Advances on continuous-time parameter synthesis

ANR PACS Deliverable 2

Étienne André, Didier Lime, Olivier H. Roux

July 7, 2017

The need for parameters to specify durations, or timing constraints, has been identified early on in the community of formal verification. In the early 1990's, the now ubiquitous formalism of timed automata [AD90] has been published and shortly after has been proposed its extension to include timing parameters: parametric timed automata, or PTAs for short [AHV93].

Unfortunately, the basic problem of deciding whether there exists a value for the parameter such that some control state is reachable is undecidable [AHV93]. Subsequent work has shown that this result is somewhat robust in the sense that the problem is undecidable, even when using only strict constraints [Doy07], integer parameters [AHV93], bounded rational parameters [Mil00], etc. All this usually using only one parameter, one clock compared to parameters, and a handful of clocks compared only to known constants (e.g. [Mil00]).

To overcome this issue, a syntactic restriction to the general formalism of PTAs has been proposed: the so called L/U parametric timed automata (L/U-PTAs), in which every parameter is restricted to be used either only as an upper bound in timing constraints, or only as a lower bound. The existence of a parameter valuation such that some control state is reachable is decidable for this formalism, thanks to its monotonicity property: increasing the value of upper bound parameters, or decreasing the value of lower bound parameters never decreases the number of possible behaviours in the model [HRSV02]. Most properties (emptiness, universality, finiteness, etc.) of the set of parameter values such that some (infinite) execution is accepted (defined by means on reaching of infinitely often visiting some control state) is decidable for L/U-PTAs with integer parameters [BT09].

We surveyed decidability results of PTAs and L/U-PTAs in [And16].

In view of these mostly negative results for such a basic problem, there has been little incentive to explore other problems such as liveness, or the algorithmics of actual synthesis of "good" parameter values.

As part of the PACS project, we have nonetheless further studied some of these other problems, investigating the emptiness and universality of the set of parameter satisfying reachability [ALR16a] or unavoidability (thus relating to liveness) [JLR15, AL17] properties. In [AL17], we have also considered more specific liveness properties: the existence of cycles and deadlocks.

Without too much surprise, everything is undecidable for PTAs and reachability properties work well with L/U-PTAs. More interestingly, we have found a more contrasted picture for liveness properties in L/U-PTAs (when the monotonicity property usually does not apply directly), with a quite thin border of

decidability for the existence of a maximal path preserving some atomic property. Decidability can be obtained by bounding the rational parameters within a topologically closed hyperrectangle. Considering unbounded or open domains leads again to undecidability [AL17].

In addition to the restriction to bounded domains, also studied in [ALR16a], or integer domains [JLR15], we have been able to define additional decidable subclasses: Integer-Point PTA (IP-PTA), in which each symbolic state contains at least one integer point, and Reset-PTA. IP-PTAs work well with reachability properties, and are different enough from L/U-PTAs, but membership to that class is undecidable. Reset-PTAs form a syntactically defined subclass of PTA, that is also a subclass of IP-PTAs, in which testing a parameter must be followed by a reset of all clocks. This condition is somewhat reminiscent of that of initialization for hybrid automata [HKPV98].

Having now several interesting subclasses of PTAs, it is natural to study their expressiveness, in order to assess what can and cannot be modeled with these formalisms. Interestingly enough, there was before the PACS project no attempt at defining the expressiveness of parametric models in the literature. In [ALR16b], we thus provided these first definitions and studied PTAs, L/U-PTAs (with or without bounded domains), and timed automata.

Finally, in [JLR15, ALR15], we have investigated the algorithmics of parameter synthesis for bounded PTAs (with no other restriction), and since this problem is in general impossible to solve completely, we have provided under-approximations, nonetheless guaranteeing that the result contains exactly all the integers values allowing the property to be satisfied [JLR15] and ensuring that those results, given as finite unions of convex polyhedra are dense with respect to the satisfaction of the property, in the sense that the rational valuations they contains are also guaranteed to be correct [ALR15].

References

[AD90] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In Mike Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer-Verlag, July 1990.

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

[AL17] Étienne André and Didier Lime. Liveness in l/u-parametric timed automata. In Axel Legay and Klaus Schneider, editors, *17th International Conference on Application of Concurrency to System Design (ACSD 2017)*, Zaragoza, Spain, June 2017. IEEE Computer Society.

[ALR15] Étienne André, Didier Lime, and Olivier H. Roux. Integer-complete synthesis for bounded parametric timed automata. In *RP*, volume 9058 of *Lecture Notes in Computer Science*. Springer, 2015.

[ALR16a] Étienne André, Didier Lime, and Olivier H. Roux. Decision problems for parametric timed automata. In *ICFEM*, volume 10009 of *Lecture Notes in Computer Science*, pages 400–416. Springer, 2016.

[ALR16b] Étienne André, Didier Lime, and Olivier H. Roux. On the expressiveness of parametric timed automata. In *FORMATS*, volume 9984 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2016.

[And16] Étienne André. What's decidable about parametric timed automata? In Cyrille Artho and Peter Csaba Ölveczky, editors, *FTSCS 2015*, volume 596 of *Communications in Computer and Information Science*, pages 52–68. Springer, 2016.

[BT09] 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.

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

[HKPV98] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

[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.

[JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. Integer parameter synthesis for timed automata. *IEEE Transactions on Software Engineering*, 41(5):445–461, 2015.

[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.

Decision Problems for Parametric Timed Automata*

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

[1] École Centrale de Nantes, IRCCyN, CNRS, UMR 6597, France
[2] Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, UMR 7030, F-93430, Villetaneuse, France

Abstract. Parametric timed automata (PTAs) allow to reason on systems featuring concurrency and timing constraints making use of parameters. Most problems are undecidable for PTAs, including the parametric reachability emptiness problem, *i. e.,* whether at least one parameter valuation allows to reach some discrete state. In this paper, we first exhibit a subclass of PTAs (namely integer-points PTAs) with bounded rational-valued parameters for which the parametric reachability emptiness problem is decidable. Second, we present further results improving the boundary between decidability and undecidability for PTAs and their subclasses.

1 Introduction

Timed automata (TAs) [AD94] are a powerful formalism that extend finite-state automata with clocks (real-valued variables evolving linearly) that can be compared with integer constants in locations ("invariants") and along transitions ("guards"); additionally, some clocks can be reset to 0 along transitions. Many interesting problems for TAs (including the reachability of a location) are decidable. However, the classical definition of TAs is not tailored to verify systems only partially specified, especially when the value of some timing constants is not yet known.

Parametric timed automata (PTAs) [AHV93] leverage this problem by allowing the specification and the verification of systems where some of the timing constants are parametric. PTAs extend TAs by allowing the use of integer- or rational-valued parameters in place of timing constants in guards and invariants. PTAs were used to model and verify a variety of case studies, from hardware circuits to communication protocols (see [And15] for a survey). This expressive power comes at the price of the undecidability of most interesting problems. The EF-emptiness problem ("does there exist a parameter valuation such that a given location is reachable?") is undecidable in general [AHV93], even when parameters are bounded [Mil00], even when only strict inequalities are used [Doy07], and

* This work is partially supported by the ANR national research program "PACS" (ANR-14-CE28-0002).

with a single integer-valued parameter [BBLS15]. However, bounding the number of parametric clocks and of parameters may yield decidability. The smallest known numbers of parametric clocks (*i. e.,* clocks compared with parameters), non-parametric clocks and parameters leading to undecidability are: three parametric clocks and one integer-valued parameter [BBLS15] or three parametric clocks and only one rational-valued parameter [Mil00], or only one parametric clock, three non-parametric clocks and one rational-valued parameter [Mil00].

In [HRSV02], L/U-PTAs are introduced as a subclass of PTAs where each parameter is either always compared to a clock as a lower bound in guards and invariants, or always as an upper bound. The EF-emptiness problem is decidable for L/U-PTAs. In [BL09], further results are proved for L/U-PTAs with integer-valued parameters: emptiness, finiteness and universality of the set of parameter valuations for which there exists an infinite accepting run are decidable. The AF-emptiness problem ("does there exist a parameter valuation for which a given location is eventually reached for any run?") is undecidable for L/U-PTAs [JLR15]. It is also shown in [JLR15] that the synthesis of the parameters reaching a given location in an L/U-PTA is intractable in practice. Two further subclasses have been defined in [BL09]: L-PTAs and U-PTAs, where all parameters are always lower bounds and upper bounds respectively.

In [JLR15], PTAs with bounded integer-valued parameters are considered. The problem of finding parameter valuations such that a given location is reachable or unavoidable becomes decidable, and two algorithms are provided that compute the exact such sets of integer valuations in a symbolic manner, *i. e.,* without performing an exhaustive enumeration. In [ALR15], it is shown that computing a parametric extrapolation of the integer hull of symbolic states allows one to synthesize (rational-valued) parameter valuations for bounded PTAs, guaranteeing the synthesis of at least all integer-valued valuations, but also sometimes most or even all rational-valued valuations.

Contribution L/U-PTAs is the only non-trivial[3] subclass of PTAs for which the EF-emptiness problem is decidable for an arbitrary number of clocks and parameters. However, other results are disappointing: undecidability of AF-emptiness, intractability of the synthesis [JLR15]. It is hence important to look for further subclasses of PTAs for which problems may be decidable. It is shown in [JLR15,ALR15] that integer points play a key role in decidability. Hence, our first contribution here is to investigate integer-points PTAs (IP-PTAs), that are PTAs where each symbolic state contains at least one integer point (*i. e.,* an integer valuation of the clocks and the parameters). Our intuition is successful: we prove that the EF-emptiness problem is decidable for bounded IP-PTAs (*i. e.,* with a bounded parameter domain), even when parameters are rational-valued. Although we show that it cannot be decided whether a bounded PTA is a (bounded) IP-PTA, we give two sufficient syntactic conditions: we show that bounded L/U-PTAs with non-strict inequalities are IP-PTAs and, more inter-

[3] The bounded integer PTAs of [JLR15] are arguably a trivial such subclass (even though the associated analysis techniques are not).

estingly, we introduce a new subclass of "reset-PTAs", that are also IP-PTAs, and for which, when bounded, the EF-emptiness problem is hence decidable too. This class is only the second syntactic subclass of PTAs (after L/U-PTAs) for which this problem is decidable.

Our second main contribution is to study several open problems for PTAs and several known subclasses (as well as the new class of IP-PTAs): we study here the emptiness and universality of reachability (EF), as well as unavoidability emptiness (AF). Emptiness is of utmost importance as, without decidability of the emptiness, exact synthesis is practically ruled out. Universality checks whether all parameter valuations satisfy a property, which is important for applications where the designer has no power on some valuations; this is the case of networks, where some latencies (*e. g.,* the transmission time of some packets) may be totally arbitrary. Among our results, we prove in particular that AF-emptiness is undecidable for both bounded IP-PTAs and bounded L/U-PTAs. Overall, we significantly enhance the knowledge we have of decidability problems for PTAs and subclasses.

Outline We first recall the necessary definitions in Section 2. Then, we introduce in Section 3 a new proof for the undecidability of the EF-emptiness problem for PTAs with a single rational-valued parameter; whereas this result is not essentially new (it has been known since [Mil00]), our original proof will be used in several other results of this paper. In addition, we extend this result (using a variant of our proof) to bounded PTAs with only non-strict inequalities which, to the best of our knowledge, is an original result. Then, we introduce the new class of IP-PTAs in Section 4, and study its properties. Finally, in part by using this new class, we prove in Section 5 several open results for L/U-PTAs and PTAs. We conclude in Section 6.

2 Preliminaries

2.1 Clocks, Parameters and Constraints

Let $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}_+$ and $\mathbb{R}_+$ denote the sets of non-negative integers, integers, non-negative rational numbers and non-negative real numbers respectively.

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* $(w(x_1), \ldots, w(x_H))$. An integer clock valuation is a valuation $w : X \to \mathbb{N}$. We write $\mathbf{0}$ for the valuation that assigns 0 to each clock. 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 (resp. clock) valuation is a valuation that assigns an integer value to each parameter (resp. clock).

In the following, we assume $\prec \in \{<, \leq\}$ and $\bowtie \in \{<, \leq, \geq, >\}$. *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

$x_i \in X$, $p_j \in P$, and $\alpha_i, \beta_j, d \in \mathbb{Z}$. *plt* denotes a parametric linear term over P, that is a linear term without clocks ($\alpha_i = 0$ for all i). A *constraint* C over $X \cup P$ is a conjunction of inequalities of the form $lt \bowtie 0$ (*i. e.,* a convex polyhedron). 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. Given $R \subseteq X$, we define the *reset* of C, denoted by $[C]_R$, as the constraint obtained from C by replacing with 0 the value of the clocks in R, and keeping the value of other clocks unchanged. We denote by $C{\downarrow_P}$ the projection of C onto P, *i. e.,* obtained by eliminating the clock variables (*e. g.,* using the Fourier-Motzkin algorithm [**?**]).

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

A *zone* is a polyhedron over a set of variables V (usually clocks) in which all constraints on variables are of the form $x \bowtie k$ (rectangular constraints) or $x_i - x_j \bowtie k$ (diagonal constraints), where $x_i \in V$, $x_j \in V$ and k is an integer. Operations on zones are well-documented (see *e. g.,* [BY04]).

A *parametric zone* is a convex polyhedron over $X \cup P$ in which all constraints on variables are of the form $x \bowtie plt$ (parametric rectangular constraints) or $x_i - x_j \bowtie plt$ (parametric diagonal constraints), where $x_i \in X$, $x_j \in X$ and *plt* is a parametric linear term over P.

2.2 Parametric Timed Automata

Syntax

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 finite set of clocks, v) P is a finite set of parameters, vi) I is the invariant, assigning to every $l \in L$ a guard $I(l)$, vii) E is a finite set of edges $e = (l, g, a, R, l')$ where $l, l' \in L$ are the source and target locations, $a \in \Sigma$, $R \subseteq X$ is a set of clocks to be reset, and g is a guard.*

We say that a PTA is *closed* if all its guards and invariants use only non-strict constraints. Note that the grammar of constraints does not include negation so this restriction is meaningful, and that $=$ defines closed constraints.

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 $v(p_i)$.

Concrete Semantics

Definition 2 (Concrete 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 $(S, s_0, \rightarrow)$, with*

- $S = \{(l, w) \in L \times \mathbb{R}_+^H \mid w|v \models I(l)\}$, $s_0 = (l_0, \mathbf{0})$
- $\rightarrow$ *consists of the discrete and (continuous) delay transition relations:*
  - *discrete transitions:* $(l, w) \xrightarrow{e} (l', w')$, *if* $(l, w), (l', w') \in S$, *there exists* $e = (l, g, a, R, l') \in E$, $\forall x \in X : w'(x) = 0$ *if* $x \in R$ *and* $w'(x) = w(x)$ *otherwise, and* $w|v \models g$.
  - *delay transitions:* $(l, w) \xrightarrow{d} (l, w + d)$, *with* $d \in \mathbb{R}_+$, *if* $\forall d' \in [0, d], (l, w + d') \in S$.

Moreover we write $(l, w) \overset{e}{\mapsto} (l', w')$ for a sequence of delay and discrete transitions where $((l, w), e, (l', w')) \in \mapsto$ if $\exists d, w'' : (l, w) \xrightarrow{d} (l, w'') \xrightarrow{e} (l', w')$.

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

Symbolic semantics Let us now recall the symbolic semantics of PTAs (see *e. g.,* [ACEF09]). A symbolic state is a pair (l, C) where $l \in L$ is a location, and C its associated parametric zone. The initial symbolic state of $\mathcal{A}$ is $\mathbf{s}_0 = (l_0, (\bigwedge_{1 \leq i \leq H} x_i = 0)^{\nearrow} \wedge I(l_0))$.

The symbolic semantics relies on the Succ operation. Given a symbolic state $\mathbf{s} = (l, C)$ and an edge $e = (l, g, a, R, l')$, $\mathsf{Succ}(\mathbf{s}, e) = (l', C')$, with $C' = \left([(C \wedge g)]_R \wedge I(l')\right)^{\nearrow} \wedge I(l')$..

A symbolic run of a PTA is an alternating sequence of symbolic states and edges starting from the initial symbolic state, of the form $\mathbf{s}_0 \overset{e_0}{\Rightarrow} \mathbf{s}_1 \overset{e_1}{\Rightarrow} \cdots \overset{e_{m-1}}{\Rightarrow} \mathbf{s}_m$, such that for all $i = 0, \ldots, m-1$, $e_i \in E$, and $\mathbf{s}_{i+1}$ belongs to $\mathsf{Succ}(\mathbf{s}_i, e)$. Given a symbolic state $\mathbf{s}$, we say that $\mathbf{s}$ is reachable if $\mathbf{s}$ belongs to a symbolic run of $\mathcal{A}$. In the following, we simply refer to symbolic states belonging to a run of $\mathcal{A}$ as symbolic states of $\mathcal{A}$.

2.3 Subclasses of PTAs

In this paper, we will sometimes consider *bounded* PTAs, *i. e.,* PTAs with a bounded parameter domain that 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 of dimension M.

Let us now recall L/U-PTAs [HRSV02,BL09].

Definition 3 (L/U-PTA [HRSV02]). *An L/U-PTA is a PTA where the set of parameters is partitioned into lower-bound parameters and upper-bound parameters. A lower- (resp. upper-)bound parameter is a parameter p that is used only in guards and invariants of the form $p \prec x$ (resp. $x \prec p$), where x is a clock.*

2.4 Decision Problems

Let $\mathcal{P}$ be a given a class of decision problems (reachability, unavoidability, etc.).

$\mathcal{P}$-emptiness problem:
INPUT: A PTA $\mathcal{A}$ and an instance ϕ of $\mathcal{P}$
PROBLEM: Is the set of parameter valuations v such that $v(\mathcal{A})$ satisfies ϕ empty?

$\mathcal{P}$-universality problem:
INPUT: A PTA $\mathcal{A}$ and an instance ϕ of $\mathcal{P}$
PROBLEM: Are all parameter valuations v such that $v(\mathcal{A})$ satisfies ϕ?

Emptiness is the most basic parametric question: is there at least one parameter valuation such that the property holds? Universality gives a robustness quality to the property and permits to effectively abstract an infinite number of verifications with concrete values.

In this paper, we mainly focus on reachability and unavoidability properties, and call them EF and AF respectively. For example, given a PTA $\mathcal{A}$ and a subset G of its locations, EF-universality asks: "are all parameter valuations v such that G is reachable in $v(\mathcal{A})$ from the initial state?" And AF-emptiness asks: "is the set of valuations v of the parameters such that G is unavoidable in $v(\mathcal{A})$ empty?"

3 Undecidability of EF-Emptiness

Let us first recall the following classical result for PTAs.

Theorem 1 ([Mil00]). *The EF-emptiness problem is undecidable for bounded PTAs.*

We provide an alternative and original proof of this result. This new construction is similar to that of Miller [Mil00], but it might be seen as a bit simpler and we will provide a complete proof. And above all, it allows us to extend it to obtain several of the main results of this paper.

Proof. We build a PTA that encodes a 2-counter machine (2CM) [Min67], such that the machine halts iff there exists some valuation of the parameters of the PTA such that it reaches a specific location.

Recall that such a machine has two non-negative counters C_1 and C_2, a finite number of states and a finite number of transitions, which can be of the form:

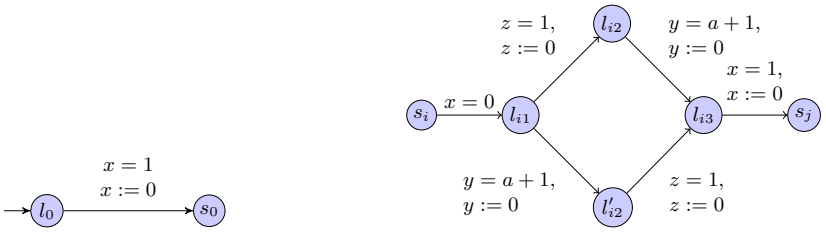
– when in state s_i, increment C_k and go to s_j;
– when in state s_i, decrement C_k and go to s_j;
– when in state s_i, if $C_k = 0$ then go to s_j, otherwise block.

The machine starts in state s_0 and halts when it reaches a particular state $l_{\texttt{halt}}$. The halting problem for 2-counter machines is undecidable [Min67].

Given such a machine $\mathcal{M}$, we now provide an encoding as a PTA $\mathcal{A}(\mathcal{M})$: each state s_i of the machine is encoded as a location of the automaton, which we also call s_i.

The counters are encoded using clocks x, y and z and one parameter a, with the following relations with the values c_1 and c_2 of counters C_1 and C_2: in any location s_i, when $x = 0$, we have $y = 1 - ac_1$ and $z = 1 - ac_2$. Note that all three clocks are parametric, *i. e.,* are compared with a in some guard or invariant. We will see that a is a rational-valued bounded parameter, typically in $[0, 1]$. The main idea of our encoding is that, to correctly simulate the machine, the parameter must be sufficiently small to encode the maximum value of the counters, *i. e.,* for $1-ac_1$ and $1-ac_2$ to stay non-negative all along the execution of the machine.

We initialize the clocks with the gadget in Figure 1a. Clearly, when in s_0 with $x = 0$, we have $y = z = 1$, which indeed corresponds to counter values 0.



(a) EF-emptiness: initial gadget

(b) EF-emptiness: increment gadget

Fig. 1: EF-emptiness: gadgets

We now present the gadget encoding the increment instruction of C_1 in Figure 1b.

The transition from s_i to l_{i1} only serves to clearly indicate the entry in the increment gadget and is done in 0 time unit.

Since we use only equalities, there are really only two paths that go through the gadget: one going through l_{i2} and one through l'_{i2}. Let us begin with the former.

We start from some encoding configuration: $x = 0$, $y = 1-ac_1$ and $z = 1-ac_2$ in s_i (and therefore the same in l_{i1}). We can enter l_{i2} (after elapsing enough time) if $1 - ac_2 \leq 1$, *i. e.,* $ac_2 \geq 0$, which implies that $a \geq 0$, and when entering l_{i2} we have $x = ac_2$, $y = 1 - ac_1 + ac_2$ and $z = 0$. Then we can enter l_{i3} if $1 - ac_1 + ac_2 \leq 1 + a$, *i. e.,* $a(c_1 + 1) \geq ac_2$. When entering l_{i3}, we then have

$x = a(c_1 + 1)$, $y = 0$ and $z = a(c_1 + 1) - ac_2$. Finally, we can go to s_j if $a(c_1 + 1) \leq 1$ and when entering s_j we have $x = 0$, $y = 1 - a(c_1 + 1)$ and $z = 1 - ac_2$, as expected.

