Logical Methods in Computer Science
Volume 22, Issue 2, 2026, pp. 11:1–11:32
https://lmcs.episciences.org/

Submitted Oct. 16, 2023
Published Apr. 29, 2026

DENSE INTEGER-COMPLETE SYNTHESIS FOR BOUNDED PARAMETRIC TIMED AUTOMATA

ÉTIENNE ANDRÉ [a,b], DIDIER LIME [c], AND OLIVIER H. ROUX [c]

[a] Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, F-93430 Villetaneuse, France

[b] Institut Universitaire de France (IUF)

[c] Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes, France

Abstract. Ensuring the correctness of critical real-time systems, involving concurrent behaviours and timing requirements, is crucial. Timed automata extend finite-state automata with clocks, compared in guards and invariants with integer constants. Parametric timed automata (PTAs) extend timed automata with timing parameters. Parameter synthesis aims at computing dense sets of valuations for the timing parameters, guaranteeing a good behaviour. However, in most cases, the emptiness problem for reachability (*i.e.,* the emptiness of the parameter valuations set for which some location is reachable) is undecidable for PTAs 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 symbolic sets of valuations containing not only all the integer points ensuring reachability, but also all the (non-necessarily integer) convex combinations of these integer points, for general PTAs with a bounded parameter domain. We also propose two further algorithms synthesizing parameter valuations guaranteeing unavoidability, and preservation of the untimed behaviour w.r.t. a reference parameter valuation, respectively. Our algorithms terminate and can output sets of valuations arbitrarily close to the complete result. We demonstrate their applicability and efficiency using the tools Roméo and IMITATOR on several benchmarks.

1. Introduction

The verification of software or hardware systems mixing time and concurrency is a notoriously difficult problem. Timed automata (TAs) [AD94] are a powerful formalism for which many interesting problems (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] overcome 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 (see [And19] for a survey).

This is an extended version of the paper by the same authors published in the proceedings of the 9th International Workshop on Reachability Problems (RP 2015) [ALR15]. This work has been supported by the ANR-NRF French-Singaporean research program ProMiS (ANR-19-CE25-0015 / 2019 ANR NRF 0092) and ANR BisoUS (ANR-22-CE48-0012).

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.46298/LMCS-22(2:11)2026

Synthesis for PTAs aims at deriving set of valuations for which a given property (such as reachability or unavoidability) holds. Ideally, this set of parameter valuations comes in a *symbolic* form, *e.g.,* using a finite union of polyhedra representing *dense* sets of valuations. This density can be useful notably in the context of robustness or implementability of a system (see, *e.g.,* [BMS13]). The goal of this work is to achieve synthesis of *dense* sets of parameter valuations, while guaranteeing the termination of our algorithms.

1.1. **Related work.** The simple problem of the emptiness of the valuations set such that some location of a PTA is reachable ("reachability-emptiness") is undecidable in both discrete and dense-time settings [AHV93], 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 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, GH21], 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, BDR08].

Subclasses of PTAs have been studied, most notably L/U-PTAs, in which the parameters are partitioned into parameters that can only be compared to a clock as an upper bound, or as a lower bound. The reachability-emptiness problem is decidable for L/U-PTAs [HRSV02], but no synthesis algorithm is provided, and there are indeed practical difficulties in proposing one [JLR15]. When further restricting to L- or U-PTAs (with only lower-bound, resp. upper-bound parameters), the integer-valued parameters can be synthesised [BLT09]. However, the full TCTL-emptiness problem was proved to be undecidable even for the restricted class of U-PTAs [ALR18].

In [ACEF09], the inverse method takes advantage of a reference parameter valuation. When it terminates, this algorithm outputs a (possibly underapproximated) set of parameter valuations for which the discrete behaviour (set of traces seen as alternating sequences of locations and actions) is exactly the same as the one of the reference valuation. The traces preservation problem (*i.e.,* given a reference parameter valuation whether there exists another valuation for which the discrete behaviour is the same) is undecidable for both general PTAs and L/U-PTAs [ALM20].

In [JLR15], the focus is on integer-valued bounded parameters (but still considering dense-time), for which many problems are obviously decidable, and symbolic algorithms are provided to compute the set of correct integer parameter valuations ensuring a reachability ("IEF") or unavoidability ("IAF") property. A drawback is that returning only integer points prevents designers to use the synthesised constraint to study the robustness or implementability of their system. Control of real-time systems with integer-valued bounded parameters was then studied in [JLR22] using similar techniques.

Finally note that an extrapolation similar to ours [ALR15] was later proposed in [BBBČ16].

1.2. **Contribution.** We propose here a terminating algorithm that computes a dense under-approximation of the set of parameter valuations ensuring reachability in bounded PTAs (*i.e.,* PTAs with a bounded parameter domain). We also propose two further algorithms ensuring unavoidability and preservation of the untimed behaviour w.r.t. a reference parameter valuation, respectively. These under-approximations are "integer-complete" in the sense

that they are guaranteed to contain at least all the correct integer valuations given in the form of a finite union of polyhedra; this is to say, only some non-integer (rational) points may be missing. To the best of our knowledge, these algorithms are the first synthesis algorithms that return a dense integer-complete result for a subclass of PTAs (namely bounded PTAs) for which the corresponding emptiness problems are undecidable; in fact, we are not aware of any terminating synthesis algorithm with a dense result with a partial completeness guarantee. Note that the synthesis algorithms for two subclasses of L/U-PTAs (namely L-PTAs and U-PTAs) [BLT09] are complete, but the setting is restricted to integer valuations.

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 valuations, interpreting the result of the IAF algorithm as dense is not correct in the sense that some non-integer parameter valuations in that result may not ensure the unavoidability property. Furthermore, since we produce rational-valued parameter valuations, we cannot use anymore the result from [JLR15] ensuring termination of the algorithms, which allows to derive a bound on clock valuations but relies on the parameters being bounded integers. One of the main *technical* contributions of this paper is therefore the derivation of a maximum-constant-based parametric extrapolation operator for bounded PTAs 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 three algorithms for reachability, unavoidability and traces preservation, and we report on them.

1.3. **Comparison with** [ALR15]**.** This manuscript is a significant extension of [ALR15]. In addition to the inclusion of all proofs (missing in [ALR15]) and of additional examples, we rephrased most definitions and explanations; we also added the new section on traces preservation (Section 4.3). We also report on experiments in the new Sections 5 and 6.

1.4. **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, RITP) in Section 4. We report on our implementations, and we relate experiments on several benchmarks using Roméo (to evaluate RIEF and RIAF) in Section 5, and using IMITATOR (to evaluate RITP) in Section 6. We conclude in Section 7.

2. Preliminaries

2.1. **Clocks, parameters and constraints.** Let $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}_+$, $\mathbb{R}_+$, $\mathbb{R}$ denote the sets of non-negative integers, integers, non-negative rationals, non-negative reals and reals respectively.

Let $\mathbb{D} \subseteq \mathbb{R}$. Given a finite variables set V, a $\mathbb{D}$-valuation over V is a function from V to $\mathbb{D}$.

2.1.1. *Constraints and convex polyhedra.* In the following, we assume $\bowtie \in \{<, \leq, \geq, >\}$.

A *constraint* CS over a set of variables V is a conjunction of inequalities of the form $lt \bowtie 0$, where lt denotes a linear term over V of the form $\sum_{1 \leq i \leq |V|} \alpha_i \nu_i + d$, with $\nu_i \in V$ and $\alpha_i, d \in \mathbb{Z}$.

A valuation η over V is a *solution* of a constraint CS if by replacing each occurrence of each variable ν in CS by $\eta(\nu)$ simplifies to the "true" Boolean value. Given a constraint CS over V, we denote by $[\![\mathsf{CS}]\!]$ the set of its solutions.

A set of valuations $\mathbf{C}$ is a *convex polyhedron*[1] if there exists a constraint CS such that $\mathbf{C} = [\![\mathsf{CS}]\!]$.

$\mathbf{C}$ is topologically closed if it is a solution to a conjunction of constraints using only non-strict inequalities.

By the Minkowski-Weyl theorem (see, *e.g.,* [Sch86]), topologically closed convex polyhedra can be equivalently defined using a set of vertices and extremal rays.

Let $\mathsf{Conv}(\mathbf{C})$ denote the convex hull of set $\mathbf{C}$ of valuations, that is $\mathsf{Conv}(\mathbf{C}) = \{\eta \mid \exists k, \exists \lambda_1, \ldots, \lambda_k \in [0,1], \exists \eta_1, \ldots, \eta_k \in \mathbf{C} \text{ s.t. } \eta = \sum_{i=1}^{k} \lambda_i \eta_i \wedge \sum_{i=1}^{k} \lambda_i = 1\}$.

Then for any convex polyhedron $\mathbf{C}$, there exist $\eta_1, \ldots, \eta_j \in \mathbf{C}$, $r_1, \ldots, r_k$ some $\mathbb{R}$-valuations, and $\mu_1, \ldots, \mu_k$ non-negative real numbers such that for all $\eta \in \mathbf{C}$, there exist $\eta' \in \mathsf{Conv}(\{\eta_1, \ldots, \eta_j\})$ and $\eta = \eta' + \sum_{i=1}^{k} \mu_i r_i$. Without loss of generality, the η_i can be uniquely chosen such that none of them belongs to the convex hull of the others, and they are then called the *vertices* of $\mathbf{C}$. Similarly, the r_i can be chosen, uniquely up to some multiplicative constant, such that none can be expressed as a non-negative combination of the others, and are then called the *extremal rays* of $\mathbf{C}$. Informally, the latter represent the limit directions in which the polyhedron is infinite.

Since we consider also strict inequalities, the above generator description is incomplete in general and we would also need to introduce *closure points* that play the role of vertices in the topological closure of the polyhedron. Their handling would be similar to what we do with vertices in the following. The reader is directed to [BHZ05] for more details, and in particular Theorem 4.4 therein for an equivalent to the Minkowski-Weyl theorem for non-necessarily closed polyhedra.

Also note that any non-closed polyhedron can be represented by a closed polyhedron with one extra dimension [HPR94].

In the sequel to keep the developments more readable we will assume all polyhedra are topologically closed and therefore deal only with vertices and rays.

2.1.2. *Clocks and parameters.* Throughout this paper, we assume a set $X = \{x_1, \ldots, x_H\}$ of *clocks*, *i.e.,* non-negative real-valued variables that evolve at the same rate. A *clock valuation* is an $\mathbb{R}_+$-valuation over X. We write $\vec{0}$ for the valuation such that for all $x \in X$, $\vec{0}(x) = 0$. 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. Given $d \in \mathbb{R}_+$, $w + d$ denotes the valuation such that $(w + d)(x) = w(x) + d$, for all $x \in X$. An *integer* clock valuation in an $\mathbb{N}$-valuation over X.

[1] Given an arbitrary order on V, a valuation over V can be seen as a real vector of size $|V|$ whose coordinates are the values associated to the variables and thus our convex polyhedra indeed represent subsets of $\mathbb{R}^{|V|}$ and the set of $\mathbb{R}$-valuations over V is a vector space over $\mathbb{R}$.

We assume a set $P = \{p_1, \ldots, p_M\}$ of *parameters*, *i.e.,* unknown constants. A *parameter valuation* v is a $\mathbb{Q}_+$-valuation over P. An *integer* parameter valuation is an $\mathbb{N}$-valuation over P.

2.1.3. *Simple clock constraints.* Let *plt* denote a *parametric linear term*, *i.e.,* a linear term over P. A *simple clock constraint* g is a constraint over $X \cup P$ defined by a conjunction of inequalities of the form $x \bowtie plt$, *i.e.,* $x \bowtie \sum_{1 \leq i \leq |P|} \alpha_i p_i + d$, with $x \in X$, $p_i \in P$, and $\alpha_i, d \in \mathbb{Z}$.

2.1.4. *Sets of valuations.* Given a set $\mathbf{C}$ of valuations over $X \cup P$, and given a parameter valuation v, $v(\mathbf{C})$ denotes the set of valuations over X obtained by replacing each parameter p in $\mathbf{C}$ with $v(p)$.

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, $w|v(p) = v(p)$.

An *integer point* is a valuation $w|v$, where w is an integer clock valuation, and v is an integer parameter valuation.

Given a set $\mathbf{C}$ of valuations over $X \cup P$, we define the *future* of $\mathbf{C}$, denoted by $\mathbf{C}^{\nearrow}$, as the set of valuations over $X \cup P$ obtained from $\mathbf{C}$ by delaying all clocks by an arbitrary amount of time: $\mathbf{C}^{\nearrow} = \{w'|v \mid \exists w, d : w \in v(\mathbf{C}) \wedge w' = w + d, d \in \mathbb{R}_+\}$. Dually, we define the *past* of $\mathbf{C}$, denoted by $\mathbf{C}^{\swarrow}$, as the set of valuations over $X \cup P$ obtained from $\mathbf{C}$ by letting time pass backward by an arbitrary amount of time: $\mathbf{C}^{\swarrow} = \{w'|v \mid \exists w, d : w \in v(\mathbf{C}) \wedge w' + d = w, d \in \mathbb{R}_+\}$.

Given $R \subseteq X$, we define the *reset* of $\mathbf{C}$, denoted by $[\mathbf{C}]_R$, as the set of valuations over $X \cup P$ obtained from $\mathbf{C}$ by resetting the clocks in R, and keeping the other clocks unchanged. That is, $w'|v \in [\mathbf{C}]_R$ iff $\exists w : X \to \mathbb{R}_+$ s.t. $w|v \in \mathbf{C}$ and $w' = [w]_R$.

We denote by $\mathbf{C}{\downarrow_P}$ the projection of $\mathbf{C}$ onto P, *i.e.,* the set of valuations over P defined as follows: $\{v \mid \exists w \text{ s.t. } w|v \in \mathbf{C}\}$.

