Integer-Complete Synthesis for Bounded Parametric Timed Automata

Étienne André, Didier Lime, Olivier Henri Roux

Integer-Complete Synthesis for Bounded Parametric Timed Automata*

Étienne André[1], Didier Lime[2], and Olivier H. Roux[2]

[1] Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, UMR 7030, France
[2] École Centrale de Nantes, IRCCyN, CNRS, UMR 6597, France

Abstract. Ensuring the correctness of critical real-time systems, involving concurrent behaviors and timing requirements, is crucial. Parameter synthesis aims at computing dense sets of valuations for the timing requirements, guaranteeing a good behavior. However, in most cases, the emptiness problem for reachability (*i.e.,* whether there exists at least one parameter valuation for which some state is reachable) is undecidable and, as a consequence, synthesis procedures do not terminate in general, even for bounded parameters. In this paper, we introduce a parametric extrapolation, that allows us to derive an underapproximation in the form of linear constraints containing all the integer points ensuring reachability or unavoidability, and all the (non-necessarily integer) convex combinations of these integer points, for general PTA with a bounded parameter domain. Our algorithms terminate and can output constraints arbitrarily close to the complete result.

1 Introduction

The verification of systems mixing time and concurrency is a notoriously difficult problem. Timed automata (TA) [AD94] are a powerful formalism for which many interesting problems (including the reachability of a location) are decidable. However, the classical definition of TA is not tailored to verify systems only partially specified, especially when the value of some timing constants is not yet known. Parametric timed automata (PTA) [AHV93] leverage this problem by allowing the specification and the verification of systems where some of the timing constants are parametric. This expressive power comes at the price of the undecidability of most interesting problems.

Related Work Parametric Timed Automata were introduced in [AHV93]. The simple problem of the existence of a parameter valuation such that some location is reachable is undecidable in both discrete and dense-time even with only three parametric clocks (*i.e.,* clocks compared to a parameter) [Mil00,BBLS15], and even with only strict constraints [Doy07]. It is decidable for a single parametric

* This is the author version of the paper of the same name accepted for publication at RP 2015. The final publication is available at http://www.springer.com. This work is partially supported by the ANR national research program "PACS" (ANR-2014).

clock in discrete and dense time [AHV93], and in discrete time with two parametric clocks and one parameter (and arbitrarily many non-parametric clocks) [BO14], or for a single parametric clock (with arbitrarily many non-parametric clocks) [BBLS15]. More complex properties expressed in parametric TCTL have been studied in [Wan96,BR07]. PTA subclasses have been studied, most notably L/U-automata, for which the problem is decidable [HRSV02], but no synthesis algorithm is provided, and there are indeed practical difficulties in proposing one [JLR15]. When further restricting to L- or U-automata, the integer-valued parameters can be synthesized however [BL09]. The trace preservation problem (*i.e.,* given a reference parameter valuation whether there exists another valuation for which the discrete behavior is the same) is undecidable for both general PTA and L/U-PTA [AM15].

In [JLR15], we focus on integer-valued bounded parameters (but still considering dense-time), for which many problems are obviously decidable, and we provide symbolic algorithms to compute the set of correct integer parameter values ensuring a reachability ("IEF") or unavoidability ("IAF") property. A drawback is that returning only integer points prevents designers to use the synthesized constraint to study the robustness or implementability of their system (see, *e.g.,* [Mar11]).

Contribution We propose terminating algorithms that compute a dense under-approximation of the set of parameter values ensuring reachability or unavoidability in bounded PTA (*i.e.,* PTA with a bounded parameter domain). These under-approximations are "integer-complete" in the sense that they are guaranteed to contain at least all the correct integer values given in the form of a finite union of polyhedra; they are also "almost-complete" in the sense that the only points that may not be included in the result are non-integer (rational) points beyond the last integer point in a convex polyhedron. To the best of our knowledge, these algorithms are the first synthesis algorithms that return an almost-complete result for a subclass of PTA (namely bounded PTA) for which the corresponding emptiness problems are undecidable; in fact, with the exception of two subclasses of L/U-PTA (namely L-PTA and U-PTA) considered over discrete parameter valuations [BL09], we are not aware of any terminating synthesis algorithm deriving a complete or an almost-complete result. Our algorithms also quantify the "size" of the resulting constraint, *i.e.,* they return all valuations except possibly some non-integer points beyond the "last" integer points.

While of great practical interest, our algorithms are in essence quite similar to those of [JLR15]. We however demonstrate that while the algorithms from [JLR15] also return a symbolic representation of the "good" integer parameter values, interpreting the result of the IAF algorithm as dense is not correct in the sense that some non-integer parameter values in that result may not ensure the unavoidability property. Furthermore, since we produce real-valued parameter values, we cannot use anymore the result from [JLR15] ensuring termination of the algorithms, which allows to derive a bound on clock values but relies on the parameters being bounded integers. The main *technical* contribution of

this paper is therefore the derivation of a maximum-constant-based parametric extrapolation operator for bounded PTA that ensures termination of our algorithms. To the best of our knowledge this operator is the first of its kind.

Finally, we have implemented the two algorithms and briefly report on them.

Outline We first recall the necessary definitions in Section 2. We present our parametric extrapolation in Section 3. We then introduce our terminating algorithms (namely RIEF, RIAF) in Section 4. We conclude in Section 5. All proofs are given in the appendix.

2 Preliminaries

2.1 Clocks, Parameters and Constraints

Throughout this paper, we assume a set $X = \{x_1, \ldots, x_H\}$ of *clocks*, *i.e.,* real-valued variables that evolve at the same rate. A clock valuation is a function $w : X \to \mathbb{R}_+$. We identify a clock valuation w with the *point* $(w(x_1), \ldots, w(x_H))$. We write $X = 0$ for $\bigwedge_{1 \leq i \leq H} x_i = 0$. Given $d \in \mathbb{R}_+$, $w + d$ denotes the valuation such that $(w + d)(x) = w(x) + d$, for all $x \in X$.

We assume a set $P = \{p_1, \ldots, p_M\}$ of *parameters*, *i.e.,* unknown constants. A parameter *valuation* v is a function $v : P \to \mathbb{Q}_+$. We identify a valuation v with the *point* $(v(p_1), \ldots, v(p_M))$. An *integer* parameter valuation is a valuation $v : P \to \mathbb{N}$.

