

Zone extrapolations in parametric timed automata^{*}

Johan Arcile[✉] and Étienne André[📧]

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract. Timed automata (TAs) are an efficient formalism to model and verify systems with hard timing constraints, and concurrency. While TAs assume exact timing constants with infinite precision, parametric TAs (PTAs) leverage this limitation and increase their expressiveness, at the cost of undecidability. A practical explanation for the efficiency of TAs is zone extrapolation, where clock valuations beyond a given constant are considered equivalent. This concept cannot be easily extended to PTAs, due to the fact that parameters can be unbounded. In this work, we propose several definitions of extrapolation for PTAs based on the M -extrapolation, and we study their correctness. Our experiments show an overall decrease of the computation time and, most importantly, allow termination of some previously unsolvable benchmarks.

Keywords: timed automata · abstraction · parameter synthesis · reachability · liveness · IMITATOR

1 Introduction

Timed automata (TAs) [AD94] represent an efficient and expressive formalism to model and verify systems mixing hard timing constraints with concurrency, being one of the most expressive decidable formalisms with timing constraints. However, TAs assume exact timing constants with infinite precision, which may not be realistic in practice; in addition, they assume full knowledge of the model, preventing verification at an early development phase. Parametric timed automata (PTAs) leverage these limitations, by allowing unknown timing constants in the model—at the cost of undecidability: the mere emptiness of the parameter valuations set for which a given (discrete) location is reachable (called *reachability emptiness*) is undecidable [AHV93].

A practical explanation for the efficiency of TAs for reachability properties is (*zone*) *extrapolation*, where clock valuations beyond a given constant are considered to be equivalent. Since the seminal work [AD94], several works improved

^{*} This is the author (and extended) version of the manuscript of the same name published in the proceedings of the 14th NASA Formal Methods Symposium (NFM 2022). The final authenticated version is available at [springer.com](https://www.springer.com). This work is partially supported by the ANR-NRF French-Singaporean research program ProMiS (ANR-19-CE25-0015).

the quality and efficiency of zone extrapolation, by considering different constants per clock [Beh+03; Beh+06] or extending extrapolation to liveness properties [Tri09; Li09]. This concept cannot be easily extended to PTAs, due to the fact that parameters can be unbounded, or that one of their bound may converge towards a constant (for example $\frac{1}{n} \leq p$, with n growing without bound.).

1.1 Related works

Extrapolation in TAs Daw and Tripakis first introduced the *extrapolation* abstraction in [DT98] as a mean to obtain a finite simulation of the state space of TAs. The extrapolation abstraction preserves reachability properties and is based on the largest constant appearing in any state of the model, which can be computed syntactically from the constants present in its guards and invariants. In [Beh+03] Behrmann *et al.* redefine this abstraction with individual clock bounds (i.e., the largest constant is computed for each clock) and will later refer to it in [Beh+06] as the *M*-extrapolation. In this latter work [Beh+06], the *M*-extrapolation is extended to a coarser abstraction based on two constants for each clock: its greater lower bound and its greater upper bound. This new form of extrapolation is referred to as the *LU*-extrapolation and still preserves reachability properties. Experiments are performed using UPPAAL [LPY97]. In 2009, Tripakis [Tri09] showed that the *M*-extrapolation is correct for checking emptiness of timed Büchi automata, i.e., checking for accepting cycles in TAs. The same year, Li [Li09] proves that this result holds true for the *LU*-extrapolation on TAs.

Parameter synthesis for PTAs Most non-trivial decision problems are undecidable for PTAs (see [And19] for a survey). As a consequence exact synthesis is usually out of reach, except for small numbers of clocks or of parameters (see, e.g., [AHV93; Ben+15; BO17]). For general subclasses (without bound on the number of variables), exact synthesis results are very scarce. Some fit in the subclasses of L/U-PTAs¹ [Hun+02], and notably in U-PTAs (resp. L-PTAs) [BL09], where each timing parameter is constrained to be always compared to a clock as an upper (resp. lower) bound, i.e., of the form $x \leq p$ (resp. $p \leq x$). The only known situations when exact reachability-synthesis (i.e., synthesis of all parameter valuations for which a given location is reachable) can be achieved for subclasses of PTAs are

1. reachability-synthesis for U-PTAs (resp. L-PTAs) over *integer-valued* timing parameters [BL09];
2. reachability-synthesis for the whole PTA class, over *bounded and integer-valued* parameters (which reduces to TAs) [JLR15]; and
3. reachability-synthesis for reset-update-to-parameters-PTAs (“R-U2P-PTAs”), in which all clocks must be updated (possibly to a parameter) whenever a clock is compared to a parameter in a guard [ALR21].

¹ While “L/U” means in both cases “lower-upper (bound)”, L/U-PTAs are a completely different concept from LU-extrapolation for (P)TAs.

On the negative side, even L/U-PTAs show negative results for synthesis: while reachability-emptiness is decidable for L/U-PTAs [Hun+02], reachability-synthesis is intractable (its result cannot be represented using a finite union of polyhedra) [JLR15]; and even in the very restricted subclass of U-PTAs without invariant, TCTL-emptiness (i.e., the emptiness of the parameter valuations set for which a TCTL formula is valid) is undecidable [ALR18].

We performed a first attempt to define an extrapolation for PTAs in [ALR15]: we adapted the M -extrapolation to the context of PTAs, although restricted to *bounded* parameter domains only. No implementation was provided. In [Bez+16], the authors also define an extrapolation very similar to [ALR15]. Compared to [ALR15], we reuse here some of the definitions of [ALR15], and we significantly extend the definition of extrapolations; we also consider several subclasses of models, as well as liveness properties; we also perform an experimental evaluation.

1.2 Contributions

We propose several definitions of extrapolation for PTAs, and study their correctness. In the context of bounded parameter domains, we extend the parametric M -extrapolation from [ALR15] to individual clock bounds. Those extrapolations are combined with results from [BL09] to cope with the issue raised by unbounded parameters. We notably consider variants of the U-PTAs and L-PTAs. We show that, on the subclass of (unbounded) PTAs on which they apply, those abstractions preserve not only reachability-synthesis but also cycle-synthesis (“liveness”). We perform experiments using the parametric timed model checker IMITATOR [And21], including on the most general class (rational-valued, possibly unbounded parameters). With the aforementioned negative theoretical results in mind, our evaluation focuses on evaluating the speed enhancement, and the increase of termination chances for our case studies. We show that, overall, extrapolation decreases the verification time and, most importantly, can effectively solve previously unsolvable benchmarks.

Outline We introduce the necessary preliminaries in Section 2. The M -extrapolation in the bounded context (partially reusing results from [ALR15]) is studied in Section 3. Section 4 adapts the M -extrapolation to the unbounded context for reachability properties. Liveness properties are discussed in Section 5. Finally, Section 6 benchmarks the abstractions, and Section 7 concludes the paper.

2 Preliminaries

2.1 Clocks, parameters and guards

Throughout this paper, we assume a set $\mathbb{X} = \{x_1, \dots, x_H\}$ of *clocks*, i.e., real-valued variables that evolve at the same rate. A clock valuation is a function

$w : \mathbb{X} \rightarrow \mathbb{R}_+$. We identify a clock valuation w with the *point* $(w(x_1), \dots, w(x_H))$. We write $\vec{0}$ for the clock valuation assigning 0 to all clocks. Given $d \in \mathbb{R}_+$, $w + d$ denotes the valuation s.t. $(w + d)(x) = w(x) + d$, for all $x \in \mathbb{X}$. Given $R \subseteq \mathbb{X}$, 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 $\mathbb{P} = \{p_1, \dots, p_K\}$ of *parameters*, i.e., unknown constants. A parameter *valuation* v is a function $v : \mathbb{P} \rightarrow \mathbb{Q}$. We identify a valuation v with the *point* $(v(p_1), \dots, v(p_K))$. Given two valuations v_1, v_2 , we write $v_1 \geq v_2$ whenever $\forall p \in \mathbb{P}, v_1(p) \geq v_2(p)$.

In the following, we assume $\bowtie \in \{<, \leq, =, \geq, >\}$. A *constraint* C over $\mathbb{X} \cup \mathbb{P}$ is a conjunction of inequalities of the form $lt \bowtie 0$, where lt is a linear term over $\mathbb{X} \cup \mathbb{P}$ of the form $\sum_{1 \leq i \leq H} \alpha_i x_i + \sum_{1 \leq j \leq K} \beta_j p_j + d$, with $x_i \in \mathbb{X}$, $p_j \in \mathbb{P}$, and $\alpha_i, \beta_j, d \in \mathbb{Z}$. We also refer to constraints as their geometrical representation, i.e., of *convex polyhedron*.

We denote by \perp the constraint over \mathbb{P} corresponding to the empty set of parameter valuations.

Given a parameter valuation v , $v(C)$ denotes the constraint over \mathbb{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 $\mathbb{X} \cup \mathbb{P}$ 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. $w|v \models C$.

We define the *time elapsing* of C , denoted by C^\nearrow , as the constraint over \mathbb{X} and \mathbb{P} obtained from C by delaying all clocks by an arbitrary amount of time. That is, $w'|v \models C^\nearrow$ iff $\exists w : \mathbb{X} \rightarrow \mathbb{R}_+, \exists d \in \mathbb{R}_+$ s.t. $w|v \models C \wedge w' = w + d$.

Given $R \subseteq \mathbb{X}$, 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 $C \downarrow_{\mathbb{P}}$ the projection of C onto \mathbb{P} , i.e., obtained by eliminating the variables not in \mathbb{P} (e.g., using Fourier-Motzkin [Sch86]).

A *simple clock guard* is an inequality of the form $x \bowtie \sum_{1 \leq i \leq K} \alpha_i p_i + z$, with $p_i \in \mathbb{P}$, and $\alpha_i, z \in \mathbb{Z}$. A *clock guard* is a constraint over $\mathbb{X} \cup \mathbb{P}$ defined by a conjunction of simple clock guards. Given a clock guard g , we write $w \models v(g)$ if the expression obtained by replacing each x with $w(x)$ and each p with $v(p)$ in g evaluates to true. We do not consider diagonal constraints (i.e., simple clock guards of the form $x - x' \bowtie \dots$) in this work.