We now examine the second path. We can enter l'_{i2} if $1 - ac_1 \leq a + 1$, *i. e.,*$a(c_1 + 1) \geq 0$, and when entering l'_{i2} we have $x = a(c_1 + 1)$, $y = 0$ and $z = 1 - ac_2 + a(c_1 + 1)$. Then we can go to l_{i3} if $1 - ac_2 + a(c_1 + 1) \leq 1 + a$, *i. e.,* $a(c_1 + 1) \leq ac_2$. When entering l_{i3}, we then have $x = ac_2$, $y = ac_2 - a(c_1 + 1)$ and $z = 0$. Finally, we can go to s_j if $ac_2 \leq 1$ and when entering s_j we have $x = 0$, $y = 1 - a(c_1 + 1)$ and $z = 1 - ac_2$, as expected.

Remark that exactly one path can be taken depending on the respective order of $c_1 + 1$ and c_2, except when both are equal or $a = 0$, in which cases both paths lead to the same configuration anyway.

Decrement is done similarly by replacing guards $y = a + 1$ with $y = 1$, and guards $x = 1$ and $z = 1$ with $x = a + 1$ and $z = a + 1$, respectively.

From s_i, to encode zero-testing C_1 and going to s_j, we only need to add a transition from s_i to s_j with guard $y = 1 \wedge x = 0$.

All those gadgets also work for C_2 by swapping y and z.

Finally, we add another location $l'_{\texttt{halt}}$ and a transition from $l_{\texttt{halt}}$ to $l'_{\texttt{halt}}$ with guard $0 < x < 1$ and $x = a$. This implies the constraint $0 < a < 1$ when reaching $l'_{\texttt{halt}}$. This is important, in order to remove the $a = 0$ value, which does not encode the counters properly. (Note that we could also do this as early as the initialization gadget; however, it is convenient to leave it here for the subsequent proofs reusing this proof.) Removing the value $a = 1$, which would be possible if both counters are always 0, is not necessary but it will be useful in subsequent proofs.

Let us now prove that the machine halts iff there exists a parameter valuation p such that $p(\mathcal{A})$ can reach $l'_{\texttt{halt}}$. Consider two cases:

1. Either the machine halts, then the automaton can go into the $l'_{\texttt{halt}}$ location, with constraints $0 < a < 1$ and, if c is the maximum value of both C_1 and C_2 over the (necessarily finite) halting run of the machine, and if $c > 0$, then $a \leq \frac{1}{c}$. The set of such valuations for a is certainly non-empty: $a = \frac{1}{2}$ belongs to it if $c = 0$ and $a = \frac{1}{c}$ does otherwise;
2. Or the machine does not halt. There are two subcases:
   (a) either the counters stay bounded. Let c be their maximal value. As before, if $c = 0$ and $0 < a \leq 1$ or $c > 0$ and $ca < 1$, then the machine is correctly encoded and the PTA cannot reach $l'_{\texttt{halt}}$. Otherwise, at some point during an incrementation of, say, C_1 we will have $a(c_1 + 1) > 1$ when taking the transition from l_{i2} to l_{i3} and the PTA will be blocked;
   (b) or one of the counters is not bounded, say C_1. Then whatever the value of $a > 0$, we have the same situation as in the previous item: the automaton blocks during some incrementation.

   In both subcases, the automaton cannot reach the $l'_{\texttt{halt}}$ location and the set of parameters such that it does is obviously empty.

□

Remark 1. We use guards with constraints $y = a + 1$ while our definition of PTAs, following [AHV93], only allows comparisons of a clock with a single parameter. Note however, and that will be true for all subsequent constructions, that transitions with $y = a + 1$ guards and $y := 0$ reset can be equivalently replaced by one transition with an $y = 1$ guard and a reset of some additional clock w, followed by a transition with a $w = a$ guard and the $y := 0$ reset (and similarly for x and z is the decrement gadget). This allows the proof to work without complex parametric expressions in guards and uses only one parametric clock and three normal clocks, with one parameter, matching the best known results with that respect [Mil00].

Now, by reusing the previous proof, we can show that the EF-emptiness problem is undecidable for closed bounded PTAs. To the best of our knowledge, this is an original result, as all existing results with bounded PTAs (*e. g.,* [Mil00,Doy07]) require strict inequalities.

Theorem 2. *The EF-emptiness problem is undecidable for closed bounded PTAs.*

Proof. See Appendix A. □

4 Integer-Points Parametric Timed Automata

In this section, we introduce integer-points parametric timed automata (IP-PTAs for short), *i. e.,* a subclass of PTAs in which any symbolic state contains at least one integer point (Section 4.1). Our first result is to prove the decidability of the EF-emptiness problem for bounded IP-PTAs (Section 4.2). Then, we compare IP-PTAs with L/U-PTAs and show that the class of bounded IP-PTAs is strictly larger than bounded L/U-PTAs with non-strict inequalities (Section 4.3). We then show that synthesis is intractable in practice, and that the same holds for bounded L/U-PTAs (Section 4.4). Finally, although we prove that the membership problem is undecidable for IP-PTAs, we exhibit a syntactic sufficient condition, that provides a new subclass of PTAs for which the EF-emptiness problem is decidable (Section 4.5).

4.1 The Class of IP-PTAs

Definition 4. *A PTA $\mathcal{A}$ is said to be an* integer points PTA *(in short* IP-PTA*) if, in any reachable symbolic state (l, C) of $\mathcal{A}$, C contains at least one integer point.*

4.2 A Decidability Result for Bounded IP-PTAs

Our main positive result is that the EF-emptiness problem is decidable for bounded IP-PTAs.

Theorem 3. *The EF-emptiness problem is decidable (and PSPACE-complete) for bounded IP-PTAs.*

Proof. We first need to recall two lemmas relating symbolic and concrete runs (proved in [HRSV02,ACEF09]).

Given a concrete (respectively symbolic) run $(l_0, \mathbf{0}) \overset{e_0}{\mapsto} (l_1, w_1) \overset{e_1}{\mapsto} \cdots \overset{e_{m-1}}{\mapsto} (l_m, w_m)$ (respectively $(l_0, C_0) \overset{e_0}{\Rightarrow} (l_1, C_1) \overset{e_1}{\Rightarrow} \cdots \overset{e_{m-1}}{\Rightarrow} (l_m, C_m)$), we define the corresponding discrete sequence as $l_0 \overset{e_0}{\Rightarrow} l_1 \overset{e_1}{\Rightarrow} \cdots \overset{e_{m-1}}{\Rightarrow} l_m$. Two runs (concrete or symbolic) are said to be equivalent if their associated discrete sequences are equal.

Lemma 1. *Let $\mathcal{A}$ be a PTA, and v be a parameter valuation. Let ρ be a run of $\mathcal{A}$ reaching a symbolic state (l, C). Then, there exists an equivalent run in the TA $v(\mathcal{A})$ reaching a concrete state (l, w) (for some w) iff $v \models C{\downarrow_P}$.*

Lemma 2. *Let $\mathcal{A}$ be a PTA, and v be a parameter valuation. Let ρ be a run of the TA $v(\mathcal{A})$ reaching a concrete state (l, w). Then there exists an equivalent run in $\mathcal{A}$ reaching a symbolic state (l, C), for some C such that $v \models C{\downarrow_P}$.*

Let $\mathcal{A}$ be a bounded IP-PTA. EF-emptiness is false for $\mathcal{A}$ iff there exists a valuation v such that a run of $v(\mathcal{A})$ reaches a location in some predefined set G. Assume there exists a valuation v such that a run of $v(\mathcal{A})$ reaches l, with $l \in G$. From Lemma 2, there exists a symbolic run of $\mathcal{A}$ reaching a symbolic state (l, C), for some C. Since $\mathcal{A}$ is an IP-PTA, C contains at least one integer point. Hence there exists an integer parameter valuation $v' \models C{\downarrow_P}$; hence from Lemma 1, there exists a concrete run of $v'(\mathcal{A})$ reaching l. This gives that EF-emptiness is false for $\mathcal{A}$ iff there exists an integer valuation v' such that a run of $v'(\mathcal{A})$ reaches a location in G.

Hence, deciding whether some valuation permits to reach l reduces to deciding whether some *integer* valuation permits to do so, which, for bounded PTAs, is PSPACE-complete [JLR15]. □

In practice, [JLR15] proposes efficient symbolic algorithms to synthesize all the integer parameter valuations that permit to reach some given location, and thus to solve EF-emptiness for IP-PTAs.

4.3 Comparison with L/U-PTAs

Let us now compare IP-PTAs and L/U-PTAs. We first need the following lemma, stating that any reachable symbolic state of an L/U-PTA contains an integer parameter valuation.

Lemma 3. *Let (l, C) be a reachable symbolic state of an L/U-PTA. Then $C{\downarrow_P}$ contains at least one integer point.*

Proof. Consider a (non-empty) reachable symbolic state (l, C) of an L/U-PTA. Let $v \models C{\downarrow_P}$. From the well-known monotonicity property of L/U-PTAs (exhibited in [HRSV02]), any parameter valuation such that the lower-bound parameters p_i^- are lower or equal to $v(p_i^-)$ and upper-bound parameters p_j^+ are greater than or equal to $v(p_j^+)$ also belong to $C{\downarrow_P}$. In particular, this is the case

of the integer parameter valuation assigning 0 to all lower-bound parameters, and assigning to upper-bound parameters p_j^+ the smallest integer greater than or equal to $v(p_j^+)$. □

The previous lemma that ensures the presence of an integer parameter valuation in any symbolic state does not guarantee that an L/U-PTA is an IP-PTA, because clocks may have non-integer values.

Proposition 1. *The class of IP-PTAs is incomparable with the class of L/U-PTAs.*

Proof.
- Consider an L/U-PTA with a transition guarded by $x > 0$ and resetting no clock, followed by a second location with invariant $x < 1$; then, necessarily, the symbolic state associated with this second location contains no integer point (as $x \in (0, 1)$ in that symbolic state).
- It is easy to exhibit an IP-PTA that is not an L/U-PTA. This is for example the case of a simple PTA with only one location, one clock x and one parameter p with a self-loop with guard $x = p$ and resetting x.

□

However, we can prove that any *closed* L/U-PTA, *i. e.,* with only non-strict inequalities, is an IP-PTA. In order to show that the class of closed L/U-PTAs is included in IP-PTAs, we need the following lemma.

Lemma 4. *Let $\mathcal{A}$ be a PTA with only non-strict inequalities. Let $\mathbf{s} = (l, C)$ be a symbolic state of $\mathcal{A}$. Then if $C{\downarrow_P}$ contains at least one integer parameter valuation, then C contains an integer point.*

Proof. Since there is at least one integer parameter valuation v in $C{\downarrow_P}$, then $v(C)$ is not empty. Since v is an integer valuation, $v(C)$ is a zone of a timed automaton with integer constants, so the vertices of $v(C)$ are integer points. Finally, there is at least one vertex in $v(C)$ because all clocks are nonnegative (and hence are bounded from below by 0), and this vertex does belong to $v(C)$ because it is topologically closed due to the non-strict constraints. So C contains at least one integer point. □

Proposition 2. *The class of IP-PTAs is strictly larger than the class of closed L/U-PTAs.*

Proof. From Lemmas 3 and 4, and Proposition 1 ($\Leftarrow$). □

The previous result also holds for bounded PTAs:

Proposition 3. *The class of bounded IP-PTAs is strictly larger than the class of closed bounded L/U-PTAs.*

Proof. Lemma 3 extends to bounded L/U-PTAs, since the bounds are integers (this would not hold otherwise). Then, the proof of Proposition 1 ($\Leftarrow$) holds with bounded IP-PTAs and closed bounded L/U-PTAs. Applying Lemma 4 concludes the proof. □

Proposition 4. *The class of bounded IP-PTAs is incomparable with the class of bounded L/U-PTAs. The class of bounded IP-PTAs is incomparable with the class of L/U-PTAs.*

Proof. The proof of Proposition 1 can be applied with bounded PTAs on either side. □

Since bounded IP-PTAs are incomparable with L/U-PTAs (for which the EF-emptiness problem is known to be decidable), and since L/U-PTAs are the only non-trivial subclass of PTAs for which this problem is known to be decidable, then Theorem 3 strictly extends the subclass of PTAs for which this problem is decidable.

4.4 Intractability of the Synthesis

Although the EF-emptiness problem is decidable for L/U-PTAs [HRSV02], the synthesis seems to pose practical problems: it was shown in [JLR15] that the solution to the EF-synthesis problem for L/U-automata, if it can be computed, cannot be represented using any formalism for which emptiness of the intersection with equality constraints is decidable. In particular, this rules out the possibility of computing the solution set as a finite union of polyhedra.

We reuse the intuition of this result and extend it to closed bounded L/U-PTAs.

Theorem 4. *If it can be computed, the solution to the EF-synthesis problem for closed bounded L/U-automata cannot be represented using any formalism for which emptiness of the intersection with equality constraints is decidable.*

Proof. We reuse the idea of [BL09] used for proving that constrained emptiness for infinite runs acceptance properties is undecidable, and reused in [JLR15, Theorem 2]. Suppose that the solution to the EF-synthesis problem for closed bounded L/U-PTAs can be represented using a formalism for which emptiness of the intersection with equality constraints is decidable. Assume a closed bounded PTA $\mathcal{A}$; for each parameter p_i of $\mathcal{A}$ that is used both as an upper bound and a lower bound, replace its occurrences as upper bounds by a fresh parameter p_i^u and its occurrences as lower bounds by a fresh parameter p_i^l. We therefore obtain a closed bounded L/U-PTA. Assume we can derive a solution to the EF-synthesis problem for this closed bounded L/U-PTA, and let K be that solution. Then, by hypothesis, we can decide whether $K \wedge \bigwedge_i p_i^l = p_i^u$ is empty or not; hence, we can solve the EF-emptiness for $\mathcal{A}$, which contradicts the undecidability of EF-emptiness for closed bounded PTAs (from Theorem 2). □

Corollary 1. *If it can be computed, the solution to the EF-synthesis problem for IP-PTAs cannot be represented using any formalism for which emptiness of the intersection with equality constraints is decidable.*

Proof. From the fact that a closed bounded L/U-PTA is an IP-PTA. □

4.5 Membership

We first show that it cannot be decided in general whether a PTA is a (bounded) IP-PTA.

Theorem 5. *It is undecidable whether a PTA is an IP-PTA, even when bounded.*

Proof. Let us consider the PTA $\mathcal{A}(\mathcal{M})$ encoding the 2-counter machine $\mathcal{M}$ proposed in our proof of Theorem 1. The PTA $\mathcal{A}(\mathcal{M})$ has only one parameter a and all the symbolic states of $\mathcal{A}(\mathcal{M})$ contain the integer value $a = 0$ except the states corresponding to the location $l'_{\texttt{halt}}$. Since all constraints are non-strict, except that of the transition leading to $l'_{\texttt{halt}}$, all reachable symbolic states, except those associated with $l'_{\texttt{halt}}$, contain an integer point. Then the PTA $\mathcal{A}(\mathcal{M})$ reaches the location $l'_{\texttt{halt}}$ if and only if $\mathcal{A}(\mathcal{M})$ is not an IP-PTA. As a consequence, this PTA is an IP-PTA iff the 2-counter machine does not halt. Finally, note that this PTA can be bounded by $0 \leq a \leq 1$, without any change in the reasoning above. □

Nevertheless, Proposition 2 provides a sufficient syntactic membership condition, since any closed L/U-PTA is an IP-PTA. In addition, we now define another new non-trivial set of restrictions leading to IP-PTAs:

Definition 5 (Reset-PTA). *A* reset-PTA *is a PTA where:*

- *all guards and invariants are conjunctions of constraints of the form* $x \leq p + k$, $x \geq p + k$, $x \leq k$, *or* $x \geq k$, *with* x *a clock,* p *a parameter, and* k *an integer;*
- *and all clocks are reset to 0 on any transition with a guard or a source location invariant in which a parameter appears.*

This kind of restriction is somewhat reminiscent of those enforced by *initialized* hybrid automata [HKPV98] to obtain decidability. We now prove that reset-PTAs are IP-PTAs, which in turn means that the EF-emptiness problem is decidable for bounded reset-PTAs. It is worth noting that, to the best our knowledge, bounded reset-PTAs and L/U-PTAs are the only non-trivial sets of syntactic restrictions of PTAs making the reachability emptiness problem decidable.

Theorem 6. *Any reset-PTA is an IP-PTA.*

Proof. See Appendix B. □

Recall that the synthesis is intractable for bounded IP-PTAs (from Corollary 1) and for bounded L/U-PTAs. In contrast, and although studying reset-PTAs in detail goes beyond the scope of this work, we highly suspect that exact synthesis can be computed for reset-PTAs (see remarks in Section 6).

5 New (Un)decidability Results for PTAs

In this section, we take advantage of the newly introduced class of IP-PTAs to solve several open problems on the more general class of PTAs; these results allow us to draw a better cartography of these problems for several subclasses of PTAs.

5.1 Undecidability of EF-Universality

We show below that, unlike L/U-PTAs, the EF-universality problem is undecidable for IP-PTAs even bounded. This result differentiates the classes of (bounded) L/U-PTAs and bounded IP-PTAs, and helps to understand better the boundary between decidability and undecidability for subclasses of PTAs.

Theorem 7. *The EF-universality problem is undecidable for bounded IP-PTAs.*

Proof. See Appendix C. □

Corollary 2. *The EF-universality problem is undecidable for IP-PTAs, for bounded PTAs, and for PTAs.*

Proof. From Theorem 7 and from the fact that a bounded IP-PTA is an IP-PTA, is a bounded PTA, and is a PTA. □

5.2 Undecidability of AF-Emptiness

It is known that AF-emptiness is undecidable for L/U-PTAs [JLR15]; reusing the encoding of the 2-counter machine proposed in our proof of Theorem 1, we now show that this result holds even for bounded L/U-PTAs.

Theorem 8. *The AF-emptiness problem is undecidable for bounded L/U-PTAs.*

Proof. See Appendix D. □

Corollary 3. *The AF-emptiness problem is undecidable for bounded IP-PTAs, for IP-PTAs and for bounded PTAs.*

Proof. The AF-emptiness problem is undecidable for bounded L/U-PTAs (Theorem 8), which immediately gives the undecidability for bounded PTAs.

Furthermore, the PTA used in the proof of Theorem 8 only uses non-strict inequalities; furthermore, $a^- = 0$ and $a^+ = 1$ is a parameter valuation solution of any symbolic state. Hence, from Lemma 4, this PTA is a bounded IP-PTA, which gives the result for bounded IP-PTAs. As a consequence, the result also holds for general IP-PTAs. □

Class	bL/U-PTAs	bIP-PTAs	L/U-PTAs	IP-PTAs	bPTAs	PTAs
EF-empt.	**Th. 10**	**Th. 3**	[HRSV02]	Th. 9	[Mil00]	[AHV93]
EF-univ.	**Th. 10**	Th. 7	[BL09]	Cor. 2	Cor. 2	Cor. 2
AF-empt.	Th. 8	Cor. 3	[JLR15]	Cor. 3	Cor. 3	[JLR15]

Table 1: Decidability results for PTAs and some subclasses

5.3 Summary

Before being able to summarize our results in Table 1, we need to prove two further missing results.

Theorem 9. *The EF-emptiness problem is undecidable for IP-PTAs.*

Proof. The proof of the undecidability of the EF-emptiness problem for general PTAs in [AHV93] can be interpreted over integer parameter valuations. Any symbolic state contains at least one integer parameter valuation (the one that is large enough to correctly encode the value of the two counters), as well as all larger parameter valuations. Furthermore, since the proof only uses non-strict inequalities (in fact only equalities), from Lemma 4, all symbolic states contain at least one integer point. Hence the PTA used in [AHV93] to encode the 2-counter machine is an IP-PTA. □

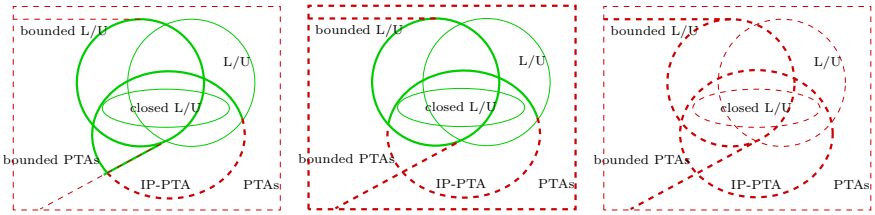
Finally, we show below (without surprise) that the EF-emptiness problem (shown to be decidable for L/U-PTAs [HRSV02]) and the EF-universality problem (shown to be decidable for integer-valued L/U-PTAs [BL09]) are also decidable for bounded L/U-PTAs.

Theorem 10. *The EF-emptiness and EF-universality problems are decidable for bounded L/U-PTAs.*

Proof. In [HRSV02,BL09], it is shown that decreasing a lower-bound parameter p_i^- or increasing an upper-bound parameter p_j^+ in an L/U-PTA $\mathcal{A}$ can only add behaviors. Hence, deciding EF-emptiness can be done by testing the reachability of the location in the TA obtained from $\mathcal{A}$ by instantiating all p_i^-s with 0 and all p_j^+s with ∞. (Recall that testing the reachability of a location in a TA is decidable [AD94].) For a bounded L/U-PTA, this can be done in a similar manner, by testing the reachability of the location in the TA obtained from $\mathcal{A}$ by instantiating all p_i^-s with their minimal value and all p_j^+s with their maximal value in the (closed) bounded parameter domain.

EF-universality can be solved similarly, except that p_i^-s are replaced with their upper bound and p_j^+s are replaced with their lower bound. □

We give a summary in Table 1. We give from left to right the (un)decidability for bounded L/U-PTAs, bounded IP-PTAs, L/U-PTAs, IP-PTAs, bounded PTAs, and PTAs. Decidability is given in bold green, whereas undecidability is given in thin red. Our contributions are depicted using a plain background, whereas existing results are depicted using a light background.



(a) EF-emptiness (b) EF-universality (c) AF-emptiness

Fig. 2: Decidability results for PTAs and subclasses

We give another summary in Figure 2. Note that bounded L/U-PTAs and L/U-PTAs are in fact incomparable of terms of expressiveness [ALR16]; they are therefore not included into each other in the figures. Decidability (resp. undecidability) is depicted in plain green (resp. dashed red); open problems are depicted in dotted black. Our contributions are depicted in thick.

