École Doctorale Galilée —
Université Paris 13 LIPN — CNRS,
UMR 7030, F-93430

On the theory and practice of updatable parametric timed automata

A thesis

in computer science

presented to the faculty of the graduate school of Université Paris 13 in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Science

by Mathias Ramparison

Supervisors:
Dr. Étienne André
and Dr. Didier Lime

September 5, 2019

Doctoral committee:

Patricia Bouyer	Directrice de recherche, LSV, ENS Cachan	Rapportrice
Jaco van de Pol	Professeur, Aarhus University	Rapporteur
Béatrice Bérard	Professeure, Université Pierre et Marie Curie	Jurée extérieure
Frédéric Herbreteau	Maître de Conférences, Bordeaux INP	Juré extérieur
Laure Petrucci	Professeure, Université Paris 13	Jurée interne
Christophe Fouqueré	Professeur, Université Paris 13	Juré interne

On the theory and practice of updatable parametric timed automata

Keywords: parametric timed automata, parameter synthesis, reachability, decidability, attack-fault trees.

Abstract

As cyber-physical systems become more and more complex, human debugging is not sufficient anymore to cover the huge range of possible behaviours. For costly critical systems where human lives can be endangered, formally proving the safety of a system is even more crucial. This is done by defining a formal specification for the system, and then performing the algorithmic verification that the system satisfies some formally specified properties. With this precise and exhaustive description of a system, the usual vagueness of human language is eliminated. In this thesis, we focus on the verification of timed concurrent systems. Time-dependent systems are very hard to verify, especially when the exact value of timing constants remains unknown. These unknown timing constants are called parameters. We study several subclasses of a parametric extension of the well-known formalism called Timed Automata. We mainly focus on the reachability decision problem, that asks whether there exists concrete values for these parameters such that a bug state can be reached in the system. We further address for these subclasses a computation problem that is to synthesise the set of parameter values for which a state is reachable. Finally, we apply our work to the security and safety of cyber-physical systems and infrastructure: we extend with parameters a classic formalism to model attack and failure scenarios called attack-fault trees, and propose an implementation of the translation of parametric attack-fault trees to parametric timed automata. This allows us to leverage the verification techniques and tools available for the latter for the analysis of (parametric) attack-fault trees.

Sur la théorie et l'application des automates paramétrés temporisés avec mises à jour.

Mots-clés: automate temporisé paramétré, synthèse de paramètres, accessibilité, décidabilité, arbres d'attaque et de défaillance.

Résumé

À mesure que les systèmes cyber-physiques deviennent de plus en plus complexes, le débogage humain ne suffit plus pour analyser le grand nombre de comportements possibles. Pour les systèmes critiques coûteux où des vies humaines peuvent être mises en danger, il est encore plus crucial de prouver formellement la sécurité d'un système. Pour ce faire, on définit une spécification formelle pour le système, puis on vérifie algorithmiquement que le système satisfait à certaines propriétés spécifiées de manière formelle. Avec cette description précise et exhaustive d'un système, le flou habituel du langage humain est éliminé. Dans cette thèse, nous nous concentrons sur la vérification des systèmes concurrents temporisés. Les systèmes dépendant du temps sont très difficiles à vérifier, en particulier lorsque la valeur exacte des constantes de synchronisation reste inconnue. Ces constantes de synchronisation inconnues sont appelées paramètres. Nous étudions plusieurs sous-classes d'une extension paramétrique d'un formalisme bien connu, les automates temporisés. Nous nous concentrons principalement sur le problème de décision de l'accessibilité, qui pose la question de l'existence de valeurs concrètes pour ces paramètres telles qu'un état de bogue peut être atteint dans le système. Nous abordons en outre pour ces sous-classes un problème de calcul consistant à synthétiser l'ensemble des valeurs de paramètres pour lesquelles un état est accessible. Enfin, nous appliquons nos travaux à la sécurité des infrastructures et des systèmes cyber-physiques : nous étendons avec des paramètres un formalisme classique pour modéliser des scénarios d'attaque et de défaillance, appelés arbres de défaillance et d'attaque, et proposons une implémentation de la traduction d'arbres de défaillance et d'attaque paramétriques en automates paramétrés temporisés. Cela nous permet de tirer parti des techniques et des outils de vérification disponibles pour ce formalisme pour l'analyse des arbres de défaillance et d'attaque (paramétriques).

Contents

Chapter 1

Introduction

PASADENA, Calif. – NASA's Mars rover Curiosity is expected to resume science investigations in a few days, as engineers quickly diagnosed a software issue that prompted the rover to put itself into a precautionary standby status over the weekend.

Curiosity initiated this automated fault-protection action, entering "safe mode" at about 8 p.m. PDT (11 p.m. EDT) on March 16, while operating on the B-side computer, one of its two main computers that are redundant to each other. It did not switch to the A-side computer, which was restored last week and is available as a back-up if needed. The rover is stable, healthy and in communication with engineers.

The safe-mode entry was triggered when a command file failed a size-check by the rover's protective software. Engineers diagnosed a software bug that appended an unrelated file to the file being checked, causing the size mismatch. NASA, March 18, 2013.

"I tested several configurations, there is no bug. My software is safe". Perhaps this is a sentence you already said. Not much to worry about in case you are wrong, as you just coded a calculator for your weekly expenses. Much more if you are working on a 2.5 billion USD NASA rover. It is supposed to work 24/7 at approximatively 76 million kilometers from Earth, and embeds a lot of different softwares, components, tools, hardwares that *must* work together like a symphony. These are developed by different teams of different people from all around the world.

Several safety questions have to be answered. How can we be sure that the brake is functional? That there is no flaw in a hardware component? That there is no bug in the code? Or, as we have seen in the above text, how can we be sure that a unrelated file will not be added to the current checked file, causing a memory overflow?

Of course, in most cases given a system there is an infinite number of possible behaviors so it is impossible to test them all. Moreover, as systems become more and more complex, human debugging is not sufficient anymore. Curiosity rover is one perfectly fitting example. This is where *formal methods* come on track. It is a mathematical way to model some system and properties often using a temporal logic. With this precise and exhaustive description of a system, usual vagueness of human language is eliminated. For instance, abstract interpretation [Cou12] allows to verify the semantics of programming languages as well as to program

static analysis.

Besides, theorem proving allows to formally prove mathematical theorems using computer science. Many tools exist, including ISABELLE/HOL [NPW02] and COQ [HH14].

Further, *model-checking* is an automated way, given as an input a *model* of a system and a property, to decide in human language whether the system satisfies the property. Many tools exist such as NuSMV [CCGR00] and are used in the industry. This work mainly focuses on models of timed systems, also called *formalisms*, that is, the way to express as accurately as possible a system with timing constraints—critical deadlines, periods—so that usual properties usually defined with a temporal logic in classical literature can be checked. This results in timed model-checking, and parametric timed model-checking when some timing constants are not precisely known, as we will present in the following.

A well-known formalism is Timed Automaton [AD94] that has been widely studied for nearly 30 years.

Timed automata (TAs) [AD94] represent a powerful formalism to model and verify systems where concurrency is mixed with hard timing constraints. Practical applications include verification of security protocols [CEHM04, JP07] and securiy analysis [KPS14].

TAs are an extension of finite-state automata with clocks, *i. e.,* real-valued variables, that can be compared to integer constants and updated to 0 along edges (called reset in the literature). TAs benefit from many decidability results such as the reachability of a discrete location which is PSPACE-complete [AD94], or the emptiness of its accepted language (and some undecidability results too, such as language inclusion). Decision problems in TAs is still a topical research subject [HSTW16, HSW16].

Although TAs seem to be able to model many interesting problems related to timed concurrent systems, several extensions were studied. For instance, TAs where clocks can be updated to integer constants have been introduced in [BDFP04] and interesting decidability results have been obtained, depending amongst other restrictions of the nature of the clock constraints (*e. g.,* diagonal-free, *i. e.,* whether clocks are compared to each other) and the updates of clocks (*e. g.,* whether it is allowed to update a clock to its current value increased by some rational constant).

In a different direction, stopping the time elapsing of at least one clock in a TA gives *stopwatch automata*, for which the reachability problem becomes undecidable [CL00].

Moreover, TAs are supported by many state-of-the-art model-checkers such as UPPAAL [LPY97] and PAT [SLDP09].

Timed automata may turn inappropriate to verify systems where the timing constants are subject to some uncertainty or can range in intervals, where clocks suffer from imprecision—often studied as robustness problems [BMS13]—or when they are simply not known at an early design stage. TAs cannot be used when constants are taken from a dense interval, particularly when it is a real value. In addition, if we want to verify a property for a set of constants using traditional model checking, we have check the property for each value of the set, which can be very efficient for a small set of values but becomes very expensive for larger ones. Finally, when a constant is supposed to be 5, what if it is practically encoded as 4.99... This highlights the need of unknown constants in our formalism, also called parameters.