2.2 Parametric timed automata

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

Definition 1 (PTA). A PTA \mathcal{A} is a tuple $\mathcal{A} = (\Sigma, L, \ell_0, L_F, \mathbb{X}, \mathbb{P}, \mathbb{D}, I, E)$, where:

1. Σ is a finite set of actions,
2. L is a finite set of locations,
3. $\ell_0 \in L$ is the initial location,
4. $L_F \subseteq L$ is a set of accepting locations,
5. \mathbb{X} is a finite set of clocks,
6. \mathbb{P} is a finite set of parameters,
7. $\mathbb{D} : \mathbb{P} \rightarrow (\mathbb{Q} \cup \{-\infty\}) \times (\mathbb{Q} \cup \{+\infty\})$ is the parameter domain,
8. I is the invariant, assigning to every $\ell \in L$ a clock guard $I(\ell)$,
9. E is a finite set of edges $e = (\ell, g, a, R, \ell')$ where $\ell, \ell' \in L$ are the source and target locations, $a \in \Sigma$, $R \subseteq \mathbb{X}$ is a set of clocks to be reset, and g is a clock guard.

Let $\mathbb{G}(\mathcal{A})$ denote the set of all simple clock guards of the PTA \mathcal{A} , i.e., all simple clock guards being a conjunct within a guard or an invariant of \mathcal{A} . Given a clock $x \in \mathbb{X}$, we denote by $\mathbb{G}^x(\mathcal{A}) \subseteq \mathbb{G}(\mathcal{A})$ the set of simple clock guards where x appears, i.e., is bound by a non-0 coefficient. A clock x of \mathcal{A} is said to be a *parametric clock* if it is compared to at least one parameter (with a non-0 coefficient) in at least one guard of $\mathbb{G}^x(\mathcal{A})$.

The parameter domain of a PTA is the admissible range of the parameters. Given p , given $\mathbb{D}(p) = (b^-, b^+)$, $\mathbb{D}^-(p)$ denotes b^- while $\mathbb{D}^+(p)$ denotes b^+ . The admissible valuations for p are therefore $[\mathbb{D}^-(p), \mathbb{D}^+(p)]$ (the domain is *closed* unless on the side of an infinite bound). A *bounded* parameter domain assigns to each parameter a minimum rational bound and a maximum rational bound. In that case, $\mathbb{D}^-(p_i) > -\infty$ and $\mathbb{D}^+(p_i) < +\infty$. A bounded parameter domain can be seen as a hyperrectangle in K dimensions. Any parameter that is not bounded is called an *unbounded* parameter. Note that an unbounded parameter can still have a lower bound or an upper bound $\in \mathbb{Q}$.

Definition 2 (bounded PTA). *A bounded PTA is a PTA the parameter domain of which is bounded. Otherwise, it is unbounded.*

Given a parameter valuation v , we denote by $v(\mathcal{A})$ the non-parametric structure where all occurrences of a parameter p_i have been replaced by $v(p_i)$. We denote as a *timed automaton* any structure $v(\mathcal{A})$, by assuming a rescaling of the constants: by multiplying all constants in $v(\mathcal{A})$ by the least common multiple of their denominators, we obtain an equivalent (integer-valued) TA, as defined in [AD94].

Example 1. Fig. 1a displays the graphical representation of a bounded PTA. We have $\mathbb{G}(\mathcal{A}) = \{x \leq 1, 1 < y, x < p\}$, $\mathbb{G}^x(\mathcal{A}) = \{x \leq 1, x < p\}$, and $\mathbb{G}^y(\mathcal{A}) = \{1 < y\}$. The valuation of parameter p can be any rational value in $[0, 5]$, hence an infinite number of possible parameter valuations. Therefore, this PTA can be seen as the abstract representation for an infinite number of TAs.

Concrete semantics of TAs Let us now recall the concrete semantics of TAs.

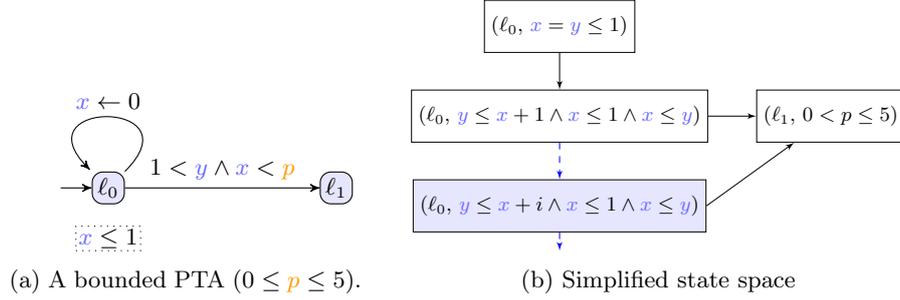


Fig. 1: Example of a bounded PTA generating an infinite state space. Blue states are a representation of an infinite sequence of states where variable i corresponds to the number of times the looping transition on ℓ_0 was taken.

Definition 3 (Semantics of a TA). Given a PTA $\mathcal{A} = (\Sigma, L, \ell_0, L_F, \mathbb{X}, \mathbb{P}, \mathbb{D}, I, E)$ and a parameter valuation v , the concrete semantics of $v(\mathcal{A})$ is given by the timed transition system (S, s_0, \rightarrow) , with

- $S = \{(\ell, w) \in L \times \mathbb{R}_{\geq 0}^H \mid w \models v(I(\ell))\}$,
- $s_0 = (\ell_0, \vec{0})$,
- \rightarrow consists of the (continuous) delay and discrete transition relations:
 - delay transitions: $(\ell, w) \xrightarrow{d} (\ell, w + d)$, with $d \in \mathbb{R}_{\geq 0}$, if $\forall d' \in [0, d], (\ell, w + d') \in S$;
 - discrete transitions: $(\ell, w) \xrightarrow{e} (\ell', w')$, if $(\ell, w), (\ell', w') \in S$, and there exists $e = (\ell, g, a, R, \ell') \in E$, such that $w' = [w]_R$, and $w \models v(g)$.

Moreover, we write $(\ell, w) \xrightarrow{(d,e)} (\ell', w')$ for a combination of a delay and discrete transition if $\exists w'' : (\ell, w) \xrightarrow{d} (\ell, w'') \xrightarrow{e} (\ell', w')$.

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 *run* of $v(\mathcal{A})$ is an alternating sequence of concrete states of $v(\mathcal{A})$ and pairs of edges and delays starting from the initial state s_0 and is of the form $s_0, (d_0, e_0), s_1, \dots, s_i, (d_i, e_i), \dots$ with $i = 0, 1, \dots$, $e_i \in E$, $d_i \in \mathbb{R}_{\geq 0}$ and $s_i \xrightarrow{(d_i, e_i)} s_{i+1}$. The set of all (finite or infinite) runs of a TA $v(\mathcal{A})$ is $\text{Runs}(v(\mathcal{A}))$. Given a concrete state $s = (\ell, w)$, we say that s is reachable in $v(\mathcal{A})$ (and by extension that ℓ is reachable, or that $v(\mathcal{A})$ visits ℓ) if s appears in a run of $v(\mathcal{A})$. An infinite run is *accepting* if it visits infinitely often (at least) one location $\ell \in L_F$.

Symbolic semantics of PTAs Let us now recall the symbolic semantics of PTAs (see e.g., [Hum+02; And+09]).

Definition 4 (Symbolic state). A *symbolic state* is a pair (ℓ, C) where $\ell \in L$ is a location, and C is a constraint over $\mathbb{X} \cup \mathbb{P}$ called its associated parametric zone.

Definition 5 (Symbolic semantics). Given a PTA $\mathcal{A} = (\Sigma, L, \ell_0, L_F, \mathbb{X}, \mathbb{P}, \mathbb{D}, I, E)$, the symbolic semantics of \mathcal{A} is the labeled transition system called parametric zone graph $\mathcal{PZG} = (E, \mathbf{S}, \mathbf{s}_0, \Rightarrow)$, with

- $\mathbf{S} = \{(\ell, C) \mid C \subseteq I(\ell)\}$,
- $\mathbf{s}_0 = (\ell_0, (\bigwedge_{1 \leq i \leq H} x_i = 0)^\nearrow \wedge I(\ell_0) \wedge \bigwedge_{1 \leq j \leq K} \mathbb{D}^-(p_j) \leq p_j \leq \mathbb{D}^+(p_j))$, and
- $((\ell, C), e, (\ell', C')) \in \Rightarrow$ if $e = (\ell, g, a, R, \ell') \in E$ and $C' = [(C \wedge g)]_R \wedge I(\ell')^\nearrow \wedge I(\ell')$, with C' satisfiable.

That is, in the parametric zone graph, nodes are symbolic states, and arcs are labeled by *edges* of the original PTA. Given $(\mathbf{s}, e, \mathbf{s}') \in \Rightarrow$, we write $\mathbf{s}' = \text{Succ}(\mathbf{s}, e)$.

Given a concrete state $s = (\ell, w)$ and a symbolic state $\mathbf{s} = (\ell', C)$, we write $s \in \mathbf{s}$ whenever $\ell = \ell'$ and $w \models C$.

Example 2. Fig. 1b displays the parametric zone graph of the PTA in Fig. 1a. Blue states represent an infinite sequence (i being the number of times the looping transition was taken). (We assume all clocks and parameters to be non-negative and, for sake of brevity, constraints of the form $x \geq 0$ may be omitted.)

Computation problems Given a class of decision problems \mathcal{P} (reachability, liveness, 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 reaches some given location?”) and liveness (i.e., “does there exist a run that visits a given location infinitely often?”).

3 M - and \vec{M} -extrapolation for bounded PTAs

3.1 Recalling M -extrapolation

In this subsection, we recall some results from [Beh+06; ALR15], where the classical “ k -extrapolation” used for the zone-abstraction of TAs is adapted to PTAs. While this part is not clearly a contribution of the current manuscript, we redefine some concepts from [ALR15], and provide several original examples.

Maximal constant of a bounded PTA First, let us formally define the *maximal constant* of a bounded PTA. The maximal constant M is the maximum value that can appear in the guards and invariants of the PTA. When those constraints are parametric expressions, we compute the maximum value that the expression can take over any parameter valuation within the (bounded) parameter domain \mathbb{D} (this maximal value is unique since expressions are linear).

