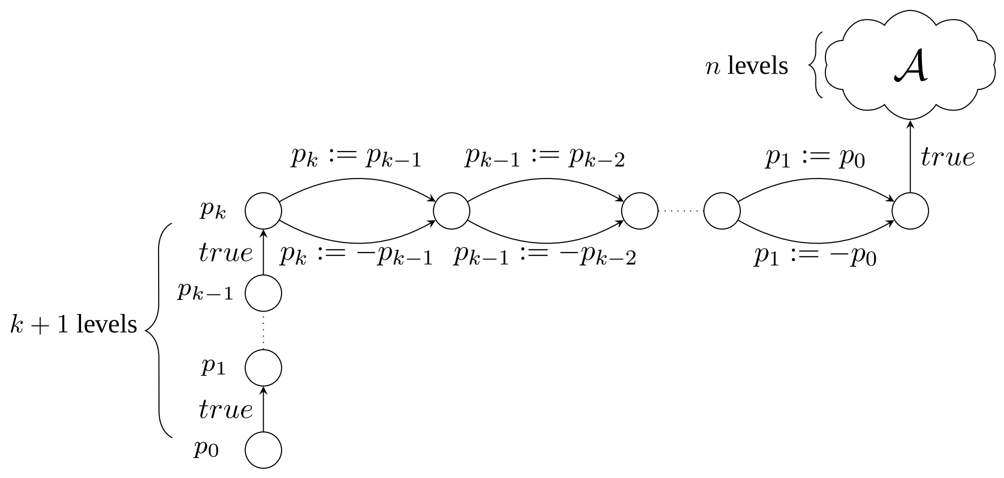


Figure 6. An equivalent ITA $\mathcal{A}'$

Let X be the set of clocks in $\mathcal{A}$ and X' be the set of clocks in $\mathcal{A}'$ (thus $X' = X \cup \{p_0, ..., p_k\}$). For any subset $Y \subseteq X$ and a valuation v, we define the restriction of v to Y as the unique valuation v on Y such that $v_{|Y}(x) = v(x)$. We now show that a configuration $s = (q, v)$ is reachable in $\mathcal{A}$ for some parameter valuation π (i.e., in $\mathcal{A}(\pi)$) iff there exists some configuration $s' = (q', v')$, such that $q' = q$ and for all $x \in X, v'_{|X}(x) = v(x)$, is reachable in $\mathcal{A}'$.
On the one hand, if there exists a path to reach s' in $\mathcal{A}'$, then by construction this path goes through a configuration (q_0, v_0) such that $(q_0, v_{0|X})$ is the initial configuration of $\mathcal{A}$ (i.e. $v_{0|X}$ is the zero valuation). Let π be the parameter valuation such that for all $i > 0, \pi(p_i) = v_0(p_i)$, then s is reachable in $\mathcal{A}(\pi)$.
On the other hand, let π be a parameter valuation and v be a clock valuation on X such that (q, v) is reachable in $\mathcal{A}(\pi)$. Then using an appropriate run in the prefix one reaches (q_0, v_0) with $v_{0|X}$ is the zero valuation and for all $i > 0, v_0(p_i) = \pi(p_i)$. Afterwards this run is extended to reach q by mimicking the run of $\mathcal{A}(\pi)$. □

Using Proposition 5.1 and Theorem 3.7, we can now give the main result of this section.

Theorem 5.2. The (polyhedral scoped) existential reachability problem is decidable for additively parametrized PITA, and belongs to 2EXPTIME. It belongs to PTIME when the number of clocks and parameters is fixed. It is PSPACE-complete when the number of levels and parameters is fixed.

Proof:
Following Proposition 5.1, every additively parametrized PITA can be transformed into an equivalent ITA, and the (unscoped) reachability problem of additively parametrized PITA is thus reduced to the reachability problem of ITA, already known to be decidable. The complexity results follow from the complexity results for ITA given in Theorem 3.7, since the size of $\mathcal{A}'$ is only linear in the size of $\mathcal{A}$: if there are n levels, N clocks, k parameters, x states and y transitions in $\mathcal{A}$, the number of levels, clocks, states and transitions in $\mathcal{A}'$ are $n + k + 1$, $N + k + 1$, $x + 2k + 1$ and $y + 3k + 1$, respectively.
With a polyhedral scope, given as a finite union of polyhedra, we need to guard the transition between the last state of the prefix and the initial state of $\mathcal{A}$, in $\mathcal{A}'$, by the given polyhedra (each polyhedra of the union could guard a different transition, as well). □

6. Reachability analysis with multiplicative parametrization

We now focus on the multiplicative case and this section is devoted to the proof of the following result:

Theorem 6.1. The (scoped) existential, universal and robust reachability problems for PITA are decidable and belong to 2EXPSPACE. The complexity reduces to PSPACE when the number of levels is fixed.

We first informally present the main ideas underlying the proof, which is based on the proof of Theorem 3.7 but extends it by the handling of parameters.

