Parametric updates in parametric timed automata*

Étienne André[1,2,3**][0000−0001−8473−9555], Didier Lime[4], Mathias Ramparison[1][0000−0001−6764−1214]

[1] Université Paris 13, LIPN, CNRS, UMR 7030, F-93430, Villetaneuse, France
[2] JFLI, CNRS, Tokyo, Japan
[3] National Institute of Informatics, Tokyo, Japan
[4] École Centrale de Nantes, LS2N, CNRS, UMR 6597, France

Abstract. Verification of timed concurrent systems is hard, especially when the exact value of timing constants remains unknown. In this work, we propose a new subclass of Parametric Timed Automata (PTAs) enjoying a decidability result; we allow clocks to be compared to parameters in guards, as in classic PTAs, but also to be updated to parameters. If we update all clocks each time we compare a clock with a parameter and each time we update a clock to a parameter, we obtain a syntactic subclass for which we can decide the EF-emptiness problem ("is the set of parameter valuations for which some given location is reachable in the instantiated timed automaton empty?") and even perform the exact synthesis of the set of rational valuations such that a given location is reachable. To the best of our knowledge, this is the first non-trivial subclass of PTAs, actually even extended with parametric updates, for which this is possible.

1 Introduction

Timed automata (TAs) are a powerful formalism to model and verify timed concurrent systems, both expressive enough to model many interesting systems and enjoying several decidability properties. In particular, the reachability of a discrete state is PSPACE-complete [1]. In TAs, clocks can be compared with constants in guards, and can be updated to 0 along edges. This can model a system where processes synchronise (are reset) together periodically.

Timed automata may turn insufficient to verify systems where the timing constants themselves are subject to some uncertainty, or when they are simply not known at the early design stage. Parametric timed automata (PTAs) [2] address this drawback by allowing parameters (unknown constants) in the timing constraints; this high expressive power comes at the cost of the undecidability of

* This work is partially supported by the ANR national research program PACS (ANR-14-CE28-0002).
** Partially supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST.

most interesting problems. In particular, the basic problem of EF-emptiness ("is the set of valuations for which a given location is reachable in the instantiated timed automaton empty?") is "robustly" undecidable: even for a single rational-valued [20] or integer-valued parameter [2,8], or when only strict constraints are used [15]. A well-known syntactic subclass of PTAs that enjoys limited decidability is L/U-PTAs [17], where the parameters set is partitioned into lower-bound and upper-bound parameters, *i.e.,* parameters that can only be compared to a clock as a lower-bound (resp. upper-bound). The EF-emptiness problem is decidable for L/U-PTAs [17,11] and for PTAs under several restrictions [13]; however, most other problems are undecidable (*e.g.,* [11,21,18,7,4]).

Contributions. We investigate parametric updates, which can model an unknown timing configuration in a system where processes need to synchronise together on common events, as in *e.g.,* programmable controller logic programs with concurrent tasks execution. We show that the EF-emptiness problem is decidable for PTAs augmented with parametric updates, with the additional condition that whenever a clock is compared to a parameter in a guard or updated to a parameter, all clocks must be updated (possibly to parameters)—this gives R-U2P-PTA. This result holds when the parameters are *bounded rationals in guards*, and possibly *unbounded rationals in updates*. Non-trivial decidable subclasses of PTAs are a rarity (to the best of our knowledge, only L/U-PTAs [17] and integer-points (IP-)PTAs [7]); this makes our positive result very welcome. In addition, not only the emptiness is decidable, but *exact synthesis* for bounded rational-valued parameters can be performed—which contrasts with L/U-PTAs and IP-PTAs as synthesis was shown intractable [18,7].

A full version of this paper with all detailed proofs is available at [6].

Related work. Our construction is reminiscent of the parametric difference bound matrices (PDBMs) defined in [22, section III.C] where the author revisit the result of the binary reachability relation over both locations and clock valuations in TAs; however, parameters of [22] are used to bound in time a run that reaches a given location, while we use parameters directly in guards and resets along the run, which make them active components of the run specifically for intersection with parametric guards, key point not tackled in [22].

Allowing parameters in clock updates is inspired by the updatable TA defined in [10] where clocks can be updated not only to 0 ("reset") but also to rational constants ("update"). In [5], we extended the result of [10] by allowing parametric updates (and no parameter elsewhere, *e.g.,* in guards): the EF-emptiness is undecidable even in the restricted setting of bounded rational-valued parameters, but becomes decidable when parameters are restricted to (unbounded) integers.

Synthesis is obviously harder than EF-emptiness: only three results have been proposed to synthesize the exact set of valuations for subclasses of PTAs, but they are all concerned with *integer*-valued parameters [11,18,5]. In contrast, we deal here with (bounded) rational-valued parameters—which makes this result the first of its kind. The idea of updating all clocks when compared to parameters comes from our class of *reset-PTAs* briefly mentioned in [7], but not thoroughly

studied. Finally, updating clocks on each transition in which a parameter appears is reminiscent of initialized rectangular hybrid automata [16], which remains one of the few decidable subclasses of hybrid automata.
Section 2 recalls preliminaries. Section 3 presents R-U2P-PTA along with our decidability result. Section 4 gives a concrete application of our result.

2 Preliminaries

Throughout this paper, we assume a set $\mathbb{X} = \{x_1, \ldots, x_H\}$ of *clocks*, *i.e.,* real-valued variables evolving at the same rate. A clock valuation is $w : \mathbb{X} \to \mathbb{R}_+$. We write $\mathbf{0}$ for the clock valuation that assigns 0 to all clocks. Given $d \in \mathbb{R}_+$, $w+d$ (resp. $w-d$) denotes the valuation such that $(w+d)(x) = w(x)+d$ (resp. $(w-d)(x) = w(x)-d$ if $w(x)-d > 0$, 0 otherwise), for all $x \in \mathbb{X}$. We assume a set $\mathbb{P} = \{p_1, \ldots, p_M\}$ of *parameters*, *i.e.,* unknown constants. A parameter *valuation* v is a function $v : \mathbb{P} \to \mathbb{Q}_+$. We identify a valuation v with the point $(v(p_1), \ldots, v(p_M))$ of $\mathbb{Q}_+^M$. Given $d \in \mathbb{N}$, $v+d$ (resp. $v-d$) denotes the valuation such that $(v+d)(p) = v(p)+d$ (resp. $(v-d)(p) = v(p)-d$ if $v(p)-d > 0$, 0 otherwise), for all $p \in \mathbb{P}$.

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

A *parametric guard* g is a constraint over $\mathbb{X} \cup \mathbb{P}$ defined as the conjunction of inequalities of the form $x \bowtie z$, where x is a clock and z is either a parameter or a constant in $\mathbb{Z}$. A *non-parametric guard* is a parametric guard without parameters (*i.e.,* over $\mathbb{X}$).

Given a parameter valuation v, $v(g)$ denotes the constraint over $\mathbb{X}$ obtained by replacing in g each parameter p with $v(p)$. We extend this notation to an *expression*: a sum or difference of parameters and constants. Likewise, given a clock valuation w, $w(v(g))$ denotes the expression obtained by replacing in $v(g)$ each clock x with $w(x)$. A clock valuation w *satisfies* constraint $v(g)$ (denoted by $w \models v(g)$) if $w(v(g))$ evaluates to true. We say that v *satisfies* g, denoted by $v \models g$, if the set of clock valuations satisfying $v(g)$ is nonempty. We say that g is *satisfiable* if $\exists w, v$ s.t. $w \models v(g)$.

