Integer Parameter Synthesis for Timed Automata[*]

A. Jovanović, D. Lime and O. H. Roux

LUNAM Université. École Centrale de Nantes - IRCCyN UMR CNRS 6597
Nantes, France

Abstract. We provide a subclass of parametric timed automata (PTA) that we can actually and efficiently analyze, and we argue that it retains most of the practical usefulness of PTA. The currently most useful known subclass of PTA, L/U automata, has a strong syntactical restriction for practical purposes, and we show that the associated theoretical results are mixed. We therefore advocate for a different restriction scheme: since in classical timed automata, real-valued clocks are always compared to integers for all practical purposes, we also search for parameter values as bounded integers. We show that the problem of the existence of parameter values such that some TCTL property is satisfied is PSPACE-complete. In such a setting, we can also of course synthesize all the values of parameters and we give symbolic algorithms, for reachability and unavoidability properties, to do it efficiently, i.e., without an explicit enumeration. This also has the practical advantage of giving the result as symbolic constraints between the parameters. We finally report on a few experimental results to illustrate the practical usefulness of the approach.

1 Introduction

Real-time systems are ubiquitous, and to ensure their correct design it seems natural to rely on the mathematical framework provided by formal methods. Within that framework, the model-checking of timed models is becoming ever more efficient. It nevertheless requires a complete knowledge of the system. Consequently, the verification can only be performed after the design stage, when the global system and its environment are known. Getting a complete knowledge of a system is often impossible and even when it is possible, it increases the complexity of the conception and the verification of systems. Moreover, if the model of the system is proved wrong or if the environment changes, this complex verification process must be carried out again. It follows that the use of parametric timed models is certainly a very interesting approach for the design of real-time systems.

However, for general parametric formalisms such as Parametric Timed Automata, the existence of a parameter value such that some state is reachable is

[*] This work was partially funded by the ANR national research program ImpRo (ANR-2010-BLAN-0317).

undecidable and there is currently no algorithm that solves the synthesis problem of parameter values except for severely restricted subclasses, whose practical usability is unclear.

It is then a challenging issue to define a subclass of parametric timed automata, which retains enough of its expressive power and such that, for both reachability and unavoidability properties, the existence of parameter values is decidable and for which there exist efficient symbolic synthesis algorithms.

1.1 Related Work

Parametric timed automata (PTA) have been introduced by Alur et al. in [3], as a way to specify parametric timing constraints. They study the parametric emptiness problem which asks if there exists a parameter valuation such that the automaton has an accepting run. The problem is proven undecidable for PTA that uses three clocks and six parameters, and applies to both dense and discrete time domains. In [9], the undecidability proof is extended for parametric timed automata that use only strict inequalities. Further in [12], Hune et al. identify a class of parametric timed automata, lower bound/upper bound (L/U) automata, for which the emptiness problem is decidable. However, their model-checking algorithm, that uses Difference Bound Matrix as data structure, might not terminate. Decidability results for L/U automata have been further investigated by Bozzelli and La Torre in [6]. They consider infinite accepting runs and liveness property, and show that main decision problems such as emptiness, finiteness and universality for the set of parameter valuations are decidable and PSPACE-complete. They also study constrained version of emptiness and universality, where parameters are constrained by linear system of equalities and inequalities, and obtain decidability if parameters of different types are not compared in the linear constraint. They show how to compute the explicit representation of the set of parameters, when all the parameters are of the same type (L-automata and U-automata).

An approach for the verification of Parametric TCTL (PTCTL) formulae has been developed in [21] by Wang, where the problem has been proved decidable. A more general problem is studied in [7], where parameters are allowed both in the model and desired property (PTCTL formula). The authors show that the model-checking problem is decidable and the parameter synthesis problem is solvable, in discrete time, over a PTA with one parametric clock, if equality is not allowed in the formulae.

In [4], the authors develop a synthesis algorithm that starts from a reference parameter valuation and derives constraints on parameters, ensuring that the behaviors of PTA are time-abstract equivalent. They also give a conjecture for the termination being true on the examples studied. Henzinger et al. in [10], study more general, hybrid, systems extended with parameters. Their state-space exploration algorithms have been implemented in the model-checking tool HyTech. In [20], the authors analyze time Petri nets with parameters in timing constraints. A property is given as a PTCTL formula, but their model-checking algorithm consists in analysis of a region graph for each parameter valuation.

In [19], the authors extend time Petri nets with inhibitor arcs with parameters, and propose an abstraction of the parametric state-space and semi-algorithms for the parametric synthesis problem, considering simple PTCTL formulae.

1.2 Contributions

L/U-automata can be seen as the most useful subclass of PTA supported by many decidability results for reachability-like properties. We show that, even for the subclass with only upper bounds (U-automata), the existence of parameter valuations such that some unavoidability property is satisfied is undecidable though. We also pinpoint some difficulties with the actual synthesis of parameter values for L/U automata and reachability properties.

We therefore propose a different way of subclassing PTA: instead of syntactical restrictions of guards and invariants we propose a novel approach based on restricting the possible values of the parameters. To obtain decidability results, we show that we have to restrict these values to bounded integers. From a practical point of view, the subclass of PTA in such setting is not that restrictive since the temporal constraints for time automata are usually defined on natural (or rational) numbers. Nevertheless, this subclass is restrictive enough to make the problems we address decidable and to allow symbolic synthesis algorithms of parameter values.

We give symbolic algorithms to synthesize the set of all parameters valuations for reachability and unavoidability properties, without having to enumerate all the possibilities. These algorithms are implemented in our tool, Roméo.