Given a simple clock guard g of the form $x \bowtie \sum_{1 \leq i \leq K} \alpha_i p_i + z$ we define $C_{maxg}(g) = \sum_{1 \leq i \leq K} \alpha_i \gamma_i + z$ where

1. $\gamma_i = \mathbb{D}^-(p_i)$ if $\alpha_i < 0$,

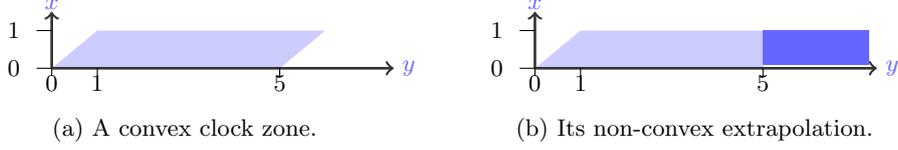


Fig. 2: Example illustrating the non-convex parametric extrapolation.

2. $\gamma_i = \mathbb{D}^+(p_i)$ if $\alpha_i > 0$, and
3. $\gamma_i = 0$ otherwise.

Example 3. Consider the simple clock guard $g : x \leq 2p_1 - p_2 + 1$ and $p_1 \in [2, 5]$, and $p_2 \in [-3, 4]$; then $C_{maxg}(g) = 2 \times 5 - (-3) + 1 = 14$.

Definition 6 (Maximal constant). Given a bounded PTA \mathcal{A} , for any clock $x \in \mathbb{X}$, the maximal constant for clock x is $C_{max}^x(\mathcal{A}) = \max_{g \in \mathbb{G}^x(\mathcal{A})} C_{maxg}(g)$ furthermore, the maximal constant of the PTA is $C_{max}(\mathcal{A}) = \max_{g \in \mathbb{G}(\mathcal{A})} C_{maxg}(g)$.

Example 4. Consider again Fig. 1a (recall that $0 \leq p \leq 5$). Then, $C_{max}^x(\mathcal{A}) = 5$, $C_{max}^y(\mathcal{A}) = 1$ and $C_{max}(\mathcal{A}) = 5$.

Bisimulation and largest constant in TAs Let us recall from [Beh+06] the notion of bisimulation based on the maximal constant M :

Lemma 1 ([Beh+06, Lemma 1]). Let \mathcal{A} be a TA. Given clock x , let $M(x)$ be an integer constant greater than or equal to $C_{max}^x(\mathcal{A})$. Let w, w' be two clock valuations. Let \equiv_M be the relation defined as $w \equiv_M w'$ iff $\forall x \in \mathbb{X}$: either $w(x) = w'(x)$ or $(w(x) > M(x) \text{ and } w'(x) > M(x))$. The relation $\mathcal{R} = \{((\ell, w), (\ell, w')) \mid w \equiv_M w'\}$ is a bisimulation relation.

Example 5. Let us recall the motivation for the use of an extrapolation, through the PTA \mathcal{A} in Fig. 1a. After i times through the loop, we get constraints in ℓ_0 of the form $y - x \leq i$. The maximal constant of the model is $C_{max}(\mathcal{A}) = 5$. After five loops, y can be greater than 5. Therefore, we can apply on y the classical k -extrapolation used for TAs (from [Beh+06]) of the corresponding zone. More specifically, we consider that when $y > k$, the bounds on y can be ignored. The obtained polyhedron is non-convex, but can be split into two convex ones, one where $y \leq k$ (the part without extrapolation) and one with $y > k$ (the part with extrapolation). This is depicted in Fig. 2 where Fig. 2a is the original clock zone (formally $y \leq x + 5 \wedge x \leq 1 \wedge x \leq y$) and Fig. 2b is its non-convex extrapolation (formally $(x \leq y \leq 5 \wedge x \leq 1) \vee (y \geq 5 \wedge 0 < x \leq 1)$).

Let us now formally recall from [ALR15] the concept of M -extrapolation for PTAs. First, we need to recall the *cylindrification* operation, which is a usual operation that consists in *unconstraining* variable x .

Definition 7 (Cylindrification [ALR15]). For a polyhedron C and variable x , we denote by $\text{Cyl}_x(C)$ the cylindrification of C along variable x , i.e., $\text{Cyl}_x(C) = \{w \mid \exists w' \in C, \forall x' \neq x, w'(x') = w(x') \text{ and } w(x) \geq 0\}$.

The (M, x) -extrapolation is an operation that splits a polyhedron into two polyhedra such that clock x is either less than or equal to M , or is strictly greater than M while being independent from the other variables.

Definition 8 ((M, x) -extrapolation [ALR15]). Let C be a polyhedron. Let $M \in \mathbb{N}$ be a non-negative integer constant and x be a clock. The (M, x) -extrapolation of C , denoted by $\text{Ext}_x^M(C)$, is defined as:

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

Given $\mathbf{s} = (\ell, C)$, we write $\text{Ext}_x^M(\mathbf{s})$ for $\text{Ext}_x^M(C)$.

We can now consistently define the M -extrapolation operator.

Definition 9 (M -extrapolation [ALR15]). Let $M \in \mathbb{N}$ be a non-negative integer constant and \mathbb{X} be a set of clocks. The (M, \mathbb{X}) -extrapolation operator $\text{Ext}_{\mathbb{X}}^M$ is defined as the composition (in any order) of all Ext_x^M , for all $x \in \mathbb{X}$. When clear from the context we omit \mathbb{X} and only write M -extrapolation.

[ALR15, Lemma 1] shows that the order of composition of (M, x) -extrapolation does not impact its results, i.e., $\text{Ext}_x^M(\text{Ext}_y^M(C)) = \text{Ext}_y^M(\text{Ext}_x^M(C))$, and [ALR15, Lemma 5] shows that given a symbolic state \mathbf{s} of a PTA and a non-negative integer M greater than the maximal constant of the PTA $C_{\max}(\mathcal{A})$, for any clock x and parameter valuation v such that $(\ell, w) \in v(\text{Ext}_x^M(\mathbf{s}))$ is a concrete state, there exists a state $(\ell, w') \in v(\mathbf{s})$ such that (ℓ, w) and (ℓ, w') are bisimilar.

3.2 Synthesis with extrapolation

We now recall the reachability-synthesis algorithm, that was formalized in [JLR15], and then enhanced with extrapolation (and “integer hull”—unused here) in [ALR15]. We adapt here to our notations a version of reachability-synthesis with the extrapolation, and write a full proof of correctness (absent from [ALR15]), also because we will use it and improve it in the remainder of the paper.

The goal of EEF given in Algorithm 1 (“E” stands for “extrapolation”, “EF” denotes reachability) is to synthesize parameter valuation solutions to the reachability-synthesis problem, i.e., the valuations for which there exists a run eventually reaching a location in T . EEF proceeds as a post-order traversal of the symbolic reachability tree, and collects all parametric constraints associated with the target locations T . In contrast to the classical reachability-synthesis algorithm EF formalized in [JLR15], it recursively calls itself (line 6) with the *extrapolation* of the successor of the current symbolic state (this difference is highlighted in yellow in Algorithm 1).

Algorithm 1: $\text{EEF}(\mathcal{A}, \mathbf{s}, T, \mathbf{P})$

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

```
1 if  $\ell \in T$  then  $K \leftarrow C \downarrow_{\mathbb{P}}$  ;
2 else
3    $K \leftarrow \perp$  ;
4   if  $\mathbf{s} \notin \mathbf{P}$  then
5     for each outgoing  $e$  from  $\ell$  in  $\mathcal{A}$  do
6        $K \leftarrow K \cup \text{EEF}(\mathcal{A}, \text{Ext}_x^M(\text{Succ}(\mathbf{s}, e)), T, \mathbf{P} \cup \{\mathbf{s}\})$  ;
7 return  $K$ 
```

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 the initial symbolic state \mathbf{s}_0 ;
- for every node n of T^∞ , if n is labeled by some symbolic state \mathbf{s} , then for all edges e of \mathcal{A} , there exists a child n' of n labeled by $\text{Succ}(\mathbf{s}, e)$ iff $\text{Succ}(\mathbf{s}, e)$ is not empty.

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

In addition, before we prove [Theorem 1](#), we need the following lemmas (adapted from [\[ALR15\]](#)).

We first recall the following lemma ([\[ALR15, Lemma 4\]](#)):

Lemma 2 ([\[ALR15, Lemma 4\]](#)). *For all parameter valuation v , non-negative integer constants M , clock x and valuations set C , $v(\text{Ext}_x^M(C)) = \text{Ext}_x^M(v(C))$.*

Lemma 3 ([\[ALR15, Lemma 5\]](#)). *Let \mathcal{A} be a PTA and \mathbf{s} be a symbolic state of \mathcal{A} . Let x be a clock, $M \in \mathbb{N}$ an integer constant greater than or equal to $C_{\max}(\mathcal{A})$, v be a parameter valuation and $(\ell, w) \in v(\text{Ext}_x^{M(x)}(\mathbf{s}))$ be a concrete state. There exists a state $(\ell, w') \in v(\mathbf{s})$ such that (ℓ, w) and (ℓ, w') are bisimilar.*

Remark 1. [Lemma 9](#) is the equivalent of [Lemma 3](#) for the \vec{M} -extrapolation.

We then prove the following [Lemma 4](#):

Lemma 4. *Let \mathcal{A} be a PTA. For all symbolic states \mathbf{s} and \mathbf{s}' , non-negative integer M greater than the maximal constant of the PTA $C_{\max}(\mathcal{A})$, and parameter valuation v , such that $v(\text{Ext}_x^M(\mathbf{s})) = v(\text{Ext}_x^M(\mathbf{s}'))$, for all states $(\ell, w) \in v(\mathbf{s})$, there exists a state $(\ell, w') \in v(\mathbf{s}')$ such that (ℓ, w) and (ℓ, w') are bisimilar.*

Proof. This is a direct consequence of [Lemmas 2](#) and [3](#).

Algorithm 1 is correct (i.e., sound and complete):

Theorem 1. *Let \mathcal{A} be a PTA with initial symbolic state \mathbf{s}_0 , and $T \subseteq L$ a set of target locations. Assume $\text{EEF}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$ terminates. We have:*

1. *Soundness: If $v \in \text{EEF}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$ then T is reachable in $v(\mathcal{A})$;*
2. *Completeness: For all v , if T is reachable in $v(\mathcal{A})$ then $v \in \text{EEF}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$.*

Proof. We reuse here large parts of the proof of [ALR15, Theorem 2], as that theorem proves the correctness of a synthesis algorithm using both extrapolation and integer hulls—while we use here only extrapolation. We give it in full details though, as our formal result will be modified for our subsequent definitions of extrapolations (e.g., Propositions 1 and 2).

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

Soundness is a direct consequence of Lemma 5.