Extending timed automata with parameters in guards in place of integers gives *parametric timed automata* (PTAs) [AHV93] and alleviates this drawback by allowing parameters (unknown constants) in the timing constraints. In the PTA literature, the main problem studied is the reachability emptiness, or EF-emptiness ("is the set of timing parameter valuations for which a given location is reachable empty?"). Nonetheless, all non-trivial problems are undecidable for PTAs (see [And19] for a survey), especially AF-emptiness ("is the set of timing parameter valuations such that a given location is unavoidable empty?") [ALR16a].

Only a few decidability results have been shown for subclasses of PTAs, such as L/U-PTAs [HRSV02, BL09, AL17] and reset-PTAs [ALR16a]. These parametric models can be checked using model-checkers such as IMITATOR [AFKS12]. Other formalisms with parameters are studied in the literature, such as Parametric Timed Petri Nets [VP99, TLR09] and are supported in model checkers such as Roméo [GLMR05].

A more detailed section presenting decision problems considered in PTAs is Section 2.4.2 and formal definitions of the aforementioned problems are given in Section 2.3.

Research goal In this thesis we will define several new subclasses of PTAs, augmented with parametric updates and focus on fresh aspects: the ability to set a clock to a parameter *i. e.,* an unknown constant that can be further instantiated *i. e.,* assigned to a concrete value (integer or rational). We will also study L/U-PTAs without invariants. For these subclasses, we will mainly focus on the decidability or undecidability of common classical properties of the literature expressed in a temporal logic. Moreover the ultimate goal will be a a computational problem that we are going to solve for a few of our subclasses: the synthesis of the set of "good" parameter valuations for which, once instantiated, a given property is satisfied. Following the initial inspiration of [CEHM04], we will try to apply these formalisms with parameters to security, through modeling timed attack scenarios.

PACS project My PhD was supported by national project ANR PACS (Parametric Analyses of Concurrent Systems). ANR PACS involves four laboratories: LIPN (Paris 13), IRIF (Paris 7), LS2N (Laboratoire des Sciences du Numérique de Nantes, formerly IRCCyN (École Centrale Nantes) and LINA (Université de Nantes)). In addition, Kim Larsen's group in Aalborg (Denmark) acts as a foreign partner. This project aims to study parameters in the context of discrete and timed/hybrid systems, both of them possibly augmented with quantitative information relating to costs (*e. g.,* energy consumption), giving cost-based models, and probabilities in discrete or timed models.

Overview of this thesis Chapter 2 introduces the notations, defines formalisms and recalls problems that constitute the basis of this thesis.

In Chapter 3 we will study the ability to update clocks to parameters in TAs, a concept we published in [ALR18b]. This work brings two new subclasses of PTAs. One with integer-valued parameters, for which we obtain decidability results for the EF, AF-emptiness/universality problems. In fact, we enumerate and even synthesize parameter valuations. Another one with rational-valued parameters,

for which the EF, AF-emptiness/universality problems become undecidable. This proof uses the reduction of the halting problem of a two counter machine.

In Chapter 4 we will study the ability to update clocks to parameters in PTAs [ALR19]. This works brings a new subclass of PTAs for which the EF-emptiness problem is decidable under the restriction that all clocks are updates whenever a clock is compared to a parameter in a guard, or updated to a parameter. This is an improvement of Chapter 3: we also use parameters in guards, but add more syntactic restrictions. The proof uses a refinement of clock regions [AD94] combined with a structure inspired of parametric bound matrices [HRSV02] in order to obtain a finite number of sets of parameters. Therefore, we are also able to synthesize parametric values we are interested in.

In Chapter 5 we will prove that the full TCTL logic is undecidable for U-PTAs without invariant *i. e.,* that one level of nesting in the formula brings undecidability, but that flat TCTL is decidable for L/U-PTAs without invariants, by resolving the last non-investigated liveness properties, resulting in [ALR18a].

In Chapter 6 we will define and implement the translation of attack-fault trees (AFTs) to a new extension of timed automata, called parametric weighted timed automata. This allows us to parametrize constants such as time and discrete costs in an AFT and then, using the model-checker IMITATOR, to compute the set of parameter values such that a successful attack is possible. Using the different sets of parameter values computed, different attack and fault scenarios can be deduced depending on the budget, time or computation power of the attacker, providing helpful data to select the most efficient counter-measure. We will present case studies. This work was published in [ALRS19].

Chapter 7 summarizes the thesis and concludes with a discussion on perspectives.

Accepted paper in international conferences resulting from these works are [ALR18b, ALR18a, ALR19, ALRS19].

Chapter 2

Preliminaries

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

Throughout this thesis, 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 $\vec{0}$ 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 $v : \mathbb{P} \to \mathbb{Q}_+$. Note that we consider positive parameter valuations only as usually done in the literature [And19]. 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 $u_{np} : \mathbb{X} \rightharpoonup \mathbb{N}$ and a clock reset [AD94] is $u_{np \to 0} : \mathbb{X} \rightharpoonup 0$. For a clock valuation w and a parameter valuation v, we denote by $[w]_{v(u)}$ the clock valuation obtained after applying $v(u)$.

2.1 Timed Automata and Parametric Timed Automata

In this first section, we are going to define the formalisms we will use and extend in this thesis.

2.1.1 Syntax

First we give the syntax of Timed Automata (TA) and an example.

2.1.1.1 Timed Automata

Definition 1 (Timed Automaton [AD94]). *A TA $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, L, l_0, \mathbb{X}, \zeta)$, where: i) Σ is a finite set of actions, ii) L is a finite set of locations, iii) $l_0 \in L$ is the initial location, iv) $\mathbb{X}$ is a finite set of clocks, v) ζ is a finite set of edges $e = \langle l, g, a, u_{np\to 0}, l' \rangle$ where $l, l' \in L$ are the source and target locations, g is a non parametric guard, $a \in \Sigma$ and $u_{np\to 0} : \mathbb{X} \rightharpoonup 0$ is a reset function.*

In a concurrent setting, timed automata can be synchronized on shared actions. It is well-known that the product of several TAs gives a TA (see *e. g.,* [Mil00]). Moreover, real-time physical systems modeled with TAs can be implemented and timed properties checked using *e. g.,* Uppaal [BLL+95] or IMITATOR [AFKS12].

Also note that we do not add invariants in our definition of TA, as we will mainly focus on the state reachability property. Invariants will be added when relevant in the definitions further chapters of this thesis. Likewise, note that we consider only diagonal-free constraints as it is as expressive as classical TA [BDFP04, BC13].

In order to illustrate Definition 1, we give the following example.

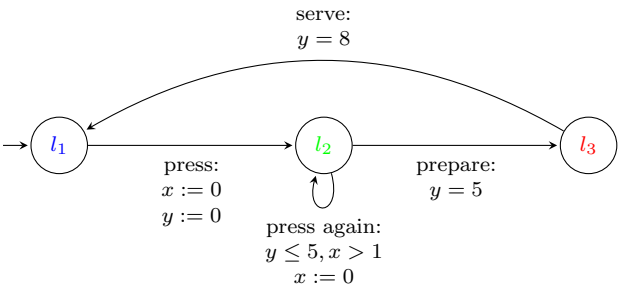


Figure 2.1: A timed automaton modelling a coffee machine

Where we have the following elements: $L = \{l_1, l_2, l_3\}$, $l_0 = l_1$, $\mathbb{X} = \{x, y\}$, and $\Sigma = \{\text{press, press again, prepare, serve}\}$. There is four edges:

- $e_1 = \langle l_1, g, a, u_{np\to 0}, l_2 \rangle$ where $u_{np\to 0}$ sets both x, y to 0,
- $e_2 = \langle l_2, g, a, u_{np\to 0}, l_2 \rangle$ where g is $y \leq 5 \wedge x > 1$ and $u_{np\to 0}$ sets x to 0,
- $e_3 = \langle l_2, g, a, u_{np\to 0}, l_3 \rangle$ where g is $y = 5$ and
- $e_4 = \langle l_3, g, a, u_{np\to 0}, l_1 \rangle$ where g is $y = 8$.

2.1.1.2 Parametric Timed Automata

As done above, we define Parametric Timed Automata (PTA) and give an example. The definition is similar but adds a set of parameters (unknown constants) $\mathbb{P}$, and allows parameters to be used in guards, therefore becoming parametric guards.

Definition 2 (Parametric Timed Automaton [AHV93])**.** *A 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) $l_0 \in L$ 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 $e = \langle l, g, a, u_{np \to 0}, l' \rangle$ where $l, l' \in L$ are the source and target locations, g is a parametric guard, $a \in \Sigma$ and $u_{np \to 0} : \mathbb{X} \rightharpoonup 0$ is a reset function.*

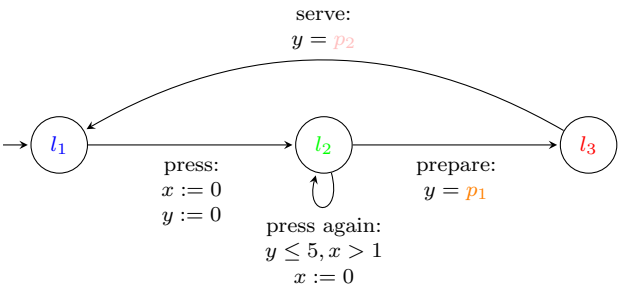


Figure 2.2: A Parametric Timed Automaton modelling a coffee machine

We have the same elements as in Figure 2.1, with the additional set $\mathbb{P} = \{p_1, p_2\}$. The two last edges become:

- $e_3 = \langle l_2, g, a, u_{np \to 0}, l_3 \rangle$ where g is $y = p_1$ and
- $e_4 = \langle l_3, g, a, u_{np \to 0}, l_1 \rangle$ where g is $y = p_2$.