6 Conclusion

In this paper, we exhibited a new subclass of PTAs (namely bounded IP-PTAs) for which the EF-emptiness problem is decidable. By showing that bounded IP-PTAs are incomparable with L/U-PTAs, we strictly extend the set of PTAs for which this problem is decidable. Although we showed that it cannot be decided whether a PTA is an IP-PTA, we introduced a new syntactic subclass of IP-PTAs, namely reset-PTAs, for which, when bounded, the EF-emptiness problem is decidable. It is worth noting that, to the best our knowledge, there is no other non-trivial set of syntactic restrictions making the reachability emptiness problem decidable for PTAs (aside from L/U-PTAs, of course).

In a second part, we considered three decision problems, and contributed in solving several open problems for PTAs and subclasses: this was achieved thanks to the results proved for IP-PTAs, and to (variations of) an original proof for the undecidability of the EF-emptiness problem for general PTAs with a single bounded rational-valued parameter and only non-strict constraints.

Future works Our new class of reset-PTAs seems promising in terms of synthesis, as the symbolic states have a very special form. Using a proper extrapolation, exact synthesis might be achievable. In addition, we are interested in extending this class to hybrid systems, and combining its restrictions with the condition of initialized hybrid automata [HKPV98]. The AF-universality problem is not treated in this paper, as it connects in an interesting manner with the problems of the existence of deadlocks or livelocks, which warrants a study on its own: in [AL16], we show in particular that the AF-universality problem is decidable for bounded L/U-PTAs with a closed parameter domain, and becomes undecidable if we lift either the assumption of boundedness or of closedness. Finally, all problems undecidable for L/U-PTAs remain open for L-PTAs and U-PTAs.

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.

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

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

AL16. Étienne André and Didier Lime. Liveness in L/U-parametric timed automata. Submitted. https://hal.archives-ouvertes.fr/hal-01304232, 2016.

ALR15. Étienne André, Didier Lime, and Olivier H. Roux. Integer-complete synthesis for bounded parametric timed automata. In *RP*, volume 9058 of *Lecture Notes in Computer Science*. Springer, 2015.

ALR16. Étienne André, Didier Lime, and Olivier H. Roux. On the expressiveness of parametric timed automata. In *FORMATS*, Lecture Notes in Computer Science. Springer, 2016. To appear.

And15. Étienne André. What's decidable about parametric timed automata? In *FTSCS*, volume 596 of *Communications in Computer and Information Science*, pages 1–17. Springer, 2015.

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.

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.

BY04. Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lecture Notes on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*. Springer–Verlag, 2004.

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

HKPV98. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

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.

JLR15. Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. Integer parameter synthesis for timed automata. *Transactions on Software Engineering*, 41(5):445–461, 2015.

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.

Min67. Marvin L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., NJ, USA, 1967.

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

A Proof of Theorem 2

Theorem 2 (recalled). *The EF-emptiness problem is undecidable for closed bounded PTAs.*

Proof. The entire encoding of the instructions of the 2-counter-machine used in the proof of Theorem 1 is a closed PTA, as only non-strict inequalities are used. In addition, the (unique) parameter a can be bounded by $[0, 1]$. However, the transition from $l_{\texttt{halt}}$ to $l'_{\texttt{halt}}$ uses strict inequalities in order to ensure that $0 < a < 1$. First, in contrast to Theorem 1, let us not remove $a = 1$, *i. e.,* the value that encodes the situation when both counters are always zero. (Recall that removing this value is not necessary, and that it was added to be used in the subsequent proofs working as variations of the proof of Theorem 1.) Now, removing $a = 0$ is necessary, as this valuation does not correctly encode the machine. Let us rewrite the transition from $l_{\texttt{halt}}$ to $l'_{\texttt{halt}}$ as in Figure 3.

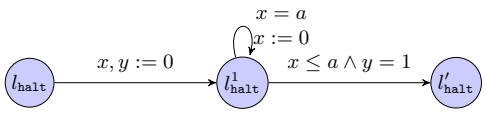


Fig. 3: Rewriting the last transition of the 2CM encoding with non-strict inequalities

Clearly, if $a = 0$, taking the self-loop on $l^1_{\texttt{halt}}$ will not allow time to elapse; and then there will be no way to leave $l^1_{\texttt{halt}}$ with $x \leq a$ and $y = 1$; hence $l'_{\texttt{halt}}$ is not reachable for $a = 0$. In contrast, if $0 < a \leq 1$, then by taking an appropriate number of times the self-loop on $l^1_{\texttt{halt}}$, we will eventually have $x \leq a$ and $y = 1$; hence $l'_{\texttt{halt}}$ will eventually be reached. To summarize:

- if $a = 0$, the machine is not correctly encoded, but there is no way to reach $l'_{\texttt{halt}}$;
- if $0 < a \leq 1$, the machine is correctly encoded, and from Theorem 1 we know that $l_{\texttt{halt}}$ is reachable iff the machine halts. Since $l'_{\texttt{halt}}$ is reachable from $l_{\texttt{halt}}$, then $l'_{\texttt{halt}}$ is reachable iff the machine halts.

Hence there exists a parameter valuation such that $l'_{\texttt{halt}}$ is reachable iff the machine halts.

B Proof of Theorem 6

Theorem 6 (recalled). *Any reset-PTA is an IP-PTA.*

Proof. We prove by induction that the symbolic states generated by reset-PTAs are zones with only non-strict constraints over the set of variables defined by the

union of clocks and parameters. To simplify the proof a bit we omit invariants but including them would raise no theoretical difficulty. We then additionally prove as part of the induction that there is no inequality involving two variables x and y in which x and y would not be of the same type (clock or parameter).

The property clearly holds for the initial symbolic state: parameters are unconstrained, all clocks are equal and their common value is greater than or equal to 0.

Now suppose this holds for some symbolic state and consider the successor of that symbolic state by some transition. Recall from the Succ definition that this successor is computed by the following operations: intersection with the guard of the transition, reset of the clocks designated in the reset set of the transition, and finally time elapsing.

Due to the restriction on constraints, all guards are themselves zones, and it is well-known that the intersection of two zones is again a zone. Similarly, the reset operation on some of the variables of a zone again leads to a zone. The time elapsing of a proper subset of the variables in a zone however is not a zone in general (the same situation arises, *e. g.,* in stopwatch automata). We therefore need to examine more closely the zone on which the time elapsing operates. Two cases arise:

1. the guard did not involve any parameter. Then, with the induction hypothesis, we still do not have any constraint between clocks and parameters after the intersection with the guard. A fortiori, we do not have any after the resets either. The time elapsing operation can be carried out by introducing a fresh non-negative variable t, performing variable substitutions $x \leftarrow x + t$ for each clock x, and finally eliminating t. This elimination can be done with the Fourier-Motzkin procedure (see *e. g.,* [Sch86]) which produces all the constraints not involving t plus those obtained by writing all the combinations of a minorant of t less or equal to a majorant of t. After the variable substitutions, t does not appear in constraints between parameters, nor in diagonal clock constraints ($x - y \leq k$ gives $(x + t) - (y + t) \leq k$, *i. e.,* again $x - y \leq k$). Since there is no constraint between clocks and parameters, t only appears in rectangular constraints that become of the form $y + t \leq k_1$ or $x + t \geq k_2$. Through the elimination procedure this gives constraints of the form $x - y \leq k_2 - k_1$, and the expected result holds.
2. the guard involves some parameters. Then before the reset we do have constraints between clocks and parameters. But then, from the definition of reset-PTAs, all clocks are reset along this transition, so these constraints are removed (as part of the elimination of clock variables) and replaced by constraints restricting the reintroduced clock variables to zero. Then, after the reset we do not have constraints between clocks and parameters anymore and the previous reasoning is again valid.

We conclude the proof by noting that in any non-empty zone with integer coefficients all vertices are integer (see the discussion in [JLR15]). And following the proof of Lemma 4, since all variables are non-negative, there is at least one

such vertex, which does belong to the zone because all constraints are non-strict. . This zone therefore contains at least an integer point. □

C Proof of Theorem 7

Theorem 7 (recalled). *The EF-universality problem is undecidable for bounded IP-PTAs.*

Proof. We start the encoding in our proof of Theorem 1. The main idea is, for all valuations of the parameter a that are not small enough to properly encode the counters (*i. e.,* for some value c of a counter, $1 - ac < 0$), to allow the PTA to directly go to an $l_{\texttt{error}}$ location. In order for our encoding to be an IP-PTA (in particular the $l_{\texttt{error}}$ symbolic states), we add a new parameter b, the value of which can be typically in $[0, 1]$.

We then reduce the problem of knowing whether the counters of the machine grow unbounded along its execution, which is undecidable [Min67], to the universality of the set of parameters that allow the encoding PTA to reach $l_{\texttt{error}}$.

First, we remove the $l'_{\texttt{halt}}$ location and the associated transition from $l_{\texttt{halt}}$, because it is no longer needed in this case and prevents the PTA to be an IP-PTA.

Instead, we create a fresh location $l_{\texttt{error}}$, and we add two transitions from l_0 (the initial location of the PTA) to the $l_{\texttt{error}}$ location:

- one with guard $x = 0 \wedge x = a$, that can only be taken when $a = 0$ and serves to "eliminate" this special case that does not correctly encode the counters.
- and one with guard $0 \leq x < 1 \wedge x = b$, that can only be taken when $b \in [0, 1)$.[4]

So, whenever $a = 0$ and $b \in [0, 1)$, the system can eventually reach $l_{\texttt{error}}$. Hence, in the following, we only need to focus on the case where $a \in (0, 1]$ and $b = 1$.

Let us now change the increment gadget (when decrementing or zero-testing, there is no upper bound constraint on a). More specifically, remark that, when incrementing C_1, the constraint that implies $a \leq \frac{1}{c_1+1}$ comes from the last transition in the path going through l_{i2}. In the other path, c_2 is already greater than or equal to c_1+1 and therefore a is already small enough to properly encode $c_1 + 1$ since it is small enough to encode c_2.

We modify the increment gadget as described in Figure 4.

In the transition from l_{i2} to l_{i3}, if a is not small enough then $x = a(c_1+1)$ will be greater than one and the final transition to s_j will not be fireable. We therefore add a direct transition from l_{i2} to $l_{\texttt{error}}$ when $x \geq b$. Recall that we only care about the case where $b = 1$, hence this transition can be understood as $x \geq 1$; now the case where $x = 1$ is problematic, as the value of a is just small enough to encode the counters, and we still can reach $l_{\texttt{error}}$, and the 2-counter machine is

[4] This case may not be necessary in the proof; however, it makes the explanation simpler, as we can now discard from our reasoning valuations such that $b \in [0, 1)$.

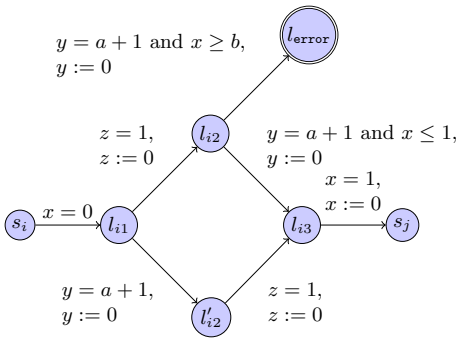


Fig. 4: EF-universality for bounded IP-PTAs: increment gadget

not properly encoded. However, since we are interested in universality, it suffices to take a valuation of a slightly smaller to properly encode the machine, as we explain more precisely below.

We now prove that the counters of the machine grow unbounded along its execution iff for all values of a and b, the encoding PTA can reach $l_{\texttt{error}}$. First recall that for $a = 0$ or $b \in [0, 1)$, it is always possible to reach $l_{\texttt{error}}$ (from the initial state). When $a > 0$ and $b = 1$, we have two cases:

- either the counters grow unbounded (say C_1 does), then whatever the value of $a > 0$, at some point we have $ac_1 > 1$. More specifically, there is an incrementation of C_1 such that $ac_1 \leq 1$ and $a(c_1 + 1) > 1$, which also implies $a(c_1 + 1) \geq b$ (since $b = 1$). Then, when executing the corresponding increment gadget, $l_{\texttt{error}}$ can be reached from l_{i2};
- or the counters stay bounded. Let c be the maximal value of the counters. Recall that when entering l_{i2} we have $x = ac_2$, $y = 1 - ac_1 + ac_2$ and $z = 0$. Then, when $y = a + 1$, we have $x = a(c_1 + 1)$. If $c_1 + 1 = c$, then $x = ca$ is the largest value that x can have in l_{i2} when $y = a+1$. Observe that, due to the non-strict inequality $x \geq b$ in the guard from l_{i2} to $l_{\texttt{error}}$, one might still reach $l_{\texttt{error}}$ for a valuation of a such that $ca = 1$. Consequently, consider the parameter valuation $a = \frac{1}{c+1}$ and $b = 1$. Then $ca < 1 = b$ and since x is in l_{i2} always at most equal to ca when $y = a + 1$, the guard to $l_{\texttt{error}}$ is never true and the of valuations for which the automaton can reach $l_{\texttt{error}}$ is not universal.

It remains to show that the constructed PTA is an IP-PTA. With the exception of $l_{\texttt{error}}$, the result is clear: $a = 0$ and $b = 0$ belongs to every reachable symbolic state, hence each symbolic state contains an integer parameter valuation, and hence from Lemma 4, all symbolic states (except $l_{\texttt{error}}$) contain at least one integer point. In addition, the two symbolic states taken by taking the two special transitions from the initial state to $l_{\texttt{error}}$ to handle $a = 0$ or $b \in [0, 1)$ also contain the integer point $x = y = a = b = 0$. Now, let us consider the other symbolic states with location $l_{\texttt{error}}$ (and reachable from some location l_{i2} due to

an increment). The projection onto the parameters of the associated constraint is $0 \leq b \leq 1 \wedge \frac{b}{i+1} \leq a \leq \frac{1}{i}$, with $i \in \mathbb{N}$ denotes the current maximum valuation of the counter. Clearly, $a = b = 0$ is a parametric integer point in this symbolic state; hence from Lemma 4 this symbolic state contains an integer point (in clocks and parameters dimensions). Hence this PTA is a (bounded) IP-PTA. □

D Proof of Theorem 8

Theorem 8 (recalled). *The AF-emptiness problem is undecidable for bounded L/U-PTAs.*

Proof. Let us consider the PTA $\mathcal{A}(\mathcal{M})$ encoding the 2-counter machine $\mathcal{M}$ proposed in our proof of Theorem 2. The PTA $\mathcal{A}(\mathcal{M})$ has only one parameter a which is used both as an upper bound and a lower bound. We add two fresh parameters a^- and a^+ and we replace the guard $y = 1 + a$ by a guard $1 + a^- \leq y$ and an invariant $y \leq 1 + a^+$ as shown for the increment gadget in Figure 5.

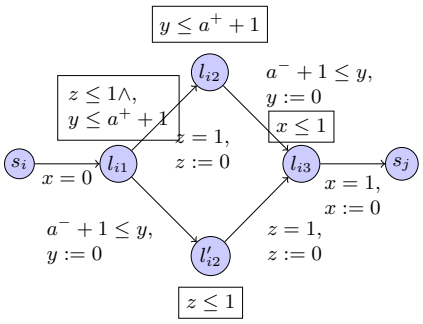


Fig. 5: AF-emptiness for bounded L/U-PTAs: increment gadget

We initialize the parameters a^- and a^+ with the gadget in Figure 6 (adapted from [JLR15]) leading to the location s_0. Clearly, starting from l_0, we have $\mathsf{AF}(s_0)$ if and only if $a^- = a^+ > 0$, because *1)* if $a^- = 0$ then it is possible to reach $l_{\texttt{sink}}$ and therefore we do not have $\mathsf{AF}(s_0)$, and *2)* any run that reaches l_1 before y is equal to a^+ can be extended by delaying a non-null amount of time into a run that will be blocked by the invariant of s_0. So all runs should enter l_1 with $y = a^+$, which is the case if and only if $a^- = a^+$. We therefore obtain an L/U-automaton with $a^- = a^+$ and $a^+ > 0$.

Let us now go back to the increment gadget of Figure 5. As in the proof of Theorem 2, when $a^- = a^+ > 0$, exactly one path can be taken depending on the respective order of $c_1 + 1$ and c_2.The invariants allow to avoid infinite delays in locations and since $a^- = a^+$, no run can be blocked inside the gadget. The same

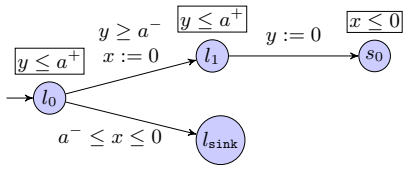


Fig. 6: AF-emptiness for bounded L/U-PTAs: initial gadget

reasoning can be made for the decrement and zero-testing gadgets. Moreover we can bound the PTA by $a^-, a^+ \in [0, 1]$ without loss of behavior.

Hence we reduce the halting problem of 2-counter machine to the AF-emptiness problem for bounded L/U-PTAs: the machine halts iff there exists a value of $a^- = a^+ > 0$, such that the location $l_{\texttt{halt}}$ is unavoidable in our bounded L/U-automaton. □

Liveness in L/U-Parametric Timed Automata

Étienne André
Université Paris 13, Sorbonne Paris Cité, LIPN
CNRS, UMR 7030, F-93430, Villetaneuse, France

Didier Lime
École Centrale de Nantes, LS2N
CNRS, UMR 6004, Nantes, France

***Abstract*—We study timed systems in which some timing features are unknown parameters. Parametric timed automata are a classical formalism for such systems but for which most interesting problems are undecidable. Lower-bound/upper-bound parametric timed automata (L/U-PTAs) achieve decidability for reachability properties by enforcing a separation of parameters used as upper bounds in the automaton constraints, and those used as lower bounds.**

We further study L/U-PTAs by considering liveness related problems. We prove that: (1) the existence of at least one parameter valuation for which there exists an infinite run in the automaton is PSPACE-complete; (2) the existence of a parameter valuation such that the system has a deadlock is however undecidable; (3) the existence of a valuation for which a run remains in a given set of locations exhibits a very thin border between decidability and undecidability.

1. Introduction

Following Lamport, properties of systems are often characterized as safety properties ("something bad will never happen") and liveness properties ("something good will eventually happen") [1]. Safety generally reduces to reachability, while liveness is more complex. The "good" behavior may not be reached for two main reasons: either there is a deadlock, a state in which the system cannot evolve anymore, or there is a livelock, an infinite path never reaching the "good" behavior. Both situations are captured by the CTL operator EG [2].

We study here those behaviors in the context of parametric timed systems, in which some timing features (*e. g.,* the duration of a task, a transmission delay in a network, the delay to trigger a watchdog, etc.) are not known and replaced by symbolic constants, called *parameters*. The objective of verification on such partially defined systems, is then to synthesize the possible valuations of parameters such that some properties are satisfied.

Parametric timed automata (PTAs) [3] have been introduced to deal with such parametric timed systems. They consist in finite automata equipped with real-valued clocks that can be compared with constants or parameters in constraints restricting if and when the edges can be taken.

The simple problem of whether there exists a valuation for each parameter such that some control location is reachable in the timed automaton obtained by replacing the parameters with those valuations (also called EF-emptiness) is undecidable for PTAs for both integer- and rational-valued parameters. Several alternative proofs refine this result in terms of the number of parameters, number of clocks compared to parameters, types of constraints, etc. (see, *e. g.,* [4], [5], [6], [7], [8]).

In order to overcome these disappointing results, lower-bound/upper-bound parametric timed automata (L/U-PTAs) are introduced as a subclass of PTAs where each parameter either always appears as an upper bound when compared to a clock, or always as a lower bound [9]. The EF-emptiness problem, and also the EF-universality problem ("Can we reach a given location, regardless of what valuations we give to the parameters?") are decidable for L/U-PTAs.

In [10], infinite acceptance properties are considered: the emptiness and the universality of the valuation set for which a given location is infinitely often traversed are decidable for integer-valued parameters. In [11], it is shown that the AF-emptiness problem ("Does there exist a valuation of the parameters, such that the system reaches a given location for all runs?") is undecidable for L/U-PTAs with integer- and rational-valued parameters.

Contribution. With the notable exception of [11], and to some extent of [10] which addresses the existence of cycles, all the works cited above focus on safety properties, through the basic problem of reachability. This is maybe not so surprising given that most results related to this simpler problem are already negative.

We nonetheless address here the problem of liveness in PTAs, and more precisely, with the negative result of [11] on AF-emptiness in mind, we start from L/U-PTAs with rational-valued parameters and further refine both the model and the properties. We prove that:

1) deciding the existence of at least one parameter valuation for which there exists an infinite run (discrete cycle) in the automaton is PSPACE-complete;
2) deciding the existence of a parameter valuation such that the system has a deadlock is however undecidable;
3) the problem of the existence of a valuation for which a run remains in a given set of locations exhibits a very thin border between decidability and undecidability: while this problem is decidable for L/U-PTAs with a bounded parameter domain with closed bounds,

it becomes undecidable if either the assumption of boundedness or of closed bounds is lifted. This result confirms that L/U-PTAs stand at the border between decidability and undecidability.

Outline. We recall the necessary preliminaries in Section 2. We then consider the problem of the existence of at least one parameter valuation for which there exists an infinite run (Section 3), for which there exists a deadlock (Section 4), and for which a run remains in a given set of locations (Section 5). We conclude and discuss perspectives in Section 6.

2. Preliminaries

2.1. Clocks, Parameters and Constraints

Let $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}_+$ and $\mathbb{R}_+$ denote the sets of non-negative integers, integers, non-negative rational numbers and non-negative real numbers respectively. Let $\mathcal{I}(\mathbb{N})$ denote the set of non-necessarily closed intervals on $\mathbb{N}$, *i. e.,* of the form $[a, b]$, $(a, b]$, $[a, b)$ or (a, b) where $a, b \in \mathbb{N}$ and $a \leq b$.

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 $(w(x_1), \ldots, w(x_H))$ of $\mathbb{R}_+^H$. We write $\vec{0}$ for the clock valuation that assigns 0 to all clocks. Given $d \in \mathbb{R}_+$, $w + d$ denotes the valuation such that $(w + d)(x) = w(x) + d$, for all $x \in X$. Given $R \subseteq X$, we define the *reset* of a valuation w, denoted by $[w]_R$, as follows: $[w]_R(x) = 0$ if $x \in R$, and $[w]_R(x) = w(x)$ otherwise.