The first novelty with respect to the unparametrized case is the need to build from the PITA a finite partition of the set $\mathbb{R}^P$ of parameter valuations. In addition the non emptiness of an item Π of this partition should be decided. We will establish that the items of this partition can be specified by a satisfiable first-order formula over $(\mathbb{R}, +, \times)$, with the parameters as variables. Intuitively, inside Π the qualitative behaviour of $\mathcal{A}$ does not depend on the precise parameter valuation. In a second step, we build a finite automaton $\mathcal{R}(\Pi)$ for each non empty Π. In $\mathcal{R}(\Pi)$, a state R, again called a class, defines a set $[\![R]\!]_\pi$ of reachable configurations of $\mathcal{T}_{\mathcal{A}(\pi)}$ for a valuation $\pi \in \Pi$. The transition relation of $\mathcal{R}(\Pi)$ contains discrete steps $R \xrightarrow{e} R'$ (for a transition e of $\mathcal{A}$) and abstract time steps $R \rightarrow Post(R)$ with the following properties:

Discrete Step (DS): If there is a transition $R \xrightarrow{e} R'$ in $\mathcal{R}(\Pi)$ then for each $\pi \in \Pi$ and each $(q, v) \in [\![R]\!]_\pi$ there exists $(q', v') \in [\![R']\!]_\pi$ such that $(q, v) \xrightarrow{e} (q', v')$.

Conversely, let $\pi \in \Pi$ and $(q, v) \in [\![R]\!]_\pi$. If there exists a transition $(q, v) \xrightarrow{e} (q', v')$ in $\mathcal{T}_{\mathcal{A}(\pi)}$ then for some R', there is a transition $R \xrightarrow{e} R'$ in $\mathcal{R}(\Pi)$ and (q', v') belongs to $[\![R']\!]_\pi$.

Time Step (TS): Let $\pi \in \Pi$ and $(q, v) \in [\![R]\!]_\pi$. There exists $d > 0$ such that $(q, v +_q d) \in [\![Post(R)]\!]_\pi$ and for each d' with $0 \leq d' \leq d$, $(q, v +_q d') \in [\![R]\!]_\pi \cup [\![Post(R)]\!]_\pi$.

Hence, we obtain a finite family of abstract time bisimulations of the transition systems $\mathcal{T}_{\mathcal{A}(\pi)}$, for all parameter valuations, which gives the decidability result.

The second novelty lies in the construction of expressions in the sets $\{E_k\}_{k \leq n}$. These expressions now contain polynomials of parameters and the main difference from the unparametrized case is the normalization operation of an expression $\sum_{i \leq k} a_i x_i + b$ which now depends on the polynomial a_k.

Example 6.2. Consider for instance expression $p_2 x_2 + x_1 - 2$ which appear in automaton $\mathcal{A}_2$ of Figure 5(a) with a comparison to 0. For a valuation where $p_2 = 0$, a normalization should yield $x_1 - 2$. If $p_2 \neq 0$, the operation should yield $-\frac{x_1 - 2}{p_2}$. In addition, the case $p_2 \neq 0$ should be split depending on the sign of p_2, since the operation could change the comparison operator involved in a guard. Therefore, we also need to define a set $PolPar$ of polynomials appearing in the denominators like p_2. $\square$

6.1. Construction of $PolPar$ and Expressions $\{E_k\}_{k \leq n}$

In the spirit of normalization, we define three operations on expressions, relatively to a level k, to help building the elements in E_k to which the active clock on level k will be compared.

Definition 6.3. Let $k \leq n$ be some level and let C be an expression in $\mathcal{L}in(Xm_{<n+1}, \mathcal{F}rac(P, \mathbb{Q}))$, $C = \sum_{i \leq n} a_i x_i + b$ with $a_k = \frac{r_k}{s_k}$, for some r_k and s_k in $\mathcal{P}ol(P, \mathbb{Q})$. We associate with C the following expressions:

- $\texttt{lead}(C, k) = r_k$;
- if $\texttt{lead}(C, k) \notin \mathbb{Q} \setminus \{0\}$, $\texttt{comp}(C, k) = \sum_{i<k} a_i x_i + b$;
- if $\texttt{lead}(C, k) \neq 0$ then $\texttt{compnorm}(C, k) = -\sum_{i<k} \frac{a_i}{a_k} x_i - \frac{b}{a_k}$.

In the previous example, comp corresponds to $x_1 - 2$ while compnorm corresponds to $-\frac{x_1-2}{p_2}$. More examples are given after the construction of $PolPar$ and $\{E_k\}_{k \leq n}$. This construction proceeds top down from level n to level 1 after initializing $PolPar$ to $\emptyset$ and E_k to $X_k \cup \{0\}$ for all k. When handling level k, we add new terms to E_i for $1 \leq i \leq k$.

1. At level k the first step consists in adding new expressions to E_k and new polynomials to $PolPar$. More precisely, let C be any expression occurring in a guard of an edge leaving a state of level k. We add $\texttt{lead}(C, k)$ to $PolPar$ when it does not belong to $\mathbb{Q}$ and we add $\texttt{comp}(C, k)$ and $\texttt{compnorm}(C, k)$ to E_k when they are defined.

2. The second step consists in iterating the following procedure until no new term is added to any E_i for $1 \leq i \leq k$.

   (a) Let $q \xrightarrow{\varphi, a, u} q'$ with $\lambda(q) \geq k$ and $\lambda(q') \geq k$, and let $C \in E_k$. Then we add $C[u]$ to E_k.