Finally, we show that the problem of the existence of bounded integer valuations for PTA such that some property is satisfied is PSPACE-complete for a significant number of properties, which include Timed Computation Tree Logic (TCTL), and also that lifting either of the boundedness or the integer assumption leads to undecidability even for reachability.

1.3 Organization of the Paper

Section 2 gives the basic definitions related to the formalism of parametric timed automata. Section 3 recalls the main positive results on L/U-automata and gives new negative results that make more precise the practical usefulness of that model. This motivates a different restriction scheme based on limiting the possible values of the parameters. Section 4 presents symbolic algorithms for the synthesis problems when valuations are searched as bounded integers. In its development this section also exhibits semi-algorithms for the general setting and the (unbounded) integer setting. Section 5 gives the computational complexities of the associated problems. Finally, section 6 discusses the performance in practice of the proposed approach, illustrated on a small but realistic case-study.

2 Parametric Timed Automata

$\mathbb{Z}$ is the set of integers and $\mathbb{N}$ the set of natural numbers. $\mathbb{R}$ is the set of real numbers. $\mathbb{R}_{\geq 0}$ is the set of non-negative real numbers and $\mathbb{R}_{>0} = \mathbb{R}_{\geq 0} \setminus \{0\}$. For any closed interval $[a, b]$ of $\mathbb{R}$ with $a, b \in \mathbb{Z}$, we denote by $[a..b]$ its intersection with $\mathbb{Z}$.

Let X be a finite set. 2^X denotes the powerset of X and $|X|$ the size of X.

A *linear expression* on X is an expression generated by the following grammar, for $k \in \mathbb{Z}$ and $x \in X$: $\lambda ::= k \mid k * x \mid \lambda + \lambda$

W.l.o.g we consider *reduced* linear expressions λ in which each element of X occurs at most once and with at most one constant term. We let $\mathsf{Coeff}(\lambda, x)$ denote the coefficient of variable $x \in X$ in λ. If x does not occur in λ then $\mathsf{Coeff}(\lambda, x) = 0$. $\mathsf{Coeff}(\lambda, x)$ is well defined since λ is reduced.

$\wedge$ denotes the logical conjunction. A *linear constraint* on x is an expression generated by the following grammar, with λ a linear expression on X, $\sim \in \{>, \geq\}$: $\gamma ::= \lambda \sim 0 \mid \gamma \wedge \gamma$.

Let $V \subseteq \mathbb{R}$. A V-valuation for X is a function from X to V. We denote by V^X the set of V-valuations on X.

For any subset $X' \subseteq X$, and a V-valuation v on X, we define the restriction of v to X' as the unique V-valuation on X' such that $v_{|X'}(x) = v(x)$. If Y is a set of valuations on X, then $Y_{|X'}$ denotes its projection on X',*i.e.*, $Y_{|X'} = \{v_{|X'} \mid v \in Y\}$.

For a linear expression (resp. constraint) λ on X and a V-valuation v of X, we denote by $v(\lambda)$ the real number (resp. boolean value) obtained by replacing in λ each element x of X by the real value $v(x)$. We denote by $\mathcal{C}(X)$ the set of linear constraints on X.

Given some arbitrary order on X, a valuation can be seen as a real vector of size $|X|$. The set of valuations satisfying some linear constraint is then a *convex polyhedron* of $\mathbb{R}^{|X|}$.

Let X (resp. P) be a finite set. We call *clocks* (resp. *parameters*) the elements of X (resp. P). A *simple* (parametric clock) constraint on X (and P) is a linear constraint on $X \cup P$ such that exactly one element x of X occurs in each conjunct of the expression (not necessarily the same for each conjunct), and $\mathsf{Coeff}(x) \in \{-1, 1\}$. We denote by $\mathcal{B}(X, P)$ the set of such simple constraints and $\mathcal{B}'(X, P)$ the set of simple constraints in which the clock variable always has coefficient -1. As before, for any V-valuation v on P, and any simple constraint γ, $v(\gamma)$ is the linear constraint on X obtained by replacing each parameter $p \in P$ by the real value $v(p)$.

We further define the null valuation $\mathbf{0}_X$ on X by $\forall x \in X, \mathbf{0}_X(x) = 0$. For any subset R of X, and any valuation on X, we denote by $v[R]$ the valuation on X such that $v[R](x) = 0$ if $x \in R$ and $v[R](x) = v(x)$ otherwise. Finally $v + d$, for $d \geq 0$, is the valuation such that $(v + d)(x) = v(x) + d$ for all $x \in X$.

Definition 1 (Parametric Timed Automaton). *A* Parametric Timed Automaton *(PTA) $\mathcal{A}$ is a tuple $(L, l_0, \Sigma, X, P, E, \mathsf{Inv})$ where: L is a finite set of* locations*; $l_0 \in L$ is the* initial *location; Σ is a finite set of* actions*; X is a finite* *set of* clocks*; P is a finite set of* parameters*; $E \subseteq L \times \Sigma \times \mathcal{B}(X,P) \times 2^X \times L$ is a finite set of* edges*: if $(l, a, \gamma, R, l') \in E$ then there is an edge from l to l' with action a, (parametric)* guard *γ and set of clocks to reset R;* $\mathsf{Inv} : L \rightarrow \mathcal{B}'(X,P)$ *assigns a (parametric)* invariant *to each location.*

For any $\mathbb{Q}$-valuation v on P, the structure $v(\mathcal{A})$ obtained from $\mathcal{A}$ by replacing each simple constraint γ by $v(\gamma)$ is a *timed automaton* with invariants [2,11] (TA).

The behavior of a PTA $\mathcal{A}$ is described by that of all the timed automata obtained by considering all possible valuations of the parameters.