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 $(v(p_1), \ldots, v(p_M))$ of $\mathbb{Q}_+^M$. An *integer* parameter valuation is such that $\forall p \in P, v(p) \in \mathbb{N}$.

In the following, we assume $\bowtie \in \{<, \leq, \geq, >\}$. Throughout this paper, *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 $x_i \in X$, $p_j \in P$, and $\alpha_i, \beta_j, d \in \mathbb{Z}$. A *constraint* C (*i. e.,* a convex polyhedron) over $X \cup P$ is a conjunction of inequalities of the form $lt \bowtie 0$. 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. $w|v \models C$.

A *guard* g is a constraint over $X \cup P$ defined by inequalities of the form $x \bowtie \sum_{1 \leq j \leq M} \beta_j p_j + d$, with $\beta_j \in \{0, 1\}$ and $d \in \mathbb{Z}$.

2.2. Parametric Timed Automata

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 finite set of clocks, v) P is a finite set of parameters, vi) I is the invariant, assigning to every $l \in L$ a guard $I(l)$, vii) E is a finite set of edges $e = (l, g, \sigma, R, l')$ where $l, l' \in L$ are the source and target locations, $\sigma \in \Sigma$, $R \subseteq X$ 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 (Concrete 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 $(S, s_0, \to)$, with*

- $S = \{(l, w) \in L \times \mathbb{R}_+^H \mid w|v \models I(l)\}$, $s_0 = (l_0, \vec{0})$
- $\to$ *consists of the discrete and (continuous) delay transition relations:*
  - *discrete transitions:* $(l, w) \xrightarrow{e} (l', w')$, *if* $(l, w), (l', w') \in S$, *there exists* $e = (l, g, \sigma, R, l') \in E$, $w' = [w]_R$, *and* $w|v \models g$.
  - *delay transitions:* $(l, w) \xrightarrow{d} (l, w + d)$, *with* $d \in \mathbb{R}_+$, *if* $\forall d' \in [0, d], (l, w + d') \in S$.

Moreover we write $(l, w) \overset{e}{\mapsto} (l', w')$ for a sequence of delay and discrete transitions where $((l, w), e, (l', w')) \in \mapsto$ if $\exists d, w'' : (l, w) \xrightarrow{d} (l, w'') \xrightarrow{e} (l', w')$.

Given a TA $v(\mathcal{A})$ with concrete semantics $(S, s_0, \to)$, we refer to the states of S as the *concrete states* of $v(\mathcal{A})$. A (concrete) *run* (resp. *infinite run*) of $v(\mathcal{A})$ is an alternating sequence of concrete states of $v(\mathcal{A})$ and edges starting from the initial concrete state s_0 of the form $s_0 \overset{e_0}{\mapsto} s_1 \overset{e_1}{\mapsto} \cdots \overset{e_{m-1}}{\mapsto} s_m$ (resp. $s_0 \overset{e_0}{\mapsto} s_1 \overset{e_1}{\mapsto} \cdots \overset{e_{m-1}}{\mapsto} s_m \overset{e_m}{\mapsto} \cdots$), such that for all $i = 0, \ldots, m-1$ (resp. $i = 0, 1, \ldots$), $e_i \in E$, and $(s_i, e_i, s_{i+1}) \in \mapsto$. 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 $v(\mathcal{A})$. By extension, we say that l is reachable in $v(\mathcal{A})$, if there exists a state (l, w) that is reachable. Given a set of locations $G \subseteq L$, we say that a run stays in G if all of its states (l, w) are such that $l \in G$. A maximal run is a run that is either infinite (*i. e.,* contains an infinite number of discrete transitions), or that cannot be extended by a discrete transition. A maximal run is deadlocked if it is finite, *i. e.,* contains a finite number of discrete transitions. By extension, we say that a TA is deadlocked if it contains at least one deadlocked run.

2.3. Subclasses of PTAs

Definition 3 (L/U-PTA)**.** *An L/U-PTA is a PTA where the set of parameters is partitioned into lower-bound parameters and upper-bound parameters, where an upper-bound (resp. lower-bound) parameter p_i is such that, for every guard*

or invariant constraint $x \bowtie \sum_{1 \leq j \leq M} \beta_j p_j + d$*, we have:* $\beta_i = 1$ *implies* $\bowtie \in \{\leq, <\}$ *(resp.* $\bowtie \in \{\geq, >\}$*).*

Recall from our definition of guard that β_i can only be 0 or 1 and therefore cannot be negative.

L/U-PTAs enjoy a well-known monotonicity property recalled in the following lemma (that corresponds to a reformulation of [9, Prop 4.2]), stating that increasing upper-bound parameters or decreasing lower-bound parameters can only add behaviors.

Lemma 1. *Let* $\mathcal{A}$ *be an L/U-PTA and* v *be a parameter valuation. Let* v' *be a valuation such that for each upper-bound parameter* p^+*,* $v'(p^+) \geq v(p^+)$ *and for each lower-bound parameter* p^-*,* $v'(p^-) \leq v(p^-)$*. Then any run of* $v(\mathcal{A})$ *is a run of* $v'(\mathcal{A})$*.*

In this paper, we will also consider *bounded* PTAs, *i. e.,* PTAs with a bounded parameter domain that assigns to each parameter an infimum and a supremum, both integers.

Definition 4 (bounded PTA)**.** *A* bounded PTA *is* $\mathcal{A}_{|bounds}$*, where* $\mathcal{A}$ *is a PTA, and* $bounds : P \to \mathcal{I}(\mathbb{N})$ *assigns to each parameter* p *an interval* $[\inf, \sup]$*,* $(\inf, \sup]$*,* $[\inf, \sup)$*, or* $(\inf, \sup)$*, with* $\inf, \sup \in \mathbb{N}$*. We use* $\inf(p, bounds)$ *and* $\sup(p, bounds)$ *to denote the infimum and the supremum of* p*, respectively. (Note that we rule out* ∞ *as a supremum.)*

We say that a bounded PTA is a closed bounded PTA *if, for each parameter* p*, its ranging interval* $bounds(p)$ *is of the form* $[\inf, \sup]$*; otherwise it is an* open bounded PTA*.*

We define similarly bounded L/U-PTAs.

2.4. Decision Problems

Let $\mathcal{P}$ be a given a class of decision problems.

$\mathcal{P}$-emptiness problem:
INPUT: A PTA $\mathcal{A}$ and an instance ϕ of $\mathcal{P}$
PROBLEM: Is the set of parameter valuations v such that $v(\mathcal{A})$ satisfies ϕ empty?

In this paper, we mainly focus on the following three decision problems:

- deadlock-existence: given a TA $v(\mathcal{A})$, is there at least one run of $v(\mathcal{A})$ that is deadlocked, *i. e.,* has no discrete successor (possibly after some delay)?
- cycle-existence: given a TA $v(\mathcal{A})$, is there at least one run of $v(\mathcal{A})$ with an infinite number of discrete transitions?
- EG: given a TA $v(\mathcal{A})$ and a subset G of its locations, is there at least one maximal run of $v(\mathcal{A})$ along which the location always remain in G?

For example, given a PTA $\mathcal{A}$, deadlock-existence-emptiness asks: "does there exist a valuation v of the parameters such that at least one run of $v(\mathcal{A})$ is deadlocked, *i. e.,* has no discrete successor". In the following, we often abbreviate deadlock-existence-emptiness and cycle-existence-emptiness as ED-emptiness and EC-emptiness, respectively.

Note that ED-emptiness is equivalent to AC-universality, where AC-universality asks whether all parameter valuations are such that all maximal runs contain an infinite number of discrete transitions. Conversely, EC-emptiness is equivalent to AD-universality (for all valuations, all runs are deadlocked). In addition, EG-emptiness is also close to both former problems: EG is true if there exists either a finite run with a deadlock staying in G, or an infinite run staying in G.

3. Cycle-Existence-Emptiness

Theorem 1. *The cycle-existence-emptiness problem is decidable for closed bounded L/U-PTAs.*

Proof. Recall that, thanks to the monotonicity property of L/U-PTAs (recalled in Lemma 1), any run possible for a valuation v of the parameters is also possible for any valuation of the parameters for which the upper-bound (resp. lower-bound) parameters are larger (resp. smaller) than or equal to that of v.

Let $\mathcal{A}_{|bounds}$ be a closed bounded L/U-PTA. Let $v_{\inf/\sup}$ be the valuation such that, for each lower-bound parameter p^-, $v_{\inf/\sup}(p^-) = \inf(p^-, bounds)$ and, for each upper-bound parameter p^+, $v_{\inf/\sup}(p^+) = \sup(p^+, bounds)$.

1) If $v_{\inf/\sup}(\mathcal{A})$ contains an infinite run (which can be checked in PSPACE [12]), then since $\mathcal{A}_{|bounds}$ is closed, $v_{\inf/\sup}$ belongs to $bounds$, and hence the set of parameter valuations that yield an infinite run is not empty.
2) On the contrary, if $v_{\inf/\sup}(\mathcal{A})$ contains no infinite run, then from the monotonicity property of L/U-PTAs (Lemma 1), no other valuation in $bounds$ gives a TA with an infinite run, as such a TA could only contain less runs. Hence the set of parameter valuations that yield an infinite run is empty.

□

The above result cannot be used as such for non-bounded L/U-PTAs as a cycle that exists for an infinite parameter valuation may not exist for any finite parameter valuation: consider the L/U-PTA in Figure 1b. This L/U-PTA has an infinite run for $p = \infty$, but for any parameter valuation (*i. e.,* different from ∞), the number of self-loops in l_0 is bounded by p, and hence finite. However, extending to rational-valued parameters a result from [10], we can still prove decidability.

Lemma 2. *Given an L/U-PTA* $\mathcal{A}$ *and a subset of its locations* G*, the problem of the existence of at least one parameter valuation* v *such that* $v(\mathcal{A})$ *has a run passing infinitely often through* G *is PSPACE-complete.*

Proof. Let us prove that there exists a rational-valued valuation satisfying the property iff there exists an integer-valued valuation doing so.

- $\Leftarrow$ Considering an integer valuation is also a rational-valued valuation, the result trivially holds.
- $\Rightarrow$ Assume there exists a rational-valued parameter valuation v for which $v(\mathcal{A})$ contains an infinite run passing

(a) (b) (c) (d)

Figure 1: Examples of PTA (a) and L/U-PTAs (b–d)

infinitely often through locations of G. Let v' be the integer parameter valuation obtained from v as follows:

$$v'(p) = \begin{cases} v(p) & \text{if } v(p) \in \mathbb{N} \\ \lceil v(p) \rceil & \text{if } p \text{ is an upper-bound parameter} \\ \lfloor v(p) \rfloor & \text{if } p \text{ is a lower-bound parameter} \end{cases}$$

From the monotonicity property of L/U-PTAs (Lemma 1), if $v(\mathcal{A})$ yields an infinite run passing infinitely often through locations of G, then $v'(\mathcal{A})$ does too.

Observe that this is not true for general PTAs: in Figure 1a, there is an infinite run passing infinitely often through l_0 iff $0 < p < 1$; therefore, there exist rational-valued valuations satisfying the property, but no integer-valued valuation.

Now, in [10, Theorem 8], it is proved that the problem of the the emptiness of the set of integer parameter valuations for which there exists an infinite run passing infinitely often through G is PSPACE-complete. This concludes the proof. □

Theorem 2. *The cycle-existence-emptiness problem is PSPACE-complete for L/U-PTAs.*

Proof. Let $\mathcal{A}$ be an L/U-PTA. The set of parameter valuations for which $\mathcal{A}$ has an infinite run is empty iff the set of parameter valuations for which $\mathcal{A}$ has an infinite run passing infinitely often through L (where L denotes all locations of $\mathcal{A}$) is empty. Hence we can directly apply our intermediate Lemma 2 to conclude that this problem is decidable and PSPACE-complete. □

Without surprise, this problem becomes undecidable for general PTAs, even when bounded. We do include the full proof of this result as it will be used later on to prove more subtle results.

Theorem 3. *The cycle-existence-emptiness problem is undecidable for (bounded) PTAs with 3 clocks and 1 parameter.*

Proof. We reduce from the boundedness problem of a 2-counter machine, which is undecidable [13]. Recall that a deterministic 2-counter machine has two non-negative counters C_1 and C_2, a finite number of states and transitions, which can be of the form:

- when in state q_i, increment C_k and go to q_j;
- when in state q_i, if $C_k = 0$ then go to q_k, otherwise go to q_j.

The machine starts in state q_0 with the counters set to 0; by definition, it halts when it reaches a specific state called q_{halt}. The boundedness problem for 2-counter machines asks whether, along the unique maximal run, the value of the counters remains smaller than some bound, and is undecidable [13].

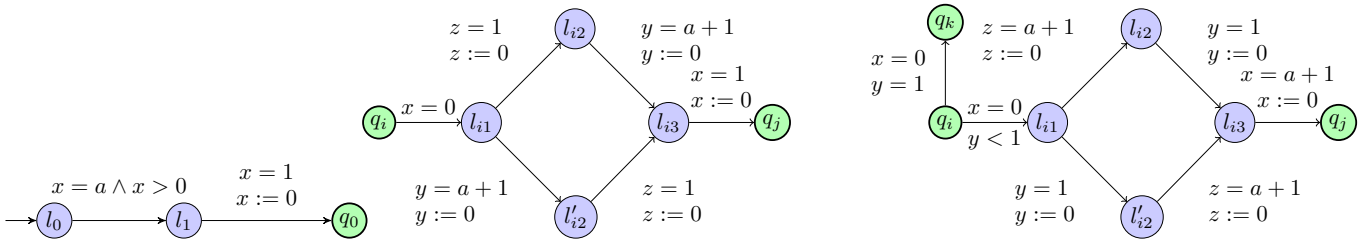
Given such a machine $\mathcal{M}$, we encode it as a PTA $\mathcal{A}(\mathcal{M})$; our encoding is adapted from an existing encoding of a 2-counter machine, used to (re)prove the undecidability of the EF-emptiness problem for bounded PTAs and then further related results, and found in [14]. However, we had to modify it in two directions:*i*) we adapt the construction so that it fits the cycle-existence-problem instead of the EF-emptiness problem; and *ii*) we change the model of the 2-counter machine (in [14], we used the model of [3], where the machine has three instructions: increment, decrement, zero-test (and block if unsatisfied)). This second part required us to modify the gadgets.

Let us now describe this encoding in details, as we will modify it in the subsequent proofs.

Each state q_i of the machine is encoded as a location of the automaton, which we call q_i. The counters are encoded using clocks x, y and z and one parameter a, with the following relations with the values c_1 and c_2 of counters C_1 and C_2: when $x = 0$, we have $y = 1 - ac_1$ and $z = 1 - ac_2$. All three clocks are parametric, *i. e.,* are compared with a in some guard or invariant of the encoding. We will see that a is a rational-valued bounded parameter, typically in $[0, 1]$ (although not bounding a has no impact on the proof).

We initialize the clocks with the gadget in Figure 2a (that also blocks the case where $a = 0$). Note that, throughout the paper, we highlight in thick green the locations of the PTA corresponding to a state of the 2CM (in contrast with other locations added in the encoding to maintain the matching between the clock values and the counter values). Since all clocks are initially 0, in Figure 2a clearly, when in q_0 with $x = 0$, we have $y = z = 1$, which indeed corresponds to counter values 0.

We now present the gadget encoding the increment instruction of C_1 in Figure 2b. The transition from q_i to l_{i1} only serves to clearly indicate the entry in the increment gadget and is done in 0 time unit. Since we use only equalities, there are really only two paths that go through the gadget: one going through l_{i2} and one through l'_{i2}. Let us begin with the former. We start from some encoding configuration: $x = 0$, $y = 1 - ac_1$ and $z = 1 - ac_2$ in q_i (and therefore the same in l_{i1}). We can enter l_{i2} (after elapsing enough time) if $1 - ac_2 \leq 1$, *i. e.,* $ac_2 \geq 0$, which implies that $a \geq 0$, and when entering l_{i2} we have $x = ac_2$, $y = 1 - ac_1 + ac_2$ and $z = 0$. Then we can enter l_{i3} if $1 - ac_1 + ac_2 \leq 1 + a$, *i. e.,* $a(c_1 + 1) \geq ac_2$. When entering l_{i3}, we then have $x = a(c_1 + 1)$, $y = 0$ and



(a) Initial gadget (b) Increment gadget (C_1) (c) 0-test and decrement gadget (C_1)

Figure 2: EC-emptiness: gadgets

$z = a(c_1+1)-ac_2$. Finally, we can go to q_j if $a(c_1+1) \leq 1$ and when entering q_j we have $x = 0$, $y = 1 - a(c_1 + 1)$ and $z = 1 - ac_2$, as expected.

We now examine the second path. We can enter l'_{i2} if $1 - ac_1 \leq a + 1$, *i. e.,* $a(c_1 + 1) \geq 0$, and when entering l'_{i2} we have $x = a(c_1 + 1)$, $y = 0$ and $z = 1 - ac_2 + a(c_1 + 1)$. Then we can go to l_{i3} if $1 - ac_2 + a(c_1 + 1) \leq 1 + a$, *i. e.,* $a(c_1 + 1) \leq ac_2$. When entering l_{i3}, we then have $x = ac_2$, $y = ac_2 - a(c_1 + 1)$ and $z = 0$. Finally, we can go to q_j if $ac_2 \leq 1$ and when entering q_j we have $x = 0$, $y = 1 - a(c_1 + 1)$ and $z = 1 - ac_2$, as expected.

Remark that exactly one path can be taken depending on the respective order of $c_1 + 1$ and c_2, except when both are equal or $a = 0$, in which cases both paths lead to the same configuration anyway (and the case $a = 0$ is excluded by Figure 2a anyway).

Decrement is done similarly by replacing guards $y = a+1$ with $y = 1$, and guards $x = 1$ and $z = 1$ with $x = a+1$ and $z = a + 1$, respectively, as shown in Figure 2c. In addition, the 0-test is obtained by simply adding a transition from q_i to q_k with guard $y = 1 \wedge x = 0$, which ensures that $C_1 = 0$. Similarly, the guard from q_i to l_{i1} ensures that decrement is done only when the counter is not null.

All those gadgets also work for C_2 by swapping y and z.

The actions associated with the transitions do not matter; we can assume a single action σ on all transitions (omitted in all figures).

Finally, we add a self-loop (with no guard) on the location $q_{\texttt{halt}}$, ensuring that whenever $q_{\texttt{halt}}$ is reachable then there exists an infinite run in the PTA.

We now prove that the value of the counters remains bounded iff there exists a parameter valuation v such that $v(\mathcal{A})$ yields an infinite run. First note that if $a = 0$ the initial gadget cannot be passed, and there is no infinite run. Assume $a > 0$. Consider two cases:

1) either the value of the counters is not bounded. Then, for any parameter valuation, at some point during an incrementation of, say, C_1 we will have $a(c_1 + 1) > 1$ when taking the transition from l_{i2} to l_{i3} and the PTA will be blocked. Therefore, there exists no parameter valuation for which there exists an infinite run.
2) or the value of the counters remains bounded. Let c be their maximal value. Let us consider two subcases:
   a) either the machine reaches $q_{\texttt{halt}}$: in that case, if $c = 0$ and $0 < a \leq 1$ or $c > 0$ and $ca < 1$, then the PTA valuated with such parameter valuations correctly simulates the machine, yielding a (unique) run reaching location $q_{\texttt{halt}}$. From there, this run is infinite thanks to the self-loop on $q_{\texttt{halt}}$. The set of such valuations for a is certainly non-empty: $a = \frac{1}{2}$ belongs to it if $c = 0$ and $a = \frac{1}{c}$ does otherwise.
   b) or the machine does not halt. Then again, for a sufficiently small parameter valuation (*i. e.,* $a < 1$ if $c = 0$ and $a \leq \frac{1}{c}$ otherwise), the machine is properly simulated, and since the machine does not halt, then the run simulating the infinite execution is infinite too. For other values of a, the machine will block at some point in an increment gadget, because a is not small enough and the guard to q_j cannot be satisfied.

   In both subcases, there exist parameter valuations for which there exists an infinite run.

Hence the value of the counters remains bounded iff there exists a parameter valuation v such that $v(\mathcal{A})$ contains an infinite run. □

Remark 1. *Throughout this paper, we allow guards and invariants of the form $x \bowtie \sum_{1 \leq j \leq M} \beta_j p_j + d$, which is more restrictive than [10] (that allows parametric coefficients different from 0 and 1, as well as diagonal constraints), but more permissive than [3], that only allows a syntax $x \bowtie p$. In fact, most papers in the literature define their own syntax (see [8] for a survey). We can adapt our proof to fit in the most restrictive syntax ($x \bowtie p$) as follows: transitions with $y = a + 1$ guards and $y := 0$ reset can be equivalently replaced by one transition with a $y = 1$ guard and a reset of some additional clock w, followed by a transition with a $w = a$ guard and the $y := 0$ reset (and similarly for x and z is the decrement gadget). This also allows the proof to work without complex parametric expressions in guards, using three additional clocks (we conjecture that a smarter encoding can be exhibited to factor these additional clocks, so as to use a single additional clock). A similar modification can be applied to all subsequent undecidability proofs.*

Finally note that the EC-emptiness problem for the class of open bounded L/U-PTAs (that does not fit in Theorems 1 and 2) remains an open problem. We conjecture that this is decidable using techniques derived from the robustness results of [15] but the adaptation appears to require rather

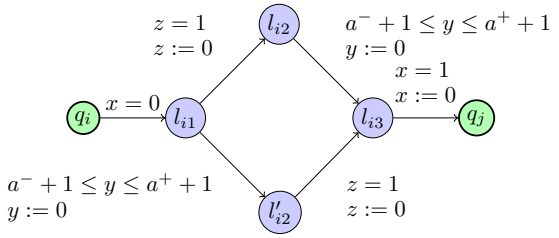
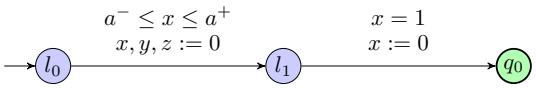
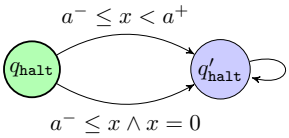


Figure 3: ED-emptiness for bounded L/U-PTAs: increment gadget



(a) Initial gadget



(b) Final gadget

Figure 4: ED-emptiness for bounded L/U-PTAs: initial and final gadgets

lenghty developments, with techniques quite different from those presented here, and is thus left to future work.