   (b) Let $q \xrightarrow{\varphi, a, u} q'$ with $\lambda(q) < k$ and $\lambda(q') \geq k$. Let $\{C, C'\}$ be a set of two expressions in E_k. We compute $C'' = C[u] - C'[u]$, choosing an arbitrary order between C and C'. This step ends by handling C'' w.r.t. $\lambda(q)$ as done for C w.r.t. k in step 1 above.

Example 6.4. For the automaton of Figure 5(a), initially, we have $PolPar = \emptyset$, $E_1 = \{x_1, 0\}$ and $E_2 = \{x_2, 0\}$. Starting with level $k = 2$, we consider in step 1 the expression $C_2 = p_2 x_2 + x_1 - 2$ appearing in the guard of the single edge leaving q_2. We compute $\texttt{lead}(C_2, 2) = p_2$, $\texttt{comp}(C_2, 2) = x_1 - 2$, and $\texttt{compnorm}(C_2, 2) = -\frac{x_1-2}{p_2}$. We obtain $PolPar = \{p_2\}$ and $E_2 = \{x_2, 0, x_1 - 2, -\frac{x_1-2}{p_2}\}$. For step 2.(a) and the same edge, we apply its update to the expressions of E_2 that contain x_2, add them to E_2, and thus obtain $E_2 = \{x_2, 0, x_1 - 2, -\frac{x_1-2}{p_2}, (p_1 - 4p_2^2)x_1 + p_2\}$.

In step 2.(b), considering the single edge from q_1 to q_2, we compute the differences between any two expressions from E_2 (after applying update which means here substituting 0 to x_2 and letting x_1 unchanged) and the resulting expressions lead, comp and compnorm, which yields:
$PolPar = \{p_2, p_2 + 1, 1 - p_1 + 4p_2^2, 1 + p_1 p_2 - 4p_2^3\}$,
$E_1 = \{x_1, 0, 2, -\frac{2(p_2+1)}{p_2}, -2 - p_2, \frac{2+p_2}{1-p_1+4p_2^2}, \frac{p_2^2-2}{p_2}, \frac{2-p_2^2}{1+p_1p_2-4p_2^3}\}$.

We proceed with level 1. Since expression $C_1 = x_1 - p_1$ occurring in the guard of the considered edge has leading coefficient equal to 1, there is no term to add to $PolPar$. We add $\texttt{compnorm}(C_1, 1) = p_1$ to E_1, hence the final result is:

$$\begin{aligned} PolPar &= \{p_2, p_2 + 1, 1 - p_1 + 4p_2^2, 1 + p_1 p_2 - 4p_2^3\} \\ E_1 &= \{x_1, 0, 2, -\tfrac{2(p_2+1)}{p_2}, -2 - p_2, \tfrac{2+p_2}{1-p_1+4p_2^2}, \tfrac{p_2^2-2}{p_2}, \tfrac{2-p_2^2}{1+p_1p_2-4p_2^3}, p_1\} \\ E_2 &= \{x_2, 0, x_1 - 2, -\tfrac{x_1-2}{p_2}, (p_1 - 4p_2^2)x_1 + p_2\} \end{aligned}$$

□

Lemma 6.5 below is used for the class automata construction. Its proof is obtained by a straightforward examination of the above procedure.

Lemma 6.5. Let C belong to E_k for some k and $c = \frac{r}{s}$ be a coefficient of C with $s \notin \mathbb{Q}$. Then there exists polynomials $P_1, \dots, P_\ell \in PolPar$ and some constant $A \in \mathbb{Q} \setminus \{0\}$ such that $s = A. \prod_{1 \leq i \leq \ell} P_i$.

Lemma 6.6 is the parametrized version of Lemma 3.2 and its (omitted) proof is almost identical.

Lemma 6.6. For a PITA $\mathcal{A}$, let H be the number of constraints in the guards, U the number of updates in the transitions (assuming $U \geq 2$) and $M = \max\{|X_k| \mid 1 \leq k \leq n\}$. The construction procedure of $\{E_k\}_{k \leq n}$ terminates and the size of every E_k is bounded by $(H+M)^{2^{n-k}} \times U^{2^{n(n-k+1)}}$.

Lemma 6.7 is the parametrized version of Lemma 3.3. However since the coefficients are now rational functions, the degree of the polynomials must also be analyzed.

Lemma 6.7. Let $\mathcal{A}$ be a PITA, and let b_0 be the maximal total number of bits for integers of an expression in $\mathcal{A}$ and d_0 the maximal degree of polynomials, occurring in $\mathcal{A}$. If b is the total number of bits of the integer constants and d the degree of a polynomial, occurring in an expression of $PolPar$ or some E_k, then $b \leq ((n+1)!)^2(n+1)2^{3n+1}b_0$ and $d \leq (n+1)!5^n d_0$.