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 an *updatable timed automaton* [BDFP04] but will be called *timed automaton* (TA) for the sake of simplicity in this thesis. In the following, we may denote as a timed automaton any structure $v(\mathcal{A})$, by assuming a rescaling of the constants: by multiplying all constants in $v(\mathcal{A})$ by their least common denominator, we obtain an equivalent timed automaton (with integer constants), as defined in [AD94].

A *bounded* PTA is a 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.

2.1.1.3 Lower/Upper bound Parametric Timed Automata

A famous subclass of PTA is Lower/Upper bound Parametric Timed Automata (L/U-PTAs) [BL09]:

Definition 3 (L/U-PTA)**.** *An L/U-PTA is a PTA where the set of parameters is partitioned into lower-bound parameters and upper-bound parameters, i. e., parameters that appear in guards are inequalities of the form $p \leq x$ or $p < x$, and of the form $p \geq x$ or $p > x$ respectively.*

Note that our definition does not include invariants as the L/U-PTAs of [HRSV02].

2.1.2 Concrete Semantics

Definition 4 (Concrete semantics of a TA)**.** *Given a 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*

- $S = \{(l, w) \in L \times \mathbb{R}_+^H\}$, $s_0 = (l_0, \vec{0})$
- $\rightarrow$ *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 $\exists d, w'' : (l, w) \overset{d}{\mapsto} (l, w'') \overset{e}{\mapsto} (l', w')$.

For instance in the TA of Figure 2.1, a possible run is : $\big(l_1, (0,0)\big) \xrightarrow[2.1]{\text{press}} \big(l_2, (0,0)\big) \xrightarrow[1.2]{\text{press again}} \big(l_2, (0, 1.2)\big) \xrightarrow[3.8]{\text{prepare}} \big(l_3, (3.8, 5)\big) \xrightarrow[3]{\text{serve}} \big(l_1, (6.8, 8)\big)$, where the delay and the action of each transition has been combined for the sake of simplicity.

In the PTA of Figure 6.3, a possible run if $p_1 = 2, p_2 = 3$: $\big(l_1, (0,0)\big) \xrightarrow[2]{\text{press}} \big(l_2, (0,0)\big) \xrightarrow[1]{\text{press again}} \big(l_2, (0,1)\big) \xrightarrow[1]{\text{prepare}} \big(l_3, (1,2)\big) \xrightarrow[1]{\text{serve}} \big(l_1, (2,3)\big)$, where the delay and the action of each transition has been combined for the sake of simplicity. The same run is impossible if $p_1 = 5, p_2 = 2$, or $p_1 < 1$.

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. Given a run ρ of $v(\mathcal{A})$, $\mathsf{time}(\rho)$ gives the total sum of the delays d along ρ.

2.2 Timed CTL

TCTL [ACD93] is the quantitative extension of CTL where temporal modalities are augmented with constraints on duration. Formulae are interpreted over timed transition systems (TTS).

Given $ap \in AP$ and $c \in \mathbb{N}$, a TCTL formula is given by the following grammar:

$$\varphi ::= \quad \top \quad | \quad ap \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \varphi \quad | \quad \mathsf{E}\varphi\mathsf{U}_{\bowtie c}\varphi \quad | \quad \mathsf{A}\varphi\mathsf{U}_{\bowtie c}\varphi$$

A reads "always", E reads "exists", and U reads "until".

Standard abbreviations include Boolean operators as well as $\mathsf{EF}_{\bowtie c}\varphi$ for $\mathsf{E}\top\mathsf{U}_{\bowtie c}\varphi$, $\mathsf{AF}_{\bowtie c}\varphi$ for $\mathsf{A}\top\mathsf{U}_{\bowtie c}\varphi$ and $\mathsf{EG}_{\bowtie c}\varphi$ for $\neg\mathsf{AF}_{\bowtie c}\neg\varphi$. ($\mathsf{F}$ reads "eventually" while G reads "globally".)

Definition 5 (Semantics of TCTL)**.** *Given a TA $v(\mathcal{A})$, the following clauses define when a state s_i of its TTS $(S, s_0, \to)$ satisfies a TCTL formula φ, denoted by $s_i \models \varphi$, by induction over the structure of φ (semantics of Boolean operators is omitted):*

1. *$s_i \models \mathsf{E}\varphi\mathsf{U}_{\bowtie c}\Psi$ if there is a maximal run ρ in $v(\mathcal{A})$ with $\sigma = s_i \xrightarrow{e_i} \cdots \xrightarrow{e_{j-1}} s_j$ $(i < j)$ a prefix of ρ s.t. $s_j \models \Psi$, $\mathsf{time}(\sigma) \bowtie c$, and if $\forall k$ s.t. $i \leq k < j$, $s_k \models \varphi$, and*
2. *$s_i \models \mathsf{A}\varphi\mathsf{U}_{\bowtie c}\Psi$ if for each maximal run ρ in $v(\mathcal{A})$ there exists $\sigma = s_i \xrightarrow{e_i} \cdots \xrightarrow{e_{j-1}} s_j$ $(i < j)$ a prefix of ρ s.t. $s_j \models \Psi$, $\mathsf{time}(\sigma) \bowtie c$, and if $\forall k$ s.t. $i \leq k < j$, $s_k \models \varphi$.*

In $\mathsf{E}\varphi\mathsf{U}_{\bowtie c}\varphi$ the classical *until* is extended by requiring that φ be satisfied within a duration (from the current state) verifying the constraint "$\bowtie c$". Given v, a PTA $\mathcal{A}$ and a TCTL formula φ, we write $v(\mathcal{A}) \models \varphi$ when $s_0 \models \varphi$.

We define *flat TCTL* as the subset of TCTL where, in $\mathsf{E}\varphi\mathsf{U}_{\bowtie c}\varphi$ and $\mathsf{A}\varphi\mathsf{U}_{\bowtie c}\varphi$, φ must be a formula of propositional logic (a Boolean combination of atomic propositions).

2.3 Problems

In this thesis, we address the following two families of decision problems, given P a class of problems (*e. g.,* reachability, unavoidability, TCTL model-checking):

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

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

We mainly focus on reachability (EF) and unavoidability (AF) [JLR15], but described several problem studied in the literature, that we are going to study as well in this thesis.

- EF-emptiness asks, given a PTA $\mathcal{A}$ and a location l whether the set of valuations v such that there is a run in $v(\mathcal{A})$ reaching l is empty? It is equivalent to AG-universality [And19]. More formally, the problem can be written as $\{v \mid \exists s_0 \xrightarrow{e_0} (l_1, w_1) \xrightarrow{e_1} \cdots \xrightarrow{e_{m-1}} (l, w) \text{ a run of } v(\mathcal{A})\} = \emptyset$?

- AF-emptiness asks, given a PTA $\mathcal{A}$ and a location l whether the set of valuations v such that all maximal runs in $v(\mathcal{A})$ reach l is empty? More formally, the problem can be written as $\{v \mid \forall s_0 \xrightarrow{e_0} (l_1, w_1) \xrightarrow{e_1} \cdots \xrightarrow{e_{m-1}} (l, w) \cdots \text{ maximal runs of } v(\mathcal{A})\} = \emptyset$? It is equivalent to EG-universality [And19].
- EF-universality asks, given a PTA $\mathcal{A}$ and a location l whether all valuations v are such that there is a run in $v(\mathcal{A})$ reaching l? More formally, the problem can be written as $\{v \mid \exists s_0 \xrightarrow{e_0} (l_1, w_1) \xrightarrow{e_1} \cdots \xrightarrow{e_{m-1}} (l, w) \text{ a run of } v(\mathcal{A})\} = \mathbb{Q}_+$? It is equivalent to AG-emptiness [And19].
- Finally, AF-universality asks, given a PTA $\mathcal{A}$ and a location l whether all valuations v are such that all maximal runs in $v(\mathcal{A})$ reach l? More formally, the problem can be written as $\{v \mid \forall s_0 \xrightarrow{e_0} (l_1, w_1) \xrightarrow{e_1} \cdots \xrightarrow{e_{m-1}} (l, w) \cdots \text{ maximal runs of } v(\mathcal{A})\} = \mathbb{Q}_+$? It is equivalent to EG-emptiness [And19].

Beyond the theoretical decision problems above, a ultimate goal is the following computation problem.