In the following, we assume $\sim \in \{<, \leq, \geq, >\}$. A *constraint* C (*i.e.,* a convex polyhedron) over $X \cup P$ is a conjunction of inequalities of the form $lt \sim 0$, where lt denotes a linear term over $X \cup P$ of the form $\sum_{1 \leq i \leq H} \alpha_i x_i + \sum_{1 \leq j \leq M} \beta_j p_j + d$, with $\alpha_i, \beta_j, d \in \mathbb{Z}$. Given a parameter valuation v, $v(C)$ denotes the constraint over X obtained by replacing each parameter p in C with $v(p)$. Likewise, given a clock valuation w, $w(v(C))$ denotes the expression obtained by replacing each clock x in $v(C)$ with $w(x)$. We say that v *satisfies* C, denoted by $v \models C$, if the set of clock valuations satisfying $v(C)$ is nonempty. Given a parameter valuation v and a clock valuation w, we denote by $w|v$ the valuation over $X \cup P$ such that for all clocks x, $w|v(x) = w(x)$ and for all parameters p, $w|v(p) = v(p)$. We use the notation $w|v \models C$ to indicate that $w(v(C))$ evaluates to true. We say that C is *satisfiable* if $\exists w, v$ s.t. $w|v \models C$. An *integer point* is $w|v$, where w is an integer clock valuation, and v is an integer parameter valuation. We define the *time elapsing* of C, denoted by $C^{\nearrow}$, as the constraint over X and P obtained from C by delaying all clocks by an arbitrary amount of time. We define the *past* of C, denoted by $C^{\swarrow}$, as the constraint over X and P obtained from C by letting time pass backward by an arbitrary amount of time (see [JLR15]). Given $R \subseteq X$, we define the *reset* of C, denoted by $[C]_R$, as the constraint obtained from C by resetting the clocks in R, and keeping the other clocks unchanged. We denote by $C{\downarrow_P}$ the projection of C onto P, *i.e.,* obtained by eliminating the clock variables.

A *guard* g is a constraint over $X \cup P$ defined by inequalities of the form $x \sim z$, where z is either a parameter or a constant in $\mathbb{Z}$. Let *plt* denote a

parametric linear term over P, that is a linear term without clocks ($\alpha_i = 0$ for all i). A *zone* C is a constraint over $X \cup P$ defined by inequalities of the form $x_i - x_j \sim plt$, where $x_i, x_j \in X \cup \{x_0\}$, where x_0 is the zero-clock always equal to 0.

A *parametric constraint* K is a constraint over P defined by inequalities of the form $plt \sim 0$. We denote by $\top$ (resp. $\bot$) the parametric constraint that corresponds to the set of all possible (resp. the empty set of) parameter valuations.

2.2 Parametric Timed Automata

Parametric timed automata (PTA) extend timed automata with parameters within guards and invariants in place of integer constants [AHV93].

Definition 1. *A PTA $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, L, l_0, X, P, I, E)$, where: i) Σ is a finite set of actions, ii) L is a finite set of locations, iii) $l_0 \in L$ is the initial location, iv) X is a set of clocks, v) P is a set of parameters, vi) I is the invariant, assigning to every $l \in L$ a guard $I(l)$, vii) E is a set of edges $e = (l, g, a, R, l')$ where $l, l' \in L$ are the source and destination locations, $a \in \Sigma$, $R \subseteq X$ is a set of clocks to be reset, and g is a guard.*

Given a parameter valuation v, we denote by $v(\mathcal{A})$ the non-parametric timed automaton where all occurrences of a parameter p_i have been replaced by $v(p_i)$.

Definition 2 (Semantics of a TA). *Given a PTA $\mathcal{A} = (\Sigma, L, l_0, X, P, I, E)$, and a parameter valuation v, the concrete semantics of $v(\mathcal{A})$ is given by the timed transition system $(Q, q_0, \Rightarrow)$, with $Q = \{(l, w) \in L \times \mathbb{R}_+^H \mid v|w \models I(l)\}$, $q_0 = (l_0, X = 0)$, $((l, w), e, (l', w')) \in \Rightarrow$ if $\exists w'' : (l, w) \xrightarrow{e} (l', w'') \xrightarrow{d} (l', w')$, with: $(l, w) \xrightarrow{e} (l', w')$, if $(l, w), (l', w') \in Q$, there exists $e = (l, g, a, R, l') \in E$, $w' = [w]_R$, and $v|w \models g$; and $(l, w) \xrightarrow{d} (l, w + d)$, with $d \in \mathbb{R}_+$, if $\forall d' \in [0, d], (l, w + d') \in Q$.*

We refer to states of a TA as concrete states. A concrete run of a TA is an alternating sequence of (concrete) states of Q and edges of the form $s_0 \overset{e_0}{\Rightarrow} s_1 \overset{e_1}{\Rightarrow} \cdots \overset{e_{m-1}}{\Rightarrow} s_m$, such that for all $i = 0, \ldots, m-1$, $e_i \in E$, and $(s_i, e_i, s_{i+1}) \in \Rightarrow$. Given a state $s = (l, w)$, we say that s is reachable (or that $v(\mathcal{A})$ reaches s) if s belongs to a run of $v(\mathcal{A})$.

Symbolic states We now recall the symbolic semantics of PTA: A *symbolic state* of a PTA $\mathcal{A}$ is a pair (l, C) where $l \in L$ is a location, and C its associated zone. A state $s = (l, C)$ is v-compatible if $v \models C$. The initial state of $\mathcal{A}$ is $s_0 = (l_0, (X = 0)^{\nearrow} \wedge I(l_0))$. The symbolic semantics relies on the Succ operation. Given a symbolic state $s = (l, C)$ and an edge $e = (l, g, a, R, l')$, $\mathsf{Succ}(s, e) = (l', C')$, with $C' = \left([(C \wedge g)]_R\right)^{\nearrow} \cap I(l')$.