A *parametric update* is a partial function $u : \mathbb{X} \rightharpoonup \mathbb{N} \cup \mathbb{P}$ which assigns to some of the clocks an integer constant or a parameter. For v a parameter valuation, we define a partial function $v(u) : \mathbb{X} \rightharpoonup \mathbb{Q}_+$ as follows: for each clock $x \in \mathbb{X}$, $v(u)(x) = k \in \mathbb{N}$ if $u(x) = k$ and $v(u)(x) = v(p) \in \mathbb{Q}_+$ if $u(x) = p$ a parameter. A non-parametric update is $u_{np} : \mathbb{X} \rightharpoonup \mathbb{N}$. For a clock valuation w and a parameter valuation v, we denote by $[w]_{v(u)}$ the clock valuation obtained after applying $v(u)$.

Given a clock x and a clock valuation w, $\lfloor w(x) \rfloor$ denotes the integer part of $w(x)$ while $frac(w(x))$ denotes its fractional part. We define the same notation for parameter valuations.

We first define a new class of parametric timed automata and further define classic parametric timed automata and timed automata.

Definition 1. *An update-to-parameter PTA (U2P-PTA) $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, L, l_0, \mathbb{X}, \mathbb{P}, \zeta)$, where: i) Σ is a finite set of actions, ii) L is a finite set of locations, iii) $l_0 \in L$*

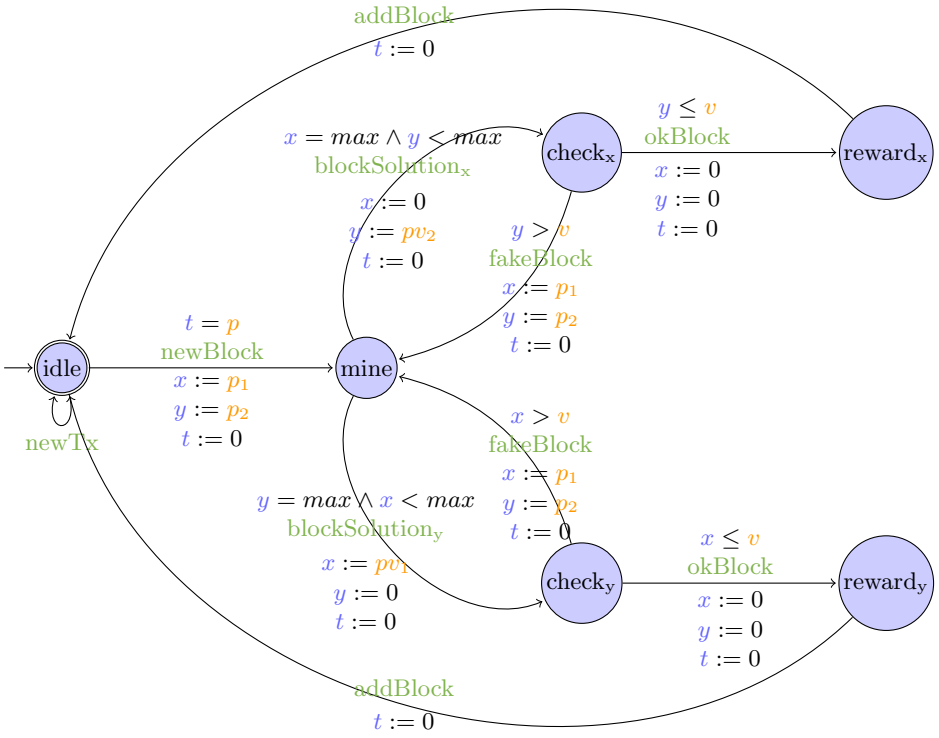

Fig. 1: A proof-of-work
modeled with a bounded R-U2P-PTA.

is the initial location, iv) $\mathbb{X}$ *is a finite set of clocks, v)* $\mathbb{P}$ *is a finite set of parameters, vi)* ζ *is a finite set of edges* $e = \langle l, g, a, u, l' \rangle$ *where* $l, l' \in L$ *are the source and target locations,* g *is a parametric guard,* $a \in \Sigma$ *and* $u : \mathbb{X} \rightharpoonup \mathbb{N} \cup \mathbb{P}$ *is a parametric update function.*

An U2P-PTA is depicted in Figure 1. Note that all clocks are updated whenever there is a comparison with a parameter (as in newBlock) or a clock is updated to a parameter (as in blockSolution$_x$). Given a parameter valuation v, we denote by $v(\mathcal{A})$ the structure where all occurrences of a parameter p_i have been replaced by $v(p_i)$. If $v(\mathcal{A})$ is such that all constants in guards and updates are integers, then $v(\mathcal{A})$ is a *updatable timed automaton* [10] but will be called *timed automaton* (TA) for the sake of simplicity in this paper.
A *bounded* U2P-PTA is a U2P-PTA with a bounded parameter domain that assigns to each parameter a minimum integer bound and a maximum integer bound. That is, each parameter p_i ranges in an interval $[a_i, b_i]$, with $a_i, b_i \in \mathbb{N}$. Hence, a bounded parameter domain is a hyperrectangle of dimension M.
A parametric timed automaton (PTA) [2] is a U2P-PTA where, for any edge $e = \langle l, g, a, u, l' \rangle \in \zeta$, $u : \mathbb{X} \rightharpoonup \{0\}$.

Definition 2 (Concrete semantics of a TA). *Given a U2P-PTA* $\mathcal{A} = (\Sigma, L, l_0, \mathbb{X}, \mathbb{P}, \zeta)$*, and a parameter valuation* v*, the concrete semantics of* $v(\mathcal{A})$ *is given by the timed transition system* $(S, s_0, \rightarrow)$*, with* $S = \{(l, w) \in L \times \mathbb{R}_+^H\}$*,* $s_0 = (l_0, \mathbf{0})$ *and* $\rightarrow$ *consists of the discrete and (continuous) delay transition relations:*

- *discrete transitions:* $(l, w) \overset{e}{\mapsto} (l', w')$*, if* $(l, w), (l', w') \in S$*, there exists* $e = \langle l, g, a, u, l' \rangle \in \zeta$*,* $w' = [w]_{v(u)}$*, and* $w \models v(g)$*.*
- *delay transitions:* $(l, w) \overset{d}{\mapsto} (l, w + d)$*, with* $d \in \mathbb{R}_+$*.*

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

Given a TA $v(\mathcal{A})$ with concrete semantics $(S, s_0, \rightarrow)$, we refer to the states of S as the *concrete states* of $v(\mathcal{A})$. A (concrete) *run* of $v(\mathcal{A})$ is a possibly infinite alternating sequence of concrete states of $v(\mathcal{A})$ and edges starting from s_0 of the form $s_0 \xrightarrow{e_0} s_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{m-1}} s_m \xrightarrow{e_m} \cdots$, such that for all $i = 0, 1, \ldots$, $e_i \in \zeta$, and $(s_i, e_i, s_{i+1}) \in \rightarrow$.

Given a state $s = (l, w)$, we say that s is reachable (or that $v(\mathcal{A})$ reaches s) if s belongs to a run of $v(\mathcal{A})$. By extension, we say that l is reachable in $v(\mathcal{A})$, if there exists a state (l, w) that is reachable.

Throughout this paper, let K denote the largest constant in a given U2P-PTA, *i.e.,* the maximum of the largest constant compared to a clock in a guard and the largest upper bound of a parameter (if the U2P-PTA is bounded).

Let us recall the notion of clock region [1].

Definition 3 (clock region). *For two clock valuations* w *and* w'*,* $\sim$ *is an equivalence relation defined by:* $w \sim w'$ *iff i) for all clocks* x*, either* $\lfloor w(x) \rfloor = \lfloor w'(x) \rfloor$ *or* $w(x), w'(x) > K$*; ii) for all clocks* x, y *with* $w(x), w(y) \leq K$*,* $frac(w(x)) \leq frac(w(y))$ *iff* $frac(w'(x)) \leq frac(w'(y))$*; iii) for all clocks* x *with* $w(x) \leq K$*,* $frac(w(x)) = 0$ *iff* $frac(w'(x)) = 0$*.*

A clock region *is an equivalence class of* $\sim$*.*

Two clock valuations in the same clock region reach the same regions by time elapsing, satisfy the same guards and can take the same transitions [1].