Definition 2 (Semantics of a PTA). *Let $\mathcal{A} = (L, l_0, \Sigma, X, P, E, \mathsf{Inv})$ be a PTA and v be a $\mathbb{R}$-valuation on P. The semantics of $v(\mathcal{A})$ is given by the timed transition system $(Q, q_0, \rightarrow)$ with:*

- *$Q = \{(l, u) \in L \times \mathbb{R}^X_{\geq 0} \mid u(v(\mathsf{Inv}(l)))$ is true$\}$;*
- *$q_0 = (l_0, \mathbf{0}_X)$ ($q_0 \in Q$ due to the special form of invariants);*
- *Time transitions: $(l, u) \xrightarrow{d} (l, u+d)$, with $d \geq 0$, iff $\forall d' \in [0, d], (l, u+d') \in Q$;*
- *Action transitions: $(l, u) \xrightarrow{a} (l', u')$, with $a \in \Sigma$, iff $(l, u), (l', u') \in Q$, there exists an edge $(l, a, \gamma, R, l') \in E$, $u' = u[R]$ and $u(v(\gamma))$ is true.*

A finite *run* is a finite sequence $\rho = q_1 a_1 q_2 a_2 \dots a_{n-1} q_n$ such that for all i, $q_i \in Q$, $a_i \in \Sigma \cup \mathbb{R}_{\geq 0}$ and $q_i \xrightarrow{a_i} q_{i+1}$. For any run ρ, we define $\mathsf{Edges}(\rho) = e_1 \dots e_m$ as the sequence of edges of the automaton taken in the discrete transitions along the run. We suppose without loss of generality that these edges are indeed thus uniquely defined. A run is *maximal* if it either is infinite or cannot be extended. We denote by $\mathsf{Runs}(v(\mathcal{A}))$ the set of runs that start in the initial state of $v(\mathcal{A})$.

We can define several interesting parametric problems on PTA. Among them we can ask: does there exist valuations for the parameters such that some property is satisfied? And, even more interesting, can we compute all of these valuations? Given a class of problems $\mathcal{P}$ (e.g. reachability, unavoidability, TCTL model-checking, control) these two questions translate into what we respectively call the $\mathcal{P}$-emptiness and the $\mathcal{P}$-synthesis problems:

$\mathcal{P}$-emptiness problem:

Inputs : A PTA $\mathcal{A}$ and an instance ϕ of $\mathcal{P}$

Problem : Is the set of valuations v of the parameters such that $v(\mathcal{A})$ satisfies ϕ empty?

$\mathcal{P}$-synthesis problem:

Inputs : A PTA $\mathcal{A}$ and an instance ϕ of $\mathcal{P}$

Problem : Compute the set of valuations v of the parameters such that $v(\mathcal{A})$ satisfies ϕ.

In this paper we mainly focus on reachability and unavoidability properties and call the corresponding problems EF and AF. Thus, given a PTA $\mathcal{A}$ and a subset G of its locations, EF-emptiness asks: does there exist a valuation v of the parameters such that G is reachable in $v(\mathcal{A})$? And AF-emptiness asks: does there exist a valuation v of the parameters such that all maximal runs in $v(\mathcal{A})$ go through G? The related synthesis problems immediately follow.

In [3], the EF-emptiness problem was proved undecidable for PTA. We give further negative results in the next section.

3 L/U automata

The following syntactic subclass of PTA, called L/U-automaton, has been proposed in [12] as a decidable subclass for the EF-emptiness problem. It relies on the notion of upper and lower bounds for parameters:

Definition 3 (Lower and upper bounds). *Let γ be a single conjunct of a simple clock constraint on the set of clocks X and the set of parameters P. Let x be the clock variable occurring in γ. γ is an upper (resp. lower) bound constraint if $\mathsf{Coeff}(x)$ is negative (resp. positive).*

p is an upper (resp. lower) bound in the PTA $\mathcal{A}$ if for each conjunct γ of each simple clock constraint in the guards and invariants of $\mathcal{A}$, either $\mathsf{Coeff}(\gamma, p) = 0$ or p is an upper (resp. lower) bound in γ.

Definition 4 (L/U-automaton). *A PTA $\mathcal{A}$ is a L/U-automaton if every parameter is either an upper bound or a lower bound in $\mathcal{A}$ but not both.*

3.1 Emptiness

EF-emptiness is PSPACE for L/U automata [12] and, more generally, emptiness, universality and finiteness of the valuation set are PSPACE-complete for infinite runs acceptance properties [6]. These good results are based on a *monotonicity* property that L/U automata have: decreasing lower bounds or increasing upper bounds only *add* behaviors. So if we set all lower bounds to 0 and all upper bounds to a large enough constant that we can compute, then the resulting timed automaton contains all the possible behaviors. This makes these automata very well-suited for reachability-like properties. For other properties however this is not enough. For AF properties, increasing lower bounds or decreasing upper bounds can suppress a run that was a counter-example to the property, and then make this property true.

We now indeed prove, with a reduction from the halting problem of 2-counter machines [17], that the AF-emptiness problem for L/U automata is undecidable. We actually prove it in a more general setting by addressing a further subclass of L/U-automata that allows only for upper bounds:

Definition 5 (L- and U-automaton). *A PTA $\mathcal{A}$ is a U-automaton (resp. L-automaton) if every parameter is an upper (resp. lower) bound in $\mathcal{A}$.*

Theorem 1. *The AF-emptiness problem is undecidable for U-automata.*

3.2 Synthesis

In [6] the authors prove that for L-automata and U-automata, the solution to the synthesis problem for infinite runs acceptance properties can be explicitly computed as a linear constraint of size doubly-exponential in the number of parameters. That is to say this solution can be expressed as a finite union of convex polyhedra.