A symbolic run of a PTA is an alternating sequence of symbolic states and edges of the form $s_0 \overset{e_0}{\Rightarrow} s_1 \overset{e_1}{\Rightarrow} \cdots \overset{e_{m-1}}{\Rightarrow} s_m$, such that for all $i = 0, \ldots, m-1$, $e_i \in E$, and s_{i+1} belongs to $\mathsf{Succ}(s_i, e)$. Given a state s, we say that s is reachable

if s belongs to a run of $\mathcal{A}$. A maximal run is a run that is either infinite, or that cannot be extended.

Given a PTA $\mathcal{A}$ and a parameter valuation v, given a concrete state (l, w) of $v(\mathcal{A})$ and a symbolic state (l', C) of $\mathcal{A}$, we write $(l, w) \in v((l', C))$ if $l = l'$ and $w \models v(C)$.

In this paper, we will consider *bounded* PTA. A bounded parameter domain assigns to each parameter a minimum integer bound and a maximum integer bound. That is, each parameter p_i ranges in an interval $[a_i, b_i]$, with $a_i, b_i \in \mathbb{N}$. Hence, a bounded parameter domain is a hyperrectangle in M dimensions.

Integer hulls We briefly recall some definitions from [JLR15]. Let C be a convex polyhedron. C is topologically closed if it can be defined using only non-strict inequalities.[1] The integer hull of a topologically closed polyhedron, denoted by $\mathsf{IH}(C)$, is defined as the convex hull of the integer vectors of C, *i.e.,* $\mathsf{IH}(C) = \mathsf{Conv}(\mathsf{IV}(C))$, where Conv denotes the convex hull, and IV the set of vectors with integer coordinates.

We treat integer hulls for finite unions of convex polyhedra in a manner similar to [JLR15]: given a (possibly non-convex) finite union of convex polyhedra $\bigcup_i C_i$, we write $\mathsf{IH}(\bigcup_i C_i)$ for the set $\bigcup_i(\mathsf{IH}(C_i))$. Given a symbolic state $s = (l, C)$, we often write $\mathsf{IH}(s)$ for $(l, \mathsf{IH}(C))$.

Decision and Computation Problems Given a class of decision problems $\mathcal{P}$ (reachability, unavoidability, etc.), we consider the problem of synthesizing the set (or part of it) of parameter valuations v such that $v(\mathcal{A})$ satisfies ϕ.

Here, we mainly focus on reachability (*i.e.,* does there exist a run that goes through some goal locations) called here EF, and unavoidability (*i.e.,* do all maximal runs go through some goal locations) called here AF.

3 Parametric Extrapolation

In this section, we present an extrapolation based on the classical k-extrapolation used for the zone-abstraction for timed automata, but this time in a parametric setting. (Proofs can be found in Appendix A.)

First, let us motivate the use of an extrapolation. Consider the PTA in Fig. 1. After a number n of times through the loop, we get constraints in l_1 of the form $0 \leq x - y \leq n \times p$, with n growing without bound. Even if the parameter p is bounded (*e.g.,* in $[0, 1]$), the time necessary to reach location l_4 is unbounded. This was not the case in [JLR15] due to the fact that parameters were integers. Hence, on this PTA, we cannot just apply the integer hull (as in [JLR15]) to ensure termination of our algorithms.

[1] We only define here the integer hull of a topologically closed polyhedron. In fact, any non-closed polyhedron can be represented by a closed polyhedron with one extra dimension [HPR94]. Direct handling of not-necessarily-closed (NNC) polyhedra raises no theoretical issue but would impair the readability of this paper (see [JLR15]).

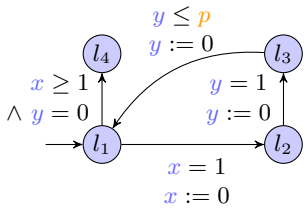
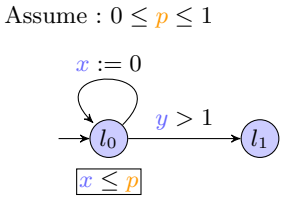


Fig. 1: Motivating PTA

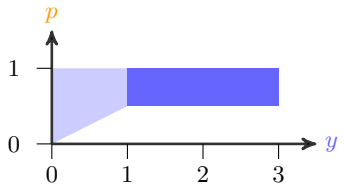
Now, we will show that the union for all values of the parameters of the classical k-extrapolation used for the zone-abstraction for timed automata leads to a non-convex polyhedron. Let us consider the PTA in Fig. 2a with a parameter p such that $0 \leq p \leq 1$. By taking n times the loop we obtain:

$$0 \leq x \leq p \quad \wedge \quad 0 \leq y - x \leq (n+1) \times p \quad \wedge \quad 0 \leq p \leq 1$$

The greatest constant of the model is $k = 1$. After one loop, y can be greater than 1. Then, for each value of p, we can apply the classical k-extrapolation used for timed automata (as recalled in [BBLP06]) of the corresponding zone. The union for all values of p of these extrapolations, projected to the plan (y, p) is depicted by the plain blue part (light and dark blue) of Fig. 2b. The obtained polyhedron is non-convex.



(a) PTA



(b) Extrapolation

Fig. 2: Example illustrating the non-convex parametric extrapolation

Let us now introduce our concept of (M, X)-extrapolation.

For any zone C and variable x, we denote by $\mathsf{Cyl}_x(C)$ the *cylindrification* of C along variable x, *i.e.*, $\mathsf{Cyl}_x(C) = \{w \mid \exists w' \in C, \forall x' \neq x, w'(x') = w(x') \text{ and } w(x) \geq 0\}$. This is a usual operation that consists in *unconstraining* variable x.

Definition 3 ((M,x)-extrapolation). *Let C be a polyhedron. Let M be a non-negative integer constant and x be a clock. The (M,x)-extrapolation of C, denoted by $\mathsf{Ext}_x^M(C)$, is defined as:*

$$\mathsf{Ext}_x^M(C) = \big(C \cap (x \leq M)\big) \cup \mathsf{Cyl}_x\big(C \cap (x > M)\big) \cap (x > M).$$

Given $s = (l, C)$, we write $\mathsf{Ext}_x^M(s)$ for $\mathsf{Ext}_x^M\big(C\big)$.