2. 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. We reuse the proof of the result of [ALR15], to only cope with extrapolation (without the integer hull defined and used in [ALR15]). Before we start, let us just recall two more results from [JLR15]:

Lemma 6 ([JLR15, Lemma 1]). *For all parameter valuation v , symbolic state \mathbf{s} and edge e , we have $\text{Succ}(v(\mathbf{s}), v(e)) = v((\text{Succ}(\mathbf{s}, e)))$.*

Lemma 7 ([JLR15, Corollary 2]). *For each parameter valuation v , reachable symbolic state \mathbf{s} , and state s , we have $s \in v(\mathbf{s})$ if and only if there is a run of $v(\mathcal{A})$ from the initial state leading to s .*

Now, the algorithm having terminated, it has explored a finite prefix $Tree$ of T^∞ . Let v be a parameter valuation. Suppose there exists a run ρ in $v(\mathcal{A})$ that reaches T . Then ρ is finite and its last state has a location belonging to T . Let e_1, \dots, e_p be the edges taken in ρ and consider the branch in the tree $Tree$ 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 $Tree$, then the tree $Tree$ has depth greater than or equal to the size of the sequence and we can apply Lemma 5 to obtain that $v \in \text{EEF}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$. Otherwise, let $\mathbf{s} = (\ell, C)$ be the symbolic state labeling the last node of the branch, e_k

be the first edge in e_1, \dots, e_p that is not present in the branch and (ℓ, w) be the state of ρ just before taking e_k . Since (ℓ, w) has a successor via e_k , then $\text{Succ}(v(\mathbf{s}), v(e_k))$ is not empty; then using [Lemma 6](#), $v(\text{Succ}(\mathbf{s}, e_k))$ is not empty; therefore, $\text{Succ}(\mathbf{s}, e_k)$ is not empty. Since the node labeled by \mathbf{s} has no child in *Tree*, it follows that either $\ell \in T$ or there exists another node on the branch that is labeled by \mathbf{s}' such that $\text{Ext}_{\mathbb{X}}^M(\mathbf{s}) = \text{Ext}_{\mathbb{X}}^M(\mathbf{s}')$.

In the former case, we can apply [Lemma 5](#) to the prefix of ρ ending in (ℓ, w) and we obtain that $v \in \text{EEF}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$.

In the latter case, we have $v(\text{Ext}_{\mathbb{X}}^M(\mathbf{s})) = v(\text{Ext}_{\mathbb{X}}^M(\mathbf{s}'))$. Using now [Lemma 4](#), there exists a state $(\ell, w') \in \mathbf{s}'$ that is bisimilar to (ℓ, w) .

Also, by [Lemma 7](#), (ℓ, w') is reachable in $v(\mathcal{A})$ via some run ρ_1 along edges $e_1 \dots e_m$, 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' be 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 T . 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.

3.3 Extending the M -extrapolation to individual bounds

Our first technical contribution is to extend the extrapolation from [\[ALR15\]](#) to *individual* clock bounds, instead of a global one, in the line of what has been proposed for non-parametric TAs [\[Beh+06\]](#).

Definition 10 (\vec{M} -extrapolation). Let $\mathbb{X} = \{x_1, \dots, x_H\}$ the set of clocks of the PTA. Let $\vec{M} = \{M(x_1), \dots, M(x_H)\}$ be a set of non-negative integer constants. The \vec{M} -extrapolation, denoted by $\text{Ext}_{\mathbb{X}}^{\vec{M}}$, is the composition (in any order) of all $\text{Ext}_x^{M(x)}$ for all $x \in \mathbb{X}$.

All we need to do for the results from [\[ALR15\]](#) to hold on the \vec{M} -extrapolation is to adapt [\[ALR15, Lemmas 1 and 5\]](#).

Lemma 8. For all polyhedra C , integers $M(x), M(x') \geq 0$ and clock variables x and x' , we have $\text{Ext}_x^{M(x)}(\text{Ext}_{x'}^{M(x')}(C)) = \text{Ext}_{x'}^{M(x')}(C) \cap \text{Ext}_x^{M(x)}(C)$.

Proof. The result comes from the following facts:

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

We now extend [\[ALR15, Lemma 5\]](#) to $\text{Ext}_{\mathbb{X}}^{\vec{M}}$:

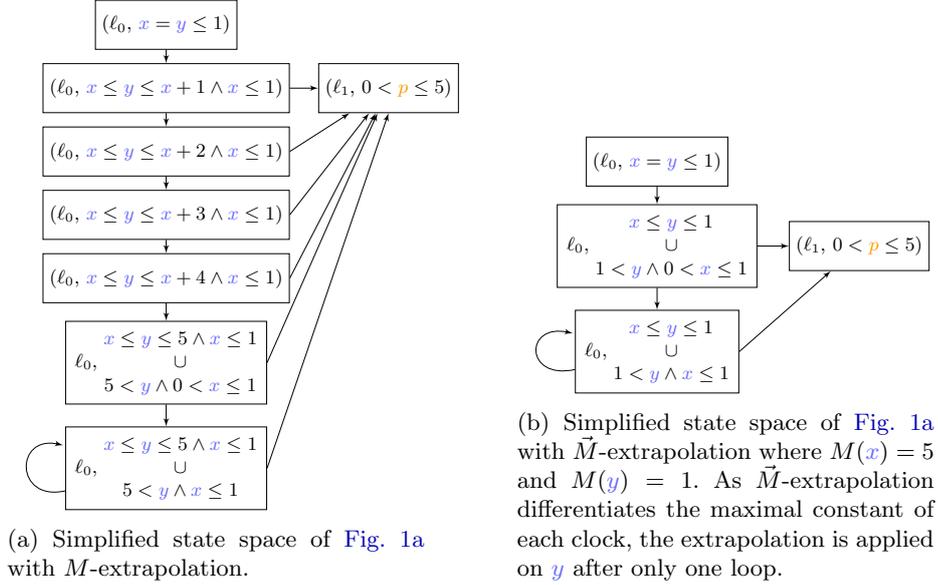


Fig. 3: Comparison between M -extrapolation and \vec{M} -extrapolation.

Lemma 9 (\vec{M} and bisimilarity). *Let \mathcal{A} be a PTA and \mathbf{s} be a symbolic state of \mathcal{A} . Let x be a clock, $M(x) \in \mathbb{N}$ an integer constant greater than or equal to $C_{max}^x(\mathcal{A})$, v be a parameter valuation and $(\ell, w) \in v(\text{Ext}_x^{M(x)}(\mathbf{s}))$ be a concrete state. There exists a state $(\ell, w') \in v(\mathbf{s})$ such that (ℓ, w) and (ℓ, w') are bisimilar.*

Proof. If $(\ell, w|v) \in \mathbf{s}$, then the results holds trivially. Otherwise, it means that there exists some clock x such that $(\ell, w|v) \in \text{Cyl}_x(\mathbf{s} \cap (x > M(x))) \cap (x > M(x))$. This implies that $v(\mathbf{s}(x > M(x))) \neq \emptyset$ and $w(x) > M(x)$. Therefore, and using the definition of Cyl_x , there exists $(\ell, w'|v) \in \mathbf{s} \cap (x > M(x))$ such that for all $x' \neq x$, $w'(x') = w(x')$. We also have $w'(x) > M(x)$, which means that $w' \equiv_M w$ and by Lemma 1, we obtain the expected result.

Given $M \in \mathbb{N}$, given a vector \vec{M} , note that, whenever $\vec{M}(x) \leq M$ for all $x \in \mathbb{X}$, then the \vec{M} -extrapolation is necessarily coarser than the M -extrapolation.

Let \vec{M} be such that, for all x , $\vec{M}(x) = C_{max}^x(\mathcal{A})$. Let $\vec{\text{EEF}}$ denote the modification of EEF where $\text{Ext}_{\mathbb{X}}^M$ is replaced with $\text{Ext}_{\mathbb{X}}^{\vec{M}}$ (line 6 in Algorithm 1). That is, instead of computing the M -extrapolation of each symbolic state, we compute its \vec{M} -extrapolation. Fig. 3 illustrates its effect on the state space of Fig. 1a.

Proposition 1. *Let \mathcal{A} be a PTA with initial symbolic state \mathbf{s}_0 , and $T \subseteq L$ a set of target locations. Assume $\vec{\text{EEF}}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$ terminates. We have:*

1. *Soundness: If $v \in \vec{\text{EEF}}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$ then T is reachable in $v(\mathcal{A})$;*
2. *Completeness: For all v , if T is reachable in $v(\mathcal{A})$ then $v \in \vec{\text{EEF}}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$.*

Proof. The result follows immediately from the proof of [Theorem 1](#), by applying [Lemma 9](#) instead of [Lemma 4](#).

4 \vec{M} -extrapolation on unbounded PTAs

In this section, we extend the \vec{M} -extrapolation to subclasses of (unbounded) PTAs. This requires to be able to identify for each clock $x \in \mathbb{X}$ a constant $M(x)$ such that given a symbolic state \mathbf{s} and a parameter valuation v , for any concrete state in $v(\text{Ext}_x^{M(x)}(\mathbf{s}))$ there exists a bisimilar state in $v(\mathbf{s})$, i.e., [Lemma 9](#) holds true.

We will consider

1. L-PTAs and U-PTAs ([Section 4.1](#)),
2. bounded PTAs with additional *unbounded* lower-bound or upper-bound parameters ([Section 4.2](#)), and
3. the full class of PTAs to which we apply extrapolation only on bounded parameters ([Section 4.3](#)).

4.1 \vec{M} -extrapolation on unbounded L-PTAs and U-PTAs

Recalling L-PTAs and U-PTAs We will use results from [\[BL09\]](#), where the authors propose a constant N for unbounded parameters such that any parameter valuation greater than N will exhibit similar behaviors in regard of infinite accepting runs. Specifically, a (different) constant N can be computed on unbounded L-PTAs and U-PTAs, which are subset of the general PTAs.

First, let us recall the definitions of L-PTAs and U-PTAs [\[BL09\]](#). An L-PTA (respectively U-PTA) is a PTA where each parameter always appears as a lower- (respectively upper-)bound when compared to a clock.

Definition 11 (L-PTA and U-PTA [\[BL09\]](#)). A PTA \mathcal{A} is an L-PTA (resp. U-PTA) if, for each guard $x \bowtie \sum_{1 \leq i \leq K} \alpha_i p_i + z$ of $\mathbb{G}(\mathcal{A})$, for all i :

- $\alpha_i = 0$, or
- $\alpha_i > 0$ and $\bowtie \in \{\geq, >\}$ (respectively $\bowtie \in \{<, \leq\}$), or
- $\alpha_i < 0$ and $\bowtie \in \{<, \leq\}$ (respectively $\bowtie \in \{\geq, >\}$).