With a different look at the idea used in [6] to prove that the *constrained* (i.e. with initial constraints) emptiness problem for infinite runs acceptance properties is undecidable for L/U-automata, we can express a new and quite strong result on the solution to the EF-synthesis problem for L/U-automata.

Theorem 2. *If it can be computed, the solution to the EF-synthesis problem for L/U-automata cannot be represented using any formalism for which emptiness of the intersection with equality constraints is decidable.*

Note that, in particular, Theorem 2 rules out the possibilty of computing the solution set as a finite union of polyhedra.

4 Integer Parametric Problems

The decidability results related to emptiness problems for L/U-automata are mixed: properties related to reachability are decidable but very simple properties that are not compatible with the monotonicity property, like unavoidability, are undecidable even for the very restricted subclass of U-automata. As for the actual synthesis of the constraints between parameters that describe the set of valuations that satisfy even the simple case of reachability properties, we have to resort to L- or U- automata, that have severely restricted modeling capacities.

We therefore advocate for different kinds of restrictions to PTA. Note that with only one irrational constant in the guards of timed automata, reachability is undecidable [16]. For all practical purposes these constants are actually always chosen as integers. Even if we insist on rationals, we can make those integers through adequate scaling and we usually have to since most tools only allow them as integers. So, instead of using syntactical restrictions in the guards and invariants of PTA, we think it makes a lot of sense to search for parameter values as *bounded integers*.

We therefore focus on synthesizing (or just proving the existence of) integer valuations for the parameters: a valuation v on a set X is an integer valuation if $\forall x \in X, v(x) \in \mathbb{Z}$. This induces new emptiness and synthesis problems that we call *integer* problems (e.g., integer EF-emptiness problem).

By insisting that these integer values should be bounded we will be (unsurprisingly) able to make all parametric problems decidable, provided the associated non-parametric problem, obtained by choosing one particular valuation, is decidable of course.

These decidability results are however only interesting for practical purposes if we can solve the corresponding synthesis problems symbolically, *i.e.*, without explicitly enumerating all the possible valuations.

To this end, we first introduce symbolic semi-algorithms to solve the synthesis problems in the general setting (possibly non integer valuations) that are based on a quite straightforward extension of the symbolic zone-based state-space exploration that is ubiquitous for timed automata [14].

4.1 Symbolic states for PTA

We therefore extend the notion of symbolic state for PTA, as well as the usual operators associated to them:

Definition 6 (Symbolic state). *A* symbolic state *of a PTA $\mathcal{A}$, with set of clocks X and set of parameters P, is a pair (l, Z) where l is a location of $\mathcal{A}$ and Z is a set of valuations on $X \cup P$.*

For state space computation, we define classical operations on valuation sets:

- future: $Z^{\nearrow} = \{v' \mid v'(x) = v(x) + d, d \geq 0 \text{ if } x \in X; v'(x) = v(x) \text{ if } x \in P\}$;
- reset of the clock variables in set $R \subseteq X$: $Z[R] = \{v[R] \mid v \in Z\}$;
- initial symbolic state of the PTA $\mathcal{A} = (L, l_0, \Sigma, X, P, E, \mathsf{Inv})$: $\mathsf{Init}(\mathcal{A}) = (l_0, \{v \in \mathbb{R}^{X \cup P} \mid v_{|X} \in \{\mathbf{0}_X\}^{\nearrow} \cap v_{|P}(\mathsf{Inv}(l_0))_{|X}\})$;
- successor by edge $e = (l, a, \gamma, R, l')$: $\mathsf{Succ}((l, Z), e) = (l', (Z \cap \gamma)[R]^{\nearrow} \cap \mathsf{Inv}(l'))$

It follows from [12] that all reachable symbolic states of a PTA are convex polyhedra. Also, the following properties trivially hold:

Property 1. The symbolic state abstraction satisfies: (1) Succ is non decreasing, and for any reachable symbolic state (l, Z): (2) Z is convex, (3) if v is an integer parameter valuation then $v(Z)$ is a (convex) zone with integer vertices and (4) for any edge e, $\mathsf{Succ}((l, Z), e) = \bigcup_{v \in Z_{|P}} \mathsf{Succ}((l, v(Z)), e)$

We can extend the Succ operator to a sequence of edges $e_1 \dots e_n$ by defining $\mathsf{Succ}((l, Z), e_1 e_2 \dots e_n) = \mathsf{Succ}(\dots \mathsf{Succ}(\mathsf{Succ}((l, Z), e_1), e_2) \dots, e_n)$. We then have:

Lemma 1. *For any valuation $v \in \mathbb{Q}^P$ and edges $e_1, \dots, e_n$, if we note $(l, Z) = \mathsf{Succ}(\mathsf{Init}(\mathcal{A}), e_1 \dots e_n)$: $v \in Z_{|P}$ iff $\exists \rho \in \mathsf{Runs}(v(\mathcal{A}))$ s.t. $\mathsf{Edges}(\rho) = e_1 \dots e_n$*

4.2 Semi-algorithms for the general synthesis problems

The following two algorithms also are natural extensions of their timed automata counterpart. The difficulty here is the handling of the parameter valuations. For $S = (l, Z)$, when non-ambiguous, we use S in place of l or Z to simplify the writing a bit.