To illustrate the (M, X)-extrapolation, we go back to the example of Fig. 2a after one loop. C is the polyhedron for $n = 1$. $\mathsf{Ext}_y^1(C)$ is depicted in Fig. 2b by the plain blue part as follows: $\big(C \cap (y \leq 1)\big)$ is in light blue and $\mathsf{Cyl}_y\big(C \cap (y > 1)\big) \cap (y > 1)$ is in dark blue. Note that for this example the $(1, y)$-extrapolation gives the same result as the union for all values of the parameter p of the classical extrapolation for timed automata. Lemma 1 follows from Definition 3.

Lemma 1. *For all polyhedra C, integers $M \geq 0$ and clock variables x and x', we have $\mathsf{Ext}_x^M\big(\mathsf{Ext}_{x'}^M(C)\big) = \mathsf{Ext}_{x'}^M\big(\mathsf{Ext}_x^M(C)\big)$.*

Proof (sketch). The result comes from the following facts:

1. $\mathsf{Cyl}_x\big(\mathsf{Cyl}_{x'}(C)\big) = \mathsf{Cyl}_{x'}\big(\mathsf{Cyl}_x(C)\big)$;
2. for $x \neq x'$, $\mathsf{Cyl}_x(C) \cap (x' \sim M) = \mathsf{Cyl}_x\big(C \cap (x' \sim M)\big)$ for $\sim \in \{<, \leq, \geq, >\}$.

We can now consistently define the (M, X)-extrapolation operator:

Definition 4 ((M, X)-extrapolation). *Let M be a non-negative integer constant and X be a set of clocks. The (M, X)-extrapolation operator Ext_X^M is defined as the composition (in any order) of all Ext_x^M, for all $x \in X$. When clear from the context we omit X and only write M-extrapolation or Ext^M.*

In the rest of this section, we prove most results on the extrapolation first on Ext_x^M. It is then straightforward to adapt them to Ext_X^M using Lemma 1.

Crucially, extrapolation preserves the projection onto P:

Lemma 2. *Let C be a constraint over $X \cup P$. Then $C{\downarrow_P} = \mathsf{Ext}_x^M(C){\downarrow_P}$.*

For the preservation of behaviors, following [BBLP06], we use a notion of simulation:

Definition 5 (Simulation [BBLP06]). *Let $\mathcal{A} = (\Sigma, L, l_0, X, I, E)$ be a TA and $\preceq$ a relation on $L \times \mathbb{R}_+^H$. Relation $\preceq$ is a (location-based) simulation if:*

- *if $(l_1, w_1) \preceq (l_2, w_2)$ then $l_1 = l_2$,*
- *if $(l_1, w_1) \preceq (l_2, w_2)$ and $(l_1, w_1) \xrightarrow{a} (l_1', w_1')$, then there exists (l_2', w_2') such that $(l_2, w_2) \xrightarrow{a} (l_2', w_2')$ and $(l_1', w_1') \preceq (l_2', w_2')$,*
- *if $(l_1, w_1) \preceq (l_2, w_2)$ and $(l_1, w_1) \xrightarrow{d} (l_1, w_1 + d)$, then there exists d' such that $(l_2, w_2) \to (l_2, w_2 + d')$ and $(l_1, w_1 + d) \preceq (l_2, w_2 + d')$.*

If $\preceq^{-1}$ is also a simulation relation then $\preceq$ is called a bisimulation.

State s_1 simulates s_2 if there exists a simulation $\preceq$ such that $s_2 \preceq s_1$. If $\preceq$ is a bisimulation, then the two states are said bisimilar.

Lemma 3 ([BBLP06, Lemma 1]). *Let M be a non-negative integer constant greater or equal to the maximum constant occurring in the time constraints of the TA. Let $\equiv_M$ be the relation defined as $w \equiv_M w'$ iff $\forall x \in X$: either $w(x) = w'(x)$ or $(w(x) > M$ and $w'(x) > M)$. The relation $\mathcal{R} = \{((l, w), (l, w'))|w \equiv_M w'\}$ is a bisimulation relation.*

Lemma 4. *For all parameter valuation v, non-negative integer constants M, clocks x and valuation set C, $v(\mathsf{Ext}_x^M(C)) = \mathsf{Ext}_x^M(v(C))$.*

We use the bounds on parameters to compute the maximum constant M appearing in all the guards and invariants of the PTA. When those constraints are parametric expressions, we compute the maximum value that the expression can take for all the bounded parameter values (it is unique since expressions are linear): *e.g.,* if a guard is $x \leq 2p_1 - p_2 + 1$ and $p_1 \in [2, 5]$, and $p_2 \in [3, 4]$ then the maximum constant corresponding to this constraint is $2 \times 5 - 3 + 1 = 8$.

Also note that bounding the parameter domain of PTA is not a strong restriction in practice – especially since the bounds can be arbitrarily large.

Lemmas 5 and 6 are instrumental in proving the preservation of all correct integer parameter values in the algorithms of Section 4, while Lemma 7 is the key to proving their termination.

Lemma 5. *Let $\mathcal{A}$ be a PTA, s be a symbolic state of $\mathcal{A}$, and M a non-negative integer constant greater than the maximal constant occurring in the PTA (including the bounds of parameters). Let x be a clock, v be a parameter valuation, and $(l, w) \in v(\mathsf{Ext}_x^M(s)))$ be a concrete state. There exists a state $(l, w') \in v(s)$ such that (l, w) and (l, w') are bisimilar.*

Extrapolation and integer hulls Here, for the sake of simplicity, and similarly to [JLR15], we consider that all polyhedra are topologically closed and, to avoid confusion, we equivalently (provided that M is (strictly) greater than the maximal constant in the PTA) define $\mathsf{Ext}_x^M(s)$ as $(s \cap (x \leq M)) \cup \mathsf{Cyl}_x(s \cap (x \geq M)) \cap (x \geq M)$.

Lemma 6. *For all integer parameter valuations v, all non-negative integer constants M, and all reachable symbolic states $s = (l, C)$, $v(\mathsf{IH}(\mathsf{Ext}_X^M(C))) = v(\mathsf{Ext}_X^M(C))$.*

Lemma 7. *In a bounded PTA, the set of constraints $\mathsf{IH}(\mathsf{Ext}_X^M(C))$ over the set of symbolic reachable states (l, C) is finite.*

4 Integer-Complete Dense Parametric Algorithms

In this section, we describe two parameter synthesis algorithms that always terminate for *bounded* PTA, and return not only all the integer points solution of the problem (à la [JLR15]) but also all real-valued points in between integer points; that is, these algorithms return a list of convex combinations of integer points, and all rational-valued points contained in each such convex combination are also solution of the problem.