L-PTAs and U-PTAs feature a well-known monotonicity property: enlarging a parameter valuation in a U-PTA (resp. decreasing in an L-PTA) can only *add* behaviors, as recalled in the following lemma:

Lemma 10 ([\[BL09\]](#)). Given a U-PTA (resp. L-PTA) \mathcal{A} , given two valuations v_1, v_2 with $v_1 \leq v_2$ (resp. $v_1 \geq v_2$), then $\text{Runs}(v_1(\mathcal{A})) \subseteq \text{Runs}(v_2(\mathcal{A}))$.

For any L-PTA \mathcal{A} , as per [\[BL09, Theorem 3\]](#), there exists a constant bound N , such that for all valuations v_1, v_2 with $v_1 \geq v_2 \geq v_N$ (where v_N denotes the parameter valuation assigning N to each parameter), if $v_2(\mathcal{A})$ provides an infinite accepting run then so does $v_1(\mathcal{A})$. Since \mathcal{A} is an L-PTA, $v_2(\mathcal{A})$ includes all the

possible executions of $v_1(\mathcal{A})$, which is given by [Lemma 10](#). That is, if $v_1(\mathcal{A})$ yields an infinite accepting run, then so does $v_2(\mathcal{A})$. Therefore, for any valuations $v \geq v_N$ and $v' \geq v_N$, $v(\mathcal{A})$ yields an infinite accepting run iff $v'(\mathcal{A})$ yields an infinite accepting run.

A dual result is shown for U-PTAs in [[BL09](#), Theorem 6]. For any U-PTA \mathcal{A} , there exists a constant bound N such that for all valuations v_1, v_2 with $v_1 \geq v_2 \geq v_N$, if $v_1(\mathcal{A})$ yields an infinite accepting run then so does $v_2(\mathcal{A})$. As \mathcal{A} is a U-PTA, $v_1(\mathcal{A})$ includes all the possible executions of $v_2(\mathcal{A})$, hence if $v_2(\mathcal{A})$ yields an infinite accepting run then so does $v_1(\mathcal{A})$. Therefore, for a given valuation $v \geq v_N$, if $v(\mathcal{A})$ yields an infinite accepting run, then so does $v'(\mathcal{A})$ for any $v' \geq v_N$. Formally:

Lemma 11 ([\[BL09, Theorems 3 and 6\]](#)). *Given a U-PTA (resp. L-PTA) \mathcal{A} with N the constant bound defined in [[BL09](#)], given two valuations $v_1 \geq v_N$ and $v_2 \geq v_N$, there exists an infinite accepting run in $v_1(\mathcal{A})$ iff there exists an infinite accepting run in $v_2(\mathcal{A})$.*

Computation of \widehat{N} Given an L-PTA (respectively U-PTA) \mathcal{A} , the value given in [[BL09](#)] is $N = k(R+1) + c + 1$ (respectively $N = 8k(R+1) + c + 1$), where k is the number of parametric clocks of \mathcal{A} , R is the number of clock regions obtained when the parameter valuation is 0 for all parameters, and c is the greatest non-parametric constant in absolute value among all linear expressions. More precisely, all linear expression being of the form $\sum_{1 \leq i \leq H} \alpha_i x_i + \sum_{1 \leq j \leq K} \beta_j p_j + d \bowtie 0$, c is the maximum over all $|d|$. Although k and c are obtained syntactically, R needs to be computed. As N acts as a lower bound, using an over-approximation of R would still guarantee the correctness of [Lemma 11](#). From [[AD94](#), Lemma 4.5], the number of clock regions is bounded by $\widehat{R} = 2^{|\mathbb{X}|} |\mathbb{X}|! \prod_{x \in \mathbb{X}} (2c_x + 2)$ with \mathbb{X} the set of clocks and c_x the greatest constant over x (either as a upper or lower bound)—which can both be obtained syntactically. We define \widehat{N} as the constant defined in [[BL09](#)] for an L-PTA (resp. U-PTA) \mathcal{A} , where we use \widehat{R} (the aforementioned over-approximation of the number of clock regions) instead of their actual number R .

Formal results We first adapt [Lemma 11](#) to our new constant \widehat{N} :

Lemma 12. *Given a U-PTA (resp. L-PTA) \mathcal{A} , given two valuations $v_1 \geq v_{\widehat{N}}$ and $v_2 \geq v_{\widehat{N}}$, there exists an infinite accepting run in $v_1(\mathcal{A})$ iff there exists an infinite accepting run in $v_2(\mathcal{A})$.*

Proof. From the fact that we use in the computation of \widehat{N} an over-approximation on the number of clock regions (with $R \leq \widehat{R}$), giving $N \leq \widehat{N}$.

We can now prove the correctness of extrapolation for unbounded L-PTAs and U-PTAs.

Let $\widehat{M} = \{M(x_1), \dots, M(x_H)\}$ such that $M(x_i)$ is the maximal constant of clock x_i when bounding all unbounded parameters with \widehat{N} . Let $\widehat{\text{EEF}}$ denote the

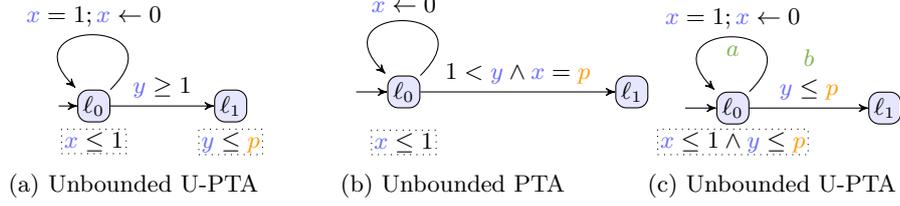
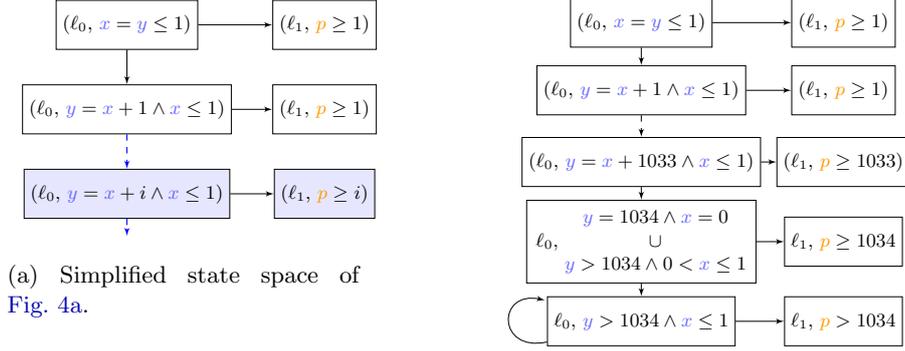


Fig. 4: Three toy PTAs



(a) Simplified state space of Fig. 4a.
(b) Simplified state space of Fig. 4a with the \widehat{M} -extrapolation where $M(x) = 1$ and $M(y) = 1034$, computed using \widehat{N} . The dashed link represents a succession of 1031 intermediate states where the value of y grows from $x + 1$ to $x + 1033$.

Fig. 5: Example of an unbounded PTA generating an infinite state space.

modification of EEF where Ext_x^M is replaced with $\text{Ext}_x^{\widehat{M}}$ (line 6 in Algorithm 1). That is, instead of computing the Ext^M -extrapolation of each symbolic state, we compute its $\text{Ext}^{\widehat{M}}$ -extrapolation.

Example 6. Fig. 5 illustrates the effects of the \widehat{M} -extrapolation on the unbounded U-PTA of Fig. 4a. Fig. 5a displays its (simplified) infinite state space. The valuation of parameter p can be any value in \mathbb{Q}_+ . Fig. 5b shows the state space obtained with the \widehat{M} -extrapolation. Note that the state space is now finite.

Proposition 2. Let \mathcal{A} be an L-PTA or U-PTA with initial symbolic state \mathbf{s}_0 , and $T \subseteq L$ a set of target locations. Assume $\widehat{\text{EEF}}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$ terminates. We have:

1. *Soundness:* If $v \in \widehat{\text{EEF}}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$ then T is reachable in $v(\mathcal{A})$;
2. *Completeness:* For all v , if T is reachable in $v(\mathcal{A})$ then $v \in \widehat{\text{EEF}}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$.

We first prove the following lemma, which adapts Lemma 1 to L-PTAs and U-PTAs.

Lemma 13. *Let \mathcal{A} be an L-PTA or a U-PTA. Given clock x , let $M(x)$ be an integer constant greater than or equal to the maximal constant $C_{max}^x(\mathcal{A})$ of clock x when bounding all unbounded parameters with \widehat{N} . For a given parameter valuation $v(\mathcal{A})$ of \mathcal{A} , let w, w' be two clock valuations. Let \equiv_M be the relation defined as $w \equiv_M w'$ iff $\forall x \in \mathbb{X}$: either $w(x) = w'(x)$ or $(w(x) > M(x) \text{ and } w'(x) > M(x))$. The relation $\widehat{\mathcal{R}} = \{((\ell, w), (\ell, w')) \mid w \equiv_M w'\}$ is a bisimulation relation.*

Proof. Any valuation $w(x) > M(x)$ implies a parameter valuation v greater than or equal to $v_{\widehat{N}}$. And we know by Lemma 12 that either for all valuation $v \geq v_{\widehat{N}}$, $v(\mathcal{A})$ accepts an infinite accepting run, or for all valuation $v \geq v_{\widehat{N}}$, $v(\mathcal{A})$ does not accept an infinite accepting run. As checking infinite accepting run can be used to reachability (for instance, by introducing an unguarded self-loop on each location matching the accepting condition), this implies that any reachable location can be reached with a clock valuation w such that for any x_i , $w(x_i) \leq M(x_i)$. As a result, relation $\widehat{\mathcal{R}}$ preserve the bisimilarity of relation \mathcal{R} from Lemma 1.

We then prove the following lemma, equivalent to Lemma 9.

Lemma 14 (\widehat{M} and bisimilarity). *Let \mathcal{A} be an L-PTA or a U-PTA and \mathbf{s} be a symbolic state of \mathcal{A} .*

Let x be a clock, $M(x) \in \mathbb{N}$ an integer constant greater than or equal to the maximal constant $C_{max}^x(\mathcal{A})$ of clock x when bounding all unbounded parameters with \widehat{N} , v be a parameter valuation and $(\ell, w) \in v(\text{Ext}_x^{M(x)}(\mathbf{s}))$ be a concrete state. There exists a state $(\ell, w') \in v(\mathbf{s})$ such that (ℓ, w) and (ℓ, w') are bisimilar.