Recall that, if $\mathbf{C}, \mathbf{C}'$ are convex polyhedra, then $\mathbf{C} \cap \mathbf{C}'$, $\mathbf{C}^{\nearrow}$, $\mathbf{C}^{\swarrow}$, $[\mathbf{C}]_R$ and $\mathbf{C}{\downarrow_P}$ are all convex polyhedra [JLR15, pp.6-7].

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 2.1. A PTA $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, L, \ell_0, X, P, PD, I, E)$, 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) X is a finite set of clocks,
(5) P is a finite set of parameters,
(6) $PD \subseteq \mathbb{Q}_+^P$ is the parameter domain, *i.e.,* the set of admissible parameter valuations,
(7) I is the invariant, assigning to every $\ell \in L$ a simple clock constraint $I(\ell)$,
(8) E is a set of edges $e = (\ell, g, a, R, \ell')$ where $\ell, \ell' \in L$ are the source and target locations, $a \in \Sigma$, $R \subseteq X$ is a set of clocks to be reset to 0, and g (the "guard") is a simple clock constraint.

Example 2.2. Consider the PTA in Fig. 1. It features 4 locations, 2 clocks and 1 parameter. The initial location is ℓ_1. The transition from ℓ_1 to ℓ_2 is guarded by "$x = 1$" and resets x to 0. This PTA has no invariant.

In this paper, we consider *bounded* PTAs. A *bounded parameter domain* is such that PD 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 $|P|$ dimensions.

Definition 2.3 (bounded PTA)**.** A bounded PTA is a PTA the parameter domain of which is bounded.

Remark 2.4. In this paper, our bounded parameter domains are simple closed intervals with integer bounds. However, we noted in [ALR22] that the nature of these intervals (open or closed) can have an impact on decidability, notably for some subclasses of PTAs [BLT09].

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 $v(p_i)$. In the following, we may denote as a *timed automaton (TA)* any such structure $v(\mathcal{A})$, by assuming a rescaling of the constants: by multiplying all constants in $v(\mathcal{A})$ by their least common denominator, we obtain an equivalent timed automaton (with integer constants, as in [AD94]).

2.2.1. *Concrete semantics.* Let us first recall the concrete semantics of TAs.

Definition 2.5 (Semantics of a TA)**.** Given a PTA $\mathcal{A} = (\Sigma, L, \ell_0, X, P, PD, I, E)$, and a parameter valuation v, the *concrete semantics* of $v(\mathcal{A})$ is given by the timed transition system $(Q, q_0, \to)$, with

- $Q = \{(\ell, w) \in L \times \mathbb{R}_+^{|X|} \mid w|v \in [\![I(\ell)]\!]\}$
- $q_0 = (\ell_0, \vec{0})$,
- delay transitions: $\big((\ell, w), d, (\ell, w + d)\big) \in \to$, with $d \in \mathbb{R}_+$, if $\forall d' \in [0, d], (\ell, w + d') \in Q$;
- discrete transitions: $\big((\ell, w), e, (\ell', w')\big) \in \to$ if $(\ell, w), (\ell', w') \in Q$, there exists $e = (\ell, g, a, R, \ell') \in E$, $w' = [w]_R$, and $w|v \in [\![g]\!]$.

We assume that $q_0 \in Q$, *i.e.,* $\vec{0}|v \in [\![I(\ell_0)]\!]$, *i.e.,* the initial clock valuation satisfies the invariant of the initial location.

We refer to the states of the concrete semantics of a TA as its *concrete states*. As usual, given two concrete states s, s', we write $s \xrightarrow{a} s'$, for $a \in \mathbb{R}_+ \cup E$, instead of $(s, a, s') \in \to$. We also further define relation $\rightarrowtail$ by $\big((\ell, w), e, (\ell', w')\big) \in \rightarrowtail$ if $\exists w'' \in \mathbb{R}_+^{|X|}, e \in E, d \in \mathbb{R}_+ : (\ell, w) \xrightarrow{d} (\ell, w'') \xrightarrow{e} (\ell', w')$. A concrete run ρ of a TA is an alternating sequence of concrete states of Q and edges of the form $s_0 \overset{e_0}{\rightarrowtail} s_1 \overset{e_1}{\rightarrowtail} \cdots \overset{e_{m-1}}{\rightarrowtail} s_m$, such that for all $i = 0, \ldots, m-1$, $e_i \in E$, and $(s_i, e_i, s_{i+1}) \in \rightarrowtail$. We say that states s_i, for $i = 0, \ldots, m$, *belong* to ρ. Given a concrete state $s = (\ell, w)$, 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, a location ℓ is reachable if there exists w such that (ℓ, w) is reachable. By extension again, a subset of locations $G \subseteq L$ is reachable if there exists $\ell \in G$ such that ℓ is reachable.

A *maximal* concrete run is a run that is either infinite, or that cannot be extended.

Given a concrete run $(\ell_0, w_0) \overset{e_0}{\rightarrowtail} (\ell_1, w_1) \overset{e_1}{\rightarrowtail} \cdots \overset{e_{m-1}}{\rightarrowtail} (\ell_m, w_m)$, its corresponding *trace* is $\ell_0 \overset{a_0}{\rightarrowtail} \ell_1 \overset{a_1}{\rightarrowtail} \cdots \overset{a_{m-1}}{\rightarrowtail} \ell_m$, where a_i is the action of e_i. The set of all traces of a TA $\mathcal{A}$ is called

its *trace set*, denoted by $\mathit{Traces}(\mathcal{A})$. That is, traces record both the locations and actions, but leave out the quantitative information such as the clock valuations and timing delays.

2.2.2. *Symbolic semantics.* We now recall the symbolic semantics of PTAs [HRSV02, ACEF09, JLR15, AMPvdP22]. Note that we usually use **bold** font to denote anything symbolic, *i.e.,* (sets of) symbolic states, and constraints.

Definition 2.6 (Symbolic state)**.** Given a PTA $\mathcal{A}$, a symbolic state of $\mathcal{A}$ is in the form of a pair $(\ell, \mathbf{C})$ where $\ell \in L$ is a location of $\mathcal{A}$, and $\mathbf{C}$ is a set of valuations.

Given a parameter valuation v, a symbolic state $\mathbf{s} = (\ell, \mathbf{C})$ is v-compatible if $v \in \mathbf{C}{\downarrow_P}$.

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

Definition 2.7 (Successor of a set of valuations)**.** Given a PTA $\mathcal{A} = (\Sigma, L, \ell_0, X, P, PD, I, E)$, given an edge $e = (\ell, g, a, R, \ell') \in E$, given a set $\mathbf{C}$ of valuations over $X \cup P$, we define

$$\mathbf{Succ}(\mathbf{C}, e) = \big([(\mathbf{C} \cap [\![g]\!])]_R \cap [\![I(\ell')]\!]\big)^{\nearrow} \cap [\![I(\ell')]\!].$$

Definition 2.8 (Symbolic semantics)**.** Given a PTA $\mathcal{A} = (\Sigma, L, \ell_0, X, P, PD, I, E)$, the symbolic semantics of $\mathcal{A}$ is the labelled transition system called *parametric symbolic state graph* $\mathbf{PSG} = (E, \mathbf{S}, \mathbf{s}_0, \Rightarrow)$, with

- $\mathbf{S} = \{(\ell, \mathbf{C}) \mid \mathbf{C} \subseteq [\![I(\ell)]\!]\}$,
- $\mathbf{s}_0 = \Big(\ell_0, \big(\{w|v \mid w = \vec{0} \wedge w|v \in [\![I(\ell_0)]\!]\}\big)^{\nearrow} \cap [\![I(\ell_0)]\!]\Big)$, and
- $\big((\ell, \mathbf{C}), e, (\ell', \mathbf{Succ}(\mathbf{C}, e))\big) \in \Rightarrow$ if $e = (\ell, g, a, R, \ell') \in E$ and $\mathbf{Succ}(\mathbf{C}, e) \neq \emptyset$.

That is, in the parametric symbolic state graph, nodes are symbolic states, and arcs are labelled by *edges* of the original PTA. In the following, we denote symbolic states using bold font ("$\mathbf{s}$").

Given $\mathbf{s} = (\ell, \mathbf{C})$ and $e = (\ell, g, a, R, \ell') \in E$, we write $\mathbf{Succ}(\mathbf{s}, e) = (\ell', \mathbf{Succ}(\mathbf{C}, e))$.

Given $\mathbf{s} = (\ell, \mathbf{C})$, as an abuse of notation, we write $\mathbf{s} = \emptyset$ whenever $\mathbf{C} = \emptyset$.

Given a PTA $\mathcal{A}$ with parametric symbolic state graph $\mathbf{PSG} = (E, \mathbf{S}, \mathbf{s}_0, \Rightarrow)$, we let $\mathbf{Init}(\mathcal{A})$ denote $\mathbf{s}_0$.

A symbolic run of a PTA is an alternating sequence of symbolic states and edges 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 $(\mathbf{s}_i, e, \mathbf{s}_{i+1}) \in \Rightarrow$. Given a symbolic state $\mathbf{s}$, we say that $\mathbf{s}$ is reachable if $\mathbf{s}$ belongs to a symbolic run of $\mathcal{A}$.

Let $(\ell_0, \mathbf{C}) = \mathbf{Init}(\mathcal{A})$. Since all basic operators on polyhedra (intersection, future, time past, resets…) preserve polyhedra as mentioned in Section 2.1.4, $\mathbf{C}$ is a convex polyhedron. For the same reason, we can recall the following property:

Property 2.9 [JLR15, Property 1]**.** *For any symbolic state $(\ell, \mathbf{C})$ reachable from $\mathbf{s}_0$ in the parametric zone graph of $\mathcal{A}$, $\mathbf{C}$ is a convex polyhedron.*

Relationship between concrete and symbolic runs. Let us recall the relationship between concrete and symbolic runs, from *e.g.,* [JLR15].

Lemma 2.10 [JLR15, Corollary 2]**.** *For each parameter valuation v, reachable symbolic state $\mathbf{s}$, and concrete 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.*

2.2.3. *Integer hulls.* We briefly recall some definitions from [JLR15]. Let $\mathbf{C}$ be a convex polyhedron.

The integer hull of a topologically closed polyhedron, denoted by $\mathsf{IH}(\mathbf{C})$, is defined as the convex hull of the integer vectors of $\mathbf{C}$, *i.e.,* $\mathsf{IH}(\mathbf{C}) = \mathsf{Conv}(\mathsf{IV}(\mathbf{C}))$, where $\mathsf{IV} = \{\eta \mid \eta \in \mathbf{C} \wedge \eta \text{ is a } \mathbb{Z}\text{-valuation}\}$ denotes the set of vectors with integer coordinates.

The interested reader is refered to [JLR15] for a discussion on how the integer hull operation can be adapted to non-necessarily closed polyhedra.

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 $\bigcup_i \mathbf{C}_i$, we write $\mathsf{IH}(\bigcup_i \mathbf{C}_i)$ for the set $\bigcup_i(\mathsf{IH}(\mathbf{C}_i))$. Given a symbolic state $\mathbf{s} = (\ell, \mathbf{C})$, we often write $\mathsf{IH}(\mathbf{s})$ for $(\ell, \mathsf{IH}(\mathbf{C}))$.

2.2.4. *Computation problems.* Given a class of decision problems (such as reachability), we consider the problem of synthesizing the set (or part of it) of parameter valuations v such that $v(\mathcal{A})$ satisfies a given problem.

Here, we mainly focus on reachability (*i.e.,* "synthesize the set of parameter valuations for which a (concrete) run visits some goal location") called EF, and unavoidability (*i.e.,* "do all maximal (concrete) runs go through some goal locations") called AF.[2]

EF-synthesis:
Input: A PTA $\mathcal{A}$ and a subset G of its locations
Problem: Synthesize the set of parameter valuations v such that G is reachable in $v(\mathcal{A})$ from the initial state.

AF-synthesis:
Input: A PTA $\mathcal{A}$ and a subset G of its locations
Problem: Synthesize the set of parameter valuations v such that all concrete runs eventually reach G in $v(\mathcal{A})$.

Finally, we will be interested in Section 4.3 in the traces preservation synthesis.

Traces preservation-synthesis:
Input: A PTA $\mathcal{A}$ and a parameter valuation v
Problem: Synthesize the set of parameter valuations v' such that $v'(\mathcal{A})$ has the same trace set as $v(\mathcal{A})$.

Since the stated problems are not computable in general, our goal here will be to synthesize sets of parameters containing all integer valuations solution to the problem, and possibly some rational valuations too.

3. Parametric extrapolation

In this section, we present an extrapolation based on the classical M-extrapolation used for the symbolic state abstraction for timed automata (see *e.g.,* [DT98]), but this time in a parametric setting.

In [JLR15], termination was guaranteed because it was possible to bound all clock valuations using a bound depending only on the integer constants of the PTA and on the bounded parameter domain, which in turns allow to bound the value of all clocks, and finally

[2]The EF and AF notations come from TCTL and denote reachability and unavoidability, respectively; they have been used in several works since [JLR15].

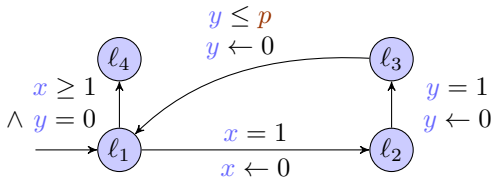


FIGURE 1. Motivating PTA

the number of possible symbolic states. Let us first show that this does not hold anymore over rational-valued parameters, which motivates the use of an extrapolation.