4.1 Parametric Reachability: RIEF

The goal of RIEF given in Algorithm 1 ("R" stands for robust, and "I" for integer hull) is to synthesize parameter valuations solution to the EF-synthesis problem,

Algorithm 1: $\mathsf{RIEF}(\mathcal{A}, s, G, S)$

input : A PTA $\mathcal{A}$, a symbolic state $s = (l, C)$, a set of target locations G, a set S of passed states on the current path
output: Constraint K over the parameters

```
1 if l ∈ G then K ← C↓_P ;
2 else
3     K ← ⊥;
4     if IH(Ext_X^M(s)) ∉ S then
5         for each outgoing e from l in A do
6             K ← K ∪ RIEF(A, Succ(s, e), G, S ∪ {IH(Ext_X^M(s))});
```

i.e., the valuations for which there exists a run eventually reaching a location in G. It is inspired by the algorithms EF and IEF introduced in [JLR15] that both address the same problem; however EF does not terminate in general, and IEF can only output integer valuations. In fact, if we replace all occurrences of $\mathsf{IH}(C)$ in Algorithm RIEF by C, we obtain Algorithm EF from [JLR15]. RIEF proceeds as a post-order traversal of the symbolic reachability tree, and collects all parametric constraints associated with the target locations G. In contrast to EF, it stores in S the *integer hulls* of the states, which ensures termination due to the finite number of possible integer hulls of k-extrapolations; however, in contrast to IEF, RIEF returns the actual states (instead of their integer hull), which yields a larger result than IEF.

As a direct consequence of Lemma 7, it is clear that RIEF explore only a finite number a symbolic states. Therefore, we have the following theorem:

Theorem 1. *For any bounded PTA $\mathcal{A}$, the computation of* $\mathsf{RIEF}(\mathcal{A}, \mathsf{Init}(\mathcal{A}), G, \emptyset)$ *terminates.*

Theorem 2. *Upon termination of* RIEF*, we have:*

1. *Soundness: If $v \in \mathsf{RIEF}(\mathcal{A}, \mathsf{Init}(\mathcal{A}), G, \emptyset)$ then G is reachable in $v(\mathcal{A})$;*
2. *Integer completeness: If v is an integer parameter valuation, and G is reachable in $v(\mathcal{A})$ then $v \in \mathsf{RIEF}(\mathcal{A}, \mathsf{Init}(\mathcal{A}), G, \emptyset)$.*

Proof. See Appendix B.

Example 1. Consider the simple PTA with a unique transition from the initial location l_0 to l_1 with guard $1 \leq x \leq 2a$. To ensure the $EF\{l_1\}$ property, we just need to be able to go through the transition from l_0 to l_1. The parametric zone C_1 obtained in l_1 is $1 \leq x \wedge 1 \leq 2a$, which implies $a \geq \frac{1}{2}$. The integer hull of C_1 is $1 \leq x \wedge 1 \leq a$, which implies $a \geq 1$.

Algorithm IEF gives the result $a \geq 1 \wedge a \in \mathbb{N}$, while algorithm RIEF gives (here) the exact result $a \geq \frac{1}{2}$.

Algorithm 2: $\mathsf{RIAF}(\mathcal{A}, s, G, S)$

input : A PTA $\mathcal{A}$, a symbolic state (l, C), a set of target locations G, a set S of passed states on the current path
output: Constraint K over the parameters

```
1  if l ∈ G then K ← C↓_P ;
2  else
3      if (l, IH(Ext^M_X(C))) ∈ S then K ← ⊥ ;
4      else
5          K ← ⊤ ;  K_Live ← ⊥;
6          for each outgoing e = (l, g, a, R, l') from l in A do
7              S' ← Succ((l, C), e);
8              K_Good ← RIAF(A, S', G, S ∪ {(l, IH(Ext^M_X(C)))});
9              K_Block ← ⊤ \ S'↓_P;
10             K ← K ∩ (K_Good ∪ K_Block);
11             K_Live ← K_Live ∪ (C ∩ g)^↙ ;
12         K ← K \ (ℝ^{X∪P} \ K_Live)↓_P;
```

4.2 Parametric Unavoidability: RIAF

RIAF (given in Algorithm 2) synthesizes parameter valuations solution to the AF-synthesis problem. It is inspired by the algorithms AF and IAF introduced in [JLR15]; however AF may not terminate, and IAF can only output integer valuations. Note also, as shown in Example 2 below, that interpreting the result of IAF as a dense set is incorrect in general, since it may contain non-integer values that do not ensure unavoidability. RIAF works as a post-order traversal of the symbolic reachability tree, keeping valuations that permit to go into branches reaching G and cutting off branches leading to a deadlock or looping without any occurrence of G. More precisely, RIAF uses three sets of valuations: *i*) K_{Good} contains the set of valuations that indeed satisfy AF, recursively computed by calling RIAF; *ii*) K_{Block} allows to cut off branches leading to deadlock or looping, by keeping only valuations in the complement of the first state in that branch; *iii*) K_{Live} is necessary to forbid reaching states from which no transition can be taken for any e, even after some delay.

The main difference between AF and RIAF is that we use the convergence condition of IAF, which operates on integer hulls instead of symbolic states, hence ensuring termination with the same reasoning as RIEF.

We state below the soundness and integer completeness of RIAF. The proofs are easily adapted from those of AF in [JLR15], by using the additional arguments provided in the proof of RIEF, and in particular Lemma 7.

Theorem 3. *Upon termination of* RIAF*, we have:*

1. *Soundness: If* $v \in \mathsf{RIAF}(\mathcal{A}, \mathsf{Init}(\mathcal{A}), G, \emptyset)$ *then* G *is inevitable in* $v(\mathcal{A})$*;*
2. *Integer completeness: If* v *is an integer parameter valuation, and* G *is inevitable in* $v(\mathcal{A})$ *then* $v \in \mathsf{RIAF}(\mathcal{A}, \mathsf{Init}(\mathcal{A}), G, \emptyset)$*.*

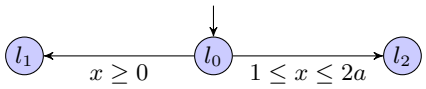


Fig. 3: Counter-example to the density of the result of IAF.