P-synthesis problem:
INPUT: a PTA $\mathcal{A}$ and an instance ϕ of P
PROBLEM: compute the set of valuations v such that $v(\mathcal{A})$ satisfies ϕ

Note that if P-emptiness is undecidable, there is no hope for a useful and effective P-synthesis procedure.

2.4 Related work

2.4.1 Updatable Timed Automata

Updatable Timed Automata (UTA), which are TAs where clocks can be updated to integer constants have been introduced in [BDFP04]. Many interesting decidability results have been obtained, depending amongst other restrictions of the nature of the clock constraints (*e. g.,* diagonal-free, *i. e.,* whether clocks are compared to each other) and the updates of clocks (*e. g.,* whether it is allowed to update a clock to its current value increased by some rational constant).

The following table summarises some results of [BDFP04]: x, y, z are clocks, c is an integer constant. $x := c$ reinitializes x to c, when $x := y$ transfers the value of y to x. $x :\in (c; +\infty)$ reinitilizes randomly x within the described interval, as does $x :\bowtie y + c$ where $\bowtie \in \{<, \leq, \geq, >\}$.

	simple constraints	and diagonal constraints
$x := c, x := y$	decidable	decidable
$x := x + 1$		undecidable
$x := y + c$		
$x := x - 1$	undecidable	
$x :\in [0; c)$	decidable	decidable
$x :\in (c; +\infty)$		undecidable
$x :\bowtie y + c$		
$x :\in (y + c; y + d)$		
$x :\in (y + c; z + d)$	undecidable	

UTAs will be the working base of the first two chapters of this thesis in which we extend UTAs with parameters in guards and in clock updates.

2.4.2 Parametric Timed Automata

In the PTA literature, the main problem studied is the reachability emptiness, or EF-emptiness : it is "robustly" undecidable in the sense that, even when varying the setting, undecidability is preserved. For example, EF-emptiness is undecidable even for a single bounded parameter [Mil00], even for a single rational-valued or integer-valued parameter [BBLS15], even with only one clock compared to parameters [Mil00], or with strict constraints only [Doy07]. More generally, all non-trivial problems are undecidable for PTAs (see [And19] for a survey). The unavoidability emptiness where we seek for valuations for which some location will *always eventually* be reached, or AF-emptiness is also undecidable [JLR15]. Similarly EF-universality and AF-universality are undecidable [ALR16a] for the general class of PTAs, while decidability results have been shown for L/U-PTAs [HRSV02, BL09, AL17]. Sometimes, the border between decidability and undecidability is quite thin: EF-synthesis is possible for bounded integer-valued PTAs, while EF-emptiness becomes undecidable if boundedness is removed [JLR15]. Other techniques are developed in order to study L/U-PTAs [KP12b]. Another subclass of PTAs named reset-PTA [ALR16a] comes with the decidability of EF-emptiness with the additional restriction that clocks are reset to 0 whenever a clock is compared to a parameter in a guard. PTAs are also studied over bounded time [KP12a]. Optimisation of time runs is also studied through *e. g.,* minimal-time reachability: synthesising a single parameter valuation for which the goal location can be reached in minimal (lower-bound) time [ABPvdP19].

2.4.3 Other formalisms using parameters

Parameters are also used in other formalisms in the literature: hybrid systems are similar to TAs but clocks can evolve at different rates. Both theory and practical applications of Parametric Hybrid Systems have been studied [Fre08, FK11, AK12].

Other formalisms with parameters are studied in the literature, such as Parametric Timed Petri Nets [VP99, TLR09] and Parametric Timed Kripke Structures [KP14].

Parametric Timed Communicating Sequential Processes (PTCSP) have also been studied in [ALS+13, ALSD14] as an extension of CSP [Hoa78] with timing parameters, a process algebra used in concurrency modeling.

Parametric Task Automata, an extension of Task Automata [NWY99, FKPY07] in which we include a list of tasks in the concrete states of a Timed Automaton have been studied in [And17], in order to model *e. g.,* periodic real-time systems.

Parametric Interrupt Timed Automata [BHJL13, BHJL16] and Polynomial Interrupt Times Automata [BHP+15] also represent an active field of research as extensions of timed automata where reachability and some variants of timed model-checking are decidable even in presence of parameters.

2.4.4 Applications of timed automata to security

Timed automata are used in the verification of security protocols by translating a language specification of a security protocol into timed automata [JP07] or to model directly protocols with timed automata [CEHM04]. Moreover, timed automata and its extensions are widely used in securiy analysis [KPS14].

Chapter 3

Timed automata with parametric updates

3.1 Introduction

In this first technical chapter, we consider an extension of TAs where we allow parameterized updates of clock variables. The key contribution is to characterize a decidability boundary for the reachability problem for this class of timed automata: the problem is undecidable when parameters are (bounded) rationals and decidable (PSPACE-complete) when parameters are restricted to be (unbounded) integers.

3.1.1 Contribution

We show that extending timed automata with parametric updates, *i. e.,* the ability to update a clock to an unknown rational constant, leads to the undecidability of the four following problems: EF-emptiness, AF-emptiness, EF-universality, AF-universality. That is, it is undecidable to determine:

- whether the set of parameter valuations for which a run leads to a given location is empty;
- whether for all parameter valuations there is a run that leads to a given location;
- whether the set of parameter valuations for which a given location is unavoidable empty;
- whether for all parameter valuations a given location is unavoidable.

In contrast, when we restrict the parameters domain to (unbounded) integers, all four problems do not only become decidable, but we can achieve exact synthesis, *i. e.,* represent the full set of valuations for which a run or all runs lead(s) to a given location.

On the one hand, our undecidability results adds to the long list of undecidable parametric extensions of timed automata.

On the other hand, our decidability result enriches the notably short list of decidable such parametric extensions: the exact synthesis of integer-valued parameters compared as upper-bounds to clocks can be achieved [BL09]; the emptiness of the valuations set for which a location is reachable is decidable both for rational-valued *L/U-PTAs* [HRSV02], and for rational-valued *integer-point PTAs*, a semantic class for which the membership is however undecidable, although [ALR16a] exhibited a syntactic subclass, namely *reset-PTAs*. And AF-universality is decidable for L/U-PTAs only if the parameters are bounded with closed bounds (*i. e.,* of the form $p \in [a, b]$). In the three latter cases (*i. e.,* L/U-PTAs and integer-point PTAs), exact synthesis cannot be achieved though [JLR15, ALR16a], which makes our synthesis result a rarity, together with only [BL09].

Finally, our formalism is supported by the parametric model checker IMITATOR [AFKS12].

3.1.2 Outline

Section 3.2 introduces our formalism of update-to-parameter timed automata. Section 3.3 proves our general undecidability result, while Section 3.4 proves the decidability when parameters become integer-valued. Section 3.5 concludes the chapter and outlines future research directions.

3.2 Update-to-parameter Timed Automata

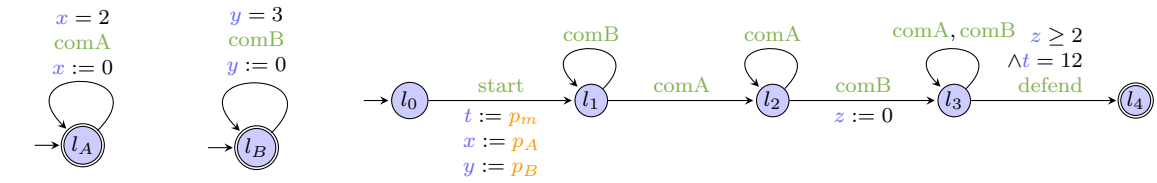
Timed automata [AD94] are an extension of finite-state automata augmented with clocks that can be compared to (usually) integer constants in guards (along edges), and that can be updated (usually) to 0 along edges. We extend this formalism by allowing clocks to be updated to *parameters*.

3.2.1 Syntax

Definition 6 (U2P-TA)**.** *An* update-to-parameter timed automaton *(U2P-TA)* $\mathcal{A}$ *is a tuple* $\mathcal{A} = (\Sigma, L, l_0, \mathbb{X}, \mathbb{P}, \zeta)$*, where:*

1. Σ *is a finite set of actions,*
2. L *is a finite set of locations,*
3. $l_0 \in L$ *is the initial location,*
4. $\mathbb{X}$ *is a finite set of clocks,*
5. $\mathbb{P}$ *is a finite set of parameters,*
6. ζ *is a finite set of edges* $e = (l, g, a, u, l')$ *where* $l, l' \in L$ *are the source and target locations,* g *is a non-parametric guard,* $a \in \Sigma$ *and* $u : \mathbb{X} \rightharpoonup \mathbb{N} \cup \mathbb{P}$ *is a parametric update function.*

Similarly to TAs, our U2P-TAs can be synchronized on shared actions, and the product of several U2P-TAs gives a U2P-TA. Their implementation within IMITATOR [AFKS12] is discussed in Section 6.5. In fact, we forbid concurrent updates of a shared clock. Note that IMITATOR does not forbid concurrent updates but will take into account only the last update written in the input file.