Example 3.1. Consider the PTA in Fig. 1. The smaller the value of p, the more loops we need to take, and therefore the longer the time to reach ℓ_4: after a number n of times through the loop, when arriving in ℓ_1 from ℓ_3 we get constraints in ℓ_1 of the form $0 \le x \le n \times p$, with n growing without bound. Indeed assume that $x \le np$ and $y = 0$ in ℓ_1. Then $x = 0$ and $y \ge 1 - np$ in ℓ_2, then $x \le np$ in ℓ_3 and finally $x \le (n+1)p$ and $y = 0$ back in ℓ_1. Since the duration of the loop (between two consecutive arrivals in ℓ_2) is exactly 1, then the time necessary to reach location ℓ_4 is arbitrarily large, depending on the value of p—even if the parameter p is bounded (*e.g.,* in $[0,1]$). This was not the case in [JLR15] due to the fact that parameters were bounded integers. Hence, on this PTA, we cannot just apply the integer hull (as in [JLR15]) to ensure termination of our algorithms.

Example 3.2. Now, we will show that the union for all valuations of the parameters of the classical M-extrapolation used for the classical symbolic state abstraction for timed automata [DT98] leads to a non-convex polyhedron. Let us consider the PTA in Fig. 2a with a parameter p such that $0 \le p \le 1$. Since when x is reset to 0 by taking the loop, y has increased by at most p due to the invariant, then by taking n times the loop we obtain:

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

Recall that the classical M-extrapolation for timed automata (see [DT98, BBLP06]) consists in exhibiting the largest constant of the model ("M"): then, each guard is either always true or always false for any clock valuation beyond this threshold. Extrapolation consists in adding (spurious) concrete states beyond M such that extrapolated polyhedra overlapping M have a finite number of possible shapes—which ensures the finiteness of the (extrapolated) symbolic state graph, and therefore termination of the related symbolic algorithms.

Here, the greatest constant of the model is $M = 1$. After one loop (and some delay), y can be greater than 1. Then, for each value of p, we can apply to the polyhedron given above the classical M-extrapolation for TAs. The union for all valuations of p of these extrapolations, projected to the plan (y, p), is depicted by the colored part (light orange and dark blue) of Fig. 2b. The right-most polyhedron (dark blue) is unbounded in y (depicted by the gradient). Note that the obtained extrapolation is non-convex. This is similar to the non-convex approximations for TAs [HKSW11].

3.1. **A parametric extrapolation.** For any polyhedron $\mathbf{C}$ over a given set of variables V, for any variable $\nu \in V$, we denote by $\mathsf{Cyl}_\nu(\mathbf{C})$ the *cylindrification* of $\mathbf{C}$ along variable ν, *i.e.,* $\mathsf{Cyl}_\nu(\mathbf{C}) = \{\eta \mid \exists \eta' \in \mathbf{C}, \forall \nu' \neq \nu, \eta'(\nu') = \eta(\nu') \text{ and } \eta(\nu) \ge 0\}$. This is a usual operation that

(A) PTA (assume $PD(p) = [0, 1]$) (B) Extrapolation

FIGURE 2. Example illustrating the non-convex parametric extrapolation

consists in *unconstraining* variable ν. The result $\mathsf{Cyl}_\nu(\mathbf{C})$ for a polyhedron $\mathbf{C}$ is a polyhedron: one just need to project out variable ν (*e.g.*, with the Fourier-Motzkin algorithm), and then add it back unconstrained.

3.1.1. *Defining extrapolation.* Let us first introduce our concept of (M, x)-extrapolation for a single clock.

Definition 3.3 ((M, x)-extrapolation)**.** Let $\mathbf{C}$ be a polyhedron. Let M be a non-negative integer constant and x be a clock. The (M, x)-extrapolation of $\mathbf{C}$, denoted by $\mathsf{Ext}_x^M(\mathbf{C})$, is defined as:

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

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

Example 3.4. To illustrate the (M, x)-extrapolation, we go back to the example of Fig. 2a. After one loop, the set $\mathbf{C}$ of valuations associated with ℓ_0 is $x \leq p \wedge y - x \leq p \wedge y \geq x \geq 0$.

According to Definition 3.3, $\mathsf{Ext}_y^1(\mathbf{C})$ is made of two parts:

- $\big(\mathbf{C} \cap [\![y \leq 1]\!]\big)$ evaluates to $0 \leq x \leq y \leq 1 \wedge y - p \leq x \leq p$, depicted in light orange in Fig. 2b after projection onto y and p (*i.e.*, $0 \leq y \leq 1 \wedge y \leq 2 \times p$).
- $\mathsf{Cyl}_y\big(\mathbf{C} \cap [\![y > 1]\!]\big) \cap [\![y > 1]\!]$ evaluates to $0 \leq x \leq p \leq 1 \wedge y > 1 \wedge x + p > 1$, depicted in dark blue in Fig. 2b after projection onto y and p (*i.e.*, $y > 1 \wedge \frac{1}{2} < p \leq 1$).

Note that for this example the $(1, y)$-extrapolation gives the same result as the union for all valuations of the parameter p of the classical extrapolation for timed automata.

Lemma 3.5 follows from Definition 3.3.

Lemma 3.5. *For each polyhedron $\mathbf{C}$, integer $M \geq 0$ and clock variables x and x', we have $\mathsf{Ext}_x^M\big(\mathsf{Ext}_{x'}^M(\mathbf{C})\big) = \mathsf{Ext}_{x'}^M\big(\mathsf{Ext}_x^M(\mathbf{C})\big)$.*

Proof. The result comes from the following facts:

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

We can now consistently define the (M, X)-extrapolation operator over a *set* of clocks:

Definition 3.6 ((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 $x \in X$.

Remark 3.7. Observe that, given a convex polyhedron $\mathbf{C}$, the size of $\mathsf{Ext}_X^M(\mathbf{C})$ is in the worst case a disjunction of $2^{|X|}$ polyhedra.

When clear from the context we omit X and only write M-extrapolation or Ext^M.

In the rest of this section, we first prove our results on Ext_x^M. It is then straightforward to adapt them to Ext_X^M using Lemma 3.5.

3.1.2. *Properties of extrapolation.* Crucially, extrapolation preserves the projection onto P:

Lemma 3.8. *Let $\mathbf{C}$ be a constraint over $X \cup P$. Then $\mathbf{C}{\downarrow_P} = \mathsf{Ext}_x^M(\mathbf{C}){\downarrow_P}$.*

Proof. First notice that $\mathbf{C} \subseteq \mathsf{Ext}_x^M(\mathbf{C})$, and therefore $\mathbf{C}{\downarrow_P} \subseteq \mathsf{Ext}_x^M(\mathbf{C}){\downarrow_P}$. Second, Ext_x^M only adds points with new clock valuations, without modifying parameter valuations. Indeed, the left-hand term only constrains $\mathbf{C}$ further and therefore does not add parameter valuations. The right-hand part first constrains $\mathbf{C}$, then unconstrains x (which does not add parameter valuations), and constrains $\mathbf{C}$ again: so no new parameter valuations can be added. By definition of Ext_x^M, for any valuation in $\mathsf{Ext}_x^M(\mathbf{C})$, there exists a valuation in $\mathbf{C}$ with the same projection on parameters. So, $\mathbf{C}{\downarrow_P} \supseteq \mathsf{Ext}_x^M(\mathbf{C}){\downarrow_P}$. □

Lemma 3.9. *For each parameter valuation v, non-negative integer constant M, clock x and valuations set $\mathbf{C}$, $v(\mathsf{Ext}_x^M(\mathbf{C})) = \mathsf{Ext}_x^M(v(\mathbf{C}))$.*

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

We will also need the following result:

Lemma 3.10. *For all edges e and sets of valuations $\mathbf{C}_1$ and $\mathbf{C}_2$, we have: if $\mathsf{Ext}_x^M(\mathbf{C}_1) = \mathsf{Ext}_x^M(\mathbf{C}_2)$ then $\mathsf{Ext}_x^M(\mathbf{Succ}(\mathbf{C}_1, e)) = \mathsf{Ext}_x^M(\mathbf{Succ}(\mathbf{C}_2, e))$.*

Proof. Assume $\mathsf{Ext}_x^M(\mathbf{C}_1) = \mathsf{Ext}_x^M(\mathbf{C}_2)$ and consider $w_1'|v \in \mathbf{Succ}(\mathbf{C}_1, e)$. Then there exists $w_1|v \in \mathbf{C}_1$ such that $w_1'|v \in \mathbf{Succ}(\{w_1|v\}, e)$.

If $w_1(x) \leq M$, then $w_1|v$ is also in $\mathbf{C}_2$ and $w_1'|v \in \mathbf{Succ}(\mathbf{C}_2, e)$.

If $w_1(x) > M$, then by definition of Ext_x^M and in particular of cylindrification, there exists some $w_2 \in \mathbf{C}_2$ such that $\forall x' \neq x, w_1(x') = w_2(x')$ and $w_2(x) > M$.

Now, either x is reset to zero along e, and then it is clear that $\mathbf{Succ}(\{w_1|v\}, e) = \mathbf{Succ}(\{w_2|v\}, e)$ and so $w_1'|v \in \mathbf{Succ}(\mathbf{C}_2, e)$.

Or x is not reset to zero along e. If we have lower bound constraints on x in guards or invariants, those constraints were and are vacuously satisfied before and after taking e, both for w_1 and w_2. And if we have any upper bound constraint on x then the successor is empty, again for both valuations. If the edge can indeed be taken (the successor is not empty), then clock x thus plays no role in determining the value of other clocks and itself will stay above M.

It follows we still have $w_1'(x) > M$ and there also exists $w_2' \in \mathbf{Succ}(\{w_2|v\}, e)$, such that $w_2'(x) > M$ and for all $x' \neq x, w_2'(x') = w_1'(x')$. This means that $w_1'|v \in \mathsf{Ext}_x^M(\mathbf{Succ}(\mathbf{C}_2, e))$.

With this, we have proved the left to right inclusion. The right to left inclusion is completely symmetrical. □

It is clear that Lemmas 3.8 to 3.10 directly extend from Ext_x^M to Ext_X^M.

3.2. **Simulations.** For the preservation of behaviours, following [BBLP06], we use a notion of simulation:

Definition 3.11 (Simulation [BBLP06])**.** Let $\mathcal{A} = (\Sigma, L, \ell_0, X, I, E)$ be a TA and $\preceq$ a relation on $L \times \mathbb{R}_+^{|X|}$. Relation $\preceq$ is a (location-based) simulation if:

- if $(\ell_1, w_1) \preceq (\ell_2, w_2)$ then $\ell_1 = \ell_2$,
- if $(\ell_1, w_1) \preceq (\ell_2, w_2)$ and $(\ell_1, w_1) \xrightarrow{a} (\ell_1', w_1')$, then there exists (ℓ_2', w_2') such that $(\ell_2, w_2) \xrightarrow{a} (\ell_2', w_2')$ and $(\ell_1', w_1') \preceq (\ell_2', w_2')$,
- if $(\ell_1, w_1) \preceq (\ell_2, w_2)$ and $(\ell_1, w_1) \xrightarrow{d_1} (\ell_1, w_1 + d_1)$, then there exists d_2 such that $(\ell_2, w_2) \xrightarrow{d_2} (\ell_2, w_2 + d_2)$ and $(\ell_1, w_1 + d_1) \preceq (\ell_2, w_2 + d_2)$.

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 3.12 [BBLP06, Lemma 1]**.** *Let M be a non-negative integer constant greater than or equal to the maximum constant occurring in the simple clock constraints appearing in the guards and invariants 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 $\mathcal{R} = \left\{\left((\ell, w), (\ell, w')\right) \mid w \equiv_M w'\right\}$ is a bisimulation relation.*

3.2.1. *Computing M in bounded PTAs.* We use the bounds on parameters to compute the maximum constant M appearing in all the guards and invariants of the PTA. When in a constraint a clock is compared to a parametric expression, we compute the maximum value of that expression over all the bounded parameter valuations. This value is finite and can be done by solving a linear program because the expression is linear and the domain of the parameters defines a polytope. M should be greater than all these valuations.

Example 3.13. Consider a guard $x \leq 2p_1 - p_2 + 1$ 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 PTAs is not a strong restriction in practice—especially since the bounds can be arbitrarily large.[3]

Lemmas 3.14 and 3.17 are instrumental in proving the preservation of all correct integer parameter valuations in the algorithms of Section 4.

Lemma 3.14. *Let $\mathcal{A}$ be a bounded PTA, $\mathbf{s}$ be a symbolic state of $\mathcal{A}$, and M a non-negative integer constant greater than the maximal constant occurring in the clock constraints of the PTA for all possible valuations of the parameters. Let x be a clock, v be a parameter valuation, and $(\ell, w) \in v(\mathsf{Ext}_x^M(\mathbf{s}))$ 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 $(\ell, w|v) \in \mathsf{Cyl}_x(\mathbf{s} \cap [\![x > M]\!]) \cap [\![x > M]\!]$. This implies that $v(\mathbf{s} \cap [\![x > M]\!]) \neq \emptyset$ and $w(x) > M$. Therefore, and using the definition of Cyl_x, there exists $(\ell, w'|v) \in \mathbf{s} \cap [\![x > M]\!]$ such that for each $x' \neq x, w'(x') = w(x')$. We also have $w'(x) > M$, which means that $w' \equiv_M w$ and by Lemma 3.12, we obtain the expected result. □

[3]Of course, the theoretical worst-case complexity of model checking depends exponentially on the size of the constants; however, appropriate abstractions such as the zone graph [BY03] are well-known to potentially alleviate this problem in practice.

Again, Lemma 3.14 directly extends from Ext_x^M to Ext_X^M.

The following lemma will be subsequently used in the proof of soundness of our synthesis algorithm (Theorem 4.3).

Lemma 3.15. *For all symbolic states* $\mathbf{s}$ *and* $\mathbf{s}'$*, non-negative integer constant* M *greater than the maximal constant occurring in the PTA (including the bounds of parameters), and parameter valuations* v*, such that* $v(\mathsf{Ext}_X^M(\mathbf{s})) = v(\mathsf{Ext}_X^M(\mathbf{s}'))$*, 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 3.9 and 3.14. □

3.3. **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 $\mathsf{Ext}_x^M(\mathbf{s})$ as $(\mathbf{s} \cap [\![x \leq M]\!]) \cup \mathsf{Cyl}_x(\mathbf{s} \cap [\![x \geq M]\!]) \cap [\![x \geq M]\!]$.