In this paper, we address the *EF-emptiness* problem: **given a U2P-PTA** $\mathcal{A}$ **and a location** l**, is the set of parameter valuations** v **such that** l **is reachable in** $v(\mathcal{A})$ **empty?**

3 A decidable subclass of U2P-PTAs

We now impose that, whenever a guard or an update along an edge contains parameters, then all clocks must be updated (to constants or parameters). Our main contribution is to prove that this restriction makes EF-emptiness decidable.

Definition 4. *An* R-U2P-PTA *is a U2P-PTA where for any* $\langle l, g, a, u, l' \rangle \in \zeta$, u *is a total function whenever:*[1] *i)* g *is a parametric guard, or ii)* $u(x) \in \mathbb{P}$ *for some* $x \in \mathbb{X}$.

The main idea for proving decidability is the following: given an R-U2P-PTA $\mathcal{A}$ we will construct a finite region automaton that bisimulates $\mathcal{A}$, as in TA [1]. Our regions will contain both clocks and parameters, and will be a finite number. Since parameters are allowed in guards, we need to construct parameter regions and more restricted clock regions. We will define a form of Parametric Difference Bound Matrices (viz., p–PDBMs for precise PDBMs, inspired by [17]) in which, once valuated by a parameter valuation, two clock valuations have the same discrete behavior and satisfy the same non-parametric guards. A p–PDBM will define the *set of clocks and parameter valuations* that satisfies it, while once valuated by a parameter valuation, a valuated p–PDBM will define the *set of clock valuations* that satisfies it. A key point is that in our p–PDBMs the parametric constraints used in the matrix will be defined from a *finite* set of predefined expressions involving parameters and constants, and we will prove that this defines a finite number of p–PDBMs. Decidability will come from this fact. We define this set ($\mathcal{PLT}$ for parametric linear term) as follows: $\mathcal{PLT} = \{frac(p_i), 1 - frac(p_i), frac(p_i) - frac(p_j), frac(p_j) + 1 - frac(p_i), 1, 0, frac(p_i) - 1 - frac(p_j), -frac(p_i), frac(p_i) - 1\}$, for all $1 \leq i, j \leq M$. Given a parameter valuation v and $d \in \mathcal{PLT}$, we denote by $v(d)$ the term obtained by replacing in d each parameter p by $v(p)$. Let us now define an equivalence relation between parameter valuations v and v'.

Definition 5 (regions of parameters). *We write that* $v \frown v'$ *if i) for all parameter* p, $\lfloor v(p) \rfloor = \lfloor v'(p) \rfloor$*; ii) for all* $d_1, d_2, d_3 \in \mathcal{PLT}$, $v(d_1) \leq v(d_2) + v(d_3)$ *iff* $v'(d_1) \leq v'(d_2) + v'(d_3)$*;*

Parameter regions are defined as the equivalence classes of $\frown$. The definition is in a way similar to [1, Definition 4.3] but also involves comparisons of sums of elements of $\mathcal{PLT}$. In fact, we will need this kind of comparisons to define our p–PDBMs. Nonetheless we do not need more complicated comparisons as in R-U2P-PTA whenever a parametric guard or update is met the update is a total function: this preserves us from the parameter accumulation, *e.g.,* obtaining expressions of the form $5frac(p_i) - 1 - 3frac(p_j)$ (that may occur in usual PTAs).

In the following, our p–PDBMs will contain pairs of the form $D = (d, \lhd)$, where $d \in \mathcal{PLT}$. We therefore need to define comparisons on these pairs.

We define an associative and commutative operator $\oplus$ as $\lhd_1 \oplus \lhd_2 = <$ if $\lhd_1 \neq \lhd_2$, or $\lhd_1$ if $\lhd_1 = \lhd_2$. We define $D_1 + D_2 = (d_1 + d_2, \lhd_1 \oplus \lhd_2)$. Following the idea of parameter regions, we define the *validity* of a comparison between pairs

[1] In the following we only consider either non-parametric, or (necessarily total) fully parametric update functions. A total update function which is not fully parametric (*i.e.,* an update of some clocks to parameters and all others to constants) can be encoded as a total fully parametric update immediately followed by a (partial) non-parametric update function.

of the form $(d_i, \lhd_i)$ within a given parameter region, *i.e.,* whether the comparison is *true* for all parameter valuations v in the parameter region R_p.

Definition 6 (validity of comparison). *Let R_p be a parameter region. Given any two linear terms d_1, d_2 over $\mathbb{P}$ (i.e., of the form $\sum_i \alpha_i p_i + d$ with $\alpha_i, d \in \mathbb{Z}$), the comparison $(d_1, \lhd_1) \lhd (d_2, \lhd_2)$ is* valid for R_p *if:*

1. *$\lhd = <$, and either i) for all $v \in R_p$, $v(d_1) < v(d_2)$ evaluates to true, or ii) for all $v \in R_p$, $v(d_1) \leq v(d_2)$ evaluates to true, $\lhd_1 = <$ and $\lhd_2 = \leq$;*
2. *$\lhd = \leq$, and either i) for all $v \in R_p$, $v(d_1) < v(d_2)$ evaluates to true, or ii) for all $v \in R_p$, $v(d_1) \leq v(d_2)$ evaluates to true, and $\lhd_1 = \lhd_2$, or $\lhd_1 = <$;*

Transitivity is immediate from the definition: if $D_1 \lhd_1 D_2$ and $D_2 \lhd_2 D_3$ are valid for R_p, $D_1(\lhd_1 \oplus \lhd_2)D_3$ is valid for R_p.

We can now define our data structure, namely p–PDBMs, inspired by the PDBMs of [17] themselves inspired by DBMs [14]. However, our p–PDBM compare differences of *fractional parts* of clocks, instead of clocks as in classical DBMs; therefore, our p–PDBMs are closer to clock regions than to DBMs and *fully contained* into clock regions of [1] A p–PDBM is a pair made of an integer vector (encoding the clocks integer part), and a matrix (encoding the parametric differences between any two clock fractional parts). Their interpretation also follows that of PDBMs and DBMs: for $i \neq 0$, the matrix cell $D_{i,0} = (d_{i,0}, \lhd_{i0})$ is interpreted as the constraint $frac(x_i) \lhd_{i0} d_{i,0}$, and $D_{0,i} = (d_{0,i}, \lhd_{0i})$ as the constraint $-frac(x_i) \lhd_{0i} d_{0,i}$. For $i \neq 0$ and $j \neq 0$, the matrix cell $D_{i,j} = (d_{i,j}, \lhd_{ij})$ is interpreted as $frac(x_i) - frac(x_j) \lhd_{ij} d_{i,j}$.

Definition 7 (p–PDBM). *Let R_p be a parameter region. A p–PDBM for R_p is a pair (E, D) with $E = (E_1, \cdots, E_H)$ a vector of H integers (or ∞ when it exceeds a possible upper-bound) which is the integer part of each clock, and D is an $(H+1)^2$ matrix where each element $D_{i,j}$ is a pair $(d_{i,j}, \lhd_{ij})$ for all $0 \leq i, j \leq H$, where $d_{i,j} \in \mathcal{PLT}$. Moreover, for all $0 \leq i \leq H$, $D_{i,i} = (0, \leq)$. In addition, for all i, j, k:*

1. *$(-1, <) \leq D_{0,i} \leq (0, \leq)$ and $(0, \leq) \leq D_{i,0} \leq (1, <)$ are valid for R_p,*
2. *For all $i \neq 0, j \neq 0$, either $(0, \leq) \leq D_{i,j} \leq (1, <)$ is valid for R_p and $(-1, <) \leq D_{j,i} \leq (0, \leq)$ is valid for R_p or $(0, \leq) \leq D_{j,i} \leq (1, <)$ is valid for R_p and $(-1, <) \leq D_{i,j} \leq (0, \leq)$ is valid for R_p.*
3. *$D_{i,j} \leq D_{i,k} + D_{k,j}$ is valid for R_p (canonical form).*
4. *If $d_{i,j} = -d_{j,i}$ and $d_{i,j} \neq \pm 1$ then $\lhd_{ij} = \lhd_{ji} = \leq$, else $\lhd_{ij} = \lhd_{ji} = <$,*

The use of *validity* ensures the consistency of the p–PDBM. We denote the set of all p–PDBMs that are *valid for* R_p by $p\text{–}\mathcal{PDBM}(R_p)$. Given a p–PDBM (E, D), it defines the subset of $\mathbb{R}^H \cup \mathbb{Q}^M$ satisfying the constraints $\bigwedge_{i,j \in [0,H]} frac(x_i) - frac(x_j) \lhd_{i,j} d_{i,j} \wedge \bigwedge_{i \in [1,H]} \lfloor x_i \rfloor = E_i$.