Example 2. Consider the PTA in Fig. 3. To ensure the $AF\{l_1\}$ property, we need to cut the transition from l_0 to l_2. The parametric zone C_2 obtained in l_2 is $1 \leq x \wedge 1 \leq 2a$, which implies $a \geq \frac{1}{2}$. The integer hull of C_2 is $1 \leq x \wedge 1 \leq a$, which implies $a \geq 1$. In order to block the path to l_2 in l_0, we intersect with the complement of projection on parameters of $\mathsf{IH}(C_2)$, *i.e.,* $a < 1$. Since there is no constraint on the transition from l_0 to l_1 the final result of our former algorithm IAF is actually $a < 1$. For integer parameters this means $a = 0$, which is correct. But if we interpret the result for real parameters, we obtain that, for instance, $a = \frac{1}{2}$ should be a valuation ensuring the property, while it is obviously not.

On the same example, RIAF gives (here) the exact result $a < \frac{1}{2}$.

4.3 Implementation in Roméo

The algorithms have been implemented in the tool Roméo [LRST09]; polyhedra operations (both convex and non-convex) are handled by the PPL library [BHZ08]. To illustrate this, we refer the reader to the scheduling example of [JLR15]. It consists in three tasks τ_1, τ_2, τ_3 scheduled using static priorities ($\tau_1 > \tau_2 > \tau_3$) in a non-preemptive manner. Task τ_1 is periodic with period a and a non-deterministic duration in $[10, b]$, where a and b are parameters. Task τ_2 only has a minimal activation time of $2a$ and has a non-deterministic duration in $[18, 28]$ and finally τ_3 is periodic with period $3a$ and a non-deterministic duration in $[20, 28]$. Each task is subject to a deadline equal to its period so that it must only have one instance active at all times. We ask for the parameter values that ensure that the system does not reach a deadline violation.[2] Algorithm IEF produces the constraint $a \geq 34, b \geq 10, a-b \geq 24$ in 7.4 s on a Core i7/Linux computer, while algorithm RIEF produces the constraint $a > \frac{562}{17}, b \geq 10, a - b > \frac{392}{17}$ in 12.7 s.

As illustrated here, the results are indeed a bit more precise but the main improvement is of course the guaranteed density of the result. Also, RIEF is generally slower than IEF and profiling shows that this is due to a decreased efficiency in computing the integer hull: we start each time from the whole symbolic state instead of starting from the successor of an already computed integer hull. This could maybe be mitigated using a cache for the constraints generated in computing the integer hulls.

5 Conclusion

Summary We introduced here an extrapolation for symbolic states that contains not only clocks but also parameters. We then proposed algorithms that always

[2] The result is therefore the complement of the result given by IEF and therefore an over-approximation containing no incorrect integer value.

terminate for PTA with bounded parameters, and output symbolic constraints that define dense sets of parameter valuations that are guaranteed to be correct and containing at least all integer points.

Synthesizing not only the integer points but also the real-valued points is of utmost importance for the robustness or implementability of the system. In fact, one can even consider any degree of precision instead of integers (*e.g.,* a degree of precision of $\frac{1}{10}$) by appropriately resizing the constants of the PTA (*e.g.,* by multiplying all constants and all parameter bounds by 10). This makes possible the synthesis of an underapproximated result arbitrarily close to the actual solution.

Future Works We proposed a first attempt to define a k-extrapolation for PTA; this can serve as a basis for further developments, *e.g.,* using better extrapolation operators such as L/U, local-L/U or local-diagonal-L/U abstractions. The approach we propose is fairly generic and could probably be adapted to more complex properties, expressed in LTL or CTL and their parametric variants. Moreover, we would like to extend in a similar manner the inverse method proposed in [ACEF09], hence ensuring termination of this algorithm with an almost-complete result. Furthermore, we use here the integer hull as an underapproximation of the result; in contrast, we could use an overapproximation using a notion (yet to be defined) of "external integer hull", and then combine both hulls to obtain two sets of "good" and "bad" parameter valuations separated by an arbitrarily small set of unknown valuations.

Acknowledgement We would like to thank anonymous reviewers for their useful comments, especially for a meaningful remark on a preliminary version of this paper together with the suggestion of the example in Fig. 1.

References

ACEF09. Étienne André, Thomas Chatain, Emmanuelle Encrenaz, and Laurent Fribourg. An inverse method for parametric timed automata. *International Journal of Foundations of Computer Science*, 20(5):819–836, 2009. 12

AD94. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994. 1

AHV93. Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *STOC*, pages 592–601. ACM, 1993. 1, 2, 4

AM15. Étienne André and Nicolas Markey. Language preservation problems in parametric timed automata. In *FORMATS*, volume 9268 of *Lecture Notes in Computer Science*, pages 27–43. Springer, 2015. 2

BBLP06. Gerd Behrmann, Patricia Bouyer, Kim G. Larsen, and Radek Pelánek. Lower and upper bounds in zone-based abstractions of timed automata. *International Journal on Software Tools for Technology Transfer*, 8(3):204–215, 2006. 6, 7

BBLS15. Nikola Beneš, Peter Bezděk, Kim G. Larsen, and Jiří Srba. Language emptiness of continuous-time parametric timed automata. In *ICALP, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2015. 1, 2

BHZ08. Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008. 11

BL09. Laura Bozzelli and Salvatore La Torre. Decision problems for lower/upper bound parametric timed automata. *Formal Methods in System Design*, 35(2):121–151, 2009. 2

BO14. Daniel Bundala and Joël Ouaknine. Advances in parametric real-time reasoning. In *MFCS*, volume 8634 of *Lecture Notes in Computer Science*, pages 123–134. Springer, 2014. 2

BR07. Véronique Bruyère and Jean-François Raskin. Real-time model-checking: Parameters everywhere. *Logical Methods in Computer Science*, 3(1:7), 2007. 2

Doy07. Laurent Doyen. Robust parametric reachability for timed automata. *Information Processing Letters*, 102(5):208–213, 2007. 1