(a) Committee A (b) Committee B (c) A PhD student's defense workflow

Figure 3.1: A motivating example of U2iP-TA

Example 1. *Consider the U2P-TA in Figure 3.1c with five locations, four clocks (x, y, z and t) and three parameters (p_m, p_A, p_B). Observe that all three parameters are used in an update along the edge from l_0 and l_1.*

As a motivating toy example, consider the case of a PhD student aiming at obtaining the authorization of her/his university in order to defend before December (assuming the system is starting at any moment). Two committees need to give their authorization sequentially (A then B), and the student must bring both authorizations to the administration two months ahead of the defense. Committee A (resp. B) meets periodically every two (resp. three) months, which is depicted in Figures 3.1a and 3.1b, assuming time units are months.

The student workflow is modeled by the U2P-TA in Figure 3.1c, synchronizing with both committees using actions comA *and* comB *(clock x is shared between committee A and the student automaton, while y is shared between B and the student). First, the student starts the process at time p_m, using the parametric update $t := p_m$. At the same time, we set the current clock of both committees to an unknown time; that is, assuming $p_A \in [0, 2]$ and $p_B \in [0, 3]$, the last occurrence of committee A (resp. B) is p_A (resp. p_B) or, put differently, the next occurrence of committee A is $2 - p_A$ (resp. $3 - p_B$). This allows us to analyze symbolically the system, by setting the clock t, that acts as a global timer, to the accurate student start date p_m, while assuming an unknown situation of the two periodic committees. Then, the student waits for the next commission A, and gets the authorization, moving to location l_2; then, (s)he waits for the next commission B, and gets the authorization, moving to location l_3. Finally, (s)he waits two more months (using $z \geq 2$) and defends in December (encoded by $t = 12$) in location l_4. The synchronization on* comA *and/or* comB *on self-loops allows the system to remain non-blocking.*

The purpose of this analysis is to understand when in the year the student may start the workflow in order to be able to defend in December, depending on the current "offset" of the committees. That is, we want to synthesize the parameter valuations for p_m, p_A and p_B such that the system may eventually reach l_4.

Throughout this section, let K denote the largest constant in a given U2P-TA, *i. e.,* the maximum of the largest constant compared to a clock in a guard or used in an update, and the largest bound of a parameter (if the U2P-TA is bounded).

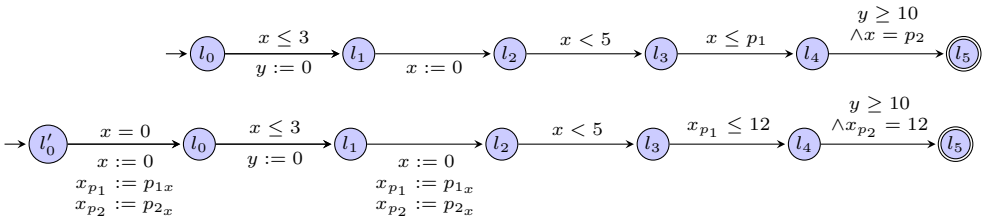


Figure 3.2: A bounded PTA $\mathcal{A}$ (above) and its equivalent $\mathsf{UtP}(\mathcal{A})$ (below)

3.3 Undecidability

In this section, we show that our extension of TAs with parametric updates leads to the undecidability of the EF-emptiness problem.

We show that any *bounded* (rational-valued) PTA [AHV93] can be transformed into a U2P-TA, and therefore that U2P-TAs are at least as expressive as (bounded) PTAs for which the EF-emptiness is known to be undecidable [Mil00].

The main idea of our proof is as follows: suppose that, in a PTA, we want to measure a (parametric) duration p. Then we can update a clock x to 0 and then test it with a guard $x = p$. But provided we know an upper bound K on p, we could, with a U2P-TA, update clock x to $K - p$ and test it with a guard $x = K$ instead. Now, since we do not allow linear expressions in updates, we instead replace $K - p$ with a new parameter p' and prove that the existence of a valuation for p' in the U2P-TA such that the property holds, is equivalent to that of a valuation for p in the initial PTA. This idea extends to other comparison operators than $=$ and its practical development requires a few clock and parameter duplications.

Let $\mathcal{A} = (\Sigma, L, l_0, \mathbb{X}, \mathbb{P}, \zeta)$ be a bounded PTA and K its largest constant. Let us define the following U2P-TA: $\mathcal{A}' = (\Sigma, L \cup \{l'_0\}, l'_0, \mathbb{X}', \mathbb{P}', \zeta')$, which has the same actions as $\mathcal{A}$. For each $x \in \mathbb{X}$, $\mathbb{X}'$ contains x and a duplicate x_p for each parameter p to which x is compared in $\mathcal{A}$. $\mathbb{P}'$ contains all parameters in $\mathbb{P}$, as well as one extra parameter per clock in $\mathbb{X}$; given a clock $x \in \mathbb{X}$, we denote by p_x its corresponding extra parameter in $\mathbb{P}'$.

Let us now build ζ', initially containing all edges of ζ, and then modified as follows. Let x be a clock. Let $e = (l_1, g, a, u, l_2)$ be an edge of $\mathcal{A}$. If $u(x) = 0$, we perform the following modifications: first, we also update x_p to p_x along e, *i. e.*, $u(x_p) = p_x$. In addition, for any edge e' comparing clock x to parameter p in its guard, we replace $x \bowtie p$ with $x \bowtie K$. All other updates and non-parametric guards remain unchanged. Finally, we add one additional location l'_0 to the locations L of $\mathcal{A}$, which will be the new initial location, and one new additional edge from l'_0 to the former inital location l_0 of $\mathcal{A}$, with guard $x = 0$ for any clock $x \in \mathbb{X}$ and which updates for all clock $x \in \mathbb{X}$, x_p to p_x.

Example 2. *An example of this construction is shown in Figure 3.2, where we assume that p_1 is bounded in $[2, 5]$ and $p_2 \in [0, 12]$—therefore $K = 12$. For example, $12 - p_{1x}$ plays the role of p_1, and $12 - p_{2x}$ plays the role of p_2.*

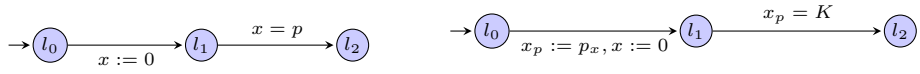
Since the initial sets of clocks $\mathbb{X}$ and $\mathbb{P}$ are finite and our set of linear constraints is finite, we only add a finite number of clocks and parameters to the new automaton. Finally, $\mathcal{A}_{RtP}$ is a U2P-TA. We denote by $\mathcal{A}_{RtP} = \mathsf{UtP}(\mathcal{A})$ this

transformation.

Note that our transformation adds to the initial system in the worst case one parameter and one clock for each comparison to a parameter, *i. e.*, $|\mathbb{P}'| + |\mathbb{X}'| \leq |\mathbb{P}| + |\mathbb{X}| + 2 \times |\mathbb{P}| \times |\mathbb{X}|$.

In order to show that EF-emptiness is undecidable for U2P-TA, we prove the following behavior: a goal location is reached by a run in a U2P-TA $\mathcal{A}$, if and only if there is a run in $\mathsf{UtP}(\mathcal{A})$ reaching it.

Consider the automaton presented in Figure 3.3a. Given a parameter valuation $v(p)$, we duplicate the clock x to x_p and update it to p_x where x is updated to 0. When x is compared to p, we replace this comparison by x_p compared to p_x, providing the automaton presented in Figure 3.3b. During an execution of Figure 3.3a accessing l_2, the time elapsed since the update of x until its comparison to p is $v(p)$. During an execution of Figure 3.3b accessing l_2, the time elapsed since the update of x_p until its comparison to p_x is $K - v(p_x)$. We define the parameter valuation $v'(p) = K - v(p_x)$. With this construction, there is a parameter valuation v such that there is a run from l_0 to l_2 in Figure 3.3a iff there is a parameter valuation v' as defined such that there is a run from l_0 to l_2 in Figure 3.3b.



(a) A PTA $\mathcal{A}$

(b) An U2P-TA $\mathsf{UtP}(\mathcal{A})$

Figure 3.3: A PTA $\mathcal{A}$ and its equivalent $\mathsf{UtP}(\mathcal{A})$

Proposition 1. *Let $\mathcal{A}$ be a bounded PTA, K its maximum constant, v be a parameter valuation, and $v' = K - v$. Let l be a goal location.*

There is a run in $v(\mathcal{A})$ reaching l iff there is a run in $v'(\mathsf{UtP}(\mathcal{A}))$ reaching l.

Proof. Let ρ be a finite run of $v(\mathcal{A})$ ending in a concrete state (l, w) and let $\sigma = e_1 \dots e_n$ be the corresponding sequence of edges taken by ρ. We build by induction on n, a run ρ' in $v'(\mathsf{UtP}(\mathcal{A}))$ ending in a concrete state (l, w') such that for all $x \in \mathbb{X}, w'(x) = w(x)$ and for all clock $x' \in \mathbb{X}' \setminus \mathbb{X}, w'(x_p) = K - v(p) + w(x)$.