Lemma 3.16. *For any integer parameter valuation* v*, any non-negative integer constant* M*, and any symbolic state* $\mathbf{s} = (\ell, \mathbf{C})$*,* $v(\mathsf{Ext}_x^M(\mathbf{C}))$ *is its own integer hull.*

Proof. For any reachable symbolic state $(\ell, \mathbf{C})$ of a PTA $\mathcal{A}$, and for any integer parameter valuation v, we have by [JLR15, Property 3] that $\mathcal{A}(v)$ is a *zone* [BY03]: a convex polyhedron defined only by inequalities of the form $\nu \bowtie d$ or $\nu - \nu' \bowtie d$, with ν, ν' (clock) variables, $d \in \mathbb{Z}$, and $\bowtie \in \{<, \leq, \geq, >\}$. Still by [JLR15, Property 3], zones also have integer vertices and are their own integer hulls.

From Lemma 3.9, $v(\mathsf{Ext}_x^M(\mathbf{C})) = \mathsf{Ext}_x^M(v(\mathbf{C}))$. Now, $[\![x \geq M]\!]$ and $[\![x \leq M]\!]$ are zones too. Furthermore, cylindrification on x preserves zones with integer coefficients too: it consists in removing any upper bound constraint on x ($x \leq U$, with U an integer) from the zone. Thus $\mathsf{Ext}_x^M(v(\mathbf{C}))$ is a union of two zones, and each of them is then its own integer hull. The result then follows from the fact that the integer hull of a union of polyhedra is defined as the union of the integer hulls of those polyhedra. □

Lemma 3.17. *For any integer parameter valuation* v*, any non-negative integer constant* M*, and any symbolic state* $\mathbf{s} = (\ell, \mathbf{C})$*,* $v(\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{C}))) = v(\mathsf{Ext}_X^M(\mathbf{C}))$*.*

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

In the other direction, using Lemma 3.16, $v(\mathsf{Ext}_X^M(\mathbf{C})) = \mathsf{IH}(v(\mathsf{Ext}_X^M(\mathbf{C})))$. Now, if $w \in \mathsf{IH}(v(\mathsf{Ext}_X^M(\mathbf{C})))$, then $w|v \in \mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{C}))$, *i.e.,* $w \in v(\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{C}))$. □

The following Proposition 3.18 is the key to proving the termination of the algorithms of Section 4.

Proposition 3.18. *In a bounded PTA, the set of constraints* $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{C}))$ *over the set of symbolic reachable states* $(\ell, \mathbf{C})$ *is finite.*

Proof. In each disjunct of $\mathsf{Ext}_X^M(\mathbf{C})$, either clocks are upper bounded by M, or (due to the cylindrification) are lower bounded by 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 of polyhedra with integer vertices with coordinates less or equal to M (and a finite set of extremal rays, taken in the finite set of the directions of clock variables), of which there can be only finitely many. □

4. Integer-complete dense parametric algorithms

In this section, we describe three parameter synthesis algorithms:

(1) RIEF, computing parameter valuations for which a location is reachable (Section 4.1);
(2) RIAF, computing parameter valuations for which a location is unavoidable (Section 4.2); and
(3) RITP, computing parameter valuations for which the trace set is identical to that of a reference valuation (Section 4.3).

These algorithms always terminate for *bounded* PTAs, and return not only all the integer valuations solution of the problem (à la [JLR15]) but also some rational valuations. In fact, contrarily to [JLR15] in which returned constraints could only be interpreted over *integer* valuations, all rational valuations in the constraints returned by our algorithms are correct solutions.

Principle of the three algorithms. All three algorithms (RIEF, RIAF and RITP) explore the standard parametric symbolic state graph (without integer hull nor extrapolation); integer hull and extrapolation are only used in the **Passed** set of visited states, to ensure termination. This is in contrast with most algorithms for timed automata, which work on extrapolated zone graphs.

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

- the root of T^∞ is labelled by $\mathbf{Init}(\mathcal{A})$;
- for every node n of T^∞, if n is labelled by some symbolic state $\mathbf{s}$, then for all edges e of $\mathcal{A}$, there exists a unique child n' of n labelled by $\mathbf{Succ}(\mathbf{s}, e)$ iff $\mathbf{Succ}(\mathbf{s}, e) \neq \emptyset$.

Remark 4.1. Definition 2.3 defines bounded PTAs as PTAs in which the parameters range in a hyperrectangle. However, all results in this section can be extended to a more permissive definition of bounded PTAs. In fact, in this section, we could consider that a PTA is bounded if its parameter domain is bounded. This includes hyperrectangles, but also more free "shapes", including non-convex or non-connected parameter domains, provided they can be expressed as a finite union of convex polyhedra.

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, *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 $\mathsf{IH}(\mathbf{C})$ in Algorithm RIEF by $\mathbf{C}$, we obtain Algorithm EF from [JLR15]. The differences with IEF (which was defined in [JLR15]) are highlighted.

RIEF is a recursive algorithm (initially called by $\mathsf{RIEF}(\mathcal{A}, \mathbf{Init}(\mathcal{A}), G, \emptyset)$) that 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, RIEF stores in the set **Passed** of visited states the *integer hulls* of (the extrapolation of) the symbolic states, which ensures termination due to the finite number

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

input : A bounded PTA $\mathcal{A}$, a symbolic state $\mathbf{s} = (\ell, \mathbf{C})$, a set of target locations G, a set **Passed** of passed states on the current path

output : Set $\mathbf{K}$ of parameter valuations guaranteeing reachability

```
1 if ℓ ∈ G then K ← C↓_P ;
2 else
3     K ← ∅;
4     if IH(Ext^M_X(s)) ∉ Passed then
5         foreach outgoing e from ℓ in A s.t. Succ(s, e) ≠ ∅ do
6             s' ← Succ(s, e);
7             K ← K ∪ RIEF(A, s', G, Passed ∪ {IH(Ext^M_X(s))});
8 return K
```

of possible integer hulls of M-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 Proposition 3.18, it is clear that RIEF explores only a finite number a symbolic states. Therefore, we have the following theorem:

Theorem 4.2. *For any bounded PTA $\mathcal{A}$, the computation of $\mathsf{RIEF}(\mathcal{A}, \mathbf{Init}(\mathcal{A}), G, \emptyset)$ terminates.*

Algorithm RIEF is a post-order depth-first traversal of some prefix of the symbolic reachability tree.

Theorem 4.3. *Let $\mathbf{K} = \mathsf{RIEF}(\mathcal{A}, \mathbf{Init}(\mathcal{A}), G, \emptyset)$. We have:*

(1) *Soundness: If $v \in \mathbf{K}$ 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 $v \in \mathbf{K}$.*

Proof. Since RIEF has terminated, it has explored a finite prefix T of T^∞.

(1) Soundness: Let $\mathbf{s}$ be symbolic state in T, and **Passed** be the set of ancestors of $\mathbf{s}$ in T. Further let **Passed′** be the set $\{(\ell, \mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{C})) \mid (\ell, \mathbf{C}) \in \mathbf{Passed}\}$.
Then, by induction, we have $\mathsf{RIEF}(\mathcal{A}, \mathbf{s}, G, \mathbf{Passed}') \subseteq \mathsf{EF}(\mathcal{A}, \mathbf{s}, G, \mathbf{Passed})$:
- if $\mathbf{s} = (\ell, \mathbf{C})$ is a leaf of T, then either $\ell \in G$ and both algorithms return the same value $\mathbf{C}{\downarrow_P}$, or the result of RIEF is empty and thus included in whatever is the result of EF.
- if it is not a leaf, then by the induction hypothesis we have inclusion for all the recursive calls and also for the union of all of them.

(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.
Let v be an *integer* parameter valuation such that 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 $\mathbf{Init}(\mathcal{A}) = \mathbf{s}_0 \xrightarrow{e_1} \mathbf{s}_1 \xrightarrow{e_2} \cdots \xrightarrow{e_p} \mathbf{s}_p$ of the

tree T^∞ obtained by following this edge sequence on the labels of the arcs in the tree. For $i \leq p$, let $\mathbf{Passed}_i = \{\mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{s}_j)) \mid 0 \leq j < i\}$.

Assume first there exists no $k \leq p$ such that $\mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{s}_k)) \in \mathbf{Passed}_k$ then, by induction on $i \leq p$, we have v in $\mathsf{RIEF}(\mathcal{A}, \mathbf{s}_i, G, \mathbf{Passed}_i)$:

- if $i = p$ then, noting $\mathbf{s}_i = (\ell_i, \mathbf{C}_i)$, we have $\ell_i \in G$ and by Lemma 2.10, $v \in \mathbf{C}_i{\downarrow_P}$ and thus in the result of RIEF as expected.
- otherwise, for $0 < i \leq p$, since we do not have $s_{i-1} \in \mathbf{Passed}_{i-1}$, we must be in the recursive case, and thus $\mathsf{RIEF}(\mathcal{A}, \mathbf{s}_{i-1}, G, \mathbf{Passed}_{i-1})$ contains $\mathsf{RIEF}(\mathcal{A}, \mathbf{s}_i, G, \mathbf{Passed}_i)$, as part of the big union on recursive calls, and by the induction hypothesis it contains v as well.

Then, in particular, for $i = 0$, we have $v \in \mathsf{RIEF}(\mathcal{A}, \mathbf{Init}(\mathcal{A}), G, \emptyset)$.

Assume now there exists k such that $\mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{s}_k)) \in \mathbf{Passed}_k$ and let $j < k$ be such that $\mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{s}_j)) = \mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{s}_k))$.

Since v is an integer parameter valuation, by Lemma 3.17, this means that $v(\mathsf{Ext}^M_X(\mathbf{s}_k)) = v(\mathsf{Ext}^M_X(\mathbf{s}_j))$. Using now Lemma 3.15, for any $(\ell_k, w_k) \in v(\mathbf{s}_k)$ there exists a state $(\ell_k, w_j) \in v(\mathbf{s}_j)$ that is bisimilar to it.

Also, by Lemma 2.10, (ℓ_k, w_j) is reachable in $v(\mathcal{A})$ via some run ρ_1 along edges $e_1 \dots e_j$. Also, since (ℓ_k, w_j) and (ℓ_k, w_k) are bisimilar, there exists a run ρ_2 that takes the same edges as the suffix of ρ starting at (ℓ_k, w_k). Let ρ' be the run obtained by merging ρ_1 and ρ_2 at (ℓ_k, w_j). 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. □

Example 4.4. Consider the simple PTA with a unique transition from the initial location ℓ_0 to ℓ_1 with guard $1 \leq x \leq 2p$. To ensure the $EF\{\ell_1\}$ property, we just need to be able to go through the transition from ℓ_0 to ℓ_1. To compute polyhedron C_1 obtained in ℓ_1, we first delay in ℓ_0, obtaining $x \geq 0$. Then we intersect with the guard, obtaining $1 \leq x \leq 2p$ (note that this implies $1 \leq 2p$). By delaying again in ℓ_1, we finally obtain that C_1 is $1 \leq x \wedge 1 \leq 2p$, which implies $p \geq \frac{1}{2}$. The integer hull of C_1 is $1 \leq x \wedge 1 \leq p$, which implies $p \geq 1$.

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

4.2. **Parametric unavoidability: RIAF.** The goal of RIAF (given in Algorithm 2) is to synthesize 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 4.6 below, that interpreting the result of IAF as a dense set is incorrect in general, since it may contain non-integer valuations that do not ensure unavoidability. The differences with IAF (which was defined in [JLR15]) are highlighted in Algorithm 2.

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:

(1) $\mathbf{K}_{good}$ contains the set of parameter valuations that indeed satisfy AF, recursively computed by calling RIAF;

Algorithm 2: $\mathsf{RIAF}(\mathcal{A}, (\ell, \mathbf{C}), G, \mathbf{Passed})$

input : A bounded PTA $\mathcal{A}$, a symbolic state $\mathbf{s} = (\ell, \mathbf{C})$, a set of target locations G, a set **Passed** of passed states on the current path

output : Set $\mathbf{K}$ of parameter valuations guaranteeing unavoidability

1 **if** $\ell \in G$ **then** $\mathbf{K} \leftarrow \mathbf{C}{\downarrow_P}$;
2 **else**
3 **if** $\left(\ell, \mathsf{IH}(\mathit{Ext}_X^M(\mathbf{C}))\right) \in \mathbf{Passed}$ **then** $\mathbf{K} \leftarrow \emptyset$;
4 **else**
5 $\mathbf{K} \leftarrow \mathbf{C}{\downarrow_P}$; $\mathbf{K}_{live} \leftarrow \emptyset$;
6 **foreach** *outgoing* $e = (\ell, g, a, R, \ell')$ *from* ℓ *in* $\mathcal{A}$ *s.t.* $\mathbf{Succ}(\mathbf{s}, e) \neq \emptyset$ **do**
7 $\mathbf{s}' \leftarrow \mathbf{Succ}(\mathbf{s}, e)$;
8 $\mathbf{K}_{good} \leftarrow \mathsf{RIAF}\Big(\mathcal{A}, \mathbf{s}', G, \mathbf{Passed} \cup \big\{\big(\ell, \mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{C}))\big)\big\}\Big)$;
9 $\mathbf{K}_{block} \leftarrow {\mathbb{Q}_+}^P \setminus \mathbf{s}'{\downarrow_P}$;
10 $\mathbf{K} \leftarrow \mathbf{K} \cap (\mathbf{K}_{good} \cup \mathbf{K}_{block})$;
11 $\mathbf{K}_{live} \leftarrow \mathbf{K}_{live} \cup (\mathbf{C} \cap [\![g]\!])^{\swarrow}$;
12 $\mathbf{K} \leftarrow \mathbf{K} \setminus (\mathbf{C} \setminus \mathbf{K}_{live}){\downarrow_P}$;
13 **return** $\mathbf{K}$

(2) $\mathbf{K}_{block}$ allows to cut off branches, for instance leading to deadlock or looping, by keeping only parameter valuations in the complement of the first state in that branch: imagine we have a guard $2 \leq x \leq p$, but selecting $p < 2$ we effectively cut the branch off. Now, for each branch we need to ensure that it always either goes to G (giving the constraint $\mathbf{K}_{good}$), or we cannot take it (constraint $\mathbf{K}_{block}$), hence the union in the algorithm (line 10). Note that for completeness we need also parameters cutting some branches that do go to G, provided they still allow at least one branch to go there: for instance in Figure 3, assuming $G = \{\ell_1, \ell_2\}$, all parameters valuations greater than $\frac{1}{2}$ ensure unavoidability since both branches go to G. But all valuations less or equal to $\frac{1}{2}$ also do because only the upper branch remains. Parameters that cut all branches are forbidden using $\mathbf{K}_{live}$.