Proof:
W.l.o.g. we assume that there is a single denominator for the rationals occurring in updates since it only induces a polynomial blow up.
Assume that before the level $n-k$ is performed, the total number of bits for integers occurring in some expression is b_k. We establish by induction that $b_k \leq \prod_{j=1}^{k}(n+2-j)^2(k+1)2^{n+2k+1}b_0$. The basis case is trivial. At level $n-k$, step 1 does induces an increasing only when operation `compnorm` is applied on a original guard whose coefficients are polynomials (instead of rational fractions). After this operation the number of bits is bounded by $(n-k+1)b_0 \leq (n-k+1)b_k$. For an expression that was already present in E_{n-k}, its coefficients are modified in order to get a common denominator by taking the product of the original denominators. After this transformation the total number of bits is bounded by $(n-k+1)2b_k$.
Examining one update applied on an expression, the total number of bits of the coefficients of the updated expression is increased by $(n-k+1)b_0$. Since an expression built after step 2.(a) has been obtained by less than 2^{n-k} updates, the total number of bits is less than $(n-k+1)2b_k + 2^{n-k}(n-k+1)b_0$.
At step 2.(b), the difference $C[u] - C'[u]$ requires to compute the lcm of two denominators (bounded by their product). So the difference operation leads to a bound $(n-k+1)4b_k + 2^{n-k+1}(n-k+1)b_0$ for the total number of bits.
The final step 2.(b) consists in multiplying a numerator and a denominator of some coefficients leading to a bound:

$$\begin{aligned}
(n-k+1)^2(4b_k + 2^{n-k+1}b_0) &\leq (n-k+1)^2\left(4\prod_{j=1}^{k}(n+2-j)^2(k+1)2^{n+2k+1}b_0 + 2^{n-k+1}b_0\right)\\
&\leq \left(\prod_{j=1}^{k+1}(n+2-j)\right)^2((k+1)2^{n+2(k+1)+1}b_0 + 2^{n+2(k+1)+1}b_0)\\
&= \left(\prod_{j=1}^{k+1}(n+2-j)\right)^2(k+2)(2^{n+2(k+1)+1}b_0)
\end{aligned}$$

for the number of bits.

Assume that before the level $n-k$ is performed, the degree of a polynomial (of parameters) occurring in some expression is at most d_k. We establish a relation between d_k and d_{k+1}. At level $n-k$, step 1 does not induce any increasing when operation `compnorm` is applied on a original guard whose coefficients are polynomials (instead of rational fractions). More precisely the numerators of rational fractions are unchanged while the denominators are numerators of some previous expressions. For an expression that was already present in E_{n-k}, its coefficient are modified in order to get a common denominator by taking the product of the original denominators. After this transformation the maximal degree is bounded by $(n-k+1)d_k$.
Let us examine an expression $C = \sum_{i \leq n-k} a_i x_i + b$ built after step 2.(a). Examining the successive updates, the numerator of coefficient a_i can be expressed as $\sum_{d \in \mathcal{D}} \prod_{j \in d} c_{d,j}$ where $\mathcal{D}$ is the set of subsets of $\{i, \ldots, n-k\}$ containing i and $c_{d,j}$ are all coefficients of the updates (i.e. coefficients of polynomials) except one coefficient of the expression built before this step. The same reasoning applies to the constant coefficient of the expression. So the degree of the a_i's and b is bounded by: $(n-k+1)(d_k+d_0)$. The denominators are denominators of expressions previously built so bounded by $(n-k+1)d_k$.
At step 2.(b), the difference $C[u] - C'[u]$ requires to compute the lcm of two denominators (bounded by their product). So the difference operation leads to a bound $(n-k+1)(2d_k+d_0)$ for the numerators of its coefficients and $(n-k+1)2d_k$ for the denominators.
The final step 2.(b) consists in multiplying a numerator and a denominator of some coefficients leading to a bound $(n-k+1)(4d_k+d_0)$. So $d_{k+1} \leq (n-k+1)5d_k$ yielding the desired bound. □

We now explain the partition construction. Starting from the finite set $PolPar$, we split the set of parameter valuations in parameter regions specified by the result of comparisons to 0 of the values of the polynomials in $PolPar$. For instance, for the set $PolPar$ computed above, the inequalities $p_2 < 0$, $p_2 + 1 = 0$, $1 - p_2 - 4p_1^2 = 0$ and $1 + p_1 p_2 - 4p_2^3 = 0$ define a set $preg$ of parameter valuations. The parameter region $preg$ is non empty since it contains $p_1 = 5$ and $p_2 = -1$. The set of such constraints yielding non empty regions can be computed by solving an existential formula of the first-order theory of reals.

Then, given a non empty parameter region $preg$, we consider the following subset of E_k for $1 \leq k \leq n$: $E_{k,preg} = \{C \in E_k \mid \text{the denominators of coefficients of } C \text{ are non null in } preg\}$. Due to Lemma 6.5, these subsets are obtained by examining the specification of $preg$.

Observe that expressions in $E_{1,preg} \setminus X_1$ belong to $\mathcal{F}rac(P, \mathbb{Q})$ and that, depending on the parameter valuation, the values of two expressions can be differently ordered. We refine $preg$ according to a linear preorder $\preceq_1$ on $E_{1,preg} \setminus X_1$ which is satisfiable within $preg$. We denote this refined region by $\Pi = (preg, \preceq_1)$ and we now build a finite automaton $\mathcal{R}(\Pi)$.