4. Deadlock-Existence-Emptiness

Theorem 4. *The deadlock-existence-emptiness problem is undecidable for closed bounded L/U-PTAs, with 3 clocks and 2 parameters.*

Proof. We will use a reduction from the halting problem of a 2-counter machine. Let us consider the encoding used in the proof of Theorem 3, that we transform into an L/U-PTA by replacing any comparison of a clock with a (say $x = a$) into $x \leq a^+ \wedge x \geq a^-$, where a^- (resp. a^+) is a lower-bound (resp. upper-bound) parameter. The crux of the proof is in the original enforcement of constraints in the encoding (in particular with location $q'_{\texttt{halt}}$) such that the deadlock property ensures that $a^- = a^+$.

We give the modified increment gadget in Figure 3 (the decrement gadget is modified in a similar fashion). We replace the initial gadget (Figure 2a) with the new one in Figure 4a. Before initializing the values of the counters, this gadget first ensures that $a^- \leq a^+$.

We also add a new location $q'_{\texttt{halt}}$ reachable from $q_{\texttt{halt}}$ as shown in the final gadget in Figure 4b. Finally, we add an unguarded transition (*i. e.,* a transition the guard of which is true) from any location of the encoding (including that of the initial gadget, but excluding $q_{\texttt{halt}}$) to location $q'_{\texttt{halt}}$. That is, it is always possible to reach $q'_{\texttt{halt}}$ from any location without condition, except from $q_{\texttt{halt}}$. From that particular location, $q'_{\texttt{halt}}$ is reachable if and only if $a^- < a^+$ or $a^- = 0$.

We assume the following bounds for the parameters: $a^-, a^+ \in [0, 1]$.

Let us show that there exists a parameter valuation for which the system contains at least one deadlock iff the 2-counter machine halts, which is undecidable [13]. Let us reason by cases on the valuations of a^- and a^+.

1) If $a^- > a^+$, the initial gadget cannot be passed, but thanks to the unguarded transitions to $q'_{\texttt{halt}}$, all runs eventually end in $q'_{\texttt{halt}}$, from which the absence of deadlock is guaranteed by the unguarded self-loop.
2) If $a^- < a^+$, the machine may not be properly simulated because some transitions do not occur at the right time and some run could reach $q_{\texttt{halt}}$ while the machine does not halt. Let us consider a run in the TA obtained with such a parameter valuation.
   a) either this run is infinite and remains in the machine (*e. g.,* it loops infinitely through the increment, decrement and 0-test gadgets of our encoding). Then there is no deadlock.
   b) or this run would block in a gadget; in that case, thanks to the unguarded transitions to $q'_{\texttt{halt}}$, this run can go to $q'_{\texttt{halt}}$, from which it is deadlock-free.
   c) or this run reaches $q_{\texttt{halt}}$ (recall that the value of x is necessarily 0 when entering $q_{\texttt{halt}}$); from there, thanks to the upper transition in Figure 4b, it can reach $q'_{\texttt{halt}}$, from which it is again deadlock-free.
3) If $a^- = a^+ = 0$, the machine may again not be properly simulated: again we could reach $q_{\texttt{halt}}$ while the machine does not halt. The situation is similar to the previous case ($a^- < a^+$) except that in $q_{\texttt{halt}}$ a run has to take the lower transition in Figure 4b to reach $q'_{\texttt{halt}}$, from which it is again deadlock-free.
4) If $a^- = a^+ > 0$:
   a) Either the machine does not halt:
      i) ...and the counters remain bounded: for some parameter valuations small enough to encode the value of the counters (typically $a^- = a^+ \leq \frac{1}{c}$, where c is the maximum value of both C_1 and C_2) then the PTA correctly simulates the infinite execution of the machine, and the system is deadlock-free. (Note that such valuations can also lead to $q'_{\texttt{halt}}$ anytime, but this is harmless since this location guarantees the absence of deadlocks.) For other valuations, at some point we have $a^- c_1 > 1$; more specifically, there is an incrementation of C_1 such that $a^- c_1 \leq 1$ and $a^-(c_1 + 1) > 1$. Hence, the run cannot continue in the encoding, but can reach $q'_{\texttt{halt}}$, from where the run is non-blocking.
      ii) ...and the counters are unbounded. Then whatever the value of $a^- > 0$, at some point we have $a^- c_1 > 1$. Then, when executing the corresponding increment gadget, $q'_{\texttt{halt}}$ can be reached from l_{i2}, from where the run is non-blocking.

   Hence if the machine does not halt, the system is deadlock-free for all parameter valuations.

b) Or the machine halts. In this case, if c is the maximum value of both C_1 and C_2 over the (necessarily finite) halting execution of the machine, and if $c > 0$, then for valuations such that $a^- = a^+ \leq \frac{1}{c}$, then there exists one run that correctly simulates the machine (beside plenty of runs that will go to $q'_{\texttt{halt}}$ due to the unguarded transitions from all locations except $q_{\texttt{halt}}$); this run that correctly simulates the machine eventually reaches $q_{\texttt{halt}}$. From $q_{\texttt{halt}}$, for such valuations, the system is deadlocked: indeed, the transitions from $q_{\texttt{halt}}$ to $q'_{\texttt{halt}}$ can only be taken if $a^- < a^+$ or $a^- = 0$. And there is no unguarded transition from $q_{\texttt{halt}}$ to $q'_{\texttt{halt}}$, which is crucial for the correctness of our encoding. The set of such valuations for which there exists a run that correctly simulates the machine is certainly non-empty: $a^- = a^+ = \frac{1}{c}$ belongs to it (if $c = 0$ then we choose, *e. g.,* $a^- = a^+ = \frac{1}{2}$). Hence, if the 2-counter machine halts, there exist parameter valuations for which a run has no discrete successor, and hence the system is not deadlock-free.

Hence the 2-counter machine halts iff the set of valuations for which the automaton has at least one deadlock is not empty. □

Corollary 1. *The deadlock-existence-emptiness problem is undecidable for open bounded L/U-PTAs, L/U-PTAs, bounded PTAs and PTAs, with 3 clocks and 2 parameters.*

Proof. Let us consider each formalism:

open bounded L/U-PTAs In the above construction, we can assume, *e. g.,* $a^- \in (0, 1]$, which does not impact the proof.

L/U-PTAs The bounds on the parameters are not required in the above construction: for valuations larger than 1 (that necessarily do not simulate correctly the machine), a gadget may block, therefore leading to $q'_{\texttt{halt}}$, from which the system is deadlock-free, hence without impacting the spirit of the proof.

bounded PTAs From the fact that a bounded L/U-PTA is a bounded PTA.

PTAs From the fact that an L/U-PTA is a PTA.

Observe that the number of parameters can be reduced to 1 for (possibly bounded) PTAs by merging a^- and a^+ into a single parameter a. □

5. EG-Emptiness

In this section, we prove that the EG-emptiness problem is decidable for closed bounded L/U-PTAs, and that lifting either closedness or boundedness leads to undecidability.

Theorem 5. *The EG-emptiness problem is decidable for closed bounded L/U-PTAs.*

We will use Lemma 1 to deal with infinite paths but it is of no use for deadlocks: by decreasing lower-bounds or increasing upper-bounds, some deadlocks can actually be removed. We will therefore also use the symbolic semantics of PTAs (see, *e. g.,* [11]), which we need first to recall.

We define the *time elapsing* of a constraint 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. That is, $C^{\nearrow} = \{w' | v \mid w \models v(C) \wedge \forall x \in X : w'(x) = w(x) + d, d \in \mathbb{R}_+\}$. Dually, 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. That is, $C^{\swarrow} = \{w' | v \mid w \models v(C) \wedge \forall x \in X : w'(x) + d = w(x), d \in \mathbb{R}_+\}$. 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 (*e. g.,* using Fourier-Motzkin).

The (sets of clock valuations satisfying the) constraints generated by PTA can be represented by subsets of $\mathbb{R}_+^{|X|}$ with a special form called *parametric zone* ([9]). A parametric zone is a convex polyhedron over $X \cup P$ in which all constraints on variables are of the form $x \bowtie plt$ (parametric rectangular constraints), or $x_i - x_j \bowtie plt$ (parametric diagonal constraints), where $x_i \in X$, $x_j \in X$ and plt is a parametric linear term over P, *i. e.,* a linear term without clocks ($\alpha_i = 0$ for all i).

A symbolic state is a pair $\mathbf{s} = (l, C)$ where $l \in L$ is a location, and C its associated parametric zone. The initial symbolic state of $\mathcal{A}$ is $\mathbf{s}_0 = \big(l_0, (\{\vec{0}\} \wedge I(l_0))^{\nearrow} \wedge I(l_0)\big)$.

The symbolic semantics relies on the Succ operation. Given a symbolic state $\mathbf{s} = (l, C)$ and an edge $e = (l, g, \sigma, R, l')$, $\mathsf{Succ}(\mathbf{s}, e) = (l', C')$, with $C' = \big([(C \wedge g)]_R \wedge I(l')\big)^{\nearrow} \wedge I(l')$. The Succ operation is effectively computable, using polyhedral operations: note that the successor of a parametric zone C is a parametric zone (see *e. g.,* [11]).

A symbolic run of a PTA is an alternating sequence of symbolic states and edges starting from the initial symbolic state, of the form $\mathbf{s}_0 \overset{e_0}{\Rightarrow} \mathbf{s}_1 \overset{e_1}{\Rightarrow} \cdots \overset{e_{m-1}}{\Rightarrow} \mathbf{s}_m$, such that for all $i = 0, \ldots, m-1$, we have $e_i \in E$, and $\mathbf{s}_{i+1} = \mathsf{Succ}(\mathbf{s}_i, e_i)$. In the following, we simply refer to symbolic states belonging to a run of $\mathcal{A}$ as symbolic states of $\mathcal{A}$.

We can now come back to the proof of Theorem 5.

Proof. Let $\mathcal{A}_{|bounds}$ be a closed bounded L/U-PTA and G be a subset of its locations. Since $\mathcal{A}$ is closed and bounded, for each parameter p, $bounds(p)$ is a closed interval $[m^-(p), m^+(p)]$.

Lemma 1 ensures that the TA $v_{\mathrm{inf/sup}}(\mathcal{A})$, where $v_{\mathrm{inf/sup}}$ is obtained by valuating lower-bound parameters p^- by $m^-(p)$ and upper-bound parameters p^+ by $m^+(p)$, includes all the runs that could be produced with other parameter valuations. Consequently, if there is an infinite path for some valuation, there is one for $v_{\mathrm{inf/sup}}$ (note that, as emphasized above, this is not true for deadlocks).

In $v_{\mathrm{inf/sup}}(\mathcal{A})$, it is decidable to find an infinite path staying in G, or conclude that none exist: this can be encoded into the CTL formula $EG(G \wedge XG)$, to be verified

on the (finite) region graph of $\mathcal{A}$ [12]. Since the region equivalence is a time-abstract bisimulation [16], this means for $\mathcal{A}$ "there exists a path that remains in G and in which every state has a discrete successor (possibly after letting some time elapse) in G". That path therefore has an infinite number of discrete actions. If we do find such a path, we can then terminate by answering yes to the EG-emptiness problem. If we do not, then in $v_{\inf/\sup}(\mathcal{A})$, all paths staying in G are finite. If we keep only discrete actions and locations, which are in finite number, the resulting paths therefore form a finite tree. Let us recall again that, thanks to Lemma 1, all the discrete paths that stay in G and can be obtained with any parameter valuation, belong to that tree.

We can now explicitly compute the symbolic states (following the symbolic semantics recalled above) for all the paths in the finite tree (not only those that are maximal). Recall that each symbolic state $\mathbf{s}$ is a pair (l, C), where l is a location and C a convex polyhedron representing all parameter valuations and clock valuations that can be reached by the given discrete path. In each of these polyhedra, we can explicitly check for the existence of a deadlock: *i*) remove all parts that are in the past of the guard of an outgoing transition in $\mathcal{A}$ (using operation $C^{\swarrow}$), and that would satisfy the target location invariant; *ii*) test for emptiness.

If the result is not empty then there exists a point in the tested set which can be decomposed into a parameter valuation and clock valuation such that, by any time elapsing from the clock valuation, none of the guards can become true. We therefore have a deadlock. It the result is empty, by the same reasoning, we can take a transition (possibly by first letting some time elapse) from all states of C, so none of them are deadlocked. Note that both operations can be performed using classical polyhedral operations.

If we find a deadlock, then we can terminate and answer yes to the EG-emptiness problem. Otherwise, we can terminate and answer no, because we have checked all the potential discrete paths staying in G for any parameter valuation. □

Note that this proof fails when the L/U-PTA is not bounded or closed. In particular, the closedness plays a key role in the sense that we are able to test the valuation $v_{\inf/\sup}$. Consider the L/U-PTA in Figure 1c. As p grows, there are more and more discrete behaviors, but there is no cycle for any parameter valuation. In [10], the authors provide a finite upper bound $N_{\mathcal{A}}$ for the upper-bound parameters such that if there exists a valuation such that the valuated L/U-PTA has an accepting run, then the valuation giving 0 to lower bound parameters and $N_{\mathcal{A}}$ to upper-bound parameters also ensures the existence of an accepting run. That bound used in this example would indeed prove the non-existence of a cycle for any parameter value, but it does not in turn allow us to derive a finite tree containing all the discrete behaviors, for any possible parameter value (a larger bound would still give more runs).

Similarly, now consider the L/U-PTA in Figure 1d. If 0 is excluded from the domain of p, we have a behavior similar to the previous example: as p gets closer and closer to 0, we have more and more discrete behaviors. And even if we could derive a lower bound à la [10] ensuring the non-existence of a cycle here, it would not give a finite tree of all the possible discrete behaviors, for any parameter value.

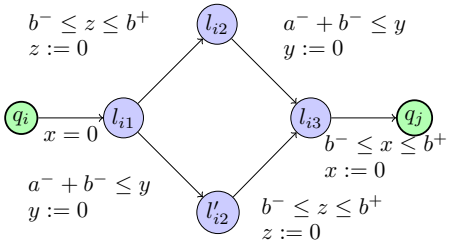


Figure 5: EG-emptiness for bounded L/U-PTAs: increment gadget

We can actually exhibit a very thin border between decidability and undecidability of L/U-PTAs by proving that, given a bounded L/U-PTA $\mathcal{A}_{|bounds}$ with a single open bound in $bounds$ or an unbounded L/U-PTA, the EG-emptiness problem becomes undecidable.

Theorem 6. *The EG-emptiness problem is undecidable for open bounded L/U-PTAs, with 4 clocks and 4 parameters.*

Proof. We will use a reduction from the halting problem of a 2-counter machine.

Let us consider the encoding used in the proof of Theorem 4, to which we will perform several modifications.

First, we force the 2-counter machine to execute in a constant 1-time unit duration as follows:

1) We replace any occurrence of "1" in the encoding with a parameter, either b^- or b^+ (depending on whether the occurrence of 1 occurs as a lower-bound or an upper-bound); hence the duration of an increment or decrement gadget is now at least b^- and at most b^+. We give the increment gadget in Figure 5. The encoding of a counter is as follows: when $x = 0$, then $y = b - ac_1$ and $z = b - ac_2$, where $a = a^- = a^+$ and $b = b^- = b^+$ (for other parameter valuations, the machine is not properly simulated). Typically, b will need to be sufficiently small compared to 1 to encode the required number of steps of the machine, and a will need to be sufficiently small compared to b to encode the maximum value of the counters. The decrement part of the "test and decrement" instruction is modified similarly.
2) We modify the zero-test part of the "test and decrement" instruction so that its duration is within $[b^-, b^+]$, as in Figure 6: only the first transition encodes the zero-test, the two other transitions forcing $[b^-, b^+]$ time units to elapse while keeping the values of the clocks unchanged, assuming $a^- = a^+$ and $b^- = b^+$ (we will see later that other valuations do not matter). Let $a = a^- = a^+$ and $b = b^- = b^+$. The zero-test requires here that $b = y \wedge x = 0$; in addition, z encodes c_2 as follows: $z = b - ac_2$. After reaching l_{i1} and waiting enough time to take the transition to l_{i2} (*i. e.,* a

duration in ac_2) we have: $z = b$ and $x = y = ac_2$. After reaching l_{i2} and waiting enough time to take the transition to q_j (*i. e.,* a duration in $b - ac_2$) we have: $z = b - ac_2$ and $x = y = b$. Resetting x gives $x = 0$, $y = b$ and $z = b - ac_2$, which was the value when performing the 0-test. So the value of the clocks remains unchanged when $b^- = b^+$, and $[b^-, b^+]$ time units have elapsed in any case.

3) We add to any location in the entire system an invariant $w \leq 1$, where w is a fresh clock that is never reset in the increment/decrement/zero-test gadgets. (These invariants are omitted in Figure 5.)

Hence, the duration of any gadget is at least b^- and therefore for any valuation $b^- > 0$ the number of operations the machine can perform is finite due to the global invariant $w \leq 1$.

Then, before starting the 2-counter machine encoding, we add an initial gadget given in Figure 7. This gadget constrains $a^- \leq a^+$, $b^- \leq b^+$, and is such that when leaving the gadget then $y, z \in [b^- \leq b^+]$ while x, w are 0. When $b^- = b^+$, this correctly encodes that the value of both counters is 0.

Then, we add a new $q'_{\texttt{halt}}$ location (without any invariant, *i. e.,* not requiring $w \leq 1$), with two transitions from $q_{\texttt{halt}}$ as depicted in Figure 8. We then add a transition (with no guard) from any location of the encoding (except $q_{\texttt{halt}}$) to $q'_{\texttt{halt}}$. That is, for any increment gadget, if the value of the parameters is not small enough to correctly simulate the machine, then the system is not deadlocked, and can lead instead to $q'_{\texttt{halt}}$. (If the value is small enough, the system can either lead to $q'_{\texttt{halt}}$ or continue in the 2-counter machine encoding.) We also add a transition to $q'_{\texttt{halt}}$ (with no guard) from all locations in the initial gadget in Figure 7.

We assume the following bounds for the parameters: $a^-, a^+, b^+ \in [0, 1]$ and $b^- \in (0, 1]$.

Let us show that the 2-counter machine halts iff the set of valuations satisfying $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ is not empty.

1) If $a^- > a^+$ or $b^- > b^+$, the initial gadget cannot be passed, and thanks to the transitions to $q'_{\texttt{halt}}$, all runs eventually reach $q'_{\texttt{halt}}$, hence $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ does not hold.
2) If $a^- < a^+$ and $b^- \leq b^+$, then the machine may not be correctly simulated: a given run will either reach $q_{\texttt{halt}}$, in which case it will also reach $q'_{\texttt{halt}}$ (as the guard from $q_{\texttt{halt}}$ to $q'_{\texttt{halt}}$ does not forbid this run), or it will loop in the machine until it eventually gets blocked (since $b^- > 0$ and because of the invariant $w \leq 1$, for any value of b^-, the maximal number of steps is $\frac{1}{b^-}$); when being blocked, it has no other option than going to $q'_{\texttt{halt}}$, thanks to the unguarded transitions from any location to $q'_{\texttt{halt}}$. Hence if $a^- < a^+$, $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ does not hold.
3) If $b^- < b^+$ (and $a^- \leq a^+$), again the machine may not be correctly simulated, and following a similar reasoning, $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ again does not hold.
4) If $a^- = a^+$ and $b^- = b^+ > 0$:
   a) Either the machine does not halt: in this case, after a maximum number of steps (typically $\frac{1}{b^-}$), a gadget will be blocked due to the invariant $w \leq 1$, and the run will end in $q'_{\texttt{halt}}$. Hence if the 2-counter machine does not halt, $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ does not hold.
   b) Or the machine halts: in this case, if c is the maximum value of both C_1 and C_2 over the (necessarily finite) halting execution of the machine, and if m is the length of this execution, and if $c > 0$, then for valuations such that $a^- = a^+ \leq \frac{b^-}{c}$ and $b^- = b^+ \leq \frac{1}{m}$, then there exists one run that correctly simulates the machine (beside plenty of runs that will go to $q'_{\texttt{halt}}$ due to the unguarded transitions); this run that correctly simulates the machine eventually reaches $q_{\texttt{halt}}$. From $q_{\texttt{halt}}$, for such valuations, the system is deadlocked: indeed, the transitions from $q_{\texttt{halt}}$ to $q'_{\texttt{halt}}$ can only be taken if $a^- < a^+$ or $b^- < b^+$. Hence $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ holds. The set of such valuations is certainly non-empty: $a^- = a^+ = \frac{1}{m \times c}$ and $b^- = b^+ = \frac{1}{m}$ belongs to it (if $c = 0$ then we choose, *e. g.,* $b^- = b^+ = 1$ and $a^- = a^+ = \frac{1}{2}$). Hence, if the 2-counter machine halts, there exist parameter valuations for which $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ holds.

Hence the 2-counter machine halts iff the set of valuations for which $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ holds is not empty. □

Remark 2. *The above construction works over 1 time unit (an invariant can be added to $q'_{\texttt{halt}}$ too), so this gives an undecidability result over bounded time as well.*

We now prove that EG-emptiness is also undecidable for unbounded L/U-PTAs. When not considering L/U-PTAs, proving an undecidability result for bounded PTAs usually gives the undecidability for unbounded PTAs, as a bounded PTA can be simulated using a PTA (by, *e. g.,* adding the bounds as a guard between a fresh location prior to the initial location and the initial location, *e. g.,* $p \in [\inf, \sup]$ becomes $\inf \leq x \leq \sup \wedge p = x$). This may not be true for L/U-PTAs, as such a construction requires to compare the clock and the parameter using an equality. In addition, our proof for unbounded L/U-PTAs uses one parameter less than for open bounded L/U-PTAs.