(3) $\mathbf{K}_{live}$ (which is a set of valuations over $X \cup P$) is necessary to forbid reaching states from which no transition can be taken for any e, even after some delay, and also to forbid cutting all branches.

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 (Proposition 3.18).

We state below the soundness and integer completeness of RIAF. The proofs are similar to those of AF in [JLR15], by using the additional arguments provided in the proof of RIEF, and in particular Proposition 3.18.

Theorem 4.5. *Let* $\mathbf{K} = \mathsf{RIAF}(\mathcal{A}, \mathbf{Init}(\mathcal{A}), G, \emptyset)$. *We have:*

(1) *Soundness: If* $v \in \mathbf{K}$ *then* G *is unavoidable in* $v(\mathcal{A})$*;*
(2) *Integer completeness: If* v *is an integer parameter valuation, and* G *is unavoidable in* $v(\mathcal{A})$ *then* $v \in \mathbf{K}$*.*

Proof. The algorithm having terminated, it has explored a finite prefix T of T^∞.

(1) Soundness: Let $\mathbf{s}$ be symbolic state in T, and **Passed** be the set of ancestors of $\mathbf{s}$ in T. Further let **Passed′** be the set $\{(\ell, \mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{C})) \mid (\ell, \mathbf{C}) \in \mathbf{Passed}\}$.
Then, by induction, we have $\mathsf{RIAF}(\mathcal{A}, \mathbf{s}, G, \mathbf{Passed}') \subseteq \mathsf{AF}(\mathcal{A}, \mathbf{s}, G, \mathbf{Passed})$:
- if $\mathbf{s} = (\ell, \mathbf{C})$ is a leaf of T, then either $\ell \in G$ and both algorithms return the same value $\mathbf{C}{\downarrow_P}$, or the result of RIAF is empty and thus included in whatever is the result of AF.
- if it is not a leaf, then by the induction hypothesis we have inclusion for all the constraints $\mathbf{K}_{good}$ from the recursive calls. Polyhedra $\mathbf{K}_{live}$ and $\mathbf{K}_{block}$ have the same value in both algorithms, and so it follows that the $\mathbf{K}$ returned by RIAF is included in the one returned by AF.

(2) Integer completeness: Let v be an *integer* parameter valuation for which the unavoidability property holds in $v(\mathcal{A})$.
Given a symbolic state $\mathbf{s}$ in T, let **Passed** be the set of all $\mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{s}'))$ for all ancestors $\mathbf{s}'$ of $\mathbf{s}$ in T. We prove by induction on the finite tree T that, for each symbolic state $\mathbf{s}$ in T such that the sequence of edges leading to $\mathbf{s}$ is feasible in $v(\mathcal{A})$, we have $v \in \mathsf{RIAF}(\mathcal{A}, \mathbf{s}, G, \mathbf{Passed})$:
- if $\mathbf{s} = (\ell, \mathbf{C})$ is a leaf then either it is because $\ell \in G$. Then, by Lemma 2.10, v is in $\mathbf{C}{\downarrow_P}$ and thus in the result of RIAF as expected.
Or it is because there exists an ancestor $\mathbf{s}'$ of $\mathbf{s}$ such that $\mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{s}')) = \mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{s}))$. Since v is an integer parameter valuation, by Lemma 3.17, this means that $v(\mathsf{Ext}^M_X(\mathbf{s}')) = v(\mathsf{Ext}^M_X(\mathbf{s}))$.
Using Lemma 3.10, we can replay the sequence of actions from $\mathbf{s}'$ to $\mathbf{s}$ from $\mathbf{s}$ and obtain a new symbolic state $\mathbf{s}''$ such that $v(\mathsf{Ext}^M_X(\mathbf{s}'')) = v(\mathsf{Ext}^M_X(\mathbf{s}))$. This implies that we can repeat this procedure and obtain a path of symbolic states of $v(\mathcal{A})$, and a corresponding concrete path using Lemma 2.10, longer than any given length. Those paths never go through G, because by construction none of the ancestors of $\mathbf{s}$ has a location in G, otherwise that node would have no successor in T.
Finally, it follows from [ACD93] that since the unavoidability property holds, we must be able to reach G within some fixed number of discrete transitions that depends only on the number of places, transitions, clocks and the maximal constant in $v(\mathcal{A})$[4].
It is therefore not possible that there exists an ancestor $\mathbf{s}'$ of $\mathbf{s}$ such that $\mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{s}')) = \mathsf{IH}(\mathsf{Ext}^M_X(\mathbf{s}))$.
Finally, it may be that none of the two previous cases hold but $\mathbf{s}$ has no successor. But then, by Lemma 2.10, any state $(\ell, w) \in \mathbf{s}$ is reachable in $v(\mathcal{A})$ via some run ρ that, as before, never goes through G and has no discrete successor, even after some delay. As before this is not possible.
- if $\mathbf{s} = (\ell, \mathbf{C})$ is not a leaf then we must be in the recursive case and there must exist at least one successor of $\mathbf{s}$.
Consider successor $\mathbf{s}'$ of $\mathbf{s}$ by edge e. There are two possibilities:
  - either e is feasible in $v(\mathcal{A})$ from some state s in $v(\mathbf{s})$; we know there is such a state because we have assumed that the sequence of edges leading to $\mathbf{s}$ is feasible in $v(\mathcal{A})$ and thus, by Lemma 2.10, $v(\mathbf{C})$ is not empty.

[4]Actually this would be the number of states in the *region graph* of $v(\mathcal{A})$ as defined in [ACD93], since a loop in that graph would imply an infinite run in $v(\mathcal{A})$.

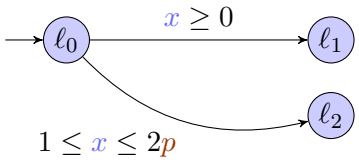


FIGURE 3. Counter-example to the density of the result of IAF.

Then by the induction hypothesis, v is in $\mathbf{K}_{good}$. We also have that the guard of e is satisfiable from s, possibly after some delay. This implies that v is also in $\mathbf{K}_{live}$ and will be at the end of the `foreach` loop.

– or e is not feasible in $v(\mathcal{A})$ from any state in $v(\mathbf{s})$. Then by Lemma 2.10, v is not in $\mathbf{s}'\downarrow_P$, and thus in $\mathbf{K}_{block}$.

The first case must happen at least once, otherwise by Lemma 2.10 we have a deadlocked run in $v(\mathcal{A})$ never reaching G. Therefore we have $v \in \mathbf{K}_{live}$ at the end of the `foreach` loop, and for all successors we have v either in $\mathbf{K}_{good}$ or in $\mathbf{K}_{block}$. It follows that v is in the returned value of $\mathbf{K}$. □

Example 4.6. Consider the PTA in Fig. 3. To ensure the $AF\{\ell_1\}$ property, we need to cut the transition from ℓ_0 to ℓ_2. With a computation similar to Example 4.4, the polyhedron C_2 obtained in ℓ_2 is $1 \leq x \wedge 1 \leq 2p$, which implies $p \geq \frac{1}{2}$. The integer hull of C_2 is $1 \leq x \wedge 1 \leq p$, which implies $p \geq 1$. In order to block the path to ℓ_2 in ℓ_0, we intersect with the complement of the projection onto the parameters of $\mathsf{IH}(C_2)$, *i.e.,* $p < 1$. Since there is no constraint on the transition from ℓ_0 to ℓ_1 the final result of the former algorithm IAF from [JLR15] is actually $p < 1$. For integer parameters this means $p = 0$, which is correct. But if we interpret the result over rational-valued parameters, we obtain that, for instance, $p = \frac{1}{2}$ should be a valuation ensuring the property—while it is obviously not.

On the same example, RIAF gives (on this example) the exact result, that is $p < \frac{1}{2}$.

4.3. **Parametric traces preservation: RITP.** We address here the traces preservation synthesis problem. That is, given a PTA $\mathcal{A}$ and a reference parameter valuation v, we seek other valuations v' such that $v'(\mathcal{A})$ has the same trace set as $v(\mathcal{A})$. In [ACEF09, ALM20], we defined a procedure TP (for "traces preservation synthesis", and also called "inverse method") that may not terminate, but is correct whenever it terminates. TP is implemented in the IMITATOR software [And21].

4.3.1. *The algorithm.* We give RITP in Algorithm 3. The differences with the original TP (formalized in [ALM20]) are highlighted.

RITP is a recursive algorithm exploring the state space in a depth-first search manner. Given a current symbolic state, its v-compatibility is first checked (line 2).

- If this state is v-compatible, the current constraint is intersected with the projection onto the parameters of the integer hull of this state (line 3). Then, if this state was not visited up to the integer hull (*i.e.,* if the integer hull of this state does not belong to the list of the integer hulls of passed states, line 4), then all its successors are computed, and the results of the recursive calls are intersected with the current constraint. Note that the integer hull of the current state is added to the list of passed states in the recursive calls—which is the crux for ensuring termination.

Algorithm 3: $\mathsf{RITP}(\mathcal{A}, v, \mathbf{s}, \mathbf{Passed})$

input : A bounded PTA $\mathcal{A}$, an integer parameter valuation v, a current symbolic state $\mathbf{s} = (\ell, \mathbf{C})$, a set **Passed** of passed states on the current path
output : Set $\mathbf{K}$ of parameter valuations guaranteeing traces preservation

1 $\mathbf{K} \leftarrow {\mathbb{Q}_+}^P$
2 **if** $v \in \mathbf{C}{\downarrow_P}$ **then**
3 $\quad \mathbf{K} \leftarrow \mathbf{K} \cap \Big(\mathsf{IH}\big(\mathsf{Ext}_X^M(\mathbf{C})\big)\Big){\downarrow_P}$
4 $\quad$ **if** $\mathsf{IH}\big(\mathit{Ext}_X^M(\mathbf{s})\big) \notin$ **Passed then**
5 $\quad\quad$ **foreach** *outgoing* e *from* ℓ *in* $\mathcal{A}$ *s.t.* $\mathbf{Succ}(\mathbf{s}, e) \neq \emptyset$ **do**
6 $\quad\quad\quad \mathbf{s}' \leftarrow \mathbf{Succ}(\mathbf{s}, e)$
7 $\quad\quad\quad \mathbf{K} \leftarrow \mathbf{K} \cap \mathsf{RITP}\Big(\mathcal{A}, v, \mathbf{s}', \mathbf{Passed} \cup \big\{\mathsf{IH}\big(\mathsf{Ext}_X^M(\mathbf{s})\big)\big\}\Big)$
8 **else**
9 $\quad \mathbf{K} \leftarrow \mathbf{K} \cap \mathsf{IH}\big(\neg(\mathbf{C}{\downarrow_P})\big)$
10 **return** $\mathbf{K}$

- Conversely, if the current state is not v-compatible, the integer hull of the negation of its projection onto the parameters is intersected with the current constraint (line 9).

The base call is $\mathsf{RITP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$.
The algorithm RITP differs from TP in four main aspects.

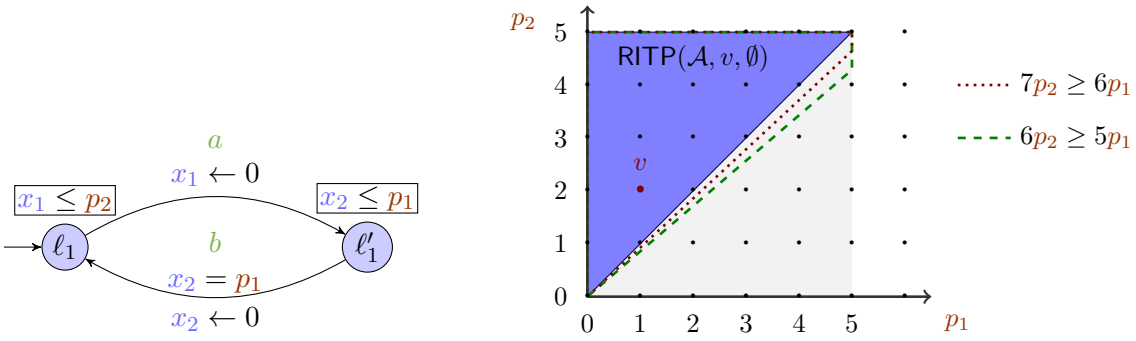
(1) The termination condition is not anymore the equality of the states, but of the *integer hull* of the extrapolation of the states (line 4): this is needed to ensure termination.
(2) RITP does not return the intersection of the v-compatible states' sets of parameter valuations, but returns the intersection of their *integer hulls* (line 3): this is needed to ensure soundness (see a counter-example in Example 4.8).
(3) RITP does not return the intersection of the negation of the v-incompatible states' sets of parameter valuations, but returns the intersection of the *integer hulls* of their negation (line 9): this is needed to ensure soundness.
(4) v must be an *integer valuation*: this is again needed to ensure soundness (see a counter-example in Example 4.9).