For EF we aggregate the valuations found when reaching the locations in G:

$$\mathsf{EF}_G(S, M) = \begin{cases} S_{|P} & \text{if } S \in G \\ \emptyset & \text{if } S \in M \\ \bigcup_{\substack{e \in E \\ S' = \mathsf{Succ}(S, e)}} \mathsf{EF}_G(S', M \cup \{S\}) & \text{otherwise} \end{cases}$$

For AF, when a path reaches G we retain all valuations that also preserve the other paths that reach G (hence the intersection). If a path cannot reach G we cut it by keeping valuations that make it impossible (in the complement of the projection on the parameters).

$$\mathsf{AF}_G(S,M) = \begin{cases} S_{|P} & \text{if } S \in G \\ \emptyset & \text{if } S \in M \\ \bigcup_{\substack{e\in E \\ S'=\mathsf{Succ}(S,e)}} \mathsf{AF}_G\big(S', M \cup \{S\}\big) \cap \bigcap_{\substack{e'\in E \\ e'\neq e \\ S''=\mathsf{Succ}(S,e')}} \mathsf{AF}_G\big(S'', M \cup \{S\}\big) \cup (\mathbb{R}^P \setminus S''_{|P}) & \text{otherwise} \end{cases}$$

In both algorithms, conditions are evaluated from top to bottom and M represents a *passed list* of symbolic states. It records the symbolic states that have already been explored on a given path. It is possible to have a global passed list shared between all paths but this complicates the writing of the algorithms, especially AF. Initially, M is empty and, for the EF-synthesis problem, for instance, the invocation of EF is, for the PTA $\mathcal{A}$ and a subset of its locations G: $\mathsf{EF}_G(\mathsf{Init}(\mathcal{A}), \emptyset)$.

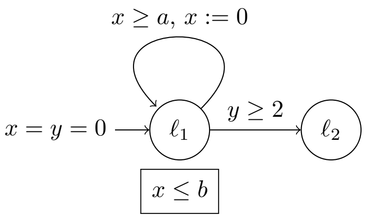

Fig. 1. The PTA $\mathcal{A}_1$ with clocks x and y and parameters a and b

The following theorem states that EF and AF are semi-algorithms for their respective synthesis problems.

Theorem 3. *For any PTA $\mathcal{A}$ and any subset of its locations G, upon termination, $\mathsf{EF}_G(\mathsf{Init}(\mathcal{A}), \emptyset)$ (resp. $\mathsf{AF}_G(\mathsf{Init}(\mathcal{A}), \emptyset)$) is the solution to the EF-synthesis (resp. AF-synthesis) problem for PTA $\mathcal{A}$ and set of locations to reach G.*

Example 1. In the PTA $\mathcal{A}_1$ in Figure 1, after $n > 0$ iterations of the loop, we get the following valuation set $Z_n = \{0 \leq x \leq b, 0 \leq y, a \leq b, 0 \leq na \leq y - x \leq (n+1)b\}$. We can see that we will never have $Z_m = Z_n$ for $m \neq n$ and therefore neither $\mathsf{EF}_{\{\ell_2\}}(\mathsf{Init}(\mathcal{A}_1), \emptyset)$ nor $\mathsf{AF}_{\{\ell_2\}}(\mathsf{Init}(\mathcal{A}_1), \emptyset)$ will terminate.

4.3 Extension for the integer synthesis problems

We now modify the two semi-algorithms to symbolically compute integer valuations. For that we use the notion of integer hull.

Let $n \in \mathbb{N}$ and let Y be a subset of $\mathbb{R}^n$. We denote by $\mathsf{Conv}(Y)$ the *convex hull* of Y, *i.e.* the smallest convex set containing Y. $\mathsf{IntVects}(Y)$ denotes the subset of all elements of Y with integer coordinates. We call those elements *integer valuations* (or vectors). The *integer hull* of Y, denoted by $\mathsf{IntHull}(Y)$ is the smallest convex set containing all the integer vectors of Y, *i.e.* $\mathsf{IntHull}(Y) = \mathsf{Conv}(\mathsf{IntVects}(Y))$.

We extend $\mathsf{IntVects}$ to symbolic states by: $\mathsf{IntVects}((l,Z)) = (l, \mathsf{IntVects}(Z))$ and extend likewise all the other operators on valuation sets.

We make the following assumption on the symbolic states of PTA.

Assumption 1 *Any non-empty symbolic state computed through the* Succ *operator contains at least one integer point.*

Though there exist pathological PTA for which this is not true, we believe that this is not a severe restriction in practice. For instance, considering only non-strict constraints as invariants, still possibly with strict guards, is an easy restriction that is enough to ensure this property. In any case, since we will compute the polyhedra and their integer hulls, Assumption 1 can be verified on-the-fly, at a very low additional cost, during the computation. Under this assumption, we now show that to address our integer parametric problems, it is sufficient to consider the integer hulls of the (valuations in the) symbolic states.

We therefore consider the semi-algorithm IEF (resp. IAF) obtained from EF (resp. AF) by replacing all occurrences of the operator Succ by ISucc with $\mathsf{ISucc}((l,Z),e) = \mathsf{IntHull}(\mathsf{Succ}(l,Z),e)$. We also extend ISucc to edge sequences in the same way as for Succ.

To prove the correctness of these two new algorithms, we rely on Lemma 2 that is the equivalent in our integer setting of Lemma 1:

Lemma 2. *For any integer valuation $v \in \mathbb{Z}^P$ and edges $e_1, \dots, e_n$, if $(l,Z) = \mathsf{ISucc}(\mathsf{Init}(\mathcal{A}), e_1 \dots e_n)$: $v \in Z_{|P}$ iff $\exists \rho \in \mathsf{Runs}(v(\mathcal{A}))$ s.t. $\mathsf{Edges}(\rho) = e_1 \dots e_n$*

Finally, we can state the main result of this subsection: IEF and IAF are correct semi-algorithms for their respective *integer* synthesis problems.