Theorem 7. *The EG-emptiness problem is undecidable for L/U-PTAs with 4 clocks and 3 parameters.*

Proof (sketch). We again use a reduction from the halting problem of a 2-counter machine. However, we must use a different PTA encoding (the encoding used in the proof of Theorem 6 does not work for unbounded L/U-PTAs, as it strongly relies on the fact that b^- be strictly positive). Instead, we propose an encoding inspired by that of a 2-counter machine proposed in [7] to prove the undecidability of the EF-emptiness problem for PTAs with a single integer-valued parameter (that can also be rational-valued). We modify the encoding of [7] to obtain an L/U-PTA, by splitting the single parameter a into a lower-bound parameter a^- and an upper-bound parameter a^+, in the spirit of Theorems 4 and 6. Then, we add a global invariant $w \leq b^+$ (where w is a fresh clock never reset, and b^+ a fresh upper-bound parameter), to ensure that, for any valuation of $b^+ > 0$,

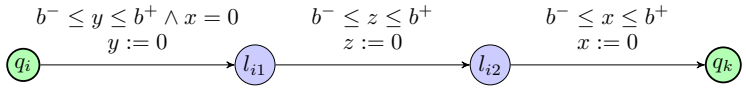


Figure 6: EG-emptiness for bounded L/U-PTAs: zero-test gadget

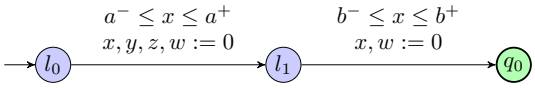


Figure 7: EG-emptiness for bounded L/U-PTAs: initial gadget

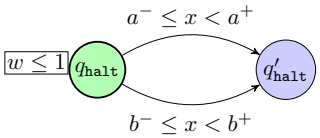


Figure 8: EG-emptiness for bounded L/U-PTAs: final gadget

the number of operations the machine can perform is finite (which requires some modifications of the gadgets to ensure that they require at least 1 time unit). The proof then follows a reasoning similar to that of Theorem 6. See Section 7.1 for a detailed proof. □

Remark 3. *The above construction works also for integer-valued parameters, so this gives an undecidability result for integer-valued parameters too. The proof also works over discrete time (with integer-valued parameters).*

6. Conclusion

Despite the vast number of undecidability results linked to the formalism of parametric timed automata, and to which we also contribute here, we have achieved some decidability for the existential parametric problem on the EG liveness property. This could be done by imposing original constraints to the classical subclass of L/U-PTAs, pertaining to the topology of the domain of the parameter values. This domain should be a closed and bounded hyperrectangle of the rational space.

The subclass together with the EG property really lies on the boundary of decidability: on the one hand, we have proved that considering unbounded, or bounded but open domains leads again to undecidability for EG. On the other hand, if we consider — instead of the EG property which asks for the existence of a maximal finite or infinite path staying in some locations — only infinite maximal paths (existence of discrete cycles), then we have proved that the problem becomes decidable (for either closed bounded domains, or unbounded domains — the case of open bounded domains remains open). And finally, if we consider only finite maximal paths (existence of deadlocks), then we have proved that the problem becomes consistently undecidable.

Future work includes *i*) studying the decidability of EC-emptiness for open bounded L/U-PTAs (possibly adapting techniques developed in [15]), *ii*) extending the EG decidability result to shapes other than hyperrectangles, and *iii*) studying actual synthesis. In addition, the decidability of problems we proved undecidable for L/U-PTAs should be studied for two subclasses of L/U-PTAs, where all parameters are upper bounds (U-PTAs) or all lower bounds (L-PTAs).

Acknowledgments

The authors thank Olivier H. Roux for fruitful discussions on the topic of parametric timed automata, as well as reviewers of a previous version of this manuscript for useful comments.

References

[1] L. Lamport, "Proving the correctness of multiprocess programs," *IEEE TSE*, vol. 3, no. 2, pp. 125–143, 1977.

[2] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Transactions on Programming Languages and Systems*, vol. 8, no. 2, pp. 244–263, 1986.

[3] R. Alur, T. A. Henzinger, and M. Y. Vardi, "Parametric real-time reasoning," in *STOC*. ACM, 1993, pp. 592–601.

[4] J. S. Miller, "Decidability and complexity results for timed automata and semi-linear hybrid automata," in *HSCC*, ser. LNCS, vol. 1790. Springer, 2000, pp. 296–309.

[5] L. Doyen, "Robust parametric reachability for timed automata," *Information Processing Letters*, vol. 102, no. 5, pp. 208–213, 2007.

[6] D. Bundala and J. Ouaknine, "Advances in parametric real-time reasoning," in *MFCS*, ser. LNCS, vol. 8634. Springer, 2014, pp. 123–134.

[7] N. Beneš, P. Bezděk, K. G. Larsen, and J. Srba, "Language emptiness of continuous-time parametric timed automata," in *ICALP, Part II*, ser. LNCS, vol. 9135. Springer, 2015, pp. 69–81.

[8] É. André, "What's decidable about parametric timed automata?" in *FTSCS*, ser. Communications in Computer and Information Science, vol. 596. Springer, 2015, pp. 1–17.

[9] T. Hune, J. Romijn, M. Stoelinga, and F. W. Vaandrager, "Linear parametric model checking of timed automata," *JLAP*, vol. 52-53, pp. 183–220, 2002.

[10] L. Bozzelli and S. La Torre, "Decision problems for lower/upper bound parametric timed automata," *Formal Methods in System Design*, vol. 35, no. 2, pp. 121–151, 2009.

[11] A. Jovanović, D. Lime, and O. H. Roux, "Integer parameter synthesis for timed automata," *IEEE TSE*, vol. 41, no. 5, pp. 445–461, 2015.

[12] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[13] M. L. Minsky, *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.

[14] É. André, D. Lime, and O. H. Roux, "Decision problems for parametric timed automata," in *ICFEM*, ser. LNCS, vol. 10009. Springer, 2016, pp. 400–416.

[15] O. Sankur, "Untimed language preservation in timed systems," in *MFCS*, ser. LNCS, vol. 6907. Springer, 2011, pp. 556–567.

[16] S. Tripakis and S. Yovine, "Analysis of timed systems using time-abstracting bisimulations," *Formal Methods in System Design*, vol. 18, no. 1, pp. 25–68, 2001.

7. Appendix

7.1. Proof of Theorem 7

Theorem 7 (recalled). *The EG-emptiness problem is undecidable for L/U-PTAs with 4 clocks and 3 parameters.*

Proof. We will again use a reduction from the halting problem of a 2-counter machine. Our proof essentially relies on a mechanism similar to the proof of Theorem 6; however, we must use a different PTA encoding (the encoding used in the proof of Theorem 6 does not work for unbounded L/U-PTAs, as it strongly relies on the fact that b^- be strictly positive), which prevents us to factor the proof as much as we would have wished.

We propose here an encoding inspired by that of a 2-counter machine proposed in [7] to prove the undecidability of the EF-emptiness problem for PTAs with a single integer-valued parameter used to encode the maximum value of the two counters (although not considered in [7], the proof also works identically with a rational-valued parameter). Two different instructions are considered:

- when in state $\mathtt{q}_i$, increment C_k and go to $\mathtt{q}_j$;
- when in state $\mathtt{q}_i$, if $C_k = 0$ then go to $\mathtt{q}_k$, otherwise decrement C_k and go to $\mathtt{q}_j$;

Starting from the initial configuration $(\mathtt{q}_0, C_1 = 0, C_2 = 0)$ the machine either reaches $q_{\mathtt{halt}}$ and halts, or loops forever. Knowing whether the machine halts is undecidable [13].

The encoding uses a single parameter a. Two clocks x and y are used to encode the value of the counters, while a third clock z is used as an auxiliary clock. Whenever $z = 0$, then $x = c_1$ and $y = c_2$.

We modify this encoding by splitting the single parameter a into a lower-bound parameter a^- and an upper-bound parameter a^+, in the spirit of previous undecidability results for L/U-PTAs in this paper (Theorems 4 and 6).

In addition, we request that the entire execution takes a time less than b^+, where b^+ is a fresh upper-bound parameter; this is achieved by adding an invariant $w \leq b^+$ to all locations (with w a fresh clock never reset after the initial gadget).

We give the modified increment gadget for the first counter in Figure 9 (invariants are omitted). Note that, if $z = 0$ when entering q_i then the time to pass this gadget is in $[a^- + 1, a^+ + 1]$.

The test and decrement gadget is similar, and given in Figure 10. We performed a slight modification to the zero-test of [7], that was executed in 0-time; we require in our construction that each gadget takes at least one time unit. Hence, we rewrote it in Figure 10 so as to force at least one time unit to elapse after the clocks are tested, and so that the final value of the clock is not changed, when $a^- = a^+$ (in the spirit of the same operation in the proof of Theorem 6): when performing the zero-test, we have $x = z = 0$ and $y = c_2$. Then after $a - c_2 + 1$ time units (with $a = a^+ = a^-$), we have $x = z = a + 1 - c_2$ and $y = a + 1$, and we can take the transition to $l_{i2''}$, resetting y. Then after c_2 time

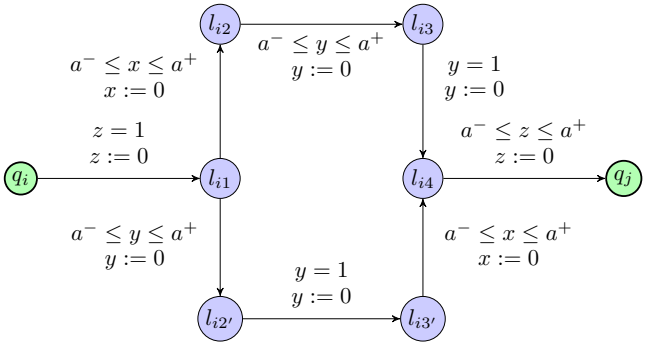


Figure 9: EG-emptiness for L/U-PTAs: increment gadget

units, we have $x = z = a + 1$ and $y = c_2$ and we can take the transition to $l_{i2''}$, resetting x and z. This gives finally $x = z = 0$ and $y = c_2$ and the time spent in the gadget is in $[a^- + 1, a^+ + 1]$, and therefore is more than one time unit. Gadgets for the second counter are symmetric.

We add before the first instruction the initial gadget given in Figure 11, constraining $a^- \le a^+$ and $b^+ > 0$, and resetting all clocks.

In addition, just as in Theorem 6, we add unguarded transitions from any location (including that of the initial gadget, but excluding $q_{\texttt{halt}}$) to a new location $q'_{\texttt{halt}}$. We also add two transitions from $q_{\texttt{halt}}$ to $q'_{\texttt{halt}}$ given in the final gadget in Figure 12.

Let us show that the 2-counter machine halts iff the set of valuations for which $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ holds is not empty. We reason on the parameter valuations.

1) If $a^- > a^+$ or $b^+ = 0$, the initial gadget cannot be passed: any run is sent to $q'_{\texttt{halt}}$ because of the transitions to $q'_{\texttt{halt}}$, and therefore $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ does not hold.
2) If $a^- < a^+$ and $b^+ > 0$, then the machine may not be correctly simulated: a given run will either reach $q_{\texttt{halt}}$, in which case it will also reach $q'_{\texttt{halt}}$ (as the guard from $q_{\texttt{halt}}$ to $q'_{\texttt{halt}}$ in Figure 12 does not forbid this run), or it will loop in the machine until it eventually gets blocked: since $b^+ > 0$, since all gadgets require at least 1 time unit, for any value of b^+ the invariant $z \le b^+$ will eventually block a transition after at most b^+ steps. When being blocked, a run has no other option than going to $q'_{\texttt{halt}}$, because of the unguarded transitions from any location to $q'_{\texttt{halt}}$. Hence if $a^- < a^+$ and $b^+ > 0$, $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ does not hold.
3) Now, assume $a^- = a^+$ and $b^+ > 0$.
   a) Either the machine does not halt: in this case, after a maximum number of steps (typically at most b^+), a gadget will be blocked due to the invariant $z \le b^+$, and the run will end in $q'_{\texttt{halt}}$ because of the unguarded transitions from any location to $q'_{\texttt{halt}}$. Hence if the 2-counter machine does not halt, $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ does not hold.
   b) Or the machine halts: in this case, if c is the maximum value of both C_1 and C_2 over the (necessarily finite) halting execution of the machine, and if m is the length of this execution, and if $c > 0$, then for valuations such that $a^- = a^+ \le c$ and sufficiently large valuations of b^+ (typically $b^+ \ge m \times (a^+ + 1)$ as a gadget can take up to $a^+ + 1$ time units), then there exists one run that correctly simulates the machine; this run eventually reaches $q_{\texttt{halt}}$. From $q_{\texttt{halt}}$, for such values, the system is deadlocked. Hence, if the 2-counter machine halts, there exist parameter valuations for which a run does not reach $q'_{\texttt{halt}}$, *i. e.,* for which $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ holds.

Hence the 2-counter machine halts iff the set of valuations for which $\mathrm{EG}(L \setminus \{q'_{\texttt{halt}}\})$ holds is not empty. □

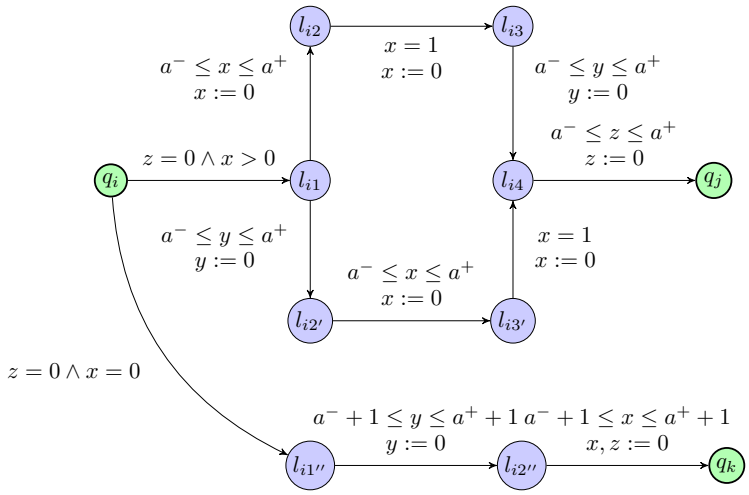


Figure 10: EG-emptiness for L/U-PTAs: test and decrement gadget

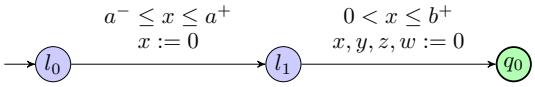


Figure 11: EG-emptiness for L/U-PTAs: initial gadget

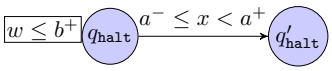


Figure 12: EG-emptiness for L/U-PTAs: final gadget

On the Expressiveness of Parametric Timed Automata*

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

[1] Université Paris 13, Sorbonne Paris Cité, LIPN
CNRS, UMR 7030, F-93430, Villetaneuse, France
[2] École Centrale de Nantes, IRCCyN, CNRS, UMR 6597, France

Abstract. Parametric timed automata (PTAs) are a powerful formalism to reason about, model and verify real-time systems in which some constraints are unknown, or subject to uncertainty. In the literature, PTAs come in several variants: in particular the domain of parameters can be integers or rationals, and can be bounded or not. Also clocks can either be compared only to a single parameter, or to more complex linear expressions. Yet we do not know how these variants compare in terms of expressiveness, and even the notion of expressiveness for parametric timed models does not exist in the literature. Furthermore, since most interesting problems are undecidable for PTAs, subclasses, such as L/U-PTAs, have been proposed for which some of those problems are decidable. It is not clear however what can actually be modeled with those restricted formalisms and their expressiveness is thus a crucial issue. We therefore propose two definitions for the expressiveness of parametric timed models: the first in terms of all the untimed words that can be generated for all possible valuations of the parameters, the second with the additional information of which parameter valuations allow which word, thus more suitable for synthesis issues. We then use these two definitions to propose a first comparison of the aforementioned PTA variants.

Keywords: parametric timed automata, L/U-PTAs, hidden parameters

1 Introduction

Designing real-time systems is a challenging issue and formal models and reasoning are key elements in attaining this objective. In this context, timed automata (TAs) [AD94] are a powerful and popular modeling formalism. They extend finite automata with timing constraints, in which clocks are compared to integer constants that model timing features of the system. In the early design phases these features may not be known with precision and therefore parametric timed

* This work is partially supported by the ANR national research program "PACS" (ANR-14-CE28-0002). This is the author version of the manuscript of the same name published in the proceedings of FORMATS 2016. This author version also includes all proofs. The final publication is available at link.springer.com.

automata (PTAs) [AHV93] allow these constants to be replaced by unknown parameters, the correct values of which will be synthesized as part of the verification process. Unfortunately, most interesting problems are undecidable for PTAs, including the basic question of the existence of values for the parameters such that a given location is reachable [AHV93] (sometimes called EF-emptiness problem).

Since the seminal definition, many variants of PTAs have been defined in the literature, both as an effort to further increase the convenience of modeling by allowing complex linear expressions on parameters in the timing constraints (such as in [HRSV02,JLR15]), or in order to better assess the frontier of decidability for PTAs. In the latter objective, parameters have been considered to be integers [AHV93,Mil00,BL09,BO14,BBLS15,JLR15,AM15] or rationals [AHV93,Mil00,HRSV02,Doy07,JLR15,AM15], possibly bounded a priori [JLR15], or even restricted to be used as either always upper bounds or always lower bounds, giving so-called L/U-PTAs [HRSV02,BL09].

This difference in the class of constraints may have a direct impact on the decidability or complexity: for example, [BO14] recently improved the complexity of [AHV93] (NEXPTIME-complete instead of non-elementary) of the EF-emptiness problem over discrete time for one parametric clock and arbitrarily many non-parametric clocks and (integer-valued) parameters, but requires non-strict inequalities and uses invariants, features not used in the constructions of [AHV93]; it is hence unclear whether the result of [AHV93] is really subsumed by [BO14].

In order to be able to compare these definitions, one must first agree on a notion of expressiveness for timed parametric models, since none exists in the literature. This is the main objective of this work.

Contribution We propose the following two definitions of expressiveness: 1) as the union over all parameter valuations of the accepting untimed words ("untimed language"); 2) as the pairs of untimed words with the parameter valuations that allow them ("constrained untimed language").

We first prove that considering rational parameter valuations or unbounded integer parameter valuations in PTAs and L/U-PTAs is actually equivalent with respect to the untimed language.

We also prove that, whereas the untimed language recognized by a PTA with a single clock and arbitrarily many parameters is regular, adding a single non-parametric clock (i. e., a clock compared at least once to a parameter), even with a single parameter, gives a language that is at least context-sensitive, hence beyond the class of regular languages.

We then compare the expressiveness, w.r.t. untimed language and constrained untimed language, of several known subclasses of PTAs with integer parameters, in particular L/U-PTAs, and PTAs with bounded parameters. It turns out that, when considering the expressiveness as the untimed language, most subclasses of PTAs with integer parameters (including PTAs with bounded parameters, and L/U-PTAs) are in fact not more expressive than TAs. However, classical PTAs remain strictly more expressive than TAs. We also show that adding fully para-

metric constraints (i. e., comparison of parametric linear terms with 0, without any clock) does not increase the expressiveness of PTAs seen as the untimed language.

We also propose and focus on a new class of PTAs in which some parameters are hidden, i. e., do not occur in the constrained untimed language. While adding hidden parameters does not increase the expressiveness w.r.t. the untimed language (since in that case all parameters can be considered as hidden), when considering the expressiveness as the constrained untimed language, we show that hidden parameters strictly extend the expressiveness of PTAs. And interestingly, for this second definition of expressiveness, L/U-PTAs with bounded parameters turn out to be incomparable with classical L/U-PTAs.

Outline We introduce the basic notions in Section 2. We propose our two definitions of expressiveness in Section 3. We then show that rational-valued parameters are not more expressive than integer-valued parameters for the untimed language (Section 4). Focusing on integer-valued parameters, we then classify PTAs, their subclasses, and their extensions with hidden parameters w.r.t. the untimed language (Section 5) and the constrained untimed language (Section 6). We conclude and outline perspectives in Section 7.

2 Preliminaries

2.1 Clocks, Parameters and Constraints

Let $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}_+$ denote the sets of non-negative integers, integers, and non-negative real numbers respectively. Let $\mathcal{I}(\mathbb{N})$ denote the set of closed intervals on $\mathbb{N}$, i. e., the set of intervals $[a, b]$ where $a, b \in \mathbb{N}$ and $a \leq b$.

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 $\mu : X \to \mathbb{R}_+$. We write $\mathbf{0}$ for the clock valuation that assigns 0 to all clocks. Given $d \in \mathbb{R}_+$, $\mu + d$ denotes the valuation such that $(\mu + d)(x) = \mu(x) + d$, for all $x \in X$. Given $R \subseteq X$, we define the *reset* of a valuation μ, denoted by $[\mu]_R$, as follows: $[\mu]_R(x) = 0$ if $x \in R$, and $[\mu]_R(x) = \mu(x)$ otherwise.

We assume a set $P = \{p_1, \ldots, p_M\}$ of *parameters*, i. e., unknown integer-valued constants (except in Section 4 where parameters can also be rational-valued). A parameter *valuation* v is a function $v : P \to \mathbb{N}$.

In the following, we assume $\prec \in \{<, \leq\}$ and $\sim \in \{<, \leq, \geq, >\}$. Throughout this paper, *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 $\alpha_i, \beta_j, d \in \mathbb{Z}$. Similarly, *plt* denotes a parametric linear term over P, that is a linear term without clocks ($\alpha_i = 0$ for all i). A *constraint* C (i. e., a convex polyhedron) over $X \cup P$ is a conjunction of inequalities of the form $lt \sim 0$. 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 μ, $\mu(v(C))$ denotes the Boolean value obtained by replacing each clock x in $v(C)$ with $\mu(x)$.

A *guard* g is a constraint over $X \cup P$ defined by inequalities of the form $x \sim plt$.

2.2 Parametric Timed Automata with Hidden Parameters

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

We actually first define an extension of PTAs (namely hPTAs) that will allow us to compare models with a different number of parameters, by considering that some of them are hidden. We will define PTAs as a restriction of hPTAs.

Definition 1 (PTA with hidden parameters). *A parametric timed automaton with hidden parameters (hereafter hPTA)* A *is a tuple* $(\Sigma, L, l_0, F, X, P, I, E)$*, 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)* $F \subseteq L$ *is a set of accepting locations, v)* X *is a finite set of clocks, vi)* $P = P_{\overline{v}} \uplus P_v$ *is a finite set of parameters partitioned into hidden parameters* $P_{\overline{v}}$ *and visible parameters* P_v*, vii)* I *is the invariant, assigning to every* $l \in L$ *a guard* $I(l)$*, viii)* E *is a finite set of edges* $e = (l, g, a, R, l')$ *where* $l, l' \in L$ *are the source and target locations,* $a \in \Sigma \cup \{\epsilon\}$ *(*ϵ *being the silent action),* $R \subseteq X$ *is a set of clocks to be reset, and* g *is a guard.*

We define a *PTA* as an hPTA in which $P = P_v$.

Observe that we allow ϵ-transitions (or silent transitions), i. e., transitions not labeled with any action.