We believe that requiring v to be an integer is harmless in practice, since a rational can be turned into an integer by appropriately resizing the constants of the PTA. However, the result of RITP still outputs in general a *dense* set of rationals (just as TP), and hence can synthesise parameter valuations infinitesimally close to v with the same set of traces—producing a (potentially partial) measure of the system robustness with respect to the set of traces (see [APP13]).

4.3.2. *Examples.*

Example 4.7. Consider the PTA $\mathcal{A}$ depicted in Fig. 4a; assume that the domain of p_1 and p_2 is $[0, 5]$. Let v be the valuation such that $v(p_1) = 1$ and $v(p_2) = 2$. For this valuation, the trace set is made of the single following infinite trace alternating between the two locations:

$$\ell_1 \overset{a}{\rightarrowtail} \ell'_1 \overset{b}{\rightarrowtail} \ell_1 \overset{a}{\rightarrowtail} \ell'_1 \overset{b}{\rightarrowtail} \cdots$$



(A) PTA (B) Sets of parameter valuations

FIGURE 4. Example for which RITP terminates but not TP

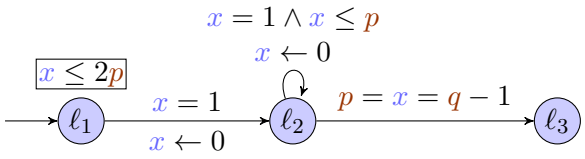


FIGURE 5. Example illustrating the need of IH in the result of RITP

TP$(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$ does not terminate, because sets of parameter valuations are generated of the form $x_1 - x_2 \geq i \times (p_1 - p_2)$, with $i \in \mathbb{N}$ growing without bound, and all these sets of parameter valuations are incomparable. In contrast, for RITP, the integer hull of the extrapolation of these polyhedra will eventually be the same (in fact, no extrapolation is needed, because both clocks are always ≤ 5 due to the invariants and the bounded parameter domain); in Fig. 4b, we give two such sets of parameter valuations, *i.e.,* $6p_2 \geq 5p_1$ and $7p_2 \geq 6p_1$, that indeed contain the same set of integer points within $[0, 5]^2$, and hence the integer hull (of their extrapolation) is equal.[5] Eventually, RITP returns $0 \leq p_1 \leq p_2 \leq 5$. In fact, for this example, it can be shown that the result output by RITP is the maximal result, *i.e.,* any valuation such that $p_1 > p_2$ has a trace set different from v (which will be a single trace made of a *finite* alternation between ℓ_1 and ℓ'_1, with the trace length proportional to the ratio between p_1 and p_2).

The following example illustrates the necessity to return the *integer hull of* the v-compatible states (line 3); otherwise, RITP would output an incorrect result.

Example 4.8. Consider the PTA $\mathcal{A}$ depicted in Fig. 5, featuring a single clock x and two parameters (actions are not depicted, because not relevant for this example—one can assume that every edge is labelled with some action a); assume that the bounded parameter domain of p and q is $[0; 2]$. Let v be such that $v(p) = 1$ and $v(q) = 2$. Then TP$(\mathcal{A}\, v, \mathbf{Init}(\mathcal{A}), \emptyset)$ synthesizes the set of valuations $\mathbf{K}$ restricted to $\mathbf{K} = \{v\}$.

Now, consider the algorithm RITP. The first time ℓ_2 is visited, the associated set of valuations $\mathbf{C}_2$ is: $\frac{1}{2} \leq p \leq 2$ (other trivial inequalities such as $x \geq 0$ are omitted; also note

[5]Strictly speaking, the sets of parameter valuations are over $X \cup P$, *i.e.,* in 4 dimensions (2 clocks and 2 parameters); for the sake of better understanding, in Fig. 4b, we project them onto P so as to represent them in 2 dimensions.

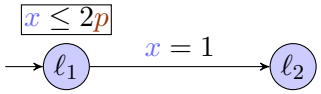


Figure 6. Example illustrating the need for an integer valuation in RITP

that the upper bound for p comes from its bounded parameter domain). Its successor is $(\ell_3, \mathbf{C}_3)$ with $\mathbf{C}_3$ being: $\frac{1}{2} \leq p \leq 1 \wedge p \leq x \wedge q = p + 1$. The projection of this set onto P is $q = p + 1 \wedge \frac{1}{2} \leq p \leq 1$.

When visiting ℓ_2 for the second time (via the self-loop over ℓ_2), the associated set of valuations $\mathbf{C}'_2$ is: $1 \leq p \leq 2$. Observe that $\mathsf{IH}(\mathbf{C}_2) = \mathsf{IH}(\mathbf{C}'_2)$. Therefore, RITP does not explore the successors of $\mathbf{C}'_2$; if RITP returned $\mathbf{C}{\downarrow_P}$ at line 3, then the result would be $q = p + 1 \wedge \frac{1}{2} \leq p \leq 1$—which is wrong.

Now, RITP actually returns $\Big(\mathsf{IH}\big(\mathsf{Ext}^M_X(\mathbf{C}_3)\big)\Big){\downarrow_P}$, which gives the expected result $p = 1 \wedge q = 2$.

The following example illustrates the necessity of requiring the input v of RITP to be an integer.

Example 4.9. Consider the PTA $\mathcal{A}$ in Fig. 6. Consider v such that $v(p) = \frac{1}{2}$. Then $\mathsf{RITP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$ outputs the set $\mathbf{K}$ of parameter valuations defined by $1 \leq p \leq 2$. Then obviously $v \notin \mathbf{K}$, which contradicts the expected soundness (see Theorem 4.16).

4.3.3. *Termination and soundness.* Termination is immediate from the finiteness of the integer hulls of extrapolations (Proposition 3.18), which makes RITP only explore a finite number of states.

We state below the soundness and integer completeness of RITP.

We first need the following lemma from [JLR15].

Lemma 4.10 [JLR15, Lemma 6]**.** *Given a symbolic state* $\mathbf{s}$ *and an edge* e*, we have* $\mathsf{IH}(\mathbf{Succ}(\mathsf{IH}(\mathbf{s}), e)) = \mathsf{IH}(\mathbf{Succ}(\mathbf{s}, e))$.

The following result derives immediately:

Lemma 4.11. *Given two symbolic states* $\mathbf{s}_1$ *and* $\mathbf{s}_2$ *such that* $\mathsf{IH}(\mathbf{s}_1) = \mathsf{IH}(\mathbf{s}_2)$*, and an edge* e*, we have* $\mathsf{IH}(\mathbf{Succ}(\mathbf{s}_1, e)) = \mathsf{IH}(\mathbf{Succ}(\mathbf{s}_2, e))$.

Proof. $\mathsf{IH}(\mathbf{Succ}(\mathbf{s}_1, e)) = \mathsf{IH}(\mathbf{Succ}(\mathsf{IH}(\mathbf{s}_1), e))$ (by Lemma 4.10), then because $\mathsf{IH}(\mathbf{s}_1) = \mathsf{IH}(\mathbf{s}_2)$, we have $\mathsf{IH}(\mathbf{Succ}(\mathbf{s}_1, e)) = \mathsf{IH}(\mathbf{Succ}(\mathsf{IH}(\mathbf{s}_2), e)) = \mathsf{IH}(\mathbf{Succ}(\mathbf{s}_2, e))$. □

The following results are immediate, yet allow our proof of Proposition 4.14 to be more clear.

Lemma 4.12. *Given two symbolic states* $\mathbf{s}_1$ *and* $\mathbf{s}_2$ *such that* $\mathsf{IH}\big(\mathit{Ext}^M_X(\mathbf{s}_1)\big) = \mathsf{IH}\big(\mathit{Ext}^M_X(\mathbf{s}_2)\big)$*, we have* $\Big(\mathsf{IH}\big(\mathit{Ext}^M_X(\mathbf{s}_1)\big)\Big){\downarrow_P} = \Big(\mathsf{IH}\big(\mathit{Ext}^M_X(\mathbf{s}_2)\big)\Big){\downarrow_P}$.

Lemma 4.13. *Given two symbolic states* $\mathbf{s}_1$ *and* $\mathbf{s}_2$ *such that* $\mathsf{IH}\big(\mathit{Ext}^M_X(\mathbf{s}_1)\big) = \mathsf{IH}\big(\mathit{Ext}^M_X(\mathbf{s}_2)\big)$*, we have* $\mathsf{IH}\Big(\Big(\mathsf{IH}\big(\mathit{Ext}^M_X(\mathbf{s}_1)\big)\Big){\downarrow_P}\Big) = \mathsf{IH}\Big(\Big(\mathsf{IH}\big(\mathit{Ext}^M_X(\mathbf{s}_2)\big)\Big){\downarrow_P}\Big)$.

Proposition 4.14 (Soundness of RITP)**.** *Upon termination of* RITP*, if* $v' \in \mathsf{RITP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$ *then* $\mathit{Traces}(v'(\mathcal{A})) = \mathit{Traces}(v(\mathcal{A}))$.

Proof. Since RITP has terminated, it has explored a finite prefix T of the symbolic reachability tree T^∞. Let $\mathbf{s}$ be symbolic state in T, and **Passed** be the set of ancestors of $\mathbf{s}$ in T. Further let **Passed′** be the set $\{(\ell, \mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{C})) \mid (\ell, \mathbf{C}) \in \mathbf{Passed}\}$.

Let us show by induction that we have $\mathsf{RITP}(\mathcal{A}, v, \mathbf{s}, \mathbf{Passed'}) \subseteq \mathsf{TP}(\mathcal{A}, v, \mathbf{s}, \mathbf{Passed})$:

- Base case: If $\mathbf{s} = (\ell, \mathbf{C})$ is a leaf of T, then
  - either $\mathbf{s}$ is v-compatible and RITP returns $(\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{C})))\downarrow_P$ while TP returns $\mathbf{C}\downarrow_P$; by Lemma 3.8 and since $\mathsf{IH}(\mathbf{C}) \subseteq \mathbf{C}$ (by definition of IH), the inclusion thus holds;
  - or $\mathbf{s}$ is v-incompatible and RITP returns $\mathsf{IH}(\neg\mathbf{C}\downarrow_P)$ while TP returns $\neg\mathbf{C}\downarrow_P$, and the inclusion holds again.
- Induction case: If $\mathbf{s}$ is not a leaf, then two cases arise.
  - If $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s})) \notin \mathbf{Passed'}$, by induction hypothesis, the inclusion holds again.
  - The case when $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s})) \in \mathbf{Passed'}$ is more delicate, as this situation did not occur in the proofs of Theorems 4.3 and 4.5. Since $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s})) \in \mathbf{Passed'}$, then there exists $\mathbf{s''} = (\ell, \mathbf{C''}) \in \mathbf{Passed'}$ such that $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s})) = \mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s''}))$, and $\mathbf{s''}$ is currently being explored by RITP, *i.e.,* a call to $\mathsf{RITP}(\mathcal{A}, v, \mathbf{s''}, \mathbf{Passed''})$ was made before, and this call led recursively to the call to $\mathsf{RITP}(\mathcal{A}, v, \mathbf{s}, \mathbf{Passed'})$. Hence, **Passed″** denotes the set of passed states when RITP was called on $\mathbf{s''}$. Let us now show by induction that whenever $\mathbf{Passed''} \subseteq \mathbf{Passed'}$ and $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s})) = \mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s''}))$ then $\mathsf{RITP}(\mathcal{A}, v, \mathbf{s''}, \mathbf{Passed''}) = \mathsf{RITP}(\mathcal{A}, v, \mathbf{s}, \mathbf{Passed'})$.

    If $\mathbf{s''}$ is a leaf state and is v-compatible, then because v is an integer valuation and because $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s})) = \mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s''}))$, then $\mathbf{s}$ is also v-compatible. For the same reason and by Lemma 4.12, $(\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s})))\downarrow_P = (\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s''})))\downarrow_P$ and both calls to RITP return the same result.

    If $\mathbf{s''}$ is a leaf state and is v-incompatible, then for the same reason and by Lemma 4.13, both calls to RITP return the same result.

    Now, let us consider the induction case. Again, if $\mathbf{s''}$ is v-compatible, then because v is an integer valuation and because $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s})) = \mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s''}))$, then $\mathbf{s}$ is also v-compatible, and the intersection line 3 gives the same result. If $\mathbf{s''} \in \mathbf{Passed''}$, then because $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s})) = \mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s''}))$ and $\mathbf{Passed''} \subseteq \mathbf{Passed'}$, we have that $\mathbf{s} \in \mathbf{Passed'}$ as well, and therefore both calls return the same result.

    Now, if $\mathbf{s''} \notin \mathbf{Passed''}$, then for each edge e, because $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s})) = \mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s''}))$, we have that $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{Succ}(\mathbf{s}, e))) = \mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{Succ}(\mathbf{s''}, e)))$ by Lemma 4.11. In addition, because $\mathbf{Passed''} \subseteq \mathbf{Passed'}$ by induction, then we have $\mathbf{Passed''} \cup \{\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s''}))\} \subseteq \mathbf{Passed'} \cup \{\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s}))\}$. Hence, by induction, both calls to RITP return the same result.

  As a consequence, discarding a state because it is equal to a former visited state (line 4) is harmless because it would give the same result. The visit of RITP to that former state explores (by definition) the same tree prefix as TP, up to the integer hulls. Hence, both algorithms explore the same tree prefix (up to the integer hulls), and therefore by induction hypothesis, $\mathsf{RITP}(\mathcal{A}, v, \mathbf{s}, \mathbf{Passed'}) \subseteq \mathsf{TP}(\mathcal{A}, v, \mathbf{s}, \mathbf{Passed})$.