If $n = 0$, then ρ' consists only of the additional initial edge of $\mathsf{UtP}(\mathcal{A})$, which clearly sets all clocks to the adequate values.

Suppose now that we have built ρ' for size n and consider a run ρ with $n + 1$ edges. Then ρ consists of a run ρ_1, ending in (l_1, w_1) with n edges followed by a delay d and finally a discrete transition along the last edge e. From the induction hypothesis, we can build an equivalent run ρ'_1 in $\mathsf{UtP}(\mathcal{A})$ ending in (l_1, w'_1), such that for all $x \in \mathbb{X}, w'_1(x) = w_1(x)$ and for all clock $x_p \in \mathbb{X}' \setminus \mathbb{X}, w'_1(x_p) = K - v(p) + w_1(x)$. Let w_2 (resp. w'_2) be the clock valuation obtained in $\mathcal{A}$ (resp. $\mathsf{UtP}(\mathcal{A})$) after the delay d. By construction, the part of the guard of e comparing clocks in $\mathbb{X}$ to constants is satisfied by w'_2 since it is the same as in $\mathcal{A}$. Further, for each clock $x \in \mathbb{X}$, such that $x \bowtie p$ along e in $\mathcal{A}$, we have instead $x_p \bowtie K$ along the modified e in $\mathsf{UtP}(\mathcal{A})$. But $w'_2(x_p) = K - v(p) + w_2(x)$, so the latter comparison is equivalent to $K - v(p) + w_2(x) \bowtie K$, *i. e.*, $w_2(x) \bowtie v(p)$. So, since the guard is satisfied in $\mathcal{A}$ by w_2, the corresponding guard is satisfied in $\mathsf{UtP}(\mathcal{A})$ by w'_2. Then clocks in $\mathbb{X}$ are updated normally, and for all clocks

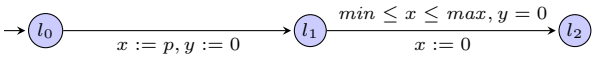


Figure 3.4: A gadget that ensures a parameter p is bounded by min and max

$x_p \in \mathbb{X}' \setminus \mathbb{X}$, we have an update to $v'(p_x) = K - v(p)$, which concludes the induction.

The other direction, starting from a run in $\mathsf{UtP}(\mathcal{A})$, is similar. □

Theorem 1. *The* EF*-emptiness problem is undecidable for bounded U2P-TAs.*

Proof. From the undecidability of EF-emptiness for bounded PTAs [Mil00]. □

We now show that this result can be extended to the full class of (unbounded) U2P-TAs.

Theorem 2. *The* EF*-emptiness problem is undecidable for U2P-TAs.*

Proof. Similarly to the proof of [ALR16b, Proposition 8], we claim that a bounded U2P-TA can be easily simulated using an unbounded U2P-TA. We present a gadget in Figure 3.4 that uses two clocks (that can be clocks used by the PTA) and two transitions that can be added before the initial location of any unbounded U2P-TA, and ensures a parameter p is bounded, *i. e.,* given two integer constants min and max we have $p \in [min, max]$. We need one gadget per parameter; these gadgets can be branched sequentially before the initial location of an unbounded U2P-TA, and all clocks must be updated to 0 before entering the *original* initial location.

The gadget works as follows: when taking the first transition from l_0 to l_1, clock x is updated to p and clock y to 0. The transition from l_1 to l_2 can be taken if and only if in a 0-delay ensured by the guard $y = 0$, we have that $x \leq max$ and $min \leq x$. This means there is a run from l_0 to l_2 if and only if there is a parameter valuation v such that $min \leq v(p) \leq max$, which in other words means that the parametric domain is bounded.

As from Theorem 1 the EF-emptiness problem is undecidable for bounded U2P-TA, and as any bounded U2P-TA can be expressed using a U2P-TA, we conclude that the EF-emptiness problem is undecidable for unbounded U2P-TA. □

Corollary 1. *The* AF*-emptiness problem is undecidable for U2P-TAs.*

Proof. The AF-emptiness problem is undecidable for PTAs as it is proven undecidable for one of its subclasses in [JLR15]. Since we can encode a PTA into a U2P-TA, it is undecidable for the former. □

Corollary 2. AF*,* EF*-universality problems are undecidable for U2P-TAs.*

Proof. In [ALR16a], EG, AG-emptiness problems are proven undecidable for PTAs. As AF, EF-universality are their equivalent respectively, they are also undecidable for PTAs, and therefore for U2P-TAs. □

3.4 Decidability

Let us now show that, when parameters are restricted to (unbounded) integers, the EF-emptiness problem becomes PSPACE-complete.

If parameters in an U2P-TA only have (possibly unbounded) integer valuations, we say it is an U2iP-TA. Note that once valuated by an integer parameter valuation v, an U2iP-TA is an updatable timed automaton with updates to integer constants, as defined in [BDFP04, Section 3.1]. Hence *clock regions* are still topical in this context [BDFP04, Section 5.1]. Let us recall the notion of clock region [AD94]. Given a clock x and a clock valuation w, recall that $\lfloor w(x) \rfloor$ denotes the integer part of $w(x)$ while $frac(w(x))$ denotes its fractional part.

Definition 7 (clock region)**.** *For two clock valuations w and w', $\sim$ is an equivalence relation defined by: $w \sim w'$ iff*

1. *for all clock x, either $\lfloor w(x) \rfloor = \lfloor w'(x) \rfloor$ or $w(x), w'(x) > K$;*
2. *for all clocks x, y with $w(x), w(y) \leq K$, $frac(w(x)) \leq frac(w(y))$ iff $frac(w'(x)) \leq frac(w'(y))$;*
3. *for all clock x with $w(x) \leq K$, $frac(w(x)) = 0$ iff $frac(w'(x)) = 0$.*

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

Two clock valuations in the same clock region reach the same region by time elapsing, satisfy the same guards and thus can take the same transitions [AD94]. It is a *bisimulation* relation.

Theorem 3. *The set of parameter valuations for which a given location is reachable is effectively computable for U2iP-TA.*

Proof. We first need an intermediate lemma:

Lemma 1. *Let $\mathcal{A}$ be an U2iP-TA. Let K be the greatest constant in $\mathcal{A}$. Let l be a goal location. Let v, v' be two rational parameter valuations s.t. for all parameter p, either $v(p) = v'(p)$ or $v(p) > K$ and $v'(p) > K$. There is a run in $v(\mathcal{A})$ reaching (l, w) iff there is a run in $v'(\mathcal{A})$ reaching (l, w') s.t. at each state, two clock valuations of ρ and ρ' are in the same clock region and location.*

Proof. By induction on the length of the run. Let v, v' be such parameter valuations.

For a run of length 0 of $v(\mathcal{A})$, there is a run of length 0 of $v'(\mathcal{A})$ reaching the initial location. If there is a run of length 0 of $v'(\mathcal{A})$, there is a run of length 0 of $v(\mathcal{A})$ reaching the initial location.

Now, suppose the result holds for every run of length i. Assume a run of $v(\mathcal{A})$ of length $i+1$, with a prefix ρ of length i reaching (l_i, w_i) followed by a state obtained using edge $e = (l_i, g, a, u, l_{i+1})$. That is, the run is of the form $\rho \xrightarrow{e} (l_{i+1}, w_{i+1})$.

By induction hypothesis, let ρ' be a run of $v'(\mathcal{A})$ reaching (l_i, w'_i) s.t. at each state, two clock valuations of ρ and ρ' are in the same clock region and location.

Now if for all clock x, no $w_i(x)$ is the result of a parametric update, then trivially $w_i \models g$ and as $w_i \sim w'_i$, $w'_i \models g$. Alternatively, suppose for some x and parameter p, we have $w_i(x) = v(p)$. If $v(p) < K + 1$ and $w_i \models g$,

since $v'(p) = v(p)$ then as $w_i \sim w'_i$, $w'_i \models g$. If $v(p) \geq K+1$ and $w_i \models g$, since $v'(p) \geq K+1$ then as $w_i \sim w'_i$, $w'_i \models g$. We treat the case of multiple updates of clocks to parameters in e the same way. Finally, we can take the transition e with the same delay. Hence $\xrightarrow{e} (l_{i+1}, w'_{i+1})$ is a run of $v'(\mathcal{A})$ of length $i+1$ reaching l_{i+1} with the same actions, locations, delays and at each state, two clock valuations of ρ and ρ' are in the same clock region and location.

The other way is a direct consequence of the previous paragraph and the definition of the clock regions.