6.2. Construction of the Class Automata

In this paragraph, we fix a non empty parameter region $\Pi = (preg, \preceq_1)$.

Class definition. A state of $\mathcal{R}(\Pi)$, called a class like before, is defined as a pair $R = (q, \{\preceq_k\}_{1 \leq k \leq \lambda(q)})$ where q is a state of $\mathcal{A}$ and $\preceq_k$ is a total preorder over $E_{k,preg}$, for $1 \leq k \leq \lambda(q)$. For a parameter valuation $\pi \in \Pi$, the class R describes the following subset of configurations in $\mathcal{T}_{\mathcal{A},\pi}$:

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

The initial state of $\mathcal{R}(\Pi)$ is the class R_0, such that $(q_0, \mathbf{0}) \in [\![R_0]\!]_\pi$, which can be straightforwardly determined by extending $\preceq_1$ to $E_{1,preg}$ with $x \preceq_1 0$ and $0 \preceq_1 x$ for all $x \in X_1$, and closing $\preceq_1$ by transitivity.

Transitions in $\mathcal{R}(\Pi)$ consist of the following discrete and time steps:

Discrete step. Let $R = (q, \{\preceq_i\}_{1\leq i\leq\lambda(q)})$ and $R' = (q', \{\preceq'_i\}_{1\leq i\leq\lambda(q')})$ be two classes and let $e : q \xrightarrow{\varphi,a,u} q'$ be a transition in $\mathcal{A}$. There is a transition $R \xrightarrow{e} R'$ if for some $\pi \in \Pi$, there are some $(q,v) \in [\![R]\!]_\pi$ and $(q',v') \in [\![R']\!]_\pi$ such that $(q,v) \xrightarrow{e} (q',v')$. In this case, we claim that for all $(q,v) \in [\![R]\!]_\pi$ there is a $(q',v') \in [\![R']\!]_\pi$ such that $(q,v) \xrightarrow{e} (q',v')$. For this, we prove in the sequel that the existence of transition $R \xrightarrow{e} R'$ is independent of $\pi \in \Pi$ and of $(q,v) \in [\![R]\!]_\pi$. It can be seen as follows.
We note $\ell = \lambda(q)$ for the level of transition e.

Firability condition. We write $\varphi = \bigwedge_{j\in J} C_j \bowtie_j 0$ with, for each j, either $C_j = a_\ell z + \sum_{i<\ell} a_i x_i + b$ (with $z \in X_\ell$) or $C_j = z - z'$ with $z, z' \in X_\ell$. We consider three subcases of the first case.
• **Subcase** $a_\ell = 0$**.** Then $C_j = \texttt{comp}(C_j, \ell) \in E_{\ell,preg}$ and using the positions of 0 and C_j w.r.t. $\preceq_\ell$, we can decide whether $C_j \bowtie_j 0$.
• **Subcase** $a_\ell \in \mathbb{Q} \setminus \{0\}$**.** Then $\texttt{compnorm}(C_j, \ell) \in E_{\ell,preg}$, hence using the sign of a_ℓ and the positions of z and $\texttt{compnorm}(C_j, \ell)$ w.r.t. $\preceq_\ell$, we can decide whether $C_j \bowtie_j 0$.
• **Subcase** $a_\ell \notin \mathbb{Q}$**.** According to the specification of $preg$, we know the sign of a_ℓ as it belongs to $PolPar$. In case $a_\ell = 0$, we decide as in the first subcase. Otherwise, we decide as in the second subcase.
The second case $C_j = z - z'$ is handled similarly.

Successor definition. To build the successor $R' = (q', \{\preceq'_i\}_{1\leq i\leq\lambda(q')})$ of R, we have to define the preorders $\{\preceq'_i\}_{1\leq i\leq\lambda(q')}$. Let $k \leq \lambda(q')$ and $g, h \in E_{k,preg}$.

1. Either $k \leq \ell$, by step 2(a) of the construction, $g[u], h[u] \in E_{k,preg}$. Then $g \preceq'_k h$ iff $g[u] \preceq_k h[u]$.

2. Or $k > \ell$, let $D = g[u] - h[u] = \sum_{i\leq\ell} a_i x_i + b$. There are again three subcases.

   • **Subcase** $a_\ell = 0$**.** Then $D = \texttt{comp}(D, \ell) \in E_{\ell,preg}$, so we can decide whether $D \preceq_\ell 0$ and $g' \preceq'_k h'$ iff $D \preceq_\ell 0$.

   • **Subcase** $a_\ell \in \mathbb{Q} \setminus \{0\}$**.** Then $\texttt{compnorm}(D, \ell) \in E_{\ell,preg}$. There are four possibilities to consider. For instance if $a_\ell > 0$ and $x_\ell \preceq_\ell \texttt{compnorm}(D, \ell)$ then $g' \preceq'_k h'$. The other cases are similar.

   • **Subcase** $a_\ell \notin \mathbb{Q}$**.** Let us write $a_\ell = \frac{r_\ell}{s_\ell}$. According to the specification of $preg$, we know the sign of a_ℓ since r_ℓ belongs to $PolPar$ and s_ℓ is a product of items in $PolPar$. In case $a_\ell = 0$, we decide $g' \preceq'_k h'$ as in the first case. Otherwise, we decide in a similar way as in the second case. For instance if $a_\ell > 0$ and $x_\ell \preceq_\ell \texttt{compnorm}(D, \ell)$ then $g' \preceq'_k h'$.