  Finally, since for all $v' \in \mathsf{TP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$, $\mathit{Traces}(v'(\mathcal{A})) = \mathit{Traces}(v(\mathcal{A}))$ from [ALM20], then for all $v' \in \mathsf{RITP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$, it holds that $\mathit{Traces}(v'(\mathcal{A})) = \mathit{Traces}(v(\mathcal{A}))$. □

Proposition 4.15 (Integer completeness of RITP)**.** *Upon termination of* RITP*, if v' is an integer parameter valuation, and $\mathit{Traces}(v'(\mathcal{A})) = \mathit{Traces}(v(\mathcal{A}))$, then $v' \in \mathsf{RITP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$.*

Proof. Let us show that, for any integer parameter valuation v', $v' \in \mathsf{RITP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$ iff $v' \in \mathsf{TP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$. The proof essentially follows the induction case of the proof of Proposition 4.14. That is, given a finite tree prefix of the symbolic reachability tree ending in a symbolic state $\mathbf{s}$, then either RITP explores the same tree prefix, and both algorithms return a similar result—up to their integer hull—which ensures the fact that an *integer* parameter valuation belongs to the result of RITP iff it belongs to the result of TP. Or RITP did not explore that prefix in full because, from a given symbolic state $\mathbf{s}$, RITP found a former state $\mathbf{s}'$ such that $\mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s})) = \mathsf{IH}(\mathsf{Ext}_X^M(\mathbf{s}'))$ and stopped exploring. But, from the induction case in the proof of Proposition 4.14, exploring the tree from $\mathbf{s}'$ would yield the same result as the exploration from $\mathbf{s}$ and would match the result of TP up to the integer hull, and therefore for any *integer* parameter valuation, the results of TP and RITP coincide.

The result finally follows from the fact that for any parameter valuation v', $\mathit{Traces}(v'(\mathcal{A})) = \mathit{Traces}(v(\mathcal{A}))$ iff $v' \in \mathsf{TP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$ [ALM20]. □

Theorem 4.16. *Upon termination of* $\mathsf{RITP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$*, we have:*

(1) *Soundness:*
   - $v \in \mathsf{RITP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$*;*
   - *if* $v' \in \mathsf{RITP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$ *then* $\mathit{Traces}(v'(\mathcal{A})) = \mathit{Traces}(v(\mathcal{A}))$*;*

(2) *Integer completeness: If* v' *is an integer parameter valuation, and* $\mathit{Traces}(v'(\mathcal{A})) = \mathit{Traces}(v(\mathcal{A}))$*, then* $v' \in \mathsf{RITP}(\mathcal{A}, v, \mathbf{Init}(\mathcal{A}), \emptyset)$.

Proof. From Propositions 4.14 and 4.15, and the fact that v is an integer valuation. □

5. Implementation and experiments using Roméo

We implemented the rational-valued algorithms (EF, AF), integer algorithms (IEF, IAF) and mixed algorithms (RIEF, RIAF) in the tool Roméo [LRST09]; polyhedra operations (both convex and non-convex) are handled by the Parma Polyhedra Library (PPL) [BHZ08]. To illustrate the practical value of our work, we study the application of our algorithms to the parametric non-preemptive model proposed in [JLR15] of the classical scheduling problem of [BFSV04] (Section 5.1), to the classical Fischer mutual exclusion protocol with the parametric model proposed in [TY01] (Section 5.2) and to the classical level crossing proposed in [BV03] (Section 5.3). We used a Core i9/Mac OS computer for our experiments.

5.1. **Scheduling benchmark.** The scheduling example of [JLR15] 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 nondeterministic duration in $[10, b]$, where a and b are parameters. Task τ_2 only has a minimal activation time of $2a$ and has a nondeterministic duration in $[18, 28]$ and finally τ_3 is periodic with period $3a$ and a nondeterministic duration in $[20, 28]$. First, we ask (using AF, IAF and RIAF algorithms) for the parameter valuations that ensure that, for all executions, all tasks are executed at least once. Algorithms AF and IAF produce the set of parameter valuations $a > 24 \wedge b \geq 10 \wedge a - b > 14$, while algorithm RIAF produces the set of parameter valuations $a \geq 25 \wedge b \geq 10 \wedge a - b \geq 15$. The three algorithms take comparable time and memory space–of about 1.0 s and 6.0 MiB.

Furthermore, 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 valuations that ensure

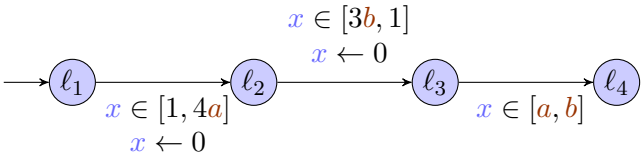


FIGURE 7. A PTA s.t. IEF ℓ_4 is false and EF and RIEF give $a \in [\frac{1}{4}, \frac{1}{3}]$ and $b \in [\frac{1}{4}, \frac{1}{3}]$ and $b \geq a$

that the system does not reach a deadline violation.[6] Algorithm EF does not terminate. Algorithm IEF produces the set of parameter valuations $a \geq 34 \wedge b \geq 10 \wedge a - b \geq 24$ in 13.5 s, while algorithm RIEF produces the set of parameter valuations $a > \frac{562}{17} \wedge b \geq 10 \wedge a - b > \frac{392}{17}$ in 14.3 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 be mitigated using a cache for the sets of parameter valuations generated in computing the integer hulls.

5.2. **Fischer's mutual-exclusion protocol.** In the following example, we first compare the algorithms when they all have a solution. Let us therefore consider a well-known benchmark in the literature of real-time verification, introduced in [Lam87, AL91]. The system consists of n identical processes which access a critical section by mutual exclusion. We use the version of this example proposed in [TY01] where A and B are parameters of the system and the correctness of the protocol depends on their values.

For n processes, mutual exclusion is checked with ROMÉO by the CTL formula $AG\Big(\big(\sum_{i=1}^{n} Critical[i]\big) \leq 1\Big)$ where $Critical[i] = 1$ represents the fact that the process i is in the critical section (0 otherwise). Recall that $AG(\varphi)$ is equivalent to $\neg EF(\neg\varphi)$.

The property holds iff $A > B$ for EF and RIEF and iff $A \geq B + 1$ for IEF. A comparison of computation times and memory consumption for the mutual exclusion checking is given in Table 1. We are in the case where EF finishes and for this example, EF has comparable performance to IEF but with a more accurate result and is more efficient than RIEF while giving the same result. The reason is that for this example, the integer hull used in RIEF adds a computation step without being useful for convergence.

In order to compare AF, IAF and RIAF, we now consider a Fischer protocol model in which the processes only make a single request to the critical section. The property to be checked is that for all executions, all processes obtain the critical section in mutual exclusion. In addition, we change the time interval of the retry from $[A, \infty)$ to $[3 \times A, 2]$, which has the effect of producing the constraints $A \in (0, \frac{2}{3}]$ and $B \in [0, \frac{2}{3})$ and $A - B > 0$ without integer solution.

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

Table 1. Results for Fischer protocol with IEF, EF, RIEF

NB of processes		4	5	6	7	8
IEF	Time (s)	0.3	1.2	5.6	23	91
	Mem. (MB)	4.4	10	36	138	510
	Result			$A - B \geq 1$		
EF	Time (s)	0.3	1.1	5.2	20	88.5
	Mem. (MB)	3.2	10	37	138	516
RIEF	Time (s)	0.3	1.5	6.5	30	118
	Mem. (MB)	4.4	13	47	177	667
	Result			$A - B > 0$		

Table 2. Results for Fischer protocol with IAF, AF, RIAF

NB of processes		4	5	6	7	8
IAF	Time (s)	0	0.1	0.1	0.2	0.4
	Mem. (MB)	0.4	0.9	1	1.8	4.4
	Result			(empty set)		
AF	Time (s)	0.5	4.3	32	238	1757
	Mem. (MB)	5	29	181	1162	6972
RIAF	Time (s)	0.5	3	21	141	975
	Mem. (MB)	4	19	94	503	2756
	Result			$A \in (0, \frac{2}{3}]$ and $A - B > 0$		

For this example, IAF produces an empty set (no integer solution), while RIAF and AF obtain the same result (*i.e.,* $A \in (0, \frac{2}{3}]$ and $A - B > 0$), but RIAF is much more efficient.

5.3. **Level crossing benchmark.** As for AF, it is easy to show, as in Fig. 7 for EF, that there may be no integer valuations of the parameters (IEF returns an empty set) but only non-integer solutions. In the next example, we will exploit this possibility and compare the algorithms when there is no integer solution, and when there is at least one integer solution.

Let us therefore consider as a third benchmark the classical level crossing example. We use the version of this example proposed in [BV03]. The system is obtained by the parallel composition of n train models synchronized with the gate controller model (actions *app*, *exit*, with n fixed) and a gate model (actions *down*, *up*).

We use a parameter a for the approach time of trains. This time is either $2a$ or is in the interval $[a-1, a+1]$ with $a \leq 10$. We use two parameters b and c for the dynamic of opening and closing of the gate. With EF formulae, we ask for the parameter valuations that ensure that the system does not reach a state such that a train is in front of an open gate.

We consider two cases: first, in case i, we choose the parameters b and c such that the parameter valuations to never have a train in front of an open gate are all non-integers. For this we use a lowering time in $[1, 5c]$, a delay to start the opening in $[3b, 5]$ and an opening time in $[c, b]$. Then, in case ii we modify the lowering time and the delay to start respectively into $[1, c]$ and $[2b, 5]$ for which there are also integer solutions.

TABLE 3. Results for level crossing

		i) without integer solution			ii) with integer solution		
	Trains	2	3	4	2	3	4
IEF	Time (s)	< 1	< 1	< 1	< 1	7	948
	Mem. (MB)	< 1	< 1	< 1	< 1	77	3735
	Result	(empty set)			$a \in [9, 10]$, $b \in [1, 2]$, $c = 1$ $a - b \geq 8$		
EF	Time (s)	2	19	**T.O.**	1	11	**T.O.**
	Mem. (MB)	< 1	203	**T.O.**	< 1	130	**T.O.**
RIEF	Time (s)	< 1	4	7	1	14	2920
	Mem. (MB)	< 1	63	168	< 1	269	13706
	Result	$a \in (\frac{36}{5}, 10]$, $b \in [\frac{1}{5}, \frac{5}{3}]$, $c \in [\frac{1}{5}, \frac{2}{3})$, $b - c \geq 0$, $b - a + 5 \times c < -6$			$a \in (8, 10]$, $b \in [1, \frac{5}{2}]$, $c \in [1, 2)$, $b - c \geq 0$ $b - a + c < -6$		

We will consider this example for $n \in \{2, 3, 4\}$ trains. Adding trains increases the explosion of the state space but does not change the parameter set result.

For the case i, Algorithm IEF produces "$\emptyset$" as a result (*i.e.,* the parameter valuations set is empty) and the result by both EF and RIEF is: $a \in (\frac{36}{5}, 10] \wedge b \in [\frac{1}{5}, \frac{5}{3}] \wedge c \in [\frac{1}{5}, \frac{2}{3}) \wedge b - c \geq 0 \wedge b - a + 5 \times c < -6$.

The case ii enlarges the space of solutions allowing some integer valuations.

The result of both EF and RIEF is $a \in (8, 10] \wedge b \in [1, \frac{5}{2}] \wedge c \in [1, 2) \wedge b - c \geq 0 \wedge b - a + c < -6$ while the result of IEF is: $a \in [9, 10] \wedge b = [1, 2] \wedge c \in [1, 1] \wedge a - b \geq 8$.

The computation times and memory space used are given in Table 3, where "**T.O.**" denotes no termination after 3 hours. The results show that RIEF is more precise than IEF and in some cases even provides valuations while IEF produces an empty set—which is a main advantage. Furthermore Table 3 shows that RIEF is most of the time faster than EF and uses less memory. Although the parameters are bounded in this example, we stopped EF (for 4 trains) after 3 hours of computation without knowing whether the algorithm would eventually terminate or not. This is not a surprise since EF has no guarantee of termination, whereas RIEF, IEF, RIAF and IAF always terminate for bounded parameters.

6. IMPLEMENTATION AND EXPERIMENTS USING IMITATOR

Regarding trace preservation, we implemented the rational-valued algorithm (TP) and its integer-complete counterpart (RITP) in the IMITATOR model checker [And21]. As in ROMÉO, polyhedra operations are handled by PPL [BHZ08].

Note that, from Theorem 4.16, RITP cannot improve the result of TP—but can potentially ensure termination where TP would not terminate.

Experiments were conducted on a Dell Precision 5570 with an Intel® Core™ i7-12700H with 32 GiB memory running Linux Mint 21 Vanessa. We used IMITATOR 3.4-alpha2 "Cheese Durian". Binary, models and expected results together with a reproducibility script are available on the long-term archiving platform Zenodo: `https://doi.org/10.5281/zenodo.13779414`.

Gaining termination. First, we apply TP and RITP to the example in Fig. 4a. Applying TP leads to an infinite loop in the algorithm due to the diverging parameter constraints, and therefore (as expected) TP does not terminate. In contrast, RITP terminates in 0.011 second, with the expected result:

$$0 \leq p_1 \leq p_2 \leq 5.$$

While this is not guaranteed by Theorem 4.16 (which only ensures integer completeness), the implementation of RITP in IMITATOR synthesizes for this example not only all integer parameter valuations, but also all rational valuations. In other words, the synthesised result is the maximal result.