HPR94. Nicolas Halbwachs, Yann-Éric Proy, and Pascal Raymond. Verification of linear hybrid systems by means of convex approximations. In *SAS*, pages 223–237, 1994. 5

HRSV02. Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 52-53:183–220, 2002. 2

JLR15. Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. Integer parameter synthesis for real-time systems. *IEEE Transactions on Software Engineering*, 41(5):445–461, 2015. 2, 3, 5, 8, 9, 10, 11, 15, 16

LRST09. Didier Lime, Olivier H. Roux, Charlotte Seidner, and Louis-Marie Traonouez. Romeo: A parametric model-checker for Petri nets with stopwatches. In *TACAS*, volume 5505 of *Lecture Notes in Computer Science*, pages 54–57. Springer, March 2009. 11

Mar11. Nicolas Markey. Robustness in real-time systems. In *SIES*, pages 28–34. IEEE Computer Society Press, 2011. 2

Mil00. Joseph S. Miller. Decidability and complexity results for timed automata and semi-linear hybrid automata. In *HSCC*, volume 1790 of *Lecture Notes in Computer Science*, pages 296–309. Springer, 2000. 1

Sch86. Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986. 15

Wan96. Farn Wang. Parametric timing analysis for real-time systems. *Information and Computation*, 130(2):131–150, 1996. 2

A Proofs of Section 3

Lemma 2 (recalled). *Let C be a constraint over $X \cup P$. Then $C{\downarrow_P} = \mathsf{Ext}_x^M(C){\downarrow_P}$.*

Proof. First notice that $C \subseteq \mathsf{Ext}_x^M(C)$, and therefore $C{\downarrow_P} \subseteq \mathsf{Ext}_x^M(C){\downarrow_P}$. Second, Ext_x^M only adds points with new clock values, without modifying parameter values: by definition of Ext_x^M, for any valuation in $\mathsf{Ext}_x^M(C)$, there exists a valuation in C with the same projection on parameters. So, $C{\downarrow_P} \supseteq \mathsf{Ext}_x^M(C){\downarrow_P}$. □

Lemma 5 (recalled). *Let $\mathcal{A}$ be a PTA, s be a symbolic state of $\mathcal{A}$, and M a non-negative integer constant greater than the maximal constant occurring in the PTA (including the bounds of parameters). Let x be a clock, v be a parameter valuation, and $(l, w) \in v(\mathsf{Ext}_x^M(s)))$ be a concrete state. There exists a state $(l, w') \in v(s)$ such that (l, w) and (l, w') are bisimilar.*

Proof. If $(l, w|v) \in s$, then the results holds trivially. Otherwise, it means that there exists some clock x such that $(l, w|v) \in \mathsf{Cyl}_x(s \cap (x > M)) \cap (x > M)$. This implies that $v(s \cap (x > M)) \neq \emptyset$ and $w(x) > M$. Therefore, and using the definition of Cyl_x, there exists $(l, w'|v) \in s \cap (x > M)$ such that for all $x' \neq x, w'(x') = w(x')$. We also have $w'(x) > M$, which means that $w' \equiv_M w$ and by Lemma 3, we obtain the expected result. □

Lemma 4 (recalled). *For all parameter valuation v, non-negative integer constants M, clocks x and valuation set C, $v(\mathsf{Ext}_x^M(C)) = \mathsf{Ext}_x^M(v(C))$.*

Proof. It is easy to prove, just by writing their definitions, that $v(C \cap C') = v(C) \cap v(C')$ and $\mathsf{Cyl}_x(v(C)) = v(\mathsf{Cyl}_x(C))$, and the result follows. □

It is clear that Lemmas 2, 4 and 5 directly extend from Ext_x^M to Ext_X^M.

Lemma 8. *For all symbolic states s and s', non-negative integer constant M greater than the maximal constant occurring in the PTA (including the bounds of parameters), and parameter valuation v, such that $v(\mathsf{Ext}_X^M(s)) = v(\mathsf{Ext}_X^M(s'))$, for all states $(l, w) \in v(s))$, there exists a state $(l, w') \in v(s'))$ such that (l, w) and (l, w') are bisimilar.*

Proof. This is a direct consequence of Lemmas 4 and 5. □

Lemma 9. *For all integer parameter valuations v, all non-negative integer constants M, and all reachable symbolic states $s = (l, C)$, $v(\mathsf{Ext}_x^M(C))$ is its own integer hull.*

Proof. We use the fact that a DBM with integer coefficients is its own integer hull. From Lemma 4, $v(\mathsf{Ext}_x^M(C)) = \mathsf{Ext}_x^M(v(C))$. Since s is reachable in a PTA, C is a parametric DBM and, since v is an integer valuation, $v(C)$ is a DBM with integer coefficients. Now, constraints $x > M$ and $x \leq M$ are DBMs with

integer coefficients too. Also cylindrification on x preserves DBMs with integer coefficients too: it consists in putting a $+\infty$ coefficient in the line and column corresponding to x. So, finally, $\mathsf{Ext}_x^M(v(C)$ is a union of two DBMs with integer coefficients, and each of them is then its own integer hull (recall that we take the integer of a union of polyhedra as the union the integer hulls). □

Lemma 6 (recalled). *For all integer parameter valuations v, all non-negative integer constants M, and all reachable symbolic states $s = (l, C)$, $v(\mathsf{IH}(\mathit{Ext}_X^M(C))) = v(\mathit{Ext}_X^M(C))$.*

Proof. Since we take the integer hull on each convex parts, it certainly holds that $\mathsf{IH}(\mathsf{Ext}_X^M(s)) \subseteq \mathsf{Ext}_X^M(s)$ and so $v(\mathsf{IH}(\mathsf{Ext}_X^M(C))) \subseteq v(\mathsf{Ext}_X^M(C))$.