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

Concrete Semantics

Definition 2 (Concrete semantics of a TA). *Given an hPTA* $\mathsf{A} = (\Sigma, L, l_0, F, X, P, I, E)$*, and a parameter valuation* v*, the concrete semantics of* $v(\mathsf{A})$ *is given by the timed transition system* $(S, s_0, \rightarrow)$*, with* $S = \{(l, \mu) \in L \times \mathbb{R}_+^H \mid \mu(v(I(l)))$ *is true*$\}$*,* $s_0 = (l_0, \mathbf{0})$*, and* $\rightarrow$ *consists of the discrete and (continuous) delay transition relations:*

- *discrete transitions:* $(l, \mu) \xrightarrow{e} (l', \mu')$*, if* $(l, \mu), (l', \mu') \in S$*, there exists* $e = (l, g, a, R, l') \in E$*,* $\mu' = [\mu]_R$*, and* $\mu(v(g))$ *is true.*
- *delay transitions:* $(l, \mu) \xrightarrow{d} (l, \mu{+}d)$*, with* $d \in \mathbb{R}_+$*, if* $\forall d' \in [0, d], (l, \mu{+}d') \in S$*.*

A *(concrete) run* is a sequence $\rho = s_1 \alpha_1 s_2 \alpha_2 \cdots s_n \alpha_n \cdots$ such that $\forall i, (s_i, \alpha_i, s_{i+1}) \in \rightarrow$. We consider as usual that concrete runs strictly alternate delays d_i and discrete transitions e_i and we thus write concrete runs in the form $\rho = s_1 \xrightarrow{(d_1, e_1)} s_2 \xrightarrow{(d_2, e_2)} \cdots$. We refer to a state of a run starting from the initial state of a TA A as a *concrete state* (or just as a *state*) of A. Note that when a run is finite, it must end with a state. The *duration* of a concrete run is the sum of all the delays d_i appearing in this run.

An *untimed* run of $v(\mathsf{A})$ is a sequence $l_1 e_1 l_2 e_2 \cdots l_n \cdots$ such that for all i there exist a clock valuation μ_i and $d_i \geq 0$ such that $(l_1, \mu_1) \overset{(d_1,e_1)}{\rightarrow} (l_2, \mu_2) \overset{(d_2,e_2)}{\rightarrow} \cdots (l_n, \mu_n) \overset{(d_n,e_n)}{\rightarrow} \cdots$ is a run of $v(\mathsf{A})$. Given a run ρ, we denote by $\mathsf{Untime}(\rho)$ its corresponding untimed run.

The *trace* of an untimed run $l_1 e_1 l_2 e_2 \cdots l_n \cdots$ is the sequence $e_1 e_2 \cdots e_n \cdots$.

The *(untimed) trace* of a concrete run ρ is the trace of $\mathsf{Untime}(\rho)$.

A run ρ is *accepted* by $v(\mathsf{A})$ if it is finite and the location of its last state belongs to F. An untimed run is accepted by $v(\mathsf{A})$ if it is finite and its last location belongs to F.

The (untimed) language of $v(\mathsf{A})$ is the set of the traces of runs accepted by $v(\mathsf{A})$.

2.3 Subclasses of Parametric Timed Automata

L/U-PTAs have been introduced as a subclass of PTAs for which the EF-emptiness problem (i. e., the existence of values for the parameters such that a given location is reachable) is decidable [HRSV02]:

Definition 3 (hL/U-PTA). *An hL/U-PTA is an hPTA where the set of parameters is partitioned into a set of lower-bound parameters P^- and a set of upper-bound parameters P^+. A parameter p belongs to P^+ (resp. P^-), if it appears in constraints $x \leq plt$ or $x < plt$ always with a non-negative (resp. non-positive) coefficient, and in constraints $x \geq plt$ or $x > plt$ always with a non-positive (resp. non-negative) coefficient.*

Just as for PTAs, we define an *L/U-PTA* as an hL/U-PTA in which $P = P_v$.

Decidability comes from the fact that in L/U-PTAs increasing the value of an upper bound parameter or decreasing that of a lower bound parameter always only increase the possible behavior:

Lemma 1 (monotonicity of hL/U-PTAs [HRSV02]). *Let A be an hL/U-PTA and v be a parameter valuation. Let v' be a valuation such that for each upper-bound parameter p^+, $v'(p^+) \geq v(p^+)$ and for each lower-bound parameter p^-, $v'(p^-) \leq v(p^-)$. Then any run of $v(\mathsf{A})$ is a run of $v'(\mathsf{A})$.*

We will often use the notation p^+ (resp. p^-) for upper (resp. lower) bound parameters in hL/U-PTAs.

Given an hL/U-PTA, we denote by $v_{0/\infty}$ the special parameter valuation (mentioned in, e. g., [HRSV02]) assigning 0 to all lower-bound parameters and ∞ to all upper-bound parameters.[3]

Let us now define a bounded PTA as a PTA where the domain of each parameter is bounded, i. e., ranges between two integer-valued constants.

[3] Technically, $v_{0/\infty}$ is not a parameter valuation, as the definition of valuation does not allow ∞. However, we will use it only to valuate an L/U-PTA (or an hL/U-PTA) with it; observe that valuating an L/U-PTA with $v_{0/\infty}$ still gives a valid TA.

Definition 4 (bounded hPTA). *A* bounded hPTA *is* $\mathsf{A}_{|bounds}$*, where* A *is an hPTA, and bounds* $: P \to \mathcal{I}(\mathbb{N})$ *assigns to each parameter* p *an interval* $[\min, \max]$*, with* $\min, \max \in \mathbb{N}$.

We define similarly bounded hL/U-PTAs.

3 Defining the Expressiveness of PTAs

In the following, we denote by $\mathcal{V}(P)$, $\mathcal{V}(P_v)$, and $\mathcal{V}(P_{\overline{v}})$ the sets of valuations of respectively all the parameters, the visible parameters, and the hidden parameters of an hPTA.

Definition 5 (untimed language of an hPTA). *Given an hPTA* A*, the* untimed language *of* A*, denoted by* $\mathsf{UL}(\mathsf{A})$ *is the union over all parameter valuations* v *of the sets of untimed words accepted by* $v(\mathsf{A})$*, i. e.,*

$$\bigcup_{v \in \mathcal{V}(P)} \Big\{ w \mid w \text{ is an untimed word accepted by } v(\mathsf{A}) \Big\}$$

TA is a subclass of PTA, hence, given a TA A, we also denote $\mathsf{UL}(\mathsf{A})$ its untimed language.

We propose below another definition of language for hPTAs, in which we consider not only the accepting untimed words, but also the parameter valuations associated with these words; this definition is more suited to compare the possibilities offered by parameter synthesis. Note that we only expose the *visible* parameter valuations.

Definition 6 (constrained untimed language of an hPTA). *Given an hPTA* A*, the* constrained untimed language *of* A*, denoted by* $\mathsf{CUL}(\mathsf{A})$ *is*

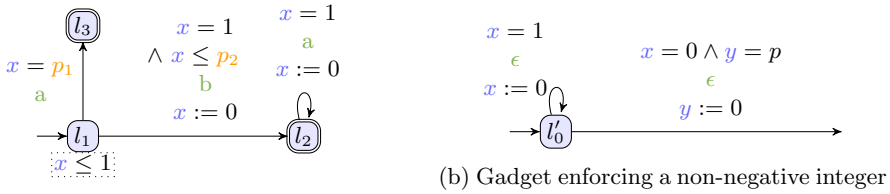
$$\bigcup_{v \in \mathcal{V}(P_v)} \Big\{ (w, v) \mid \exists v' \in \mathcal{V}(P_{\overline{v}}) \text{ s.t. } w \text{ is an untimed word accepted by } v(v'(\mathsf{A})) \Big\}$$

Note that since P_v and $P_{\overline{v}}$ are disjoint, we can write indifferently $v(v'(\mathsf{A}))$ and $v'(v(\mathsf{A}))$.

We use the word "constrained" because another way to represent the constrained language of an hPTA is in the form of a set of elements (w, K), where w is an untimed word, and K is a parametric constraint such that for all v in K, then w is an untimed word accepted by $v(v'(\mathsf{A}))$ for some $v' \in \mathcal{V}(P_{\overline{v}})$.

Example 1. Let us consider the hPTA A of Figure 1a, where $P_v = \{p_1\}$ and $P_{\overline{v}} = \{p_2\}$.

- Its untimed language is $\mathsf{UL}(\mathsf{A}) = \{a\} \cup \{ba^n \mid n \in \mathbb{N}\}$ that we note with the rational expression $\mathsf{UL}(\mathsf{A}) = a + ba^*$.



(a) A PTA

(b) Gadget enforcing a non-negative integer value for p

Fig. 1: An example of PTA, and a PTA gadget

– Its constrained untimed language is $\mathsf{CUL}(\mathsf{A}) = \left\{(a, p_1 = i) \mid 0 \leq i \leq 1\right\} \bigcup \left\{(ba^n, p_1 = i) \mid i \in \mathbb{N}, n \in \mathbb{N}\right\}$ that we can also note $\mathsf{CUL}(\mathsf{A}) = \left\{(a, p_1 \leq 1), (ba^*, p_1 \geq 0)\right\}$, with $p_1 \in \mathbb{N}$. Note that both the parameter p_2 and the fact that p_2 must be at least 1 to go to l_2 are hidden.

Definition 7 (regular constrained language). *The constrained untimed language of an hPTA* A *is* regular *if for all visible parameter valuations* $v \in \mathcal{V}(P_v)$*, the language* $\{w \mid (w, v) \in \mathsf{CUL}(\mathsf{A})\}$ *is regular.*

Remark 1. Since valuating a PTA with any rational parameter valuation gives a TA, the constrained untimed language of any PTA is regular in the sense of Definition 7.

Note that the idea of combining the untimed language with the parameter valuations leading to it is close to the idea of the behavioral cartography of parametric timed automata [AF10], that consists in computing parameter constraints together with a "trace set", i. e., the untimed language (that also includes in [AF10] the locations).

In the following, a *class* refers to an element in the set of TAs, bounded L/U-PTAs, L/U-PTAs, bounded PTAs and PTAs, and their counterparts with hidden parameters. An *instance* of a class is a model of that class.

A first class is strictly more expressive than a second one w.r.t. the untimed language if *i*) for any instance of the second one, their exists an instance of the first one that has the same untimed language, and *ii*) there exists an instance of the first one for which no instance of the second one has the same untimed language. Two classes are equally expressive w.r.t. the untimed language if for any instance of either class, their exists an instance of the other class that has the same untimed language. The comparison of the expressiveness w.r.t. the constrained untimed language can be defined similarly, with the additional requirement that the two instances must contain the same visible parameters (possibly after some renaming).

4 An Equivalence Between Integer and Rational Parameters

In the literature, some works focus on integer parameters [BO14,BBLS15,BL09], some others on rational parameters [HRSV02,Doy07], and also some propose constructions working in both settings [AHV93,Mil00,JLR15,AM15].

In this section, we prove that considering rational parameter valuations or unbounded integer parameter valuations in PTAs and L/U-PTAs is actually equivalent with respect to untimed languages.[4]

First, remark that any PTA with rational parameter valuations can be constrained to accept only non-negative integer parameter valuations. We just need to insert a copy of the gadget in Figure 1b for each parameter p before the initial location. We connect them to each other in sequence, in any order, and x and y can be clocks from the original PTA. In that gadget x is zero only when y is a non-negative integer and therefore p must be a non-negative integer to permit the exit from l'_0. Clearly, when considering only non-negative integer parameter valuations, both PTAs have the same untimed language.

With the above construction, we can filter out non-integer valuations. We can actually go a bit further and establish the following result:

Lemma 2. *For each PTA* A*, there exists a PTA* A' *such that:*

1. *for all rational parameter valuations* v *of* A *there exists an integer parameter valuation* v' *of* A' *such that* $v(\mathsf{A})$ *and* $v'(\mathsf{A}')$ *have the same untimed language.*
2. *for all integer parameter valuations* v' *of* A' *there exists a rational parameter valuation* v *of* A *such that* $v(\mathsf{A})$ *and* $v'(\mathsf{A}')$ *have the same untimed language.*

Proof. The idea of the proof is to scale all the expressions to which clocks are constrained so that they are integers. However, since we do not know in advance by how much we have to scale, we use an additional parameter to account for this scaling factor.

Let A be a PTA. Let p be a fresh parameter and let A'' be the PTA obtained from A by replacing every inhomogeneous (i. e., constant) term c in the linear expressions of guards and invariants by $c * p$. For instance, the constraint $x \leq 3p_1 + 2p_2 + 7$ becomes $x \leq 3p_1 + 2p_2 + 7p$.

We now build A' as follows: we add a new location (which will be the initial location of A'), from which two transitions, labeled ϵ and resetting all clocks, exit. The first one has guard $x \neq 0 \wedge x = p$ and goes to the initial location of A''. The second has guard $x = 0 \wedge x = p$ and goes to the initial location of an exact copy of A. By construction the first one can be taken only if $p \neq 0$ and the second one only if $p = 0$.

1. Let v be a rational parameter valuation of A. Let m be the least common multiple (LCM) of the denominators of the values assigned to parameters

[4] Comparing constrained languages would make no sense since obviously the parameter valuations cannot match in general in the rational and integer settings.

by v. Let v' be defined as: $\forall p_i \neq p, v'(p_i) = m * v(p_i)$ and $v'(p) = m$. Then, by construction, v' is an integer valuation of A', $v'(p) \neq 0$ and $v'(\mathsf{A}'')$ is a TA that is scaled by m from the TA $v(\mathsf{A})$. Then by [AD94, Lemma 4.1], $v(\mathsf{A})$ and $v'(\mathsf{A}'')$ have the same untimed runs up to renaming. And finally, $v(\mathsf{A})$ and $v'(\mathsf{A}')$ have the same untimed language.

2. The opposite direction works similarly: let v' be an integer parameter valuation of A'. If $v'(p) = 0$, then in A'' we can only go to the copy of A. We can therefore choose $v(p_i) = v'(p_i)$ and obtain the same untimed language. If $v'(p) \neq 0$, we define v by $v(p_i) = \frac{v'(p_i)}{v'(p)}$. Then v is a rational parameter valuation of A and $v(\mathsf{A})$ is a scaled down version of $v'(\mathsf{A}'')$, which therefore has the same untimed runs. And again, $v(\mathsf{A})$ and $v'(\mathsf{A}')$ have the same untimed language.

□

First remark that, in order to show the equivalence between integer- and rational-valued parameters, we provided a construction that added one additional parameter, and possibly some parametric clocks. This is consistent with the fact that PTAs with integer parameters typically have decidability results for slightly more parametric clocks and parameters than with rational parameters. For instance, the existence of a rational parameter valuation such that a given location is reachable is undecidable for PTAs with 1 parametric clock (a clock compared to parameters) and 3 normal clocks [Mil00], while the existence of an *integer* parameter valuation is decidable in that setting [BBLS15].

Second, in the construction, we need the integer parameters to be unbounded because the LCM can be arbitrarily big.

Finally, this result is not directly applicable to L/U-PTAs as we cannot ensure that the parameterized scaling factor would be the same for upper bound inhomogeneous terms as for lower bound ones. However, for L/U-PTAs, we can derive the same result from the monotonicity property:

Lemma 3. *For an L/U-PTA* A*, the set of untimed runs produced with only integer parameter valuations or with all rational parameter valuations is the same.*

Proof. Clearly the set of untimed runs produced by considering only integer parameter valuations is included in the one obtained by considering all rational parameter valuations.

In the other direction: let v be a rational parameter valuation of A and let v' be the integer parameter valuation obtained from v by rounding up the values for upper bound parameters, and rounding down for lower bound parameters. Then, by Lemma 1, $v'(\mathsf{A})$ contains all the untimed runs of $v(\mathsf{A})$. □

Here also we need integer parameters to be unbounded because the rational parameter valuations can themselves be arbitrarily big and we get accordingly big integers when rounding up.

We can now conclude the following:

Proposition 1. *PTAs (resp. L/U-PTAs) with rational parameters and PTAs (resp. L/U-PTAs) with unbounded integer parameters are equivalent with respect to the untimed language.*

When the parameters are bounded, we will see in Proposition 2 that the integer setting leads to regular languages. So, when bounded, PTAs with rational parameters are obviously strictly more expressive than their integer parameter counterpart. For L/U-PTAs, using again the monotonicity property, we trivially see that the valuation setting all upper-bound parameters to the maximal value allowed by the bounded domain, and lower-bound parameters to the minimal value gives all the untimed runs that are possible with other valuations. That "extremal" valuation is an integer valuation by definition. So, even when bounded, L/U-PTAs are still equally expressive in the rational and integer settings.

5 Expressiveness as the Untimed Language

5.1 PTAs in the Hierarchy of Chomsky

Let us show that (without surprise) Turing-recognizable languages (type-0 in Chomsky's hierarchy) can be recognized by PTAs (with enough clocks and parameters).

Lemma 4. *Turing-recognizable languages are also recognizable by PTAs.*

Proof. Consider a Turing-machine: it can be simulated by a 2-counter machine (with labelled instructions), which can in turn be simulated by a PTA. The transitions of the encoding PTA can be easily labeled accordingly (using also ϵ transitions). Assume that a word is accepted by the machine when it halts (i. e., it reaches $l_{\texttt{halt}}$). If the machine does not halt, $l_{\texttt{halt}}$ is reachable for no parameter valuation, hence the language of the machine is empty and that of the encoding PTA also. If the machine halts, $l_{\texttt{halt}}$ is reachable for parameter valuations correctly encoding the machine (i. e., depending on the proof, large enough or small enough to correctly encode the maximum value of the two counters). Hence, by taking the union over all parameter valuations of all untimed words accepted by the encoding PTA, one obtains exactly the language recognized by the machine. □

Lemma 4 only holds with enough clocks and parameters, typically 3 parametric clocks and 1 integer-valued or rational-valued parameter [BBLS15], or 1 parametric clock, 3 non-parametric clocks and 1 rational-valued parameter [Mil00].

For lower numbers, either decidability of the EF-emptiness problem is ensured (in which case the language cannot be type-0), or this problem remains open.

Let us point out a direct consequence of a result of [AM15] on PTAs with a single (necessarily parametric) clock.

Lemma 5. *The untimed language recognized by a PTA with a single clock and arbitrarily many parameters is regular.*

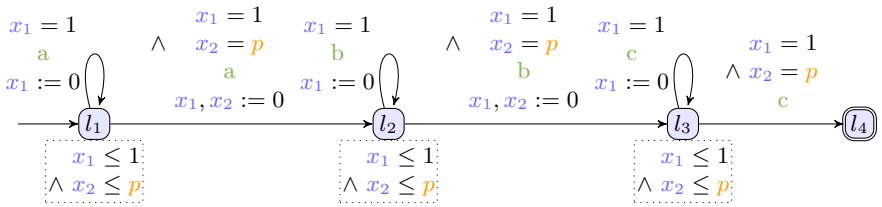


Fig. 2: A PTA with untimed language $a^n b^n c^n$

Proof. In [AM15, Theorem 20], we proved that the parametric zone graph (an extension of the zone graph for PTAs, following e. g., [JLR15]) of a PTA with a single (necessarily parametric) clock and arbitrarily many parameters is finite. This gives that the language recognized by a PTA with a single clock is regular. □

We now show that adding to the setting of Lemma 5 a single non-parametric clock, even with a single parameter, may give a language that is at least context-sensitive, hence beyond the class of regular languages.

Theorem 1. *PTAs with 1 parametric clock, 1 non-parametric clock and 1 parameter can recognize languages that are context-sensitive.*

Proof. Consider the PTA A in Figure 2. Consider an integer parameter valuation v such that $v(p) = i$, with $i \in \mathbb{N}$. The idea is that we use the parameter to first count the number of as, and then ensure that we perform an identical number of bs and cs; such counting feature is not possible in TAs (at least not for any value of i as is the case here). Clearly, due to the invariant $x_1 \leq 1$ in l_1, one must take the self-loop on l_1 every 1 time unit; then, one can take the transition to l_2 only after i such loops. The same reasoning applies to locations l_2 and l_3. Hence, the language accepted by the TA $v(\mathsf{A})$ is $a^{i+1}b^{i+1}c^{i+1}$.

Hence the union over all parameter valuations of the words accepted by A is $\{a^n b^n c^n \mid n \geq 1\}$. This language is known to be in the class of context-sensitive languages (type-1 in Chomsky's hierarchy), hence beyond the class of regular languages (type-3). □

This result is interesting for several reasons. First, it shows that adding a single clock, even non-parametric, to a PTA with a single clock immediately increases its expressiveness. Second, it falls into the interesting class of PTAs with 2 clocks, for which many problems remain open: the PTA exhibited in the proof of Theorem 1 (1 parametric clock and 1 non-parametric) falls into the class of 1 parametric clock, arbitrarily many non-parametric clocks and arbitrarily many integer-valued parameters, for which the EF-emptiness is known to be decidable [BBLS15]. When replacing the integer-valued with a rational-valued parameter (which does not fundamentally change our example), it also falls into the class of 1 parametric clock, 1 non-parametric clock and 1 rational-valued parameter, for which the EF-emptiness is known to be open [And15]. In both cases, it gives a lower bound on the class of languages recognized by such a PTA.

5.2 Comparison of Expressiveness

In this section, we compare the expressiveness of PTAs w.r.t. their untimed language UL.

First, we show in the following lemma that the untimed language of an L/U-PTA is equal to that of the same L/U-PTA valuated with $v_{0/\infty}$.

Lemma 6. *Let* A *be an L/U-PTA. Then:* $\mathsf{UL}(\mathsf{A}) = \mathsf{UL}(v_{0/\infty}(\mathsf{A}))$.

Proof. $\subseteq$ Let us first show that any accepting run of A for some parameter valuation is also an accepting run of $v_{0/\infty}(\mathsf{A})$, in the spirit of [HRSV02]. Let v be a parameter valuation. Let ρ be an accepting run of $v(\mathsf{A})$. Observe that, by definition, the guards and invariants of $v_{0/\infty}(\mathsf{A})$ are more relaxed than that of $v(\mathsf{A})$. Hence, any transition of ρ is also enabled in $v_{0/\infty}(\mathsf{A})$. Hence, ρ is also an accepting run of $v_{0/\infty}(\mathsf{A})$.