Proof. The result follows immediately from the proof of Lemma 9, by applying Lemma 13 instead of Lemma 1.

We can proceed with the proof of Proposition 2:

Proof. The result follows immediately from the proof of Theorem 1, by applying Lemma 14 instead of Lemma 4.

4.2 \vec{M} -extrapolation on PTAs with unbounded lower or upper bound parameters

The method described previously can be adapted to a subclass of PTAs that can be turned into L-PTAs or U-PTAs (only) for the sake of computing the constant bound \widehat{N} . Let us first define this subclass:

Definition 12 (bPTA+L and bPTA+U). *Let \mathcal{A} be a PTA. \mathcal{A} is a bounded PTA with unbounded lower-(resp. upper-)bound parameters, or bPTA+L (resp. bPTA+U), if for each guard $x \bowtie \sum_{1 \leq i \leq K} \alpha_i p_i + z$ of $\mathbb{G}(\mathcal{A})$, for all i :*

- $\mathbb{D}(p_i) \in \mathbb{Q} \times \mathbb{Q}$ (i.e., p_i is bounded), or
- $\alpha_i = 0$, or
- $\alpha_i > 0$ and $\bowtie \in \{\geq, >\}$ (respectively $\bowtie \in \{<, \leq\}$), or

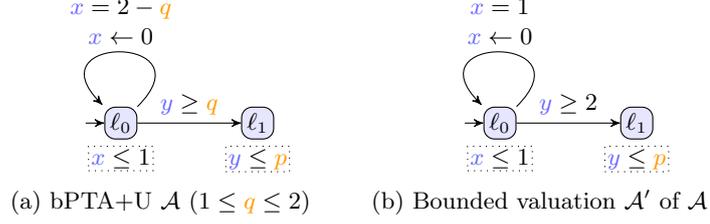


Fig. 6: A bPTA+U and its bounded valuation.

– $\alpha_i < 0$ and $\bowtie \in \{<, \leq\}$ (respectively $\bowtie \in \{\geq, >\}$).

Let \mathcal{A} be a bPTA+L (resp. bPTA+U). We denote by $\overline{\mathcal{A}}$ the L-PTA (resp. U-PTA) obtained from \mathcal{A} by valuating the bounded parameters as follows: we replace each bounded parameter p_i within a guard or invariant with its lower bound $\mathbb{D}^-(p_i)$ if it appears negatively ($\alpha_i < 0$) or with its upper bound $\mathbb{D}^+(p_i)$ otherwise. Clearly, if \mathcal{A} is a bPTA+L (resp. bPTA+U) then $\overline{\mathcal{A}}$ is an L-PTA (resp. U-PTA).

We first valuate bounded parameters to turn a bPTA+L (resp. bPTA+U) \mathcal{A} into an L-PTA (resp. U-PTA). This is obtained by transforming \mathcal{A} such that, in every guard and invariant, any bounded parameter of positive coefficient α_i is replaced with its upper bound and any bounded parameter of negative coefficient α_i with its lower bound.

Definition 13 (Bounded valuation of a bPTA+L or bPTA+U). Let \mathcal{A} be a bPTA+L (resp. bPTA+U). Let $\overline{\mathcal{A}}$ be the modification of \mathcal{A} where for each guard $x \bowtie \sum_{1 \leq i \leq K} \alpha_i p_i + z \in \mathbb{G}(\mathcal{A})$, for each bounded $p_i \in \mathbb{P}$,

1. if $\alpha_i < 0$, p_i is replaced by $\mathbb{D}^-(p_i)$,
2. if $\alpha_i > 0$, p_i is replaced by $\mathbb{D}^+(p_i)$, and
3. p_i is replaced with 0 otherwise.

Example 7. To illustrate Definition 12 we modify Fig. 4a by adding a bounded parameter. Fig. 6a is a bPTA+U \mathcal{A} with q bounded between 1 and 2, and p unbounded. Fig. 6b is the bounded valuation \mathcal{A}' of \mathcal{A} , as defined in Definition 13. Note that in this example \mathcal{A}' does not describe a behavior that belongs to \mathcal{A} , as parameter q is valuated to 1 in the guard where it occurs with a negative sign, while it is valuated to 2 in the guard where it occurs with a positive sign. It will nevertheless be useful to determine a constant bound for \mathcal{A} .

Correctness of the transformation Trivially, we get that the PTA $\overline{\mathcal{A}}$ is an L-PTA (or U-PTA).

Lemma 15. Let \mathcal{A} be a bPTA+L (resp. bPTA+U). Then $\overline{\mathcal{A}}$ is an L-PTA (resp. U-PTA).

Proof. Assume \mathcal{A} is a bPTA+L (resp. bPTA+U). When building $\overline{\mathcal{A}}$, any occurrence of a bounded parameter is replaced by its constant bounds. In addition, all unbounded parameters from \mathcal{A} are, by [Definition 12](#), lower-bound (resp. upper-bound) parameters. Therefore, the only remaining parameters in $\overline{\mathcal{A}}$ are lower-bound (resp. upper-bound) parameters. Therefore, $\overline{\mathcal{A}}$ is an L-PTA (resp. U-PTA).

Method Our method is then as follows: given a bPTA+L (resp. bPTA+U) \mathcal{A} ,

1. we construct the L-PTA (resp. U-PTA) $\overline{\mathcal{A}}$, and
2. we then compute the bound \widehat{N} on the obtained L-PTA (resp. U-PTA) $\overline{\mathcal{A}}$ (using the technique given in [Section 4.1](#)).

Let \overline{N} denote this result.

Let $\overline{M} = \{M(x_1), \dots, M(x_H)\}$ such that $M(x_i)$ is the maximal constant of clock x_i when bounding in \mathcal{A} all unbounded parameters with \overline{N} . Let $\overline{\text{EEF}}$ denote the modification of EEF where $\text{Ext}_{\mathbb{X}}^M$ is replaced with $\text{Ext}_{\mathbb{X}}^{\overline{M}}$ ([line 6](#) in [Algorithm 1](#)). That is, instead of computing the Ext^M -extrapolation of each symbolic state, we compute its $\text{Ext}^{\overline{M}}$ -extrapolation, where \overline{M} was obtained using the \overline{N} computed on the L-PTA (or U-PTA) when valuating the bounded parameters with their bounds.

Correctness

Proposition 3. *Let \mathcal{A} be a bPTA+L or bPTA+U with initial symbolic state \mathbf{s}_0 , and $T \subseteq L$ a set of target locations. Assume $\overline{\text{EEF}}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$ terminates. We have:*

1. *Soundness: If $v \in \overline{\text{EEF}}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$ then T is reachable in $v(\mathcal{A})$;*
2. *Completeness: For all v , if T is reachable in $v(\mathcal{A})$ then $v \in \overline{\text{EEF}}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$.*

Lemma 16. *The bounded valuation $\overline{\mathcal{A}}$ of a PTA \mathcal{A} guarantees for each constraint in the model to give the greatest possible constant bound for all valuations in the set of bounded parameters of \mathcal{A} .*

Proof. In any given guard, as each upper bounded parameter of positive sign is set to its upper bound and each lower bounded parameter of negative sign is set to its lower bound, there can be no other valuation of bounded parameters such that any guard or invariant displays a greater constant part.

Note that $\overline{\mathcal{A}}$ might not even be in the set of PTA obtained when setting values for bounded parameters, as it is possible that a given parameter is replaced by its lower bound in some guard, and by its upper bound in some other. It guarantees, however, that the value of the constant bound for any of the PTA obtained by valuating bounded parameters is no greater than \overline{N} .

We can proceed with the proof of [Proposition 3](#):

Proof. Let \mathcal{A}' be any bounded valuation of \mathcal{A} . By definition, \mathcal{A}' is either an L-PTA or a U-PTA. From [Lemma 16](#), we know that \bar{N} is greater than the constant bound of \mathcal{A}' . By [Proposition 2](#), we know that the extrapolation of \mathcal{A}' is sound and complete when defining $M(x)$ as the maximal constant of clock x when bounding all unbounded parameters with \hat{N} . As $\bar{N} > \hat{N}$, the extrapolation is still sound and complete for any bounded valuation of \mathcal{A} .

4.3 Partial \vec{M} -extrapolation on general PTAs

Finally, it is possible to perform a *partial* extrapolation on any PTA \mathcal{A} , by extrapolating only the clocks that are only compared to the set of bounded parameters \mathbb{P}_{bound} of \mathcal{A} . That is, for a given guard or invariant g of the form $x \bowtie \sum_{1 \leq i \leq K} \alpha_i p_i + z$, the maximum value $C_{max}(g) = \sum_{1 \leq i \leq K} \alpha_i \gamma_i + z$ where

1. $\gamma_i = \mathbb{D}^-(p_i)$ if $\alpha_i < 0$,
2. $\gamma_i = \mathbb{D}^+(p_i)$ if $\alpha_i > 0$, and
3. $\gamma_i = 0$ otherwise.

Note that γ_i may be ∞ or $-\infty$ if p_i is not an unbounded parameter. As a result, the maximal constant of any clock $x_i \in \mathbb{X}$ compared to unbounded parameter is equal to ∞ . Therefore, $M(x_i) \in \vec{M} = \infty$ —which amounts to never applying extrapolation on x_i .

Let \mathbb{X}_b denote the set of clocks compared to no unbounded parameter (i.e., compared in guards and invariants only to constants or bounded parameters).

Let $\vec{M}_b = \{M(x_1), \dots, M(x_H)\}$ such that $M(x_i)$ is the maximal constant of clock x_i (i.e., ∞ if $x_i \notin \mathbb{X}_b$). Let $\text{Ext}_{\mathbb{X}_b}^M$ denote the composition (in any order) of all $\text{Ext}_x^{M(x)}$, for all $x \in \mathbb{X}_b$. Let pEEF (“p” stands for “partial”) denote the modification of EEF where $\text{Ext}_{\mathbb{X}}^M$ is replaced with $\text{Ext}_{\mathbb{X}_b}^M$ ([line 6](#) in [Algorithm 1](#)).