Theorem 4. *For any PTA $\mathcal{A}$ and any subset of its locations G, upon termination, $\mathsf{IEF}_G(\mathsf{Init}(\mathcal{A}), \emptyset)$ (resp. $\mathsf{IAF}_G(\mathsf{Init}(\mathcal{A}), \emptyset)$) is the solution to the integer EF-synthesis (resp. AF-synthesis) problem for PTA $\mathcal{A}$ and set of locations to reach G.*

Example 2. Let us go back to the PTA $\mathcal{A}_1$ in Figure 1. After n iterations of the loop, we still get the same valuation set $Z_n = \{0 \leq x \leq b, 0 \leq y, \leq na \leq y - x \leq (n+1)b\}$. This is because Z_n is its own integer hull. So, again neither $\mathsf{IEF}_{\{\ell_2\}}(\mathsf{Init}(\mathcal{A}_1), \emptyset)$ nor $\mathsf{IAF}_{\{\ell_2\}}(\mathsf{Init}(\mathcal{A}_1), \emptyset)$ will terminate.

4.4 Termination for the bounded integer synthesis problems

To ensure termination of semi-algorithms IEF and IAF, we now consider that we are searching for bounded integer parameter valuations, *i.e.*, given *a priori* some $M, N \in \mathbb{N}$, we search for valuations in $[-M..N]^P$. Again, this induces new emptiness and synthesis problems that we call (M,N)-*bounded integer* problems (e.g., $(100,100)$-bounded integer EF-emptiness problem).

First remark that, in a TA with $|L|$ locations and $R(m)$ regions (m being the maximal constant appearing in the constraints of the TA), if some location ℓ is reachable, then there exists a run that leads to ℓ and visits at most $|L| \times R(m)$ states. Since it takes at most 1 time unit to go from one region to another, the duration of this run is at most $|L| \times R(m)$ time units. So, if we add invariants $x \leq |L| \times R(m)$ for all clocks x in all the locations of the TA, we obtain an equivalent TA, with respect to location reachability and unavoidability. Since $R(m)$ is non-decreasing with m, this is also true if we increase the value of m.

Now, in our bounded integer parameters setting, we can compute a constant upper bound for each parametric linear expression used in the guards and invariants of the automaton. Let K be the maximum of those upper bounds and of the constants in the non-parametric constraints of the TA. Using the reasoning above, we can then add for all clocks x the invariant $x \leq |L| \times R(K)$ to all locations of our PTA and obtain an equivalent PTA, with respect to location reachability and unavoidability.

For such a PTA $\mathcal{A}$ with bounded clocks and for any valuation $v \in [-M..N]^P$, $v(\mathcal{A})$ is a TA with bounded clocks for which the finiteness of the number of zones computed with the Succ operator is thus ensured.

Let us define an extension of the Init operator that accepts a bound on the values of the parameters in the initial symbolic state (and therefore in the whole computation): for any $M, N \in \mathbb{N}$, $\mathsf{Init}_{M,N}(\mathcal{A}) = (l_0, \{v \in \mathbb{R}^{X \cup P} \mid v_{|X} \in \{\mathbf{0}_X\}^{\nearrow} \cap v_{|P}(\mathsf{Inv}(l_0))_{|X}$ and $v_{|P} \in [-M..N]^P\})$.

Theorem 4 can be naturally adapted to this setting in the following form:

Theorem 5. *For any $M, N \in \mathbb{N}$, any PTA $\mathcal{A}$ and any subset of its locations G, upon termination, $\mathsf{IEF}_G(\mathsf{Init}_{M,N}(\mathcal{A}), \emptyset)$ (resp. $\mathsf{IAF}_G(\mathsf{Init}_{M,N}(\mathcal{A}), \emptyset)$) is the solution to the (M, N)-bounded integer EF-synthesis (resp. AF-synthesis) problem for PTA $\mathcal{A}$ and set of locations to reach G.*

To prove the termination of our computations, we rely on Lemma 3, which states that computing the integer hull of a symbolic state is equivalent to separately computing each of its subsets corresponding to integer parameters and then taking the convex hull of their union.

Lemma 3. *For any symbolic state (l, Z) of the PTA $\mathcal{A}$ s.t. $\forall v \in \mathsf{IntVects}(Z_{|P})$, $v(Z)$ is convex and has integer vertices: $\mathsf{IntHull}(Z) = \mathsf{Conv}(\bigcup_{v \in \mathsf{IntVects}(Z_{|P})} v(Z))$*

We can finally prove that, in this setting, the semi-algorithms do terminate:

Theorem 6. *For any $M, N \in \mathbb{N}$, any PTA $\mathcal{A}$ and any subset of its locations G, Algorithms $\mathsf{IEF}_G(\mathsf{Init}_{M,N}(\mathcal{A}), \emptyset)$ and $\mathsf{IAF}_G(\mathsf{Init}_{M,N}(\mathcal{A}), \emptyset)$ terminate.*

Example 3. Consider once again the PTA $\mathcal{A}_1$ in Figure 1. We now suppose that both parameters are bounded and take their values, say in [0..3]. Then as seen above, we add the invariants $x \leq 4$ and $y \leq 4$ to both locations (4 is less than the bound proposed above but enough in this simple case and keeps the computation understandable). This preserves location-based reachability and unavoidability properties. Now, after $n > 0$ iterations of the loop with the "normal" Succ operator, we have the valuation set $Z_n = \{0 \leq a \leq 3, 0 \leq b \leq 3, a \leq b, 0 \leq x \leq$