$\supseteq$ Conversely, let us show that, for any accepting run of $v_{0/\infty}(\mathsf{A})$, there exists a parameter valuation v such that this run is also an accepting run of $v(\mathsf{A})$. It suffices to show that, for a given run, there exists one parameter valuation accepting this run, as we define UL as the union over all parameter valuations. Let $\rho : s_0 \overset{(e_0,d_0)}{\to} s_1 \overset{(e_1,d_1)}{\to} \cdots \overset{(e_{m-1},d_{m-1})}{\to} s_m$ be an accepting run of $v_{0/\infty}(\mathsf{A})$. Let d be the duration of this run. Let $k = \lceil d \rceil + 1$. Let $v_{0/k}$ be the parameter valuation assigning 0 to all lower-bound parameters, and k to all upper-bound parameters. Now, observe that $v_{0/\infty}(\mathsf{A})$ and $v_{0/k}(\mathsf{A})$ are identical TAs, with the exception that some guards and invariants in $v_{0/k}(\mathsf{A})$ may include additional constraints of the form $x \leq i \times k$ or $x < i \times k$ (for some clock x and some $i > 0, i \in \mathbb{N}$). Since the duration of ρ is strictly less than k, then no clock will reach value k and therefore this run cannot be impacted by these additional constraints; hence, ρ is an accepting run of $v_{0/k}(\mathsf{A})$ too. □

Proposition 2. *TAs, L/U-PTAs and bounded PTAs are equally expressive w.r.t. the union of untimed languages.*

Proof. **L/U-PTAs = TAs** Direct from Lemma 6, and the fact that any TA is an L/U-PTA with no parameter.

bounded PTAs = TAs The untimed language of a PTA is the union of the untimed language of the TAs over all possible parameter valuations. As we consider integer-valued parameters, there is a finite number of valuations in a bounded PTA. Since the language recognized by a TA is a regular language, and the class of regular languages is closed under finite union, then bounded PTAs also recognize regular languages, and are therefore equally expressive with TAs.

□

Proposition 3. *L/U-PTAs and hL/U-PTAs are equally expressive w.r.t. the union of untimed languages.*

Proof. Consider an L/U-PTA A. Let A_h be the hL/U-PTA that is identical to A and contains no hidden parameters (i. e., $P_v = P$ and $P_{\overline{v}} = \emptyset$). Then $\mathsf{UL}(\mathsf{A}_h) = \mathsf{UL}(\mathsf{A})$.

Conversely, consider an hL/U-PTA A_h with visible parameters P_v and hidden parameters $P_{\overline{v}}$. Let A be the L/U-PTA such that $P = P_v \cup P_{\overline{v}}$. Then $\mathsf{UL}(\mathsf{A}) = \mathsf{UL}(\mathsf{A}_h)$. □

Proposition 4. *PTAs are strictly more expressive than TAs w.r.t. the union of untimed languages.*

Proof. Since the untimed words recognized by TA form a regular language [AD94], then the PTA exhibited in Theorem 1 recognizes a language not recognized by any TA. Conversely, any TA is a PTA (with no parameter) which gives that the expressiveness of PTAs is strictly larger than that of TAs.

□

In the following, we show that neither hidden parameters nor fully parametric linear constraints increase the expressive power of PTAs w.r.t. the union of untimed languages.

Proposition 5. *PTAs and hPTAs are equally expressive w.r.t. the union of untimed languages.*

Proof. Following the same reasoning as in Proposition 3. □

Impact of the syntax of the guards Recall that our guards and invariants are of the form $x \sim plt$, with plt a parametric linear term. Several alternative definitions exist in the literature. In addition to the PTAs defined in Definition 1, we consider here two other definitions, one that can be seen as the most restrictive (and used in e. g., [AHV93]), and one that is very permissive, with even constraints involving no clocks. We denote by a *simple guard* 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 $\mathbb{Z}$. We define an *AHV93-PTA* as a PTA the guards and invariants of which are all conjunctions of simple guards. We define a PTA with fully parametric constraints (*fpc-PTA*) as a PTA the guards and invariants of which are conjunctions of inequalities either of the form $x \sim plt$ ("guards"), or $plt \sim 0$ ("fully parametric guards"). Let us show that all three definitions are equivalently expressive w.r.t. the untimed language.

Proposition 6. *PTAs and AHV93-PTAs are equally expressive w.r.t. the union of untimed languages.*

Proof. See Appendix A. □

This result extends in a straightforward manner to fpc-PTAs.

Proposition 7. *PTAs and fpc-PTAs are equally expressive w.r.t. the union of untimed languages.*

Proof. See Appendix B. □

(a) Bounding a PTA

(b) PTA accepting a for any valuation

Fig. 3: A PTA gadget and a PTA

6 Expressiveness as the Constrained Untimed Language

In this section, we compare the expressiveness of PTAs w.r.t. their visible constrained untimed language.

Proposition 8. *Bounded PTAs are strictly less expressive than PTAs w.r.t. the constrained untimed language.*

Proof. Bounded PTAs can easily be simulated using a non-bounded PTA, by bounding the parameters using one clock and appropriate extra locations and transitions prior to the original initial location of the PTA. For example, if x is reset when entering l'_1, the gadget in Figure 3a ensures that $p \in [\min, \max]$. All such gadgets (one per parameter) must be added in a sequential manner, resetting x prior to each gadget, and resetting all clocks when entering the original initial location after the last gadget.

Now, it is easy to find a PTA that has a larger constrained untimed language than any bounded PTA. This is the case of any PTA for which a word is accepting for parameter valuations arbitrarily large (e. g., Figure 3b). □

We now show that, interestingly, this result does not extend to L/U-PTAs, i. e., bounded L/U-PTAs are not strictly less expressive than but incomparable with L/U-PTAs.

Proposition 9. *Bounded L/U-PTAs are incomparable with L/U-PTAs w.r.t. the constrained untimed language.*

Proof.
- Let us show that the constrained untimed language of a given bounded L/U-PTA cannot be obtained for any L/U-PTA. Consider a bounded U-PTA with a single parameter p^+ with bounds such that $p^+ \in [0, 1]$, and accepting a for any valuation of $p^+ \in [0, 1]$. From Lemma 1, if this run is accepted in an L/U-PTA A', then this run is also accepted for any valuation v' such that $v'(p^+) \geq 0$, including for instance $v'(p^+) > 1$. Hence accepting a only for valuations of $p^+ \in [0, 1]$ cannot be obtained in an L/U-PTA, and therefore no L/U-PTA yields this constrained untimed language.
- This converse is immediate: assume an L/U-PTA with a single parameter p^+, accepting a for any valuation of $p^+ \in [0, \infty)$. From the definition of bounded (L/U-)PTAs, all parameters must be bounded, and therefore there exists no

bounded L/U-PTA that can accept a run for $p^+ \in [0, \infty)$. Hence no bounded L/U-PTA yields this constrained untimed language.

□

We now show that hidden parameters do not extend the expressiveness of L/U-PTAs (proof is in Appendix C).

Proposition 10. *hL/U-PTAs are equally expressive with L/U-PTAs w.r.t. the constrained untimed language.*

Hidden parameters however strictly extend the expressiveness of PTAs.

Lemma 7. *There exists an hPTA* A *such that* $\mathsf{CUL}(\mathsf{A})$ *is not regular.*

Proof. Assume a PTA with no parameter. Its constrained untimed language is a set of pairs (w, v), where v is a degenerate parameter valuation (i. e., a valuation $v : \emptyset \to \mathbb{N}$ as this PTA contains no parameter). The projection of this set of pairs onto the words (i. e., $\{w \mid (w, v) \in \mathsf{CUL}(\mathsf{A})\}$) yields a regular language, as a PTA without parameters is a TA, the class of language recognized by which is that of regular languages. Now consider an hPTA where all parameters are hidden. This time, from Theorem 1 the projection of its constrained untimed language onto the words yields a language that goes beyond the class of regular languages. Hence there exists an hPTA for which the constrained untimed language is not regular. □

Remark 2. The idea used in the proof of Lemma 7 uses a PTA with no (visible) parameter. But such a result can be generalized to a PTA with an arbitrary number of visible parameters: assume such a PTA, and assume one of its parameter valuations v. We can extend this PTA into a PTA A' with a single hidden parameter such that, for the valuation v (of the visible parameters), the PTA will produce $a^n b^n c^n$ using the construction in Theorem 1. Hence, the constrained untimed language of A' is not regular.

Proposition 11. *hPTAs are strictly more expressive than PTAs w.r.t. the constrained untimed language.*

Proof. From Remark 1 and Lemma 7. □

Let us finally show that PTAs and fpc-PTAs (involving additionally $plt \sim 0$) are not more expressive than AHV93-PTAs with hidden parameters.

Proposition 12. *PTAs and fpc-PTAs are not more expressive than AHV93-PTAs with hidden parameters w.r.t. the constrained untimed language.*

Proof. In Propositions 6 and 7, we used a construction to show the equivalent expressiveness of the untimed language of PTAs, fpc-PTAs and AHV93-PTAs. This construction transforms a PTA or an fpc-PTA into an AHV93-PTAs. Since we use extra parameters in this construction, it suffices to hide these extra parameters, and we therefore obtain an AHV93-PTA with the same CUL as the original (fpc-)PTA. □

Remark 3. In fact, we highly suspect that PTAs and fpc-PTAs are not more expressive than AHV93-PTAs (without hidden parameters). We cannot use our construction of Propositions 6 and 7 as it adds extra parameters, and therefore cannot ensure the equality of the CUL. However, we could propose an alternative construction using no extra parameter, at the cost of (many) extra parametric clocks (typically two per different guard in the PTA) and many extra locations. The idea would be to replace each guard with a gadget to be synchronized with the original PTA: this gadget will ensure that the sum of parameters is indeed achieved, using an extra clock counting each parameter. For negative parameter coefficients, this can be achieved thanks to another clock nondeterministically reset, and that must be equal to p_2 whenever the original clock x is reset. Typically, to ensure $x \sim p_1 - p_2$, we nondeterministically reset x_1, x_2; then whenever we reset x, then we must have $x_2 = p_2$. Finally, the guard $x \sim p_1 - p_2$ becomes $x_1 \sim p_1$. Proposing a formal construction is among our future works.

7 Conclusion and Perspectives

In this paper, we proposed a first attempt at defining the expressiveness of parametric timed automata, also introducing the notion of hidden parameters to compare models with different numbers of parameters. When considering the union over all parameter valuations of the untimed language, it turns out that all subclasses of PTAs with integer parameters are not more expressive than TAs. However, PTAs are strictly more expressive than TAs (from 1 parametric clock and 1 non-parametric clock); extending PTAs with hidden parameters or fully parametric constraints does not increase their expressiveness. In addition, integer-valued or rational-valued parameters turn out to be equivalent.

When considering the set of accepting untimed words together with their associated parameter valuations, then subclasses of PTAs with integer parameters have a varying expressiveness. An interesting result is that bounded L/U-PTAs turn out to be incomparable with L/U-PTAs. In addition, hidden parameters strictly extend the expressiveness of PTAs.

Future works We compared so far general formalisms; it now remains to be studied what consequences on decidability the forms of guards and invariants together with a fixed number of clocks and parameters may have: a ultimate goal would be to unify the wealth of (un)decidability results from the literature with all different syntactic contexts.

We showed that rational-valued parameters are not more expressive than integer-valued parameters; our construction makes use of an extra parameter. It remains to be shown whether this construction is optimal or not.

Finally, forbidding ϵ-transitions may also change our comparison of formalisms, as such silent transitions have an impact on the expressiveness of TAs (see, e. g., [BPDG98]).

References

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

AF10. Étienne André and Laurent Fribourg. Behavioral cartography of timed automata. In *RP*, volume 6227 of *Lecture Notes in Computer Science*, pages 76–90. Springer, 2010.

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

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.

And15. Étienne André. What's decidable about parametric timed automata? In *FTSCS*, volume 596 of *Communications in Computer and Information Science*, pages 1–17. Springer, 2015.

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.

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.

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.

BPDG98. Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2-3):145–182, 1998.

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

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.

JLR15. Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. Integer parameter synthesis for timed automata. *Transactions on Software Engineering*, 41(5):445–461, 2015.

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.

Appendix

A Proof of Proposition 6

Proposition 6 (recalled). *PTAs and AHV93-PTAs are equally expressive w.r.t. the union of untimed languages.*

Proof. The first direction (AHV93-PTAs are not more expressive than PTAs) is trivial: any AHV93-PTA is a PTA.

Let us consider the other direction. We transform a PTA A into an equivalent AHV93-PTA, that will contain two additional parametric clocks (x_1, x_2, that can be existing clocks of the PTA) and several additional parameters (one per different parametric linear term).

Consider a constraint of the form $x \sim \sum_i \beta_i p_i$ appearing in a guard or an invariant. The idea is to create an additional parameter p, and ensure that $\sum_i \beta_i p_i = p$. We transform the equality $\sum_i \beta_i p_i = p$ into $\sum_j \beta_{i_j} p_{i_j} = \sum_k \beta_{i_k} p_{i_k} + p$, where $\beta_{i_j}, \beta_{i_k} \in \mathbb{N}$, that is we move to the right-hand side all negative coefficients so that they become positive. Then, we create the gadget as in Figure 4. Note that we make use of the shortcut $x_1 = \beta_i \times p_i$ for β_i consecutive transitions with guard $x_1 = p_i$ and resetting x_1 (recall that all our coefficients are now non-negative, since we moved the negative coefficient to the other side of the equality).

This gadget consists in the product of two PTA parts, that synchronize as follows: they start in their initial location (i. e., l_i^1 and l_i^2 respectively) with their respective clock (x_1 and x_2) equal to 0. Then, we synchronize the PTA parts through strong synchronization with renaming (via the a transitions), where the transitions to the respective final location (i. e., l_f^1 and l_f^2 respectively) can only be synchronized if both PTA parts are ready to take it together; note that they must reach the location preceding the a transition exactly at the same time due to the urgent invariants. We assume that a is a fresh label not used in the original PTA, and that the a label is renamed into ϵ when synchronized.

Through this synchronization, this ensures that the duration of the upper PTA part ($\sum_j \beta_{i_j} p_{i_j}$) is equal to that of the lower PTA part ($\sum_k \beta_{i_k} p_{i_k} + p$).

All these transitions are labeled with ϵ. At the end of this gadget, we necessarily have $p = \sum_i \beta_i p_i$.

A' is obtained from A as follows: first, we replace all occurrences in A of a constraint $x \sim plt$ with $x \sim p$ (where p is the additional parameter created when handling plt using the above construction). Second, we add before the initial location of A all necessary gadgets in a sequential manner, and we reset all clocks on the transition leading to the original initial location of A.

The initial location of A' is the initial location of the first gadget (i. e., one in the form of Figure 4). Since all transitions in the initial gadget are labeled with ϵ (recall that we use a synchronization with renaming so that the a transitions become ϵ when synchronized), the untimed language of A' is not impacted. □

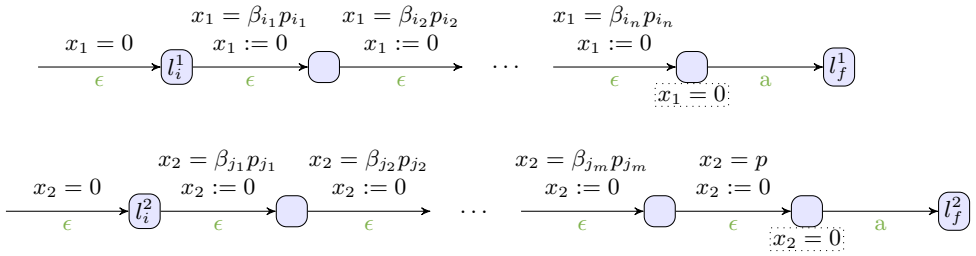


Fig. 4: Encoding a parametric constraint

B Proof of Proposition 7

Proposition 7 (recalled). *PTAs and fpc-PTAs are equally expressive w.r.t. the union of untimed languages.*

Proof. Fpc-PTAs may also contain constraints of the form $plt \sim 0$. The same construction as in Proposition 6 can be used, except that in the PTA the constraint $plt \sim 0$ is replaced with $p \sim 0$. Technically, the constraint $p \sim 0$ is not allowed in an AHV93-PTA; it can be simulated using an extra clock equal to 0, and to be compared with p. (Ensuring this clock is equal to 0 may require to duplicate the current location and to add an ϵ-transition.) □

Remark 4. The above construction requires two additional clocks and as many additional parameters as the number of (different) parametric constraints in the PTA. Clearly, our two additional clocks in the initial gadgets are not necessary, as the other clocks used in the PTA can be used instead. However, for fpc-PTAs, the clock used to simulate "$p \sim 0$" is necessary in our construction; and so seem to be the ϵ-transitions, the extra locations, and all extra parameters.

C Proof of Proposition 10

Proposition 10 (recalled). *hL/U-PTAs are equally expressive with L/U-PTAs w.r.t. the constrained untimed language.*

Proof. $\subseteq$ Let $\mathsf{A}_h = (\Sigma, L, l_0, F, X, P, I, E)$ be an hL/U-PTA, where $P = P_{\overline{v}} \uplus P_v$. Let $v^{\overline{p}}_{0/\infty}$ be the partial parameter valuation assigning 0 to any lower-bound parameter of $P_{\overline{v}}$ and ∞ to any upper-bound parameter of $P_{\overline{v}}$ (and not defined for parameters in P_v). As an abuse of notation, we denote by $v^{\overline{p}}_{0/\infty}(\mathsf{A}_h)$ the PTA where each occurrence of a parameter $p \in P_{\overline{v}}$ has been replaced with $v^{\overline{p}}_{0/\infty}(p)$. The obtained automaton is still parametric, as parameters in P_v are left untouched; also note that the obtained PTA contains no hidden parameter. We will show in the following that $\mathsf{CUL}(\mathsf{A}_h) = \mathsf{CUL}(v^{\overline{p}}_{0/\infty}(\mathsf{A}_h))$.

– Let $(w, v) \in \mathsf{CUL}(\mathsf{A}_h)$. Then there exists a valuation v' of the hidden parameters such that w is accepted by $v'(v(\mathsf{A}_h))$. By the monotonicity property ([HRSV02]) of the L/U-PTA $v(\mathsf{A}_h)$, replacing upper bounds in the parameters assigned by v' by $+\infty$ and lower bounds by 0 gives a TA in which w is still accepted: w is accepted by $v^{\overline{p}}_{0/\infty}(v(\mathsf{A}_h))$. Finally, the hidden and visible parameter sets being disjoint, this means w is accepted by $v(v^{\overline{p}}_{0/\infty}(\mathsf{A}_h))$, i. e., $(w, v) \in \mathsf{CUL}(v^{\overline{p}}_{0/\infty}(\mathsf{A}_h))$.
– $(w, v) \in \mathsf{CUL}(v^{\overline{p}}_{0/\infty}(\mathsf{A}_h))$. Then, as before, w is accepted by $v^{\overline{p}}_{0/\infty}(v(\mathsf{A}_h))$ and $v(\mathsf{A}_h)$ is an L/U-PTA. Then using the bound of [HRSV02, Proposition 4.4], which we will call $\mathcal{N}$ here, we know that if v' is the valuation of hidden parameters that assigns 0 to lower bound parameters and $\mathcal{N}$ to upper bound parameters then w is accepted by $v'(v(\mathsf{A}_h))$, which in turn means that $(w, v) \in \mathsf{CUL}(\mathsf{A}_h)$.

$\supseteq$ The converse is trivial: let A be an L/U-PTA $\mathsf{A} = (\Sigma, L, l_0, F, X, P, I, E)$. Then A is in itself an hL/U-PTA with no hidden parameters, and therefore there exists an hL/U-PTA (itself) with the same constrained untimed language as A.

□

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]).

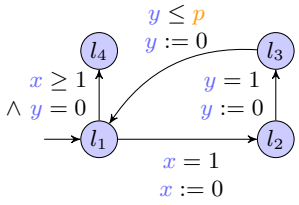
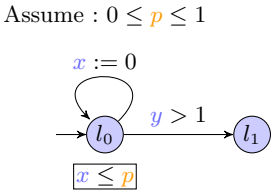


Fig. 1: Motivating PTA

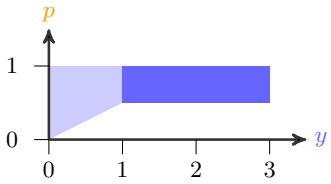
Now, we will show that the union for all values of the parameters of the classical k-extrapolation used for the zone-abstraction for timed automata leads to a non-convex polyhedron. Let us consider the PTA in Fig. 2a with a parameter p such that $0 \leq p \leq 1$. By taking n times the loop we obtain:

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

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 $\mathsf{Ext}_x^M(C)$.

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^M_X(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^M_X(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

if $l \in G$ **then** $K \leftarrow C{\downarrow_P}$;
else
  if $(l, \mathsf{IH}(\mathit{Ext}^M_X(C))) \in S$ **then** $K \leftarrow \bot$;
  else
    $K \leftarrow \top$; $K_{Live} \leftarrow \bot$;
    for *each outgoing* $e = (l, g, a, R, l')$ *from* l *in* $\mathcal{A}$ **do**
      $S' \leftarrow \mathsf{Succ}((l, C), e)$;
      $K_{Good} \leftarrow \mathsf{RIAF}(\mathcal{A}, S', G, S \cup \{(l, \mathsf{IH}(\mathsf{Ext}^M_X(C)))\})$;
      $K_{Block} \leftarrow \top \setminus S'{\downarrow_P}$;
      $K \leftarrow K \cap (K_{Good} \cup K_{Block})$;
      $K_{Live} \leftarrow K_{Live} \cup (C \cap g)^{\swarrow}$;
    $K \leftarrow K \setminus (\mathbb{R}^{X \cup P} \setminus K_{Live}){\downarrow_P}$;

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* $v \in \mathsf{RIAF}(\mathcal{A}, \mathsf{Init}(\mathcal{A}), G, \emptyset)$*.*

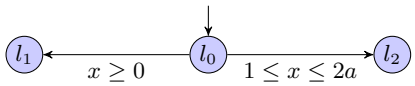


Fig. 3: Counter-example to the density of the result of IAF.

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 $\mathsf{IH}(\mathsf{Ext}^M_X(s)) = \mathsf{IH}(\mathsf{Ext}^M_X(s'))$.
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 $v(\mathsf{IH}(\mathsf{Ext}^M_X(s))) = v(\mathsf{IH}(\mathsf{Ext}^M_X(s')))$. Since v is an integer parameter valuation, by Lemma 6, this is $v(\mathsf{Ext}^M_X(s)) = v(\mathsf{Ext}^M_X(s'))$. 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. □