Proposition 4. *Let \mathcal{A} be a PTA with initial symbolic state \mathbf{s}_0 , and $T \subseteq L$ a set of target locations. Assume $\text{pEEF}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$ terminates. We have:*

1. *Soundness: If $v \in \text{pEEF}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$ then T is reachable in $v(\mathcal{A})$;*
2. *Completeness: For all v , if T is reachable in $v(\mathcal{A})$ then $v \in \text{pEEF}(\mathcal{A}, \mathbf{s}_0, T, \emptyset)$.*

Proof. The proof is the same as for [Proposition 1](#).

Example 8. In [Fig. 4b](#) (with p being unbounded), which is a variation of [Fig. 1a](#) where p is now equal to x in the transition to ℓ_1 , x is compared to the unbounded parameter p which is neither a lower bound nor an upper bound parameter. Therefore, this PTA is not in any of the previous classes on which it is possible to compute a constant bound. However, we can apply a partial extrapolation, i.e., the extrapolation is only applied on y , for which there exists a maximal constant $C_{max}^y(\mathcal{A}) < \infty$. The analysis using `IMITATOR` returns quickly (in < 0.1 s) the expected result $p \in [0, 1]$, while it cannot be solved with a standard exploration (i.e., the algorithm would not terminate).

Of course, we have even less guarantee of termination in the case where only some clocks are extrapolated, but this can still help termination when comparing to the case without any extrapolation.

5 Beyond reachability in bPTA+L and bPTA+U

We saw in Section 4 that it was possible to apply extrapolation on unbounded PTAs, thanks to a result from [BL09], notably unbounded L-PTAs and U-PTAs with additional bounded parameters. However, we only proved correctness of this method for reachability properties. In this section, we study liveness and trace preservation properties.

5.1 Liveness

In the context of unbounded parameters, the \widehat{M} -extrapolation cannot be used directly to check liveness properties, as it might produce false positives. The U-PTA in Fig. 4c exemplifies why the parametric extrapolation is not correct for cycle synthesis on unbounded PTAs. With this automaton, the state space is infinite with y growing without bound: after i loops, we have $y = x + i \leq p$. The expected result of a cycle synthesis is \perp (no valuation yields a cycle), but an exploration of the state space would not terminate. If we try applying the \widehat{M} -extrapolation, we obtain $M(x) = 1$ and $M(y) = 522$ as greatest constants, computed using \widehat{N} (Section 4.1). After 522 loops, the valuation of y can be greater than $M(y)$, and we obtain a self-looping state where $y > 522$ and $p > 523$. As a result, the \widehat{M} -extrapolation will synthesize a cycle for $p > 523$, while there should be none. This behavior is due to the invariant $y \leq p$ being removed by the cylindrification of clock y . Note that this is not possible with bounded parameters (or general TAs) because any invariant $y \leq t$, with t a given constant, would necessarily contradict the constraint $y > M$. Indeed, M being by definition the greatest constant of clock y , $M \geq t$ and thus $y > M \cap y \leq t = \emptyset$.

Observe that the model in Fig. 4c is a U-PTA. From [BL09, Theorem 6], we know that there exists a maximal constant (similar to our \widehat{N} computed in Section 4.1) such that there exists no accepting cycle for any parameter valuation whenever the TA obtained from the U-PTA by valuating its parameters with \widehat{N} yields no accepting cycle. This is not a contradiction with our example: in our method, we do not only use \widehat{N} to value parameters, but we also apply extrapolation, which involves cylindrification (Definition 8). This is the cylindrification operator which is responsible for the incorrectness of the extrapolation.

A solution to fix that issue is to ensure the invariant is not ignored, by bounding p by the constant \widehat{N} (522 in this case). In general, bounding all parameters by \widehat{N} ensures no false positive are present, but might include false negative in the form of upper bounds (those we introduced to bound the parameters). However, we know from [BL09, Theorems 3 and 6] that in an L-PTA or a U-PTA, if there is an infinite accepting run for a parameter valuation v with $v(p) \geq \widehat{N}$, then this run exists for all valuations v' with $v'(p) \geq \widehat{N}$. Therefore, in a U-PTA, the upper bound on p can be removed on any results that contains “ $p = \widehat{N}$ ”. This method can be applied on the classes of models on which we have defined a extrapolation using the constant bound \widehat{N} (i.e., bPTA+L and bPTA+U).

In the case of our example from Fig. 4c, this means constraining the model with $p \leq 522$. As a result, the \widehat{M} -extrapolation will synthesize no cycles, which is correct. Now, imagine a model with the same constant bound over parameter $\widehat{N} = 522$, but such that the expected result is $400 < p$. The \widehat{M} -extrapolation on the constrained model will synthesize $400 < p \leq 522$ —which contains $p = 522$. We can then remove the upper bound on p and obtain the correct result, i.e., $400 < p$.

6 Experiments

We implemented all aforementioned extrapolations in IMITATOR [And21]; all operations on parametric zones are computed by polyhedral operations, using PPL [BMZ08]. We consider the full class of PTAs, over (potentially unbounded) rational-valued parameters. We applied the extrapolation on the bPTA+L/bPTA+U subclass from Section 4.2 when it was possible, and the partial \vec{M} -extrapolation from Section 4.3 otherwise (i.e., extrapolation is applied to each clock whenever possible), to a library of standard PTA benchmarks [AMP21]. Experiments were performed using an Intel Core i5-4690K with a clock rate of 4 GHz.²

We tabulate our results in Table 1. The first and main outcome is the two lines for “all models” (in bold): on the entire benchmark set (119 models and 177 properties), the average execution time is 954s without extrapolation, and 824s with; in addition, the normalized average (always taking 1 for the slowest of both algorithms and rescaling the second one accordingly) is 0.89 without and 0.91 with. Both metrics are complementary, as the average favors models with large verification times, while normalized average gives the same weight to all models, including those of very small verification times. The outcome is that the extrapolation decreases the average time by 14%, and increases the normalized average time by 1.5%, which remains near-to-negligible. On the larger models (> 5s), extrapolation allows for a very similar decrease of 14% in average, and even a small decrease of 0.6% for the normalized average time.

We only tabulate in Table 1 results with the most significant difference, i.e., with a gap of more than 1s with a ratio $\frac{\min}{\max} > 2$ (and only one property per model). Put it differently, other models show little difference between both versions. “**reach**” denotes reachability synthesis; “**liveness**” denotes the synthesis of valuations leading to at least one infinite run.

Recall that, even on the most restrictive syntactic subclass of PTAs we considered (L-PTAs and U-PTAs), synthesis over rational-valued parameters is intractable, and therefore our algorithms (including with extrapolation) come with no guarantee of termination. On the entire benchmarks set, 39 properties (over 33 models) do not terminate without extrapolation; this figure reduces

² Source, benchmarks, raw results and full table are available on the long-term archiving platform Zenodo at doi.org/10.5281/zenodo.5824264. We used a fork of IMITATOR 3.1 “Cheese Artichoke” extended with extrapolation functions (exact version: **v3.1.0+extrapolation**).

Table 1: Execution times (in seconds) for our experiments. T.O. denotes an execution unfinished after 3,600 seconds. (We therefore use this value for means computation.) Normalized mean is the ratio to the worst execution times. Cells color represents the difference in performance for a given row: the lighter the better.

| Model | Property | No extrapolation (s) | M -extrapolation (s) |
|--|--------------|----------------------|------------------------|
| FischerPS08-4 | safety | 10.6 | 4.8 |
| FMTV_2 | reach | 0.7 | 2.3 |
| fischerPAT3 | safety | 1.9 | 0.8 |
| SLAF14.5 | safety | 12.6 | 74.4 |
| spsmall | safety | 0.4 | 19.3 |
| SSLAF13_test2 | safety | 2869.8 | 1399.1 |
| synthRplus | reach | T.O. | 0.2 |
| Cycle1 | liveness | T.O. | 0.001 |
| infinite-5 | liveness | T.O. | 0.006 |
| infinite-5.6 | liveness | T.O. | 0.004 |
| exUnoloop | acc liveness | 1.1 | 7.7 |
| Mean (models from Table 1 only) | | 1572.5 | 137.1 |
| Normalized mean (models from Table 1 only) | | 0.697 | 0.490 |
| Mean (all models) | | 954.4 | 823.8 |
| Normalized mean (all models) | | 0.891 | 0.905 |

to 33 properties (over 29 models) when applying extrapolation. (No analysis terminating without extrapolation would lead to non-termination when adding extrapolation.)

On the models where there is a significant difference between with and without extrapolation, tabulated in Table 1, the extrapolation is sometimes significantly faster, sometimes significantly slower. Most importantly, extrapolation allows termination of some so far unsolvable models. The slower cases are due to the fact that our implementation in IMITATOR needs to keep each symbolic state *convex*—this is required by the internal polyhedral structure. Therefore, when a clock is extrapolated, this increases the number of states in the state space (a given extrapolated symbolic state can be potentially split into up to $2^{|\mathbb{X}|}$ new symbolic states via a single outgoing transition).

These experiments highlight the main drawback of our implementation, that is, extrapolated symbolic states have to be split into convex sub-states, sometimes ended up doing more computation in the process than without any extrapolation. (We will discuss it in the conclusion.) Despite that drawback, the extrapolation can still significantly decrease computation time. Furthermore, a main benefit of our extrapolation is that it can lead to a better termination, allowing to turn infinite state spaces into finite ones; this allows us to solve previously unsolvable benchmarks (`synthRplus`, `Cycle1`, `infinite-5`, `infinite-5.6`).

All in all, our experiments suggest that, despite a few models (tabulated in Table 1) where the presence or absence of extrapolation has a significant difference of execution time, adding extrapolation remains overall harmless, with even an average decrease of 14% in the execution time. Most importantly, it allows to solve so far unsolvable benchmarks—which we consider as the main

outcome. This suggests to use extrapolation by default for parameter synthesis in PTAs using IMITATOR.

7 Conclusion and perspectives

7.1 Conclusion

In this paper, we proposed several definitions of zone extrapolation for parametric timed automata. We notably improve the parametric M -extrapolation from [ALR15] by allowing each clock to have its own bound and combining it with results from [BL09] in order to address unbounded subclasses of PTAs. We proposed a first implementation (in IMITATOR), and showed that, while extrapolation is harmless for most models, it can also decrease the computation time of larger models and, most importantly, can lead to termination (with exact synthesis) of previously unsolvable benchmarks. Considering the difficulty of parameter synthesis for timed models, we consider it a non-trivial and promising step.

7.2 Future works

We now discuss future works.