$4, 0 \leq y \leq 4, x \leq b$, $na \leq y - x \leq (n+1)b\}$. If we do not suppose that a and b are integers, we still never have $Z_m = Z_n$ for any $m \neq n$. If we do suppose they are integers, we compute each time $Z'_n = \mathsf{IntHull}(Z_n)$. We have $Z'_0 = Z_0$, $Z'_1 = Z_1 \cap \{y \leq a+3, y \leq b+2\}$, $Z'_2 = Z_2 \cap \{x \leq b-2a+2, y \leq a+3, y-x \leq a+2\}$, $Z'_3 = Z_3 \cap \{a \leq 1, y \leq a+3, y \leq b+3a\}$, $Z'_4 = Z_4 \cap \{y - x = 4a, x \leq 3 - 3a, x \leq b - a\}$, and when $n \geq 5$, $Z'_n = Z'_{n+1} = \{a = 0, x = y, 0 \leq x \leq b, b \leq 3\}$. And therefore $\mathsf{IEF}_{\{\ell_2\}}(\mathsf{Init}_{0,3}(\mathcal{A}_1), \emptyset)$ terminates and its result is $b \in [1..3]$. Similarly, $\mathsf{IAF}_{\{\ell_2\}}(\mathsf{Init}_{0,3}(\mathcal{A}_1), \emptyset)$ terminates and its result is $a \in [1..3]$ and $b \in [a..3]$.

5 Complexity of the Integer Parametric Problems

When the possible values of the parameters are integer and bounded, we can enumerate all of the possible valuations in exponential time. And therefore, for all classes of problems $\mathcal{P}$ that are EXPTIME for TA, the $\mathcal{P}$-synthesis problem (and of course the $\mathcal{P}$-emptiness) can be solved in exponential time. Also, since the $\mathcal{P}$ problem for TA is always a special case of the $\mathcal{P}$-emptiness problem for PTA, for problems that are complete for some complexity class containing EXPTIME, we can deduce that the corresponding bounded integer emptiness problem is complete for the same complexity class. For instance, the reachability control problem is EXPTIME-complete for TA [13]. The corresponding parametric emptiness problem is: for a PTA $\mathcal{A}$ with actions partitioned between controllable and uncontrollable, does there exist a parameter valuation v such that there exists a controller for $v(\mathcal{A})$ that enforces the reachability of some location whatever the uncontrollable actions that occur? This problem is therefore EXPTIME-complete for bounded integer parameters.

For simpler problems, we have a better and a bit surprising result, using the classical construction of Savitch giving PSPACE=NPSPACE [18]:

Theorem 7. *The $\mathcal{P}$-emptiness problem for PTA with bounded integer parameters is PSPACE-complete for any class of problems $\mathcal{P}$ that is PSPACE-complete for TA.*

In particular the whole TCTL model-checking, including reachability and unavoidability, is PSPACE-complete for TA [1] and as a consequence, the corresponding emptiness problem, which includes EF-emptiness and AF-emptiness, is PSPACE-complete for PTA with bounded integer parameters.

Finally, it is important to remark that we cannot easily lift either of the boundedness or the integer assumptions: the EF-emptiness problem for PTA with *bounded rational* parameter values is undecidable [16], and Theorem 8 follows from the undecidability proof of [3]:

Theorem 8. *The EF-emptiness problem for PTA with* possibly unbounded integer *parameter values is undecidable.*

6 On Performance in Practice: Task Set Schedulability

The PTA in Figure 1 demonstrates that it is very easy to find an example for which the symbolic computation does not terminate without the bounded integer parameters restriction but one could object that this PTA models nothing real (if $a = 0$, there are zeno runs for instance). We now show with a very simple but realistic case-study that this restriction is also useful for real applications.

Consider the scheduling problem adapted from [8] for a non-preemptive setting: we have three real-time tasks τ_1, τ_2 and τ_3. τ_1 is periodic with period a and has an execution time $C_1 \in [10, b]$. τ_2 is sporadic: it has only a minimal delay between two activations and that delay is $2a$. The execution time of τ_2 is $C_2 \in [c, d]$, with $c \leq d$. Finally, τ_3 is periodic with period $3a$ and has an execution time $C_3 \in [20, 28]$. These three tasks are scheduled using a non-preemptive[1] priority policy defined by $\tau_1 > \tau_2 > \tau_3$. We say that the system is schedulable if each task always has at most one instance running, which is a safety property.

We can model this problem with a parametric time Petri net (which, as it will be bounded by the property can be seen as a subclass of PTA [19]) in Roméo. Schedulability is verified using implementations of the semi-algorithms EF and IEF presented in section 4. The symbolic computations use the state class abstraction of [5,19], which is specific to time Petri nets, and do not require extrapolation. It is however very similar to the zone-based abstraction and trivially satisfies Property 1. The rest of the results carry over to that abstraction without any difficulty. We use a machine with an Intel Core I7 at 2.3 GHz and 8 Gb RAM.

Table 1 provides some insight on the performance of Algorithm IEF and a comparison to Algorithm EF. The only difference in the implementations of the two algorithms is the application or not of the integer hull operator. The table shows the total time for the verifications, the part of it used for computing the integer hull for Algorithm IEF, and the memory consumptions. DNF means that the computation did not finish within 90 min (memory was not a problem here). The constraint generated for the first column is $a \geq 44$, for the second $a - b \geq 24, b \geq 10$, and for the third $a - b \geq 24, b \geq 10, 0 \leq c \leq 28$. For the fourth column the constraint is much more complex so we will not reproduce it here. Note that in all these cases some parameters are unbounded so an explicit enumeration of all possible parameter values coupled with an efficient (DBM-based or discrete-time decision diagram-based) verification was not possible (and termination of Algorithm IEF was actually not guaranteed).

With Table 2 we illustrate the smooth scaling of our approach with the value of upper bounds. Not that the performance of Algorithm IEF is actually worse when all parameters are bounded (compare with the fourth column of Table 1). This is due to the fact that our implementation uses inclusion for convergence, which is favored by the reduced number of constraints in the absence of upper bounds. In this setting, termination is guaranteed however.