Given a parameter valuation v, we denote by $(E, v(D))$ the *valuated* p–*PDBM*, *i.e.,* the set of clock valuations defined by:

$$\bigwedge_{i,j \in [0,H]} frac(x_i) - frac(x_j) \lhd_{i,j} v(d_{i,j}) \wedge \bigwedge_{i \in [1,H]} \lfloor x_i \rfloor = E_i.$$

For a clock valuation w, we write $w \in (E, v(D))$ if it satisfies all constraints of $(E, v(D))$. Intuitively, our p–PDBMs are partitioned into three types.

(1) The *point* p*–PDBM* is a clock region defined by *only parameters* which contains only one clock valuation; it represents the unique clock valuation (for a given parameter valuation) obtained after a total parametric update in an U2P-PTA. Each clock is valuated to a parameter and each difference of clocks is valuated to a difference of parameters (it corresponds to constraints of the form $x = p$ and $x - y = p_i - p_j$).

Let v be a parameter valuation. We assume $\lfloor v(p_2) \rfloor = \lfloor v(p_1) \rfloor = k \in \mathbb{N}$ and $frac(v(p_1)) > frac(v(p_2))$. The p–PDBM obtained after an update $u(x) = v(p_2)$ and $u(y) = v(p_1)$ is represented using the following pair (where the indices $\mathbf{0}, \mathbf{x}, \mathbf{y}$ are shown for the sake of comprehension)

$$(E, D) = \left(\binom{k}{k}, \begin{pmatrix} & \mathbf{0} & \mathbf{x} & \mathbf{y} \\ \mathbf{0} & (0, \leq) & (-frac(p_2), \leq) & (-frac(p_1), \leq) \\ \mathbf{x} & (frac(p_2), \leq) & (0, \leq) & (frac(p_2) - frac(p_1), \leq) \\ \mathbf{y} & (frac(p_1), \leq) & (frac(p_1) - frac(p_2), \leq) & (0, \leq) \end{pmatrix} \right)$$

Once valuated with v, it contains a unique clock valuation. We represent it as the black dot in Figure 2.

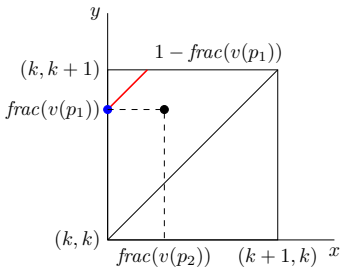

Fig. 2: Graphical representations of p–PDBMs and [1] regions

(2) In contrast, a *border* p*–PDBM* is a clock region which can contain several clock valuations satisfying some possibly parametric constraints, or contain at least one clock valuation satisfying non-parametric constraints (as the corner-point region of [1]). In particular, the initial clock region $\{0^H\}$ and any clock region that is a single integer clock valuation is a p–PDBM. A *border* p*–PDBM* is characterized by at least one clock x s.t. $D_{x,0} = D_{0,x} = (0, \leq)$ and can be seen as a subregion of an open line segment or a corner point region of [1, fig. 9 example 4.4]. After an immediate update of x to k, the above p–PDBM (E, D) becomes

$$(E, D) = \left(\binom{k}{k}, \begin{pmatrix} & \mathbf{0} & \mathbf{x} & \mathbf{y} \\ \mathbf{0} & (0, \leq) & (0, \leq) & (-frac(p_1), \leq) \\ \mathbf{x} & (0, \leq) & (0, \leq) & (-frac(p_1), \leq) \\ \mathbf{y} & (frac(p_1), \leq) & (frac(p_1), \leq) & (0, \leq) \end{pmatrix} \right)$$

We represent it once valuated with v as the blue dot in Figure 2. The open line segment of [1, fig. 9 example 4.4] can be represented as

$$\left(\binom{k}{k}, \begin{pmatrix} & \mathbf{0} & \mathbf{x} & \mathbf{y} \\ \mathbf{0} & (0, \leq) & (0, \leq) & (0, <) \\ \mathbf{x} & (0, \leq) & (0, \leq) & (0, <) \\ \mathbf{y} & (1, <) & (1, <) & (0, \leq) \end{pmatrix} \right)$$

and is depicted as the vertical left black line in Figure 2.

(3) A *center* p*–PDBM* is a clock region which can contain several clock valuations satisfying some possibly parametric constraints (as the open region of [1]). A *center* p*–PDBM* is characterized by at least one clock y s.t. $D_{y,0} = (1, <)$ and for all x s.t. $D_{0,x} = (0, \lhd_{ox})$, then we have $\lhd_{ox} = <$ and can be seen as a subregion of an open region of [1, fig. 9 example 4.4]. After some time elapsing,

and *before* any clock valuation reaches the next integer $k+1$—therefore the next *border* p*–PDBM*—, the above p–PDBM (E, D) becomes

$$(E, D) = \left(\binom{k}{k} , \begin{pmatrix} & \mathbf{0} & \mathbf{x} & \mathbf{y} \\ \mathbf{0} & (0, \leq) & (0, <) & (-frac(p_1), <) \\ \mathbf{x} & (1 - frac(p_1), <) & (0, \leq) & (-frac(p_1), \leq) \\ \mathbf{y} & (1, <) & (frac(p_1), \leq) & (0, \leq) \end{pmatrix} \right)$$

We represent it once valuated with v as the red line in Figure 2. The open region of [1, fig. 9 example 4.4] can be represented as

$$\left(\binom{k}{k} , \begin{pmatrix} & \mathbf{0} & \mathbf{x} & \mathbf{y} \\ \mathbf{0} & (0, \leq) & (0, <) & (0, <) \\ \mathbf{x} & (1, <) & (0, \leq) & (0, <) \\ \mathbf{y} & (1, <) & (1, <) & (0, \leq) \end{pmatrix} \right)$$

and is depicted as the top left black triangle in Figure 2.

Remark that sets of the form $\{frac(w(x)) \mid 0 \leq frac(w(x)) \leq 1\}$ are in contradiction with Definition 7 (4) and therefore cannot be part of a p–PDBM, as in the regions of [1]. Basically, only the first p–PDBM after a (necessarily total) parametric clock update will be a *point* p*–PDBM*; any following p–PDBM will be a *border* p*–PDBM* or a *center* p*–PDBM* until the next (total) parametric update.

The differentiation made in the previous paragraph between *border* p*–PDBM* and *center* p*–PDBM* is intended to give an intuition to the reader about the inclusion of p–PDBMs into [1] clock regions. Technical details are not relevant for a good understanding of this paper but are given in[6].

In the following Section 3.1, we are going to define operations on p–PDBMs (*i.e.,* update of clocks, time elapsing and guards satisfaction), and will show that the set of p–PDBMs is stable under these operations.

3.1 Operations on p–PDBMs

Non-parametric update. To apply a non-parametric update on a p–PDBM, following classical algorithms for DBMs [9], we define an update operator.