Scalability test. We also conducted a scalability test to evaluate the overhead induced by the computation of the integer hull when it brings actually no gain at all (*i.e.,* when comparing the integer hull of the states, instead of the actual states, does not bring any gain in terms of size of the state space, and therefore neither in terms of termination). We use as benchmark for comparison a model of the Carrier Sense Multiple Access/Collision Detection (CSMA/CD) protocol [KNSW07] belonging to the IMITATOR benchmarks library [AMvdP21]. This protocol is a network protocol used in Ethernet to manage data transmission by detecting collisions on a shared communication channel. The idea is as follows: devices first check whether the channel is free, then transmit and, if a collision occurs, they stop, wait for a random time (bounded by an exponential bound), and try again. The model is made of the parallel composition of three automata, and has three parameters: λ (which encodes the length of a message), σ (which encodes the propagation time of a message), and TS (which encodes the duration of a time slot). The models are parametrised by the value of the exponential backoff ("*bc*"); the size of the model grows exponentially with *bc*.

We fix as reference valuation v such that $v(\lambda) = 808$, $v(\sigma) = 26$ and $v(TS) = 52$. When calling TP or RITP with this parameter valuation, the number of visited states is the same in both algorithms for each value of *bc*, and the result is exactly the same for any value of *bc*:

$$15 \times TS < \lambda < 16 \times TS \wedge 0 < \sigma < TS.$$

We give the results in Table 4, where "**T.O.**" denotes no termination after 5 minutes. We give from left to right the exponential backoff (*i.e.,* the scalability factor of this benchmark), the numbers of automata, of clocks, of parameters of the model, the number of locations for all three automata, the number of symbolic states explored (identical for TP and RITP); and the computation time for TP and RITP respectively.

From Table 4, one can infer that the overhead remains acceptable, with a factor almost constant and close to 4 w.r.t. the execution time. While this is not negligible, we believe this is an acceptable price to pay in order to guarantee termination.

7. Conclusion

7.1. **Summary.** We introduced here an extrapolation for symbolic states in parametric timed automata (PTAs) that contains not only clocks but also parameters. We then proposed three algorithms that always terminate for PTAs with bounded (but dense) parameter domains, and output symbolic sets of parameter valuations that define dense sets

TABLE 4. Scalability test for RITP when IH does not bring state space reduction

bc	$\lvert\mathcal{A}\rvert$	$\lvert X\rvert$	$\lvert P\rvert$	$\lvert L\rvert$	$\lvert\mathbf{S}\rvert$	TP (s)	RITP (s)
1	3	3	3	$3 \times 8 \times 8$	511	0.19	0.74
2	3	3	3	$3 \times 17 \times 17$	979	0.38	1.50
3	3	3	3	$3 \times 34 \times 34$	2,249	0.74	2.97
4	3	3	3	$3 \times 67 \times 67$	6,017	1.88	7.70
5	3	3	3	$3 \times 132 \times 132$	16,221	4.95	20.69
6	3	3	3	$3 \times 261 \times 261$	37,177	12.28	47.48
7	3	3	3	$3 \times 518 \times 518$	79,637	28.18	107.35
8	3	3	3	$3 \times 1031 \times 1031$	165,105	72.72	**T.O.**

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 rational-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.

Our experiments using ROMÉO and IMITATOR show that our algorithms ensure termination for case studies that were otherwise unsolvable using these model checking engines, and incur an acceptable overhead otherwise.

7.2. **Future works.** We proposed a first M-extrapolation for PTAs; this can serve as a basis for further developments, *e.g.,* using better extrapolation operators such as LU, local-LU or local-diagonal-LU abstractions, following the recent line of works on extrapolations and abstractions over timed automata zones (*e.g.,* [HSW13, HSW16, BGH+22]). 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.

Furthermore, investigating the possible combination of the integer hull with recent heuristics specifically designed for parametric timed formalisms, such as state merging over symbolic states (*e.g.,* [AMPvdP22]) or more complex versions of our parametric timed extrapolation [AA25] (proposed after [ALR15]), is on our agenda.

Finally, 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.

ACKNOWLEDGMENT

We would like to thank the three anonymous reviewers for useful comments. We would like to thank an anonymous reviewer for a meaningful remark on a preliminary version of this paper together with the suggestion of the example in Fig. 1. We finally thank Johan Arcile for useful remarks on the manuscript.

References

[AA25] Johan Arcile and Étienne André. Zone extrapolations in parametric timed automata. *Innovations in Systems and Software Engineering*, 21:707–726, 2025. doi:10.1007/s11334-024-00554-5.

[ACD93] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, May 1993. doi:10.1006/inco.1993.1024.

[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, 10 2009. doi:10.1142/S0129054109006905.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994. doi:10.1016/0304-3975(94)90010-8.

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *STOC*, pages 592–601, New York, NY, USA, 1993. ACM. doi:10.1145/167088.167242.

[AL91] Martín Abadi and Leslie Lamport. An old-fashioned recipe for real time. In J. W. de Bakker, Cornelis Huizing, Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX Workshop on Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 1991. doi:10.1007/BFB0031985.

[ALM20] Étienne André, Didier Lime, and Nicolas Markey. Language preservation problems in parametric timed automata. *Logical Methods in Computer Science*, 16, 1 2020. URL: https://lmcs.episciences.org/6042.

[ALR15] Étienne André, Didier Lime, and Olivier H. Roux. Integer-complete synthesis for bounded parametric timed automata. In Mikołaj Bojańczyk, Sławomir Lasota, and Igor Potapov, editors, *RP*, volume 9328 of *Lecture Notes in Computer Science*, pages 7–19. Springer, 9 2015. doi:10.1007/978-3-319-24537-9_2.

[ALR18] Étienne André, Didier Lime, and Mathias Ramparison. TCTL model checking lower/upper-bound parametric timed automata without invariants. In David N. Jansen and Pavithra Prabhakar, editors, *FORMATS*, volume 11022 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2018. doi:10.1007/978-3-030-00151-3_3.

[ALR22] Étienne André, Didier Lime, and Olivier H. Roux. Reachability and liveness in parametric timed automata. *Logical Methods in Computer Science*, 18(1):31:1–31:41, February 2022. doi:10.46298/lmcs-18(1:31)2022.

[AMPvdP22] Étienne André, Dylan Marinho, Laure Petrucci, and Jaco van de Pol. Efficient convex zone merging in parametric timed automata. In Sergiy Bogomolov and David Parker, editors, *FORMATS*, volume 13465 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2022. doi:10.1007/978-3-031-15839-1_12.

[AMvdP21] Étienne André, Dylan Marinho, and Jaco van de Pol. A benchmarks library for extended timed automata. In Frédéric Loulergue and Franz Wotawa, editors, *TAP*, volume 12740 of *Lecture Notes in Computer Science*, pages 39–50. Springer, 2021. doi:10.1007/978-3-030-79379-1_3.

[And19] Étienne André. What's decidable about parametric timed automata? *International Journal on Software Tools for Technology Transfer*, 21(2):203–219, April 2019. doi:10.1007/s10009-017-0467-0.

[And21] Étienne André. IMITATOR 3: Synthesis of timing parameters beyond decidability. In Rustan Leino and Alexandra Silva, editors, *CAV*, volume 12759 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2021. doi:10.1007/978-3-030-81685-8_26.

[APP13] Étienne André, Laure Petrucci, and Giuseppe Pellegrino. Precise robustness analysis of time Petri nets with inhibitor arcs. In Víctor Braberman and Laurent Fribourg, editors, *FORMATS*, volume 8053 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 8 2013. doi:10.1007/978-3-642-40229-6_1.

[BBBČ16] Peter Bezděk, Nikola Beneš, Jiří Barnat, and Ivana Černá. LTL parameter synthesis of parametric timed automata. In Rocco De Nicola and eva Kühn, editors, *SEFM*, volume 9763 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2016. doi:10.1007/978-3-319-41591-8_12.

[BBLP06] Gerd Behrmann, Patricia Bouyer, Kim Guldstrand 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. doi:10.1007/s10009-005-0190-0.

[BBLS15] Nikola Beneš, Peter Bezděk, Kim Gulstrand Larsen, and Jiří Srba. Language emptiness of continuous-time parametric timed automata. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 69–81. Springer, July 2015. doi:10.1007/978-3-662-47666-6_6.

[BDR08] Véronique Bruyère, Emmanuel Dall'Olio, and Jean-Francois Raskin. Durations and parametric model-checking in timed automata. *ACM Transactions on Computational Logic*, 9(2):12:1–12:23, 2008. doi:10.1145/1342991.1342996.

[BFSV04] Giacomo Bucci, Andrea Fedeli, Luigi Sassoli, and Enrico Vicario. Timed state space analysis of real-time preemptive systems. *IEEE Transactions on Software Engineering*, 30(2):97–111, 2004. doi:10.1109/TSE.2004.1265815.

[BGH+22] Patricia Bouyer, Paul Gastin, Frédéric Herbreteau, Ocan Sankur, and B. Srivathsan. Zone-based verification of timed automata: Extrapolations, simulations and what next? In Sergiy Bogomolov and David Parker, editors, *FORMATS*, volume 13465 of *Lecture Notes in Computer Science*, pages 16–42. Springer, 2022. doi:10.1007/978-3-031-15839-1_2.

[BHZ05] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. Not necessarily closed convex polyhedra and the double description method. *Formal Aspects of Computing*, 17(2):222–257, 2005. doi:10.1007/S00165-005-0061-1.

[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. doi:10.1016/j.scico.2007.08.001.

[BLT09] 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. doi:10.1007/s10703-009-0074-0.

[BMS13] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robustness in timed automata. In Parosh Aziz Abdulla and Igor Potapov, editors, *RP*, volume 8169 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 9 2013. Invited paper. doi:10.1007/978-3-642-41036-9_1.

[BO14] Daniel Bundala and Joël Ouaknine. Advances in parametric real-time reasoning. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *MFCS, Part I*, volume 8634 of *Lecture Notes in Computer Science*, pages 123–134. Springer, 2014. doi:10.1007/978-3-662-44522-8.

[BR07] Véronique Bruyère and Jean-François Raskin. Real-time model-checking: Parameters everywhere. *Logical Methods in Computer Science*, 3(1:7):1–30, 2007. doi:10.2168/LMCS-3(1:7)2007.

[BV03] Bernard Berthomieu and François Vernadat. State class constructions for branching analysis of time Petri nets. In Hubert Garavel and John Hatcliff, editors, *TACAS*, volume 2619 of *Lecture Notes in Computer Science*, pages 442–457. Springer, 2003. doi:10.1007/3-540-36577-X_33.

[BY03] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In Jöörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer, 2003. doi:10.1007/978-3-540-27755-2_3.

[Doy07] Laurent Doyen. Robust parametric reachability for timed automata. *Information Processing Letters*, 102(5):208–213, 2007. doi:10.1016/j.ipl.2006.11.018.

[DT98] Conrado Daws and Stavros Tripakis. Model checking of real-time reachability properties using abstractions. In Bernhard Steffen, editor, *TACAS*, volume 1384 of *Lecture Notes in Computer Science*, pages 313–329. Springer, 1998. doi:10.1007/BFb0054180.

[GH21] Stefan Göller and Mathieu Hilaire. Reachability in two-parametric timed automata with one parameter is EXPSPACE-complete. In Markus Bläser and Benjamin Monmege, editors, *STACS*, volume 187 of *LIPIcs*, pages 36:1–36:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.36.

[HKSW11] Frédéric Herbreteau, Dileep Kini, B. Srivathsan, and Igor Walukiewicz. Using non-convex approximations for efficient analysis of timed automata. In Supratik Chakraborty and Amit

Kumar, editors, *FSTTCS*, volume 13 of *LIPIcs*, pages 78–89. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPIcs.FSTTCS.2011.78.

[HPR94] Nicolas Halbwachs, Yann-Éric Proy, and Pascal Raymond. Verification of linear hybrid systems by means of convex approximations. In Baudouin Le Charlier, editor, *SAS*, volume 864 of *Lecture Notes in Computer Science*, pages 223–237. Springer, 1994. doi:10.1007/3-540-58485-4_43.

[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. doi:10.1016/S1567-8326(02)00037-1.

[HSW13] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Lazy abstractions for timed automata. In Natasha Sharygina and Helmut Veith, editors, *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 990–1005. Springer, 2013. doi:10.1007/978-3-642-39799-8_71.

[HSW16] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Better abstractions for timed automata. *Information and Computation*, 251:67–90, 2016. doi:10.1016/j.ic.2016.07.004.

[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. doi:10.1109/TSE.2014.2357445.

[JLR22] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. Control of real-time systems with integer parameters. *IEEE Transactions on Automatic Control*, 67(1):75–88, 2022. doi:10.1109/TAC.2020.3046578.

[KNSW07] Marta Z. Kwiatkowska, Gethin Norman, Jeremy Sproston, and Fuzhi Wang. Symbolic model checking for probabilistic timed automata. *Information and Computation*, 205(7):1027–1077, 2007.

[Lam87] Leslie Lamport. A fast mutual exclusion algorithm. *ACM Transactions on Computer Systems (TOCS)*, 5(1):1–11, 1987. doi:10.1145/7351.7352.

[LRST09] Didier Lime, Olivier H. Roux, Charlotte Seidner, and Louis-Marie Traonouez. Romeo: A parametric model-checker for Petri nets with stopwatches. In Stefan Kowalewski and Anna Philippou, editors, *TACAS*, volume 5505 of *Lecture Notes in Computer Science*, pages 54–57. Springer, March 2009. doi:10.1007/978-3-642-00768-2_6.

[Mil00] Joseph S. Miller. Decidability and complexity results for timed automata and semi-linear hybrid automata. In Nancy A. Lynch and Bruce H. Krogh, editors, *HSCC*, volume 1790 of *Lecture Notes in Computer Science*, pages 296–309. Springer, 2000. doi:10.1007/3-540-46430-1_26.

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

[TY01] Stavros Tripakis and Sergio Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001. doi:10.1023/A:1008734703554.

[Wan96] Farn Wang. Parametric timing analysis for real-time systems. *Information and Computation*, 130(2):131–150, 1996. doi:10.1006/inco.1996.0086.