[1] A running task cannot be interrupted even if another task with a greater priority is ready.

	$a \in [0, \infty)$ $b = 20$ $c = 18$ $d = 28$	$a \in [0, \infty)$ $b \in [10, \infty)$ $c = 18$ $d = 28$	$a \in [0, \infty)$ $b \in [10, \infty)$ $c \in [0, 28]$ $d = 28$	$a \in [0, \infty)$ $b \in [10, \infty)$ $c = 18$ $d \in [18, \infty)$	$a \in [0, \infty)$ $b \in [10, \infty)$ $c \geq 0$ $d \geq c$
IEF Time	1 s	2.8 s	27 s	840 s	DNF
Int. Hull	0.2 s (20%)	0.4 s (14%)	2.9 s (11%)	146 s (17%)	–
IEF Mem.	15 Mo	35 Mo	153 Mo	1289 Mo	–
EF Time	1.5 s	6.4 s	DNF	DNF	DNF
EF Mem.	19.6 Mo	55 Mo	–	–	–

Table 1. Usefulness of the Integer Hull

	$a \in [0, 100]$	$a \in [0, 1000]$	$a \in [0, 10000]$
IEF Time	1079 s	1150 s	1178 s
Int. Hull	166 s (15.4%)	167 s (14.5%)	168 s (14.3%)
IEF Mem.	1598 Mo	1667 Mo	1667 Mo

Table 2. Scaling a's upper bound for $b \in [10, 100], c = 18$ and $d \in [18, 100]$

7 Conclusion

We have presented novel results for the parametric verification of timed systems modeled as parametric timed automata. Our new negative results show that even when severely restricting the form of the parametric constraints we encounter undecidabilty for many interesting problems. So we have proposed instead to restrict the codomain of the valuations to bounded integers.

This is completely orthogonal to previous restriction schemes in the sense that it does not enforce any syntactic restriction on PTA, thus simplifying the modeling activity. Also experimental evidence shows that the symbolic approach we propose to avoid an explicit enumeration of all the possible parameter values is robust to scaling the bounds of the parameters (and improves on convergence even without any bounds in some cases).

Also, in this setting, most problems are of course decidable and we have proved that, for instance, emptiness for TCTL properties, which include reachability and unavoidability, is PSPACE-complete. We have also proved that lifting the boundedness or the integer assumption leads to undecidability. We have exhibited symbolic algorithms that allow to avoid the explicit enumeration of all possible valuations and implemented them in our tool Roméo [15].

Our current lines of work on this problem include improving the computation of the integer hulls, the search for less restrictive codomains for parameter valuations, and extension of this work for parametric timed games and PTA with stopwatches.

Acknowledgments The authors thank Claude Jard and Enea Zaffanella for the related discussions, as well as an anonymous reviewer from CONCUR'12 for her/his very useful comments.

References

1. R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, May 1993.
2. R. Alur and D. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
3. R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *ACM Symposium on Theory of Computing*, pages 592–601, 1993.
4. E. André, T. Chatain, E. Encrenaz, and L. Fribourg. An inverse method for parametric timed automata. In *RP workshop on Reachability Problems*, volume 223, pages 29–46, Liverpool, U.K., 2008.
5. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE trans. on soft. eng.*, 17(3):259–273, 1991.
6. L. Bozzelli and S. L. Torre. Decision problems for lower/upper bound parametric timed automata. *Formal Methods in System Design*, 35(2):121–151, 2009.
7. V. Bruyère and J.-F. Raskin. Real-time model-checking: Parameters everywhere. *Logical Methods in Computer Science*, 3(1):1–30, 2007.
8. G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario. Time state space analysis of real-time preemptive systems. *IEEE Trans. on Soft. Eng.*, 30(2):97–111, Feb. 2004.
9. L. Doyen. Robust parametric reachability for timed automata. *Information Processing Letters*, 102(5):208–213, 2007.
10. T. A. Henzinger, P.-H. Ho, and H. Wong-toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:460–463, 1997.
11. T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Inform. and Computation*, 111(2):193–244, 1994.
12. T. Hune, J. Romijn, M. Stoelinga, and F. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 52-53:183–220, 2002.
13. M. Jurdzinski and A. Trivedi. Reachability-time games on timed automata. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *34th International Colloquium on Automata, Languages and Programming (ICALP 2007)*, volume 4596 of *LNCS*, pages 838–849, Wroclaw, Poland, July 2007. Springer.
14. K. G. Larsen, P. Pettersson, and W. Yi. Model-Checking for Real-Time Systems. In *Fundamentals of Computation Theory*, volume 965 of *LNCS*, pages 62–88, 1995.
15. D. Lime, O. H. Roux, C. Seidner, and L.-M. Traonouez. Romeo: A parametric model-checker for Petri nets with stopwatches. In *15th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2009)*, volume 5505 of *LNCS*, pages 54–57, York, UK, Mar. 2009. Springer.
16. J. S. Miller. Decidability and complexity results for timed automata and semi-linear hybrid automata. In *Hybrid Systems: Computation and Control*, volume 1790 of *LNCS*, page 296–309. Springer, 2000.
17. M. Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall, 1967.
18. W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, Apr. 1970.
19. L.-M. Traonouez, D. Lime, and O. H. Roux. Parametric model-checking of stopwatch Petri nets. *Journal of Universal Computer Science*, 15(17):3273–3304, 2009.
20. I. Virbitskaite and E. Pokozy. Parametric behaviour analysis for time Petri nets. In *5th International Conference on Parallel Computing Technologies*, volume 1662 of *LNCS*, pages 134–140, London, UK, 1999. Springer-Verlag.
21. F. Wang. Parametric timing analysis for real-time systems. *Information and Computation*, 130(2):131–150, November 1996.