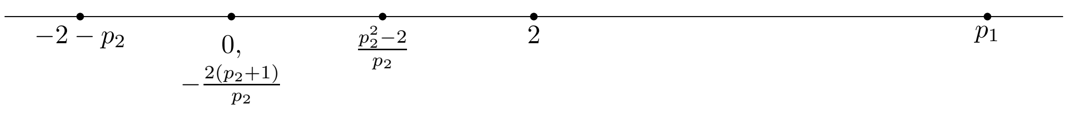
Time step. For $R = (q, \{\preceq_k\}_{1\leq k\leq\ell})$, there is a transition $R \xrightarrow{succ} Post(R)$, where $Post(R) = (q, \{\preceq'_k\}_{1\leq k\leq\ell})$ is the time successor of R, defined as follows. Intuitively, all preorders below $\ell = \lambda(q)$ are fixed, so $\preceq'_i = \preceq_i$ for each $i < \ell$. On level ℓ, the value of the active clock simply progresses along the one dimensional time line, where the expressions are ordered. More precisely, let $\sim$ be the equivalence relation $\preceq_\ell \cap \preceq_\ell^{-1}$ induced by the preorder. A $\sim$-equivalence class groups expressions yielding the same value, and on these classes, the (total) preorder becomes a (total) order. Let V be the $\sim$-equivalence class containing $\texttt{act}(q)$.

1. Either $V = \{\texttt{act}(q)\}$. If V is the greatest $\sim$-equivalence class, then $\preceq'_\ell = \preceq_\ell$ (and $Post(R) = R$). Otherwise, let V' be the next $\sim$-equivalence class. Then $\preceq'_\ell$ is obtained by merging $V = \{\texttt{act}(q)\}$ and V', and preserving $\preceq_\ell$ elsewhere.

2. Or V is not a singleton. Then we split V into $V \setminus \{\texttt{act}(q)\}$ and $\{\texttt{act}(q)\}$ and "extend" $\preceq_\ell$ by $V \setminus \{\texttt{act}(q)\} \preceq'_\ell \{\texttt{act}(q)\}$.

To conclude, observe that the automaton $\mathcal{R}(\Pi)$ defined above has the properties **(DS)** and **(TS)** mentioned previously, and is hence a finite time abstract bisimulation of $\mathcal{T}_{\mathcal{A},\pi}$, for all parameter valuations $\pi \in \Pi$.

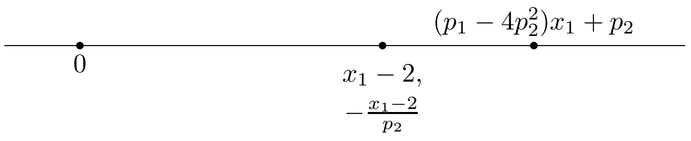
Example 6.8. The construction of $\mathcal{R}(\Pi)$ is illustrated on the automaton $\mathcal{A}_2$ from Figure 5(a), for the region $\Pi = (preg, \preceq_1)$, where $preg$ was defined above by: $p_2 < 0$, $p_2 + 1 = 0$, $1 - p_2 - 4p_1^2 = 0$ and $1 + p_1 p_2 - 4p_2^3 = 0$. For $\preceq_1$, we first remove from E_1 the expressions with null denominator: $E_{1,preg} = \{x_1, 0, 2, -\frac{2(p_2+1)}{p_2}, -2 - p_2, \frac{p_2^2-2}{p_2}, p_1\}$ and we consider the ordering on $E_{1,preg} \setminus \{x_1\}$ specified by the line below.



A part of the resulting class automaton $\mathcal{R}(\Pi)$, including the run corresponding to the one in Figure 5(b), is depicted in Figure 7, where dashed lines indicate (abstract) time steps.

The initial class is $R_0 = (q_0, Z_0)$ where Z_0 is $\preceq_1$ extended with $x_1 = 0$. Denoting (slightly abusively) extensions with the symbol $\wedge$, the time successors of the initial state are obtained by moving x_1 to the right along the line: $R_0^1 = (q_0, \preceq_1 \wedge\, 0 < x_1 < \frac{p_2^2-2}{p_2})$, $R_0^2 = (q_0, \preceq_1 \wedge\, x_1 = \frac{p_2^2-2}{p_2}), \ldots$, up to $R_0^7 = (q_0, \preceq_1 \wedge\, x_1 > p_1)$. Transition a can be fired from all classes up to R_0^5 (but not from R_0^6 and R_0^7 where the constraint $x_1 < p_1$ is not satisfied). In Figure 7, we represent only the one from $R_0^5 = (q_0, Z_1)$ with $Z_1 = \preceq_1 \wedge\, 2 < x_1 < p_1$, corresponding to the run in Figure 5(b).

Along this run, the ordering $\preceq_2$ is determined by region Π and Z_1, on $E_{2,preg} \setminus \{x_2\} = \{0, x_1 - 2, -\frac{x_1-2}{p_2}, (p_1 - 4p_2^2)x_1 + p_2\}$. It is illustrated on the line below.