A main limitation of our implementation in IMITATOR (discussed in Section 6) is that it only handles *convex* parametric zones. Using the non-convex polyhedral structures offered by PPL [BMZ08] may dramatically reduce the number of symbolic states. However, they are much more costly than their convex counterparts—this should be experimentally compared.

Another perspective on implementation concerns the computation of the constant bounds \widehat{N} , for which one needs to compute the number R of clock regions. Our current implementation uses its over-approximation \widehat{R} . Computing the actual number of clock regions before applying the extrapolation may considerably reduce the analysis time for larger models.

The main limitation of parametric extrapolation is that termination of synthesis for PTAs cannot be guaranteed, even for bounded PTAs. Although the motivation behind extrapolation is to replace infinite sequences by cycles, this is not possible for parameters converging towards a constant. A perspective would be to exhibit a subclass of PTAs for which it is possible to extrapolate on parameters themselves the constant towards which they converge.

Finally, we plan to go beyond this work by adapting the LU -extrapolation from [Beh+06] to PTAs, a theoretically coarser abstraction for which implementation is not trivial. Algorithms from [HSW16] may prove useful to this purpose.

Acknowledgements

The authors would like to thank the reviewers for their comments, and Dylan Marinho for his help in providing the models and automation tools that were used for the benchmarking presented in this paper.

References

- [AD94] Rajeev Alur and David L. Dill. “A theory of timed automata”. In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975. DOI: [10.1016/0304-3975\(94\)90010-8](https://doi.org/10.1016/0304-3975(94)90010-8) (cit. on pp. 1, 5, 15).
- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. “Parametric real-time reasoning”. In: *STOC* (May 16–18, 1993). Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, United States: ACM, 1993, pp. 592–601. DOI: [10.1145/167088.167242](https://doi.org/10.1145/167088.167242) (cit. on pp. 1, 2, 4).
- [ALR15] Étienne André, Didier Lime, and Olivier H. Roux. “Integer-Complete Synthesis for Bounded Parametric Timed Automata”. In: *RP* (Sept. 21–23, 2015). Ed. by Mikołaj Bojańczyk, Sławomir Lasota, and Igor Potapov. Vol. 9328. LNCS. Warsaw, Poland: Springer, Sept. 2015, pp. 7–19. DOI: [10.1007/978-3-319-24537-9](https://doi.org/10.1007/978-3-319-24537-9) (cit. on pp. 3, 7–12, 24).
- [ALR18] Étienne André, Didier Lime, and Mathias Ramparison. “TCTL model checking lower/upper-bound parametric timed automata without invariants”. In: *FORMATS* (Sept. 4–6, 2018). Ed. by David N. Jansen and Pavithra Prabhakar. Vol. 11022. Lecture Notes in Computer Science. Beijing, China: Springer, 2018, pp. 1–17. DOI: [10.1007/978-3-030-00151-3_3](https://doi.org/10.1007/978-3-030-00151-3_3) (cit. on p. 3).
- [ALR21] Étienne André, Didier Lime, and Mathias Ramparison. “Parametric updates in parametric timed automata”. In: *Logical Methods in Computer Science* 17.2 (May 2021), 13:1–13:67. DOI: [10.23638/LMCS-17\(2:13\)2021](https://doi.org/10.23638/LMCS-17(2:13)2021) (cit. on p. 2).
- [AMP21] Étienne André, Dylan Marinho, and Jaco van de Pol. “A Benchmarks Library for Extended Timed Automata”. In: *TAP* (June 21–25, 2021). Ed. by Frédéric Loulergue and Franz Wotawa. Vol. 12740. Lecture Notes in Computer Science. virtual: Springer, 2021, pp. 39–50. DOI: [10.1007/978-3-030-79379-1_3](https://doi.org/10.1007/978-3-030-79379-1_3) (cit. on p. 22).
- [And+09] Étienne André, Thomas Chatain, Emmanuelle Encrenaz, and Laurent Fribourg. “An Inverse Method for Parametric Timed Automata”. In: *International Journal of Foundations of Computer Science* 20.5 (Oct. 2009), pp. 819–836. DOI: [10.1142/S0129054109006905](https://doi.org/10.1142/S0129054109006905) (cit. on p. 6).
- [And19] Étienne André. “What’s decidable about parametric timed automata?”. In: *International Journal on Software Tools for Technology Transfer* 21.2 (Apr. 2019), pp. 203–219. DOI: [10.1007/s10009-017-0467-0](https://doi.org/10.1007/s10009-017-0467-0) (cit. on p. 2).
- [And21] Étienne André. “IMITATOR 3: Synthesis of timing parameters beyond decidability”. In: *CAV* (July 18–23, 2021). Ed. by Rustan Leino and Alexandra Silva. Vol. 12759. Lecture Notes in Computer Science. virtual: Springer, 2021, pp. 1–14. DOI: [10.1007/978-3-030-81685-8_26](https://doi.org/10.1007/978-3-030-81685-8_26) (cit. on pp. 3, 22).
- [Beh+03] Gerd Behrmann, Patricia Bouyer, Emmanuel Fleury, and Kim Guldstrand Larsen. “Static Guard Analysis in Timed Automata Verification”. In: *TACAS* (Apr. 7–11, 2003). Ed. by Hubert Garavel and John Hatcliff. Vol. 2619. Lecture Notes in Computer Science. Warsaw, Poland: Springer, 2003, pp. 254–277. DOI: [10.1007/3-540-36577-X_18](https://doi.org/10.1007/3-540-36577-X_18) (cit. on p. 2).
- [Beh+06] Gerd Behrmann, Patricia Bouyer, Kim Guldstrand Larsen, and Radek Pelánek. “Lower and upper bounds in zone-based abstractions of timed

- automata”. In: *International Journal on Software Tools for Technology Transfer* 8.3 (2006), pp. 204–215. DOI: [10.1007/s10009-005-0190-0](https://doi.org/10.1007/s10009-005-0190-0) (cit. on pp. 2, 7, 8, 12, 24).
- [Ben+15] Nikola Beneš, Peter Bezděk, Kim Gulstrand Larsen, and Jiří Srba. “Language Emptiness of Continuous-Time Parametric Timed Automata”. In: *ICALP, Part II* (July 6–10, 2015). Ed. by Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann. Vol. 9135. Lecture Notes in Computer Science. Kyoto, Japan: Springer, July 2015, pp. 69–81. DOI: [10.1007/978-3-662-47666-6_6](https://doi.org/10.1007/978-3-662-47666-6_6) (cit. on p. 2).
- [Bez+16] Peter Bezděk, Nikola Beneš, Jiří Barnat, and Ivana Černá. “LTL Parameter Synthesis of Parametric Timed Automata”. In: *SEFM* (July 4–8, 2016). Ed. by Rocco De Nicola and Eva Kühn. Vol. 9763. Lecture Notes in Computer Science. Vienna, Austria: Springer, 2016, pp. 172–187. DOI: [10.1007/978-3-319-41591-8_12](https://doi.org/10.1007/978-3-319-41591-8_12) (cit. on p. 3).
- [BL09] Laura Bozzelli and Salvatore La Torre. “Decision problems for lower/upper bound parametric timed automata”. In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151. DOI: [10.1007/s10703-009-0074-0](https://doi.org/10.1007/s10703-009-0074-0) (cit. on pp. 2, 3, 14, 15, 21, 24).
- [BMZ08] Roberto Bagnara, Hill Patricia M., and Enea Zaffanella. “The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems”. In: *Science of Computer Programming* 72.1–2 (2008), pp. 3–21. DOI: [10.1016/j.scico.2007.08.001](https://doi.org/10.1016/j.scico.2007.08.001) (cit. on pp. 22, 24).
- [BO17] Daniel Bundala and Joël Ouaknine. “On parametric timed automata and one-counter machines”. In: *Information and Computation* 253 (2017), pp. 272–303. DOI: [10.1016/j.ic.2016.07.011](https://doi.org/10.1016/j.ic.2016.07.011) (cit. on p. 2).
- [DT98] Conrado Daws and Stavros Tripakis. “Model Checking of Real-Time Reachability Properties Using Abstractions”. In: *TACAS* (Mar. 28–Apr. 4, 1998). Ed. by Bernhard Steffen. Vol. 1384. Lecture Notes in Computer Science. Lisbon, Portugal: Springer, 1998, pp. 313–329. DOI: [10.1007/BFb0054180](https://doi.org/10.1007/BFb0054180) (cit. on p. 2).
- [HSW16] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. “Better abstractions for timed automata”. In: *Information and Computation* 251 (2016), pp. 67–90. DOI: [10.1016/j.ic.2016.07.004](https://doi.org/10.1016/j.ic.2016.07.004) (cit. on p. 24).
- [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. “Linear parametric model checking of timed automata”. In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220. DOI: [10.1016/S1567-8326\(02\)00037-1](https://doi.org/10.1016/S1567-8326(02)00037-1) (cit. on pp. 2, 3, 6).
- [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. “Integer Parameter Synthesis for Real-Time Systems”. In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461. DOI: [10.1109/TSE.2014.2357445](https://doi.org/10.1109/TSE.2014.2357445) (cit. on pp. 2, 3, 9–11).
- [Li09] Guangyuan Li. “Checking Timed Büchi Automata Emptiness Using LU-Abstractions”. In: *FORMATS* (Sept. 14–16, 2009). Ed. by Joël Ouaknine and Frits W. Vaandrager. Vol. 5813. Lecture Notes in Computer Science. Budapest, Hungary: Springer, 2009, pp. 228–242. DOI: [10.1007/978-3-642-04368-0_18](https://doi.org/10.1007/978-3-642-04368-0_18) (cit. on p. 2).
- [LPY97] Kim Gulstrand Larsen, Paul Pettersson, and Wang Yi. “UPPAAL in a Nutshell”. In: *International Journal on Software Tools for Technology*

- Transfer* 1.1-2 (1997), pp. 134–152. DOI: [10.1007/s100090050010](https://doi.org/10.1007/s100090050010) (cit. on p. 2).
- [Sch86] Alexander Schrijver. *Theory of linear and integer programming*. New York, NY, USA: John Wiley & Sons, Inc., 1986 (cit. on p. 4).
- [Tri09] Stavros Tripakis. “Checking timed Büchi automata emptiness on simulation graphs”. In: *ACM Transactions on Computational Logic* 10.3 (2009), 15:1–15:19. DOI: [10.1145/1507244.1507245](https://doi.org/10.1145/1507244.1507245) (cit. on p. 2).