Given a p–PDBM (E, D) and u_{np} a non-parametric update function that updates a clock x to $k \in \mathbb{N}$, $update((E, D), u_{np})$ defines a new p–PDBM by *i*) updating E_x to k; *ii*) setting the fractional part of x to 0: $D_{x,0} := D_{0,x} := (0, \leq)$; *iii*) updating the new difference between fractional parts with all other clocks i, which is the range of values i can currently take: $D_{x,i} := D_{0,i}$ and $D_{i,x} := D_{i,0}$.

Intuitively, we update in (E, D) the lower and upper bounds of some clocks to $(0, \leq)$ and the difference between two clocks $D_{i,j}$ to $D_{0,j}$ if x_i is updated: that is, the new difference between two clocks if one has been updated is just the lower/upper bound of the one that is not updated. This allows us to conserve the canonical form as we only "moved" some cells in D that already verified the canonical form. Therefore $update((E, D), u_{np})$ is a p–PDBM.

The following lemma states that the update operator behaves as expected.

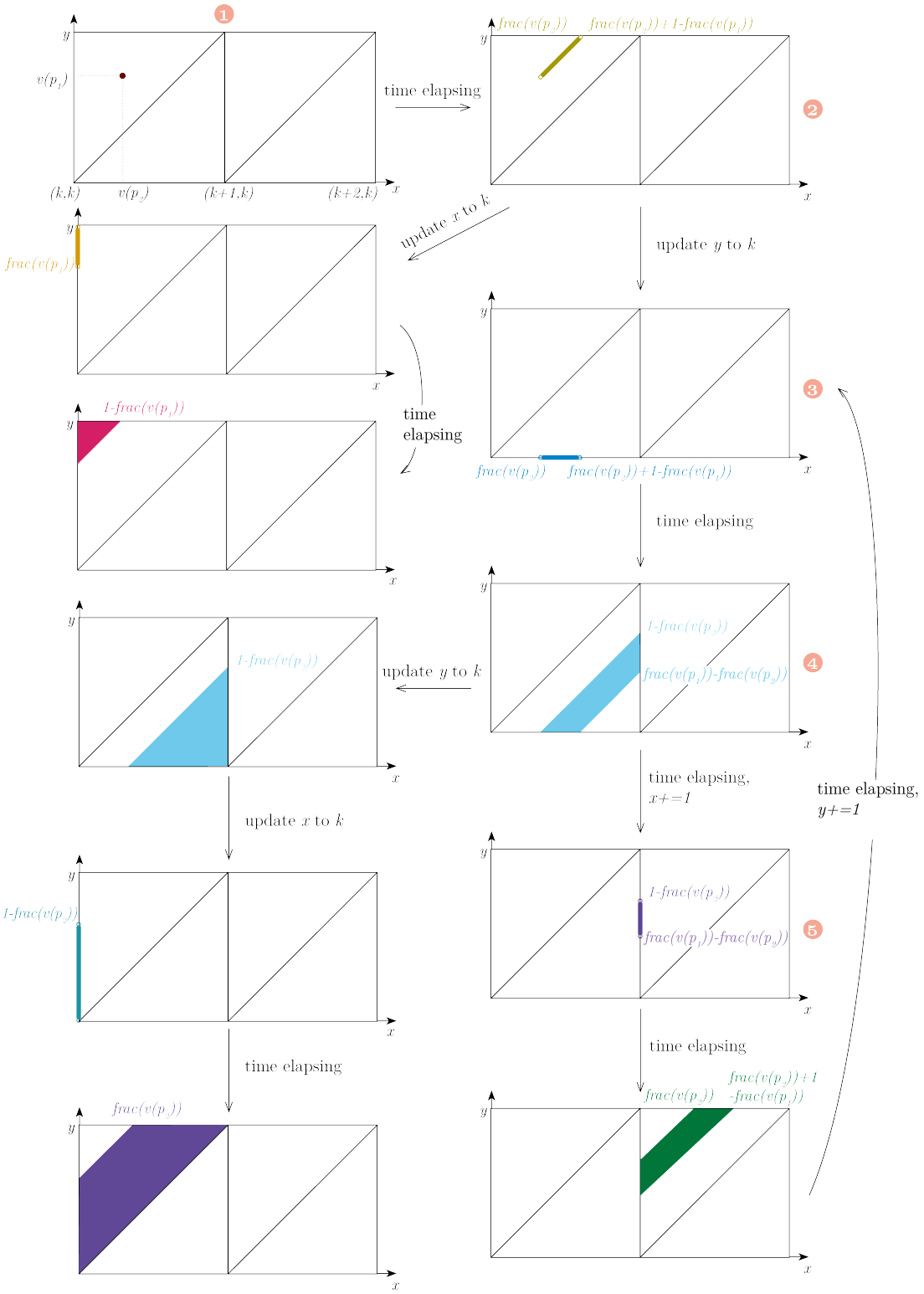

Fig. 3: Representation of p–PDBMs in two dimensions with two clocks x, y, two parameters p_1, p_2 and v s.t. $\lfloor v(p_1) \rfloor = \lfloor v(p_2) \rfloor$ and $frac(v(p_1)) > frac(v(p_2))$.

Lemma 1 (semantics of *update* on p–$\mathcal{PDBM}(R_p)$). *Let R_p be a parameter region and $(E, D) \in p$–$\mathcal{PDBM}(R_p)$. Let $v \in R_p$. Let u_{np} be a non-parametric update. For all clock valuations w, $w \in update((E, v(D)), u_{np})$ iff $w' \in (E, v(D))$ for some w' s.t. $w = [w']_{u_{np}}$.*

Proof idea. The technical part is ($\Rightarrow$). The idea is to prove that, given $w' \in update((E, v(D)), u_{np})$ there is a non-empty set of clock valuations w s.t. $w' = [w]_{u_{np}}$ that is precisely defined by the constraints in $(E, v(D))$.

Parametric update. Given $(E, D) \in p$–$\mathcal{PDBM}(R_p)$ we write $\overline{update}((E, D), u)$ to denote the update of (E, D) by u, when u is a total parametric update function, *i.e.,* updating the set of clocks exclusively to parameters. We therefore obtain a *point* p–PDBM, containing the parametric set of constraints defining a unique clock valuation. The semantics is straightforward.

Time elapsing. Given a parameter region R_p, recall that constraints satisfied by parameters are known, and we can order elements of $\mathcal{PLT}$. Thanks to this order, within a p–PDBM (E, D) the clocks with the (possibly parametric) largest fractional part *i.e.,* the clocks that have a larger fractional part than any other clock, can always be identified by their bounds in D. For a p–PDBM (E, D), we define the set of clocks with the largest fractional part (LFP) as $\mathsf{LFP}_{R_p}(D) = \{x \in [1, H] \mid 0 \leq D_{x,i}$ is valid for R_p, for all $0 \leq i \leq H\}$. Clocks belonging to LFP are the first to reach the upper bound 1 by letting time elapse.

Note that several clocks may have the largest fractional parts (up to some syntactic replacements , in that case they satisfy the same constraints in (E, D)).

Let $(E, D) \in p$–$\mathcal{PDBM}(R_p)$ and $x \in \mathsf{LFP}_{R_p}(D)$. To formalize time elapsing until the largest fractional part $frac(x)$ reaches 1, we define a time elapsing operator that will decline in two variants depending on the input: *border* p–*PDBM* or *center/point* p–*PDBM*.

Given a *border* p–*PDBM* (E, D) with $E_x = k$, $TE((E, D))$ defines a new *center* p–*PDBM* by *i*) setting $D_{x,0} := (1, <)$ as x is the first one that will reach $k + 1$; *ii*) updating the upper bound of all other clocks i, which has increased: $D_{i,0} := D_{i,x} + (1, <)$; *iii*) updating all lower bounds as they have to leave the *border*: $D_{0,i} := D_{0,i} + (0, <)$ (x included). This gives the range of possible clock valuations *before* $frac(x)$ reaches 1. Intuitively it represents the transformation from an open line segment or the corner-point region of [1] into an open region of [1].