□

We can now go back to the proof of Theorem 3. Let $\mathcal{A}$ be an U2iP-TA and K be the greatest constant in $\mathcal{A}$. Now let v be a (integer) parameter valuation. Since $v(\mathcal{A})$ is an updatable timed automaton, the reachability of a given state (l, w) is decidable [BDFP04, Section 5]. It is sufficient to enumerate all integer valuations s.t. for each parameter p, $v(p) \leq K+1$. Indeed, from Lemma 1 a parameter valuation v with $v(p) > K+1$ allows to take the same transitions and reach the same guards as the parameter valuation v' s.t. for all $p' \neq p$, $v(p') = v'(p')$ and $v'(p) = K+1$ so we can replace such parameter valuations by a valuation v' as defined previously. In conclusion, there is a finite number of parameter valuations to test to obtain the full set of valuations for which the goal location is reachable. □

Proposition 2. *The* EF*-emptiness problem is PSPACE-complete for U2iP-TAs.*

Proof. Since we can synthesize exactly the set of parameter valuations for which the goal location is reachable using Theorem 3, the decidability of the EF-emptiness follows immediately.

Let us now have a look at the complexity of the EF-emptiness problem for U2iP-TA. First, since a TA is a special case of U2iP-TA with no parametric update, we have the PSPACE-hardness for EF-emptiness in our U2iP-TA [AD94]. Now, let G be a set of goal locations of $\mathcal{A}$. Consider the non-deterministic Turing machine that:

1. takes $\mathcal{A}$, G and K as input
2. non-deterministically "guesses" an integer valuation v bounded by $K+1$ and writes it to the tape
3. overwrite on the tape each parameter p by $v(p)$, giving the updatable TA $v(\mathcal{A})$
4. solves reachability in $v(\mathcal{A})$ for G
5. accepts iff the result of the previous step is "yes".

The machine accepts iff there is an integer valuation v bounded by $K+1$ and a run in $v(\mathcal{A})$ reaching a location $l \in G$.

The size of the input is $|\mathcal{A}| + |G| + |K|$, using $|.|$ to denote the size in bits of the different objects. There are at most $(K+1)^M$ possible valuations, where M is the number of parameters in $\mathcal{A}$. Storing the valuation at step 2 uses at most $M \times |K+1|$ additional bits, which is polynomial w.r.t. the size of the input. Step 4 also needs polynomial space from [BDFP04]. So globally this non-deterministic machine runs in polynomial space. Finally, by Savitch's theorem we have PSPACE = NPSPACE [Sav70], and the expected result. □

The following result is direct from Theorem 3:

Corollary 3. *The* EF*-universality problem is decidable for U2iP-TAs.*

Proof. Using Lemma 1 (see proof of Theorem 3) given an U2iP-TA $\mathcal{A}$ and its greatest constant in $\mathcal{A}$, there is a finite number of parameter valuations to test. Therefore given a goal location l, it is sufficient to test whether for all parameter valuations, there is a run reaching l in the valuated instance of $\mathcal{A}$. □

We state also the two following corollaries that fulfill the last unknown decision problems considered in this chapter for U2P-TAs.

Corollary 4. *The set of parameter valuations for which a given location is unavoidable is effectively computable for U2iP-TA.*

Proof. Let $\mathcal{A}$ be an U2iP-TA and v a parameter valuation. As we use in our construction the same clock regions as in [AD94], suppose there is a run in $v(\mathcal{A})$ reaching a location l, then all runs going through the same clock regions are equivalent—they satisfy the same guards, and end in the same region after an update and after letting time elapse. Moreover, using the construction of the region automaton of [AD94], it is sufficient to test whether all runs in the region automaton of $\mathcal{A}$ reach l, which are a finite number. Using the same reasoning as in the proof of Theorem 3 we obtain our result. □

Corollary 4 leads to the decidability of the AF-emptiness problem. Following the same reasoning as in Theorem 3, we state the last but not least result of this chapter:

Corollary 5. *The* AF*-emptiness and* AF*-universality problems are decidable for U2iP-TAs.*

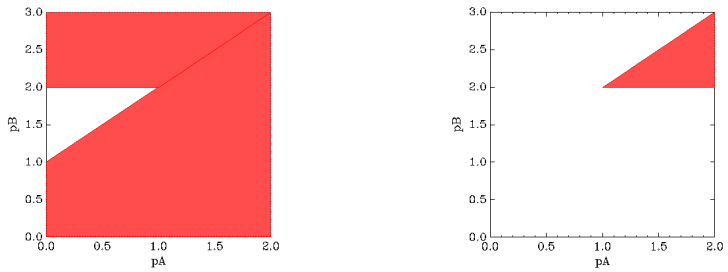
Proof. Given an U2iP-TA $\mathcal{A}$ and using the same reasoning as in the previous proof and the region automaton of [AD94], we can test whether all runs in this region automaton reach l, which are a finite number. As there is a finite number of parameter valuations to test, we can compute the set of parameter valuations such that all runs reach l (*i. e.*,, AF-synthesis) from Corollary 4. Testing the emptiness of the obtained set of parameter valuations gives AF-emptiness. Given a goal location l, it is sufficient to test whether for all parameter valuations, there is a run reaching l in the valuated instance of $\mathcal{A}$ to decide AF-universality. □

Implementation in IMITATOR

U2P-TAs (and naturally U2iP-TA) are natively supported by IMITATOR [AFKS12], a parametric model checker taking as input extensions of parametric timed automata.

Passing Example 1 as input and using the reachability synthesis algorithm, IMITATOR synthesizes the following constraint:

$$p_B + 4 \geq p_m \wedge p_B \geq p_A + 1 \wedge p_B \leq 3$$
$$\vee$$
$$p_m \leq p_B + 7 \wedge p_A \leq 2 \wedge p_B \leq p_A + 1$$



(a) $p_m = 6$ (b) $p_m = 9$

Figure 3.5: Graphical visualization in two dimensions of the parameter synthesis of Example 1

The first conjunction of inequalities states that, if the committee B is the next to meet (which is encoded by $p_B \geq p_A + 1$, and could also be written as $3 - p_B \leq 2 - p_A$), then the month p_m at which the student starts the process should be less than 4 plus the number of months since the last occurrence of committee B. (The last inequality simply recalls that p_B is less than or equal to 3). The second conjunction of inequalities states that, if the committee A is the next to meet, then the month p_m at which the student starts the process should be less than 7 plus the number of months since the last occurrence of committee B.

For any such valuation, there exists a run of the system (*i. e.,* a configuration of the committees dates respecting their respective periods) such that the student may defend in December. Also note that, if we add proper invariants[1], then the system becomes completely deterministic and the valuations for which there exists a run reaching l_4 are also such that all runs reach l_4 (since there exists only one run), and therefore the student is *guaranteed* to be able to defend in December for any of these valuations.

We can also study a situation where the system is only partially parameterized: assume $p_m = 6$, *i. e.,* the student will start the process in June in any case. The constraint encoding the current state of committees A and B is given by:

$$p_A \leq 2 \wedge p_B \leq p_A + 1$$
$$\vee$$
$$p_B \geq 2 \wedge p_B \leq 3 \wedge p_B \geq p_A + 1$$

A graphical visualization (output by IMITATOR) is given in Figure 3.5a (plain red depicts good valuations, *i. e.,* for which the student may defend in December).

Alternatively, if $p_m = 9$ (*i. e.,* the student starts the process in September), then the constraint on p_A and p_B is as follows:

$$p_B \geq 2 \wedge p_A \leq 2 \wedge p_A + 1 \geq p_B$$

A graphical visualization is given in Figure 3.5b.

Finally note that this entire example is not restricted to integer-valued parameters (rational-valued months can be used to denote finer time grain, *e. g.,*

[1]Precisely, $x \leq 2$ in committee A, $y \leq 3$ in committee B, and $t \leq 12$ in the student automaton.

days or even hours), and it therefore falls in the undecidable case of Theorem 1. Nevertheless, IMITATOR terminates here with an exact (sound and complete) result.

3.5 Conclusion

In this chapter we defined two new formalisms to model concurrent timed systems with uncertainty: U2P-TA for which we proved that the EF-emptiness problem is undecidable, even for bounded parameters, and U2iP-TA for which we proved that the EF-emptiness problem is PSPACE-complete. This discrepancy between integer-valued and rational-valued was already spotted in parametric timed automata: the EF-emptiness is decidable for integer-valued parameters with 1 parametric clock (*i. e.,* a clock compared to a parameter in at least one guard) and 3 non-parametric clocks [AHV93], while it becomes undecidable over rational-valued parameters [Mil00]. Similarly, the discrepancy between (rational-valued) bounded parameters and unbounded parameters is reminiscent of the recent result we showed for EG-emptiness ("is the set of valuations for which at least one maximal run remains in a given set of locations empty?"): this problem is decidable for bounded L/U-PTAs (a parameter is either used as an upper bound or a lower bound in guards) with rational-valued parameters, while it becomes undecidable for the full class of L/U-PTAs [AL17]. Furthermore, we extended our undecidability results to the EF-universality, AF-emptiness and AF-universality problems for U2P-TA, but also our decidability results to these same problems for U2iP-TA. This chapter therefore handles a wide range of decision problems for U2P-TA. We assume that the decidability could be extended to the full TCTL model checking following a similar reasoning.