Firing transition a produces the class $R_1 = (q_1, Z_1, \preceq_2 \wedge x_2 = 0)$. Transition b is then fired from the (second) time successor of R_1 for which $x_2 = -\frac{x_1-2}{p_2}$. $\square$

We are now in position to prove Theorem 6.1.

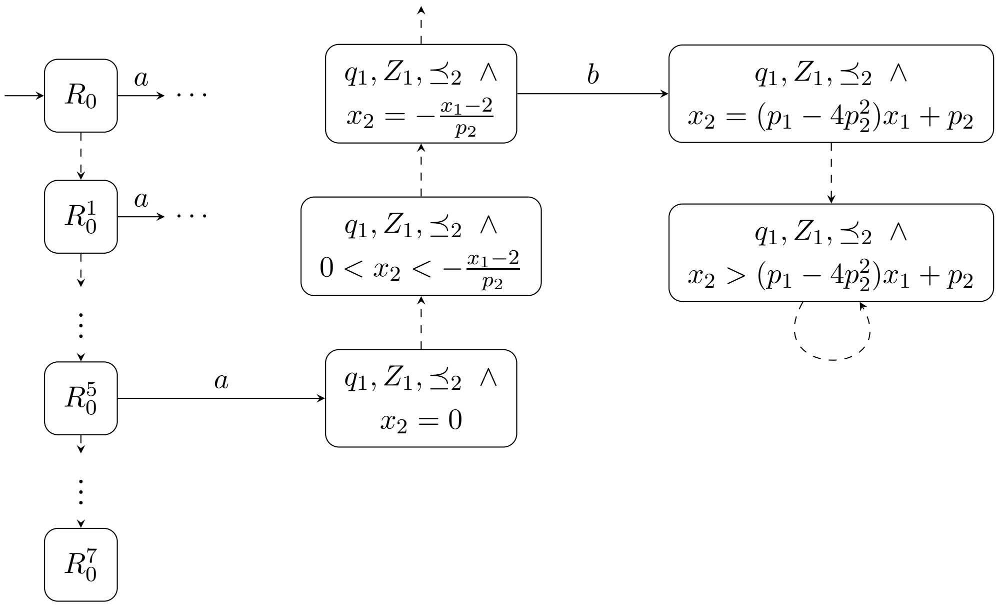

R_0
a
$\cdots$
R_0^1
a
$\cdots$
R_0^5
a
R_0^7
$q_1, Z_1, \preceq_2 \ \wedge$ $x_2 = 0$
$q_1, Z_1, \preceq_2 \ \wedge$ $0 < x_2 < -\frac{x_1-2}{p_2}$
$q_1, Z_1, \preceq_2 \ \wedge$ $x_2 = -\frac{x_1-2}{p_2}$
b
$q_1, Z_1, \preceq_2 \ \wedge$ $x_2 = (p_1 - 4p_2^2)x_1 + p_2$
$q_1, Z_1, \preceq_2 \ \wedge$ $x_2 > (p_1 - 4p_2^2)x_1 + p_2$

Figure 7. A part of $\mathcal{R}(\Pi)$ for $\mathcal{A}_2$

Proof:
Starting from a PITA $\mathcal{A}$, we use the above construction, whose termination is guaranteed by lemma 6.6, to design a nondeterministic procedure for existential reachability of a given state q:

1. Build $PolPar$ and $\{E_k\}_{1 \le k \le n}$.
2. Guess a parameter region $(preg, \preceq_1)$.
3. Check non emptiness of $(preg, \preceq_1)$.
4. Build the class automaton $\mathcal{R}(preg, \preceq_1)$ and check whether q occurs in some class.

For universal reachability of q, in step 4, one checks whether q does not occur in any class. This gives us a nondeterministic procedure for the complementary problem. For robust reachability in step 2, one guesses an open parameter region *i.e.*, only specified by strict inequalities.

We now analyze the complexity of these procedures. Due to lemmas 6.6 and 6.7, the first step is performed in 2EXPTIME and in PTIME when the number of clocks is fixed. Guessing a parameter region has the same complexity.
The satisfiability problem for a first-order formula is in PSPACE [24]. Due to lemma 6.6, the number s of (in)equalities specifying the region fulfills $s = O((H+M)^{2^n} \times U^{2^{n^2}})$ with the previous notations. Let b be the total number of bits of the integers occurring in a constraint of the specification of the region. Due to lemma 6.7, $b \le ((n+1)!)^2(n+1)2^{3n+1}b_0$. Let d be the maximal degree of the polynomials occurring in the specification of the region. Due to the same lemma, $d \le (n+1)!5^n d_0$. So the emptiness problem for a region is decided in 2EXPSPACE which becomes PSPACE when the number of levels is fixed.