The time elapsing operator also operates the transformation from an open region of [1] to the upper open line segment or the corner-point region of [1]. Given a *center/point* p–*PDBM* (E, D) where $E_x = k$, $TE((E, D))$ defines a new *border* p–*PDBM* by *i*) setting $D_{x,0} := D_{0,x} := (0, \leq)$ (intuitively both became $(1, \leq)$) and $E_x = k + 1$ (if $E_x \leq K + 1$), as x is now in the upper *border*; *ii*) updating the upper and lower bounds of all other clocks i: $D_{i,0} := D_{i,x} + (1, \leq)$ and $D_{0,i} := D_{x,i} + (-1, \leq)$; *iii*) updating the new difference between fractional parts with all other clocks i, which is the range of values i can currently take (as in the update operator): $D_{x,i} := D_{0,i}$ and $D_{i,x} := D_{i,0}$.

Although we perform some additions such as $D_{j,i} + (1, <)$, we do not create new expressions that are not in $\mathcal{PLT}$. In fact, this addition is performed on a

negative term (*e.g.*, $frac(p) - 1$), as x_i is a clock with the largest fractional part and adding 1 transforms it into another term of $\mathcal{PLT}$. The intuition is similar when performing additions such as $D_{i,j} + (-1, \leq)$: as x_i is a clock with the largest fractional part, $d_{i,j}$ is a positive term. The canonical form is also preserved by the last setting operations of the algorithm, as in the update operator. Therefore $TE((E, D))$ is a p–PDBM.

Proposition 1 (semantics of p–PDBM under TE). *Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$. Let $v \in R_p$. There exists $w' \in TE((E, v(D)))$ iff there exist $w \in (E, v(D))$ and a delay δ s.t. $w' = w + \delta$.*

Proof idea. This proof is quite technical. Intuitively, we bound the difference of each upper bound $v(d_{i,0})$ and $w(x_i)$ and each lower bound $v(d_{0,i})$ and $w(x_i)$. This allows us to take a delay δ inside these bounds that allows us to reach the next p–PDBM.

Running example: Figure 3 represents graphically different p–PDBMs obtained after an update $u(x) = v(p_2)$ and $u(y) = v(p_1)$ (figure 1). Time elapsing before $y \in \mathsf{LFP}$ reaches the next integer gives the *center* p–PDBM (figure 2)

$$(E, D) = \left(\binom{k}{k}, \begin{pmatrix} & \mathbf{0} & \mathbf{x} & \mathbf{y} \\ \mathbf{0} & (0, \leq) & (-frac(p_2), <) & (-frac(p_1), <) \\ \mathbf{x} & (frac(p_2) + 1 - frac(p_1), <) & (0, \leq) & (-frac(p_1) + frac(p_2), \leq) \\ \mathbf{y} & (1, <) & (frac(p_1) - frac(p_2), \leq) & (0, \leq) \end{pmatrix} \right)$$

After an update of y to k prior to reaching $k+1$, the *border* p–PDBM obtained is (figure 3)

$$(E, D) = \left(\binom{k}{k}, \begin{pmatrix} & \mathbf{0} & \mathbf{x} & \mathbf{y} \\ \mathbf{0} & (0, \leq) & (-frac(p_2), <) & (0, \leq) \\ \mathbf{x} & (frac(p_2) + 1 - frac(p_1), <) & (0, \leq) & (frac(p_2) + 1 - frac(p_1), <) \\ \mathbf{y} & (0, \leq) & (-frac(p_2), <) & (0, \leq) \end{pmatrix} \right)$$

Time elapsing before $x \in \mathsf{LFP}$ reaches the next integer gives the *center* p–PDBM (figure 4)

$$(E, D) = \left(\binom{k}{k}, \begin{pmatrix} & \mathbf{0} & \mathbf{x} & \mathbf{y} \\ \mathbf{0} & (0, \leq) & (-frac(p_2), <) & (0, <) \\ \mathbf{x} & (1, <) & (0, \leq) & (frac(p_2) + 1 - frac(p_1), <) \\ \mathbf{y} & (1 - frac(p_2), <) & (-frac(p_2), <) & (0, \leq) \end{pmatrix} \right)$$

When $x \in \mathsf{LFP}$ reaches $k+1$, the *border* p–PDBM obtained is (figure 5)

$$(E, D) = \left(\binom{k+1}{k}, \begin{pmatrix} & \mathbf{0} & \mathbf{x} & \mathbf{y} \\ \mathbf{0} & (0, \leq) & (0, \leq) & (-frac(p_1) + frac(p_2), <) \\ \mathbf{x} & (0, \leq) & (0, \leq) & (-frac(p_1) + frac(p_2), <) \\ \mathbf{y} & (1 - frac(p_2), <) & (1 - frac(p_2), <) & (0, \leq) \end{pmatrix} \right)$$

Non-parametric guard. From [1, Section 4.2] we have that either every clock valuation of a clock region satisfies a guard, or none of them does. Note that a p–PDBM for R_p is contained into a clock region of [1, Section 4.2], therefore we have that if $w \in (E, v(D))$ satisfies a non-parametric guard g, then for all $w' \in (E, v(D))$ we also have w' satisfies g.

Let $v \in R_p$. We define $v \in guard_\forall(g, E, D)$ iff for all $w \in (E, v(D))$, $w \models g$. As any two $v, v' \in R_p$ satisfy the same constraints, it is straightforward that if $v \in guard_\forall(g, E, D)$, then for all $v' \in R_p$, $v' \in guard_\forall(g, E, D)$.

Parametric guard. Using a projection on parameters does not create new constraints on parameters that are not already in a parameter region R_p. Indeed, a parametric guard g only adds new constraints of the form $x \bowtie p$ which gives again a comparison between elements of $\mathcal{PLT}$. Therefore, these new constraints already belong to $\mathcal{PLT}$ and we can decide whether the set of clock valuations satisfying this set of constraints is non-empty *i.e.,* given $v \in R_p$, $v(g)$ is satisfied by some clock valuation $w \in (E, v(D))$. This is a key point in the overall process of proving the decidability of our R-U2P-PTAs. Note that there will also be additional constraints involving clocks (with other clocks, constants or parameters), but they will not be relevant as we immediately update all clocks, therefore replacing these constraints with new constraints encoding the clock updates.

Let $v \in R_p$. We define $v \in p\text{-}guard_{\exists}(g, E, D)$ iff there is a $w \in (E, v(D))$ s.t. $w \models v(g)$.[2] Again, as any two $v, v' \in R_p$ satisfy the same constraints, it is straightforward that if $v \in p\text{-}guard_{\exists}(g, E, D)$, then for all $v' \in R_p$, $v' \in p\text{-}guard_{\exists}(g, E, D)$.

Now that we have defined useful operations on p–PDBMs, we are going, given a parameter region R_p, to construct a finite region automaton in which for any run, there is an equivalent concrete run in the R-U2P-PTA.

3.2 Parametric region automaton

Let $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$, we say $(E', D') \in \mathsf{Succ}((E, D)) \Leftrightarrow \exists\ i \geq 0$ s.t. $(E', v(D')) = TE^i((E, D))$. In other words, (E', D') is obtained after applying $TE((E, D))$ a finite number of times. $\mathsf{Succ}((E, D))$ is also called the *time successors* of (E, D).

In order to finitely simulate an R-U2P-PTA, we create a parametric region automaton.