In the other direction, using Lemma 9, $v(\mathsf{Ext}_X^M(C)) = \mathsf{IH}(v(\mathsf{Ext}_X^M(C)))$. Now, if $w \in \mathsf{IH}(v(\mathsf{Ext}_X^M(C)))$, then $w|v \in \mathsf{IH}(\mathsf{Ext}_X^M(C))$, *i.e.,* $w \in v(\mathsf{IH}(\mathsf{Ext}_X^M(C))$. □

Lemma 7 (recalled). *In a bounded PTA, the set of constraints $\mathsf{IH}(\mathit{Ext}_X^M(C))$ over the set of symbolic reachable states (l, C) is finite.*

Proof. In each disjunct of $\mathsf{Ext}_X^M(C)$, clocks are either upper bounded by M or the only constraint, due to the cylindrification, is a lower bound of M. Therefore, vertices of these disjuncts all have coordinates less or equal to M. When taking the integer hull of all these disjuncts separately we obtain a finite union a polyhedra with integer vertices with coordinates less or equal to M (and a finite set of extremal rays[3], taken in the finite set of the directions of clock variables), of which there can be only finitely many. □

B Proof of Theorem 2 (RIEF)

In order to prove the soundness and completeness of Algorithm 1, we inductively define, as in [JLR15], the *symbolic reachability tree* of $\mathcal{A}$ as the possibly infinite directed labeled tree T^∞ such that:

- the root of T^∞ is labeled by $\mathsf{Init}(\mathcal{A})$;
- for every node n of T^∞, if n is labeled by some symbolic state S, then for all edges e of $\mathcal{A}$, there exists a child n' of n labeled by $\mathsf{Succ}(S, e)$ iff $\mathsf{Succ}(S, e)$ is not empty.

Algorithm RIEF is a post-order depth-first traversal of some prefix of that tree.

Theorem 2 (recalled). *Upon termination of* RIEF*, we have:*

1. *Soundness: If $v \in \mathsf{RIEF}(\mathcal{A}, \mathsf{Init}(\mathcal{A}), G, \emptyset)$ then G is reachable in $v(\mathcal{A})$;*
2. *Integer completeness: If v is an integer parameter valuation, and G is reachable in $v(\mathcal{A})$ then $v \in \mathsf{RIEF}(\mathcal{A}, \mathsf{Init}(\mathcal{A}), G, \emptyset)$.*

[3] Informally speaking, extremal rays are the directions in which the polyhedron is infinite, see, *e.g.,* [Sch86] for details.

Proof. 1. Soundness: this part of the proof is almost exactly the same as in [JLR15] so we do not repeat it. The only difference is that, with the same proof, we actually have a slightly stronger result that holds for any finite prefix of T^∞ instead of exactly the one computed by EF:

Lemma 10. *Let T be a finite prefix of T^∞, on which we apply algorithm* RIEF*. Let n be a node of T labeled by some symbolic state s, and such that the subtree rooted at n has depth N. We have: $v \in \mathsf{RIEF}(\mathcal{A}, s, G, M)$, where M contains the symbolic states labeling nodes on the path from the root, iff there exists a state (l, w) in $v(s)$ and a run ρ in $v(\mathcal{A})$, with less than N discrete steps, that starts in (l, w) and reaches G.*

Soundness is a direct consequence of Lemma 10.

2. Integer completeness: The proof of this part follows the same general structure as that of EF in [JLR15] but with additional complications due to the use of the extrapolation. For the sake of clarity, we rewrite it completely here, taking care of the modified convergence scheme.
Before we start, let us just recall two more results from [JLR15]:

Lemma 11 ([JLR15, Lemma 1]). *For all parameter valuation v, symbolic states s and edges e, we have $\mathsf{Succ}(v(s), v(e)) = v((\mathsf{Succ}(s, e)))$.*

Lemma 12 ([JLR15, Corollary 2]). *For each parameter valuation v, reachable symbolic state S, and state s, we have $s \in v(S)$ if and only if there is a run of $v(A)$ from the initial state leading to s.*

Now, the algorithm having terminated, it has explored a finite prefix T of T^∞. Let v be an *integer* parameter valuation. Suppose there exists a run ρ in $v(\mathcal{A})$ that reaches G. Then ρ is finite and its last state has a location belonging to G. Let $e_1, \ldots, e_p$ be the edges taken in ρ and consider the branch in the tree T obtained by following this edge sequence on the labels of the arcs in the tree as long as possible. If the whole edge sequence is feasible in T, then the tree T has depth greater or equal to the size of the sequence and we can apply Lemma 10 to obtain that $v \in \mathsf{RIEF}(\mathcal{A}, \mathsf{Init}(\mathcal{A}), G, \emptyset)$. Otherwise, let $s = (l, C)$ be the symbolic state labeling the last node of the branch, e_k be the first edge in $e_1, \ldots, e_p$ that is not present in the branch and (l, w) be the state of ρ just before taking e_k. Using Lemma 11, $v(\mathsf{Succ}(s, e_k))$ is not empty so $\mathsf{Succ}(s, e_k)$ is not empty. Since the node labeled by s has no child in T, it follows that either $l \in G$ or there exists another node on the branch that is labeled by s' such that $\mathsf{IH}(\mathsf{Ext}_X^M(s)) = \mathsf{IH}(\mathsf{Ext}_X^M(s'))$.
In the former case, we can apply Lemma 10 to the prefix of ρ ending in (l, w) and we obtain that $v \in \mathsf{RIEF}(\mathcal{A}, \mathsf{Init}(\mathcal{A}), G, \emptyset)$.
In the latter case, we have $v(\mathsf{IH}(\mathsf{Ext}_X^M(s))) = v(\mathsf{IH}(\mathsf{Ext}_X^M(s')))$. Since v is an integer parameter valuation, by Lemma 6, this is $v(\mathsf{Ext}_X^M(s)) = v(\mathsf{Ext}_X^M(s'))$. Using now Lemma 8, there exists a state $(l, w') \in s'$ that is bisimilar to (l, w).
Also, by Lemma 12, (l, w') is reachable in $v(\mathcal{A})$ via some run ρ_1 along edges $e_1 \ldots e_m$, with $m < k$. Also, since (l, w') and (l, w) are bisimilar, there exists

a run ρ_2 that takes the same edges as the suffix of ρ starting at (l, w). Let ρ' the run obtained by merging ρ_1 and ρ_2 at (l, w'). Run ρ' has strictly less discrete actions than ρ and also reaches G. We can thus repeat the same reasoning as we have just done. We can do this only a finite number of times (because the length of the considered run is strictly decreasing) so at some point we have to be in some of the other cases and we obtain the expected result. □