Observe now that the class automaton $\mathcal{R}(preg, \preceq_1)$ is isomorphic to the class automaton of the ITA $\mathcal{A}(\pi)$ that would be obtained from $\mathcal{A}$ with any parameter valuation π in $\Pi = (preg, \preceq_1)$. It has been proved in Section 3 that this automaton can be built in polynomial time w.r.t. the size of the representation of any class. As the size of the representation of a class of a PITA has the same order as the one of the corresponding ITA (dominated by the doubly exponential number of expressions) and the construction algorithms perform similar operations, this yields a complexity of 2EXPTIME and PSPACE when the number of levels is fixed.
So the dominating factor of this nondeterministic procedure is the emptiness check done in 2EXPSPACE. By Savitch's theorem this procedure can be determinized with the same complexity. □

7. Conclusion

While seminal results on parametrized timed models leave little hope for decidability in the general case, we provide here an expressive formalism for the analysis of parametric reachability problems. Our setting includes a restricted form of stopwatches and polynomials in the parameters occurring as both additive and multiplicative coefficients of the clocks in guards and updates. We plan to investigate which kind of timed temporal logic would be decidable on PITA.

References

[1] Alur R, Courcoubetis C, Halbwachs N, Henzinger TA, Ho PH, Nicollin X, et al. The algorithmic analysis of hybrid systems. Theoretical Computer Science. 1995;138:3–34.

[2] Alur R, Dill DL. Automata for modeling real-time systems. In: ICALP'90. Springer; 1990. p. 322–335.

[3] Bérard B, Haddad S. Interrupt Timed Automata. In: FoSSaCS'09. vol. 5504 of LNCS. Springer; 2009. p. 197–211.

[4] Bérard B, Haddad S, Sassolas M. Interrupt Timed Automata: Verification and Expressiveness. Formal Methods in System Design. 2012;40(1):41–87.

[5] Alur R, Henzinger TA, Vardi MY. Parametric real-time reasoning. In: ACM Symp. on Theory of Computing. ACM; 1993. p. 592–601.

[6] Miller JS. Decidability and Complexity Results for Timed Automata and Semi-linear Hybrid Automata. In: HSCC'00. vol. 1790 of LNCS. Springer; 2000. p. 296–309.

[7] Doyen L. Robust Parametric Reachability for Timed Automata. Information Processing Letters. 2007;102(5):208–213.

[8] Bérard B, Fribourg L. Automated Verification of a Parametric Real-Time Program: The ABR Conformance Protocol. In: CAV'99. vol. 1633 of LNCS. Springer; 1999. p. 96–107.

[9] André É, Chatain Th, Encrenaz E, Fribourg L. An Inverse Method for Parametric Timed Automata. Int J of Foundations of Comp Sci. 2009;20(5):819–836.

[10] André É, Fribourg L, Kühne U, Soulat R. IMITATOR 2.5: A Tool for Analyzing Robustness in Scheduling Problems. In: FM'12. vol. 7436 of LNCS. Springer; 2012. p. 33–36.

[11] Jovanović A, Lime D, Roux OH. Integer Parameter Synthesis for Real-Time Systems. IEEE Transactions on Software Engineering (TSE). 2015;41(5):445–461.

[12] Traonouez LM, Lime D, Roux OH. Parametric Model-Checking of Stopwatch Petri Nets. Journal of Universal Computer Science (JUCS). 2009;15(17):3273–3304.

[13] Lime D, Roux OH, Seidner C, Traonouez LM. Romeo: A Parametric Model-Checker for Petri Nets with Stopwatches. In: TACAS'09. vol. 5505 of LNCS. Springer; 2009. p. 54–57.

[14] Alur R, Henzinger TA, Ho PH. Automatic Symbolic Verification of Embedded Systems. IEEE Transactions on Software Engineering. 1996;22:181–201.

[15] Henzinger TA, Ho PH, Wong-Toi H. HyTech: A Model-Checker for Hybrid Systems. Software Tools for Technology Transfer. 1997;1:110–122.

[16] Hune T, Romijn J, Stoelinga M, Vaandrager F. Linear Parametric Model Checking of Timed Automata. J of Logic and Alg Prog. 2002;52-53:183–220.

[17] Bozzelli L, La Torre S. Decision problems for lower/upper bound parametric timed automata. Formal Methods in System Design. 2009;35(2):121–151.

[18] Bundala D, Ouaknine J. Advances in Parametric Real-Time Reasoning. In: MFCS'14. vol. 8634 of LNCS. Springer; 2014. p. 123–134.

[19] Bruyère V, Dall'Olio E, Raskin J. Durations and parametric model-checking in timed automata. ACM Trans Comput Log. 2008;9(2).

[20] Bylander T. The Computational Complexity of Propositional STRIPS Planning. Artificial Intelligence. 1994;69:165–204.

[21] Zaslavsky T. Facing up to Arrangements: Face-Count Formulas for Partitions of Space by Hyperplanes. AMS Memoirs. 1975;1(154).

[22] Roos C, Terlaky T, Vial JP. Theory and Algorithms for Linear Optimization. An Interior Point Approach. Wiley-Interscience, John Wiley & Sons Ltd; 1997.

[23] Courcoubetis C, Yannakakis M. Minimum and Maximum Delay Problems in Real-Time Systems. Formal Methods in System Design. 1992;1(4):385–415.

[24] Canny JF. Some algebraic and geometric computations in PSPACE. In: ACM Symp. on Theory of Computing. ACM; 1988. p. 460–467.