Definition 8 (Parametric region automaton). *Let R_p be a parameter region. For an R-U2P-PTA $\mathcal{A} = (\Sigma, L, l_0, \mathbb{X}, \mathbb{P}, \zeta)$, given (E_0, D_0) the initial* p–*PDBM where all clocks are* 0*, the* parametric region automaton $\mathcal{R}(\mathcal{A})$ *over R_p is the tuple $(L', \Sigma, L'_0, \zeta')$ where: i) $L' = L \times p\text{–}\mathcal{PDBM}(R_p)$ ii) $L'_0 = (l_0, (E_0, D_0))$ iii) $\zeta' = \{\big((l, (E, D)), e, (l', (E', D')\big) \in L' \times \zeta \times L' \mid$ either $\exists e = \langle l, g, a, u_{np}, l' \rangle \in \zeta$, g is a non-parametric guard, $\exists (E'', D'') \in \mathsf{Succ}((E, D))$, $R_p \subseteq guard_{\forall}(g, (E'', D''))$ and $(E', D') = update(E'', D'', u_{np})$, or $\exists e = \langle l, g, a, u, l' \rangle \in \zeta$, g is a parametric guard, $\exists (E'', D'') \in \mathsf{Succ}((E, D))$, $R_p \subseteq p\text{-}guard_{\exists}(g, (E'', D''))$ and $(E', D') = \overline{update}(E'', D'', u)$.$\}$*

Let R_p be a parameter region, $\mathcal{A}$ be an R-U2P-PTA and $\mathcal{R}(\mathcal{A}) = (L', \Sigma, L'_0, \zeta')$. A run in $\mathcal{R}(\mathcal{A})$ is an untimed sequence
$\sigma : (l_0, (E_0, D_0))e_0(l_1, (E_1, D_1))e_1 \cdots (l_i, (E_i, D_i))e_i(l_{i+1}, (E_{i+1}, D_{i+1}))e_{i+1} \cdots$ such

[2] Remark that here is why our construction works for EF-emptiness, but cannot be used for, *e.g.,*, AF-emptiness ("is there a parameter valuation such that all runs reach a goal location l"): unlike $guard_{\forall}(g, E, D)$, not all clock valuations in a p–PDBM $(E, v(D))$ can satisfy a parametric guard if $v \in p\text{-}guard_{\exists}(g, E, D)$.

that for all i we have $\big((l_i,(E_i,D_i)),e_i,(l_{i+1},(E_{i+1},D_{i+1}))\big) \in \zeta'$, which we also write $(l_i,(E_i,D_i)) \xrightarrow{e_i} (l_{i+1},(E_{i+1},D_{i+1}))$ where e_i. Note that we label our transitions with the edges of the R-U2P-PTA.

3.3 Decidability of EF-emptiness and synthesis

Using our construction of the parametric region automaton $\mathcal{R}(\mathcal{A})$ for a given R-U2P-PTA $\mathcal{A}$, we state the next proposition.

Proposition 2. *Let R_p be a parameter region. Let $\mathcal{A}$ be an R-U2P-PTA and $\mathcal{R}(\mathcal{A})$ its parametric region automaton over R_p. There is a run $\sigma : (l_0,(E_0,D_0)) \xrightarrow{e_0} (l_1,(E_1,D_1)) \xrightarrow{e_1} \cdots (l_{f-1},(E_{f-1},D_{f-1})) \xrightarrow{e_{f-1}} (l_f,(E_f,D_f))$ in $\mathcal{R}(\mathcal{A})$ iff for all $v \in R_p$ there is a run $\rho : (l_0,w_0) \xrightarrow{e_0} (l_1,w_1) \xrightarrow{e_1} \cdots (l_{f-1},w_{f-1}) \xrightarrow{e_{f-1}} (l_f,w_f)$ in $v(\mathcal{A})$ s.t. for all $0 \leq i \leq f$, $w_i \in (E_i, v(D_i))$.*

From Proposition 2, if there is a run reaching a goal location in an instantiated R-U2P-PTA, then for another parameter valuation in the same parameter region there is a run in the instantiated R-U2P-PTA with the same locations and transitions (but possibly different delays), reaching the same location.

Theorem 1. *Let $\mathcal{A}$ be an R-U2P-PTA. Let R_p be a parameter region and $v \in R_p$. If there is a run $\rho = (l_0,w_0) \xrightarrow{e_0} \cdots \xrightarrow{e_{i-1}} (l_i,w_i)$ in $v(\mathcal{A})$, then for all $v' \in R_p$ there is a run $\rho' = (l_0,w'_0) \xrightarrow{e_0} \cdots \xrightarrow{e_{i-1}} (l_i,w'_i)$ in $v'(\mathcal{A})$ with for all i, there is $(E_i,D_i) \in p\text{–}\mathcal{PDBM}(R_p)$ s.t. $w_i \in (E_i,v(D_i))$ and $w'_i \in (E_i,v'(D_i))$.*

Note that there is a finite number of p–PDBMs for each parameter region R_p. Let $(E,D) \in p\text{–}\mathcal{PDBM}(R_p)$ and consider $\mathcal{PLT}$: D is an $(H+1)^2$ matrix made of pairs $(d,\triangleleft)$ where $d \in \mathcal{PLT}$ and $\triangleleft \in \{\leq,<\}$. Therefore the number of possible$D$ is bounded by $(2 \times (2 + 3 \times \binom{M}{2} + 4 \times M))^{(H+1)^2}$. Moreover the number of possible values for E is unbounded, but only a finite subset of all values needs to be explored, *i.e.,* those smaller than $K+1$: indeed, following classical works on timed automata [1,10], (integer) values exceeding the largest constant used in the guards or the parameter bounds are equivalent.

To test EF-emptiness given a bounded R-U2P-PTA $\mathcal{A}$ and a goal location l, we first enumerate all parameter regions (which are in finite number), and apply for each R_p the following process: we pick $v \in R_p$ (*e.g.,* using a linear programming algorithm [19]). Then, we consider $v(\mathcal{A})$ which is an updatable timed automaton and test the reachability of l in $v(\mathcal{A})$ [10]. Then EF-emptiness is false if and only if there is v and a run in $v(\mathcal{A})$ reaching l.

Theorem 2. *The EF-emptiness problem is PSPACE-complete for bounded R-U2P-PTAs.*

Given a goal location l and a bounded R-U2P-PTA $\mathcal{A}$, we can exactly synthesize the parameter valuations v s.t. there is a run in $v(\mathcal{A})$ reaching l by enumerating each parameter region (of which there is a finite number) and test

if l is reachable for one of its parameter valuations. The result of the synthesis is the union of the parameter regions for which one valuation (and, from our results, all valuations in that region) indeed reaches the goal location in the instantiated TA.

Corollary 1. *Given a bounded R-U2P-PTA $\mathcal{A}$ and a goal location l we can effectively compute the set of parameter valuations v s.t. there is a run in $v(\mathcal{A})$ reaching l.*

Remark 1. By bounding parameter valuations in guards but not those used in updates, we still have a finite number of parameter regions. Indeed, an integer vector E with components E_x greater than $\lfloor K \rfloor + 1$ is equivalent to an integer vector E' with $E'_x = E_x$ if $E_x < \lfloor K \rfloor + 1$ and $E'_x = \lfloor K \rfloor + 1$ if $E_x \geq \lfloor K \rfloor + 1$. Moreover for all p, we have to replace each parameter valuation v used in an update by $v(p) = v'(p)$ if $v(p) \leq K$ and $v'(p) = K + 1$ if $v(p) > K$.

4 Case study

We implemented EFsynth for R-U2P-PTAs in IMITATOR, a parametric model checker for (extensions of) PTAs [3].

Our class is the first for which synthesis is possible over bounded rational parameters. We believe our formalism is useful to model several categories of case studies, notably distributed systems with a periodic (global) behavior for which the period is unknown: this can be encoded using a parametric guard while resetting all clocks—possibly to other parameters.

Consider the R-U2P-PTA in Figure 1 with six locations, three clocks compared to parameters (x, y, t), one constant (max) and six parameters (p, p_1, p_2, v, pv_1, pv_2).

We consider the case of a network of peers exchanging transactions grouped by blocks, *e.g.,* a blockchain, using the Proof-of-Work as a mean to validate new blocks to add. In this simplified example, we consider a set of two peers (represented by x, y) which have different computation power (represented by p_1, p_2). Peers write new transactions on the current block (newTx). If it is full ($t = p$), both peers try to add a new block (newBlock) to write the transaction on it. We update x to p_1, y to p_2, and t to 0 as the peers have a different computation power, and they start "mining" the block (find a solution to a computation problem). Either x or y will eventually offer a solution to the problem ($\text{blockSolution}_\text{x}$ if $x = max$ or $\text{blockSolution}_\text{y}$ if $y = max$). If y offers a solution, x will check whether the solution is correct: x is updated to pv_1 to represent its rapidity to verify an offer. x can refuse the offer if the verification is too long (fakeBlock if $x > v$) therefore the mining step restarts. x can approve the offer (okBlock if $x \leq v$), y is rewarded and the block is added to the blockchain (addBlock).

We are interested in a malicious peer x that wants to avoid y to be rewarded for every new block. Therefore x asks: *"what are the possible computation power configurations and verification rapidity so that y is eventually rewarded"* ($EF(\text{reward}_\text{y})$-synthesis), considered as a bug state in the automaton.

We run this R-U2P-PTA using IMITATOR [3]. We set $max = 30$ units of time and also the upper bound of p and $1 \geq v > 0$ unit of time. IMITATOR computes a disjunction of constraints so that reward_y is unreachable: we keep two relevant ones; *i*) $p_1 \geq p_2$: x has strictly more computation power than y in which case x always offers a block solution, or has the same computation power than y in which case the systems blocks. x should invest heavily into hardware to keep its computation power high; *ii*) $pv_1 > v$: the malicious peer x is always faster to verify the solution offered by y and refuses it. The blockchain is probably compromised.

Using a parameter valuation respecting one of the previous constraints guarantees that y is never rewarded.

5 Conclusion and perspectives

Our class of bounded R-U2P-PTAs is one of the few subclasses of PTAs (actually even extended with parametric updates) to enjoy decidability of EF-emptiness. In addition, R-U2P-PTAs is the first "subclass" of PTAs to allow exact synthesis of bounded *rational*-valued parameters.

Beyond reachability emptiness, we aim at studying unavoidability-emptiness and language preservation emptiness, as well as their synthesis.

Finally, we would like to investigate whether our parametric updates can be applied to decidable hybrid extensions of TAs [16,12].

Acknowledgements. We would like to thank anonymous reviewers for constructive remarks.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science **126**(2), 183–235 (Apr 1994). https://doi.org/10.1016/0304-3975(94)90010-8
2. Alur, R., Henzinger, T.A., Vardi, M.Y.: Parametric real-time reasoning. In: Kosaraju, S.R., Johnson, D.S., Aggarwal, A. (eds.) STOC. pp. 592–601. ACM, New York, NY, USA (1993). https://doi.org/10.1145/167088.167242
3. André, É., Fribourg, L., Kühne, U., Soulat, R.: IMITATOR 2.5: A tool for analyzing robustness in scheduling problems. In: Giannakopoulou, D., Méry, D. (eds.) FM. Lecture Notes in Computer Science, vol. 7436, pp. 33–36. Springer (Aug 2012). https://doi.org/10.1007/978-3-642-32759-9_6, http://www.lsv.fr/Publis/PAPERS/PDF/AFKS-fm12.pdf
4. André, É., Lime, D.: Liveness in L/U-parametric timed automata. In: ACSD. pp. 9–18. IEEE (2017). https://doi.org/10.1109/ACSD.2017.19
5. André, É., Lime, D., Ramparison, M.: Timed automata with parametric updates. In: Juhás, G., Chatain, T., Grosu, R. (eds.) ACSD. pp. 21–29. IEEE (2018). https://doi.org/10.1109/ACSD.2018.000-2
6. André, É., Lime, D., Ramparison, M.: Parametric updates in parametric timed automata. CoRR **abs/1904.08824** (2019), http://arxiv.org/abs/1904.08824
7. André, É., Lime, D., Roux, O.H.: Decision problems for parametric timed automata. In: ICFEM. Lecture Notes in Computer Science, vol. 10009, pp. 400–416. Springer (2016). https://doi.org/10.1007/978-3-319-47846-3_25
8. Beneš, N., Bezděk, P., Larsen, K.G., Srba, J.: Language emptiness of continuous-time parametric timed automata. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP, Part II. Lecture Notes in Computer Science, vol. 9135, pp. 69–81. Springer (Jul 2015). https://doi.org/10.1007/978-3-662-47666-6_6
9. Bengtsson, J., Yi, W.: Timed automata: Semantics, algorithms and tools. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) Lectures on Concurrency and Petri Nets, Advances in Petri Nets. Lecture Notes in Computer Science, vol. 3098, pp. 87–124. Springer (2003). https://doi.org/10.1007/978-3-540-27755-2_3
10. Bouyer, P., Dufourd, C., Fleury, E., Petit, A.: Updatable timed automata. Theoretical Computer Science **321**(2-3), 291–345 (2004). https://doi.org/10.1016/j.tcs.2004.04.003
11. Bozzelli, L., La Torre, S.: Decision problems for lower/upper bound parametric timed automata. Formal Methods in System Design **35**(2), 121–151 (2009). https://doi.org/10.1007/s10703-009-0074-0
12. Brihaye, T., Doyen, L., Geeraerts, G., Ouaknine, J., Raskin, J., Worrell, J.: Time-bounded reachability for monotonic hybrid automata: Complexity and fixed points. In: Hung, D.V., Ogawa, M. (eds.) ATVA. Lecture Notes in Computer Science, vol. 8172, pp. 55–70. Springer (2013). https://doi.org/10.1007/978-3-319-02444-8_6
13. Bundala, D., Ouaknine, J.: Advances in parametric real-time reasoning. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS. Lecture Notes in Computer Science, vol. 8634, pp. 123–134. Springer (2014). https://doi.org/10.1007/978-3-662-44522-8
14. Dill, D.L.: Timing assumptions and verification of finite-state concurrent systems. In: Sifakis, J. (ed.) Automatic Verification Methods for Finite State Systems 1989. Lecture Notes in Computer Science, vol. 407, pp. 197–212. Springer (1989). https://doi.org/10.1007/3-540-52148-8_17

15. Doyen, L.: Robust parametric reachability for timed automata. Information Processing Letters **102**(5), 208–213 (2007). https://doi.org/10.1016/j.ipl.2006.11.018
16. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? Journal of Computer and System Sciences **57**(1), 94–124 (1998). https://doi.org/10.1006/jcss.1998.1581
17. Hune, T., Romijn, J., Stoelinga, M., Vaandrager, F.W.: Linear parametric model checking of timed automata. Journal of Logic and Algebraic Programming **52-53**, 183–220 (2002). https://doi.org/10.1016/S1567-8326(02)00037-1
18. Jovanović, A., Lime, D., Roux, O.H.: Integer parameter synthesis for real-time systems. IEEE Transactions on Software Engineering **41**(5), 445–461 (2015). https://doi.org/10.1109/TSE.2014.2357445
19. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica **4**(4), 373–396 (1984). https://doi.org/10.1007/BF02579150
20. Miller, J.S.: Decidability and complexity results for timed automata and semi-linear hybrid automata. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC. Lecture Notes in Computer Science, vol. 1790, pp. 296–309. Springer (2000). https://doi.org/10.1007/3-540-46430-1_26
21. Quaas, K.: MTL-model checking of one-clock parametric timed automata is undecidable. In: André, É., Frehse, G. (eds.) SynCoP. EPTCS, vol. 145, pp. 5–17 (2014). https://doi.org/10.4204/EPTCS.145.3
22. Quaas, K., Shirmohammadi, M., Worrell, J.: Revisiting reachability in timed automata. In: LICS. pp. 1–12. IEEE Computer Society (2017). https://doi.org/10.1109/LICS.2017.8005098