The fact that we allow update to parameters in the (possibly parametric) timed extensions of finite-state automata is quite new and, to the best of our knowledge, has not been investigated until now. Despite having an undecidability result when the parameter domain is rational, we believe this new formalism, improved with parameters allowed in guards, could become decidable even over rational-parameters if we add a few semantic restrictions. Indeed, reset-PTAs have been studied in [ALR16a] and are a promising subclass of PTA to extend. For this purpose, we would like to explore PTAs in which update to parameters is also allowed, and under which conditions the EF-emptiness problem could become decidable. Moreover, the semantic restrictions of reset-PTAs (a clock is updated to 0 whenever it is compared to a parameter) is in a way reminiscent to *initialized* rectangular hybrid automata (a variable is updated whenever its dynamic changes) presented in [HKPV98] and it would be interesting to study these systems in which we involve parameters. Therefore, extending our result to hybrid automata is also an interesting perspective.

Finally, beyond the toy aspect of Example 1, we believe that U2iP-TAs can be used to model scheduling problems for real-time systems subject to uncertainty, notably in the tasks offsets, as this is where we used parameters in Figure 3.1.

Now that we have studied parametric updates in TAs, naturally we will try to add parametric updates in PTAs, as PTAs are a powerful extension of TAs. This is the subject of the next Chapter 4.

Chapter 4

Parametric updates in parametric timed automata

4.1 Introduction

In this chapter, we consider an extension of PTAs and establish the decidability of the EF-emptiness problem (that asks if the set of parameter valuations for which a given location is reachable in the resulting TA is empty), and the corresponding parameter synthesis problem EF-synthesis. The key to the decidability is to impose a restriction on the transitions in the PTA in which a clock is compared to a parameter in the guard, or a clock is updated to a parameter in the update.

Recall that the EF-emptiness problem is decidable for L/U-PTAs [HRSV02, BL09] and for PTAs under several restrictions [BO14]; however, most other problems are undecidable (*e. g.,* [BL09, Qua14, JLR15, ALR16a, AL17]).

4.1.1 Contribution

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 (*i. e.,* U2P-PTA), 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 [HRSV02] and integer-points (IP-)PTAs [ALR16a]); 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 [JLR15, ALR16a].

4.1.2 Related Work

Our construction is reminiscent of the parametric difference bound matrices (PDBMs) defined in [QSW17, 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 [QSW17] 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 [QSW17]. Related DBMs with an additional parameter are studied such as shrunk DBMs [SBM14, BMRS19] and infinitesimally enlarged DBMs [San15].

Allowing parameters in clock updates is inspired by the updatable TA formalism defined in [BDFP04] where clocks can be updated not only to 0 ("reset") but also to rational constants ("update"). In [ALR18b], we extended the result of [BDFP04] 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 [BL09, JLR15, ALR18b]. 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 [ALR16a], but not thoroughly studied. Finally, updating clocks on each transition in which a parameter appears is reminiscent of the initialized rectangular hybrid automata [HKPV98], which remains one of the few decidable subclasses of hybrid automata.

Section 4.2 recalls preliminaries. Section 4.3 presents R-U2P-PTA along with our decidability result. Section 4.7 gives a concrete application of our result.

4.2 Preliminaries

Throughout this chapter, we assume $\triangleleft \in \{<, \leq\}$ and $\bowtie \in \{<, \leq, \geq, >\}$.

Given a clock x and a clock valuation w, recall that $\lfloor w(x) \rfloor$ 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.

Definition 8. *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) $l_0 \in L$ 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 4.1. 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).

Recall that K denotes 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

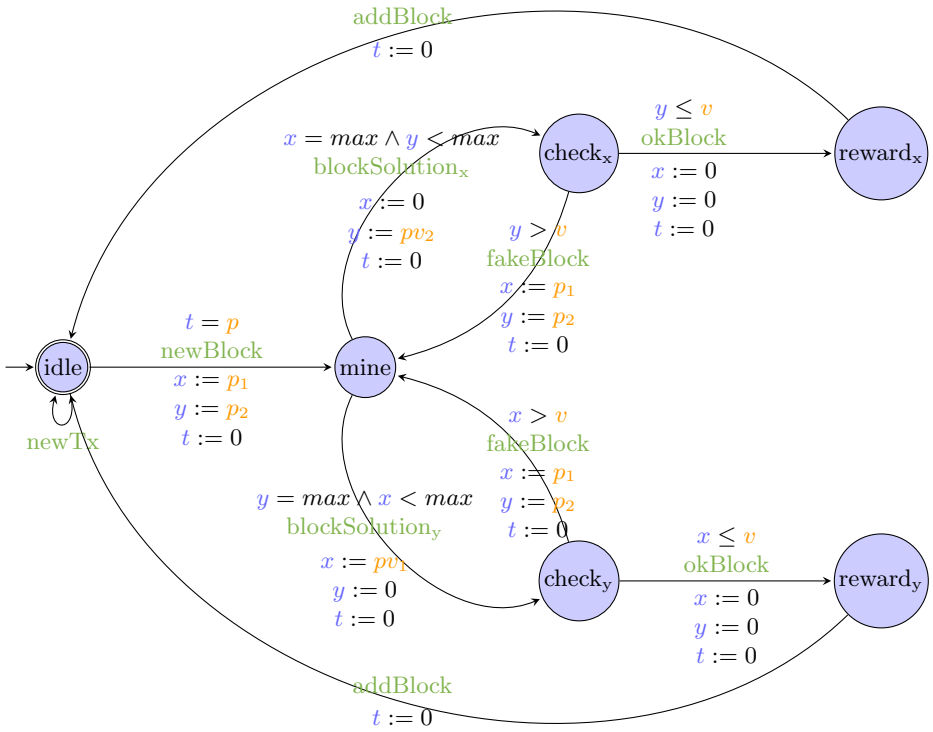


Figure 4.1: A proof-of-work modeled with a bounded R-U2P-PTA.

upper bound of a parameter (if the U2P-PTA is bounded). Also recall that two clock valuations in the same clock region (cf. Definition 7) and reach the same regions by time elapsing, satisfy the same guards and can take the same transitions [AD94]. In this chapter, we address the EF-emptiness problem.

4.3 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 9. *An* R-U2P-PTA *is a U2P-PTA where for any edge* $\langle l, g, a, u, l' \rangle \in \zeta$, u *is a total function whenever:*[1]

1. g *is a parametric guard, or*
2. $u(x) \in \mathbb{P}$ *for some* $x \in \mathbb{X}$.

[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.

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 [AD94]. 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 [HRSV02]) 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 10 (regions of parameters)**.** *We write that $v \frown v'$ if*

1. *for all parameter p, $\lfloor v(p) \rfloor = \lfloor v'(p) \rfloor$;*
2. *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 $\frown$, and we will use the notation R_p for parameter regions. The set of all *parameter regions* is denoted by $\mathcal{R}_p$. The definition is in a way similar to Definition 7 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 updated 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 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 11 (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*
   (a) *for all $v \in R_p$, $v(d_1) < v(d_2)$ evaluates to true regardless of $\lhd_1, \lhd_2$, or*
   (b) *for all $v \in R_p$, $v(d_1) \leq v(d_2)$ evaluates to true, $\lhd_1 = <$ and $\lhd_2 = \leq$;*

2. $\triangleleft = \leq$, *and either*

   (a) *for all* $v \in R_p$, $v(d_1) < v(d_2)$ *evaluates to true regardless of* $\triangleleft_1, \triangleleft_2$, *or*

   (b) *for all* $v \in R_p$, $v(d_1) \leq v(d_2)$ *evaluates to true, and* $\triangleleft_1 = \triangleleft_2$, *or* $\triangleleft_1 = <$.

Transitivity is immediate from the definition: if $D_1 \triangleleft_1 D_2$ and $D_2 \triangleleft_2 D_3$ are valid for R_p, $D_1(\triangleleft_1 \oplus \triangleleft_2)D_3$ is valid for R_p.

The following lemma derives from Definition 11:

Lemma 2 (validity of addition)**.** *Let* $d_1, d_2, d_3, d_4 \in \mathcal{PLT}$. *Let* R_p *be a parameter region. If* $(d_1, \triangleleft_1) \leq (d_2, \triangleleft_2)$ *and* $(d_3, \triangleleft_3) \leq (d_4, \triangleleft_4)$ *are valid for* R_p *then* $(d_1, \triangleleft_1) + (d_3, \triangleleft_3) \leq (d_2, \triangleleft_2) + (d_4, \triangleleft_4)$ *is valid for* R_p.

Proof. Four cases show up: for all $v \in R_p$,

- $v(d_1) < v(d_2)$ and $v(d_3) < v(d_4)$, then clearly $v(d_1) + v(d_3) < v(d_2) + v(d_4)$ and we have our result from Definition 11 (2a).
- $v(d_1) < v(d_2)$ and $v(d_3) \leq v(d_4)$, then $v(d_1) + v(d_3) < v(d_2) + v(d_4)$ and we have our result from Definition 11 (2a).
- $v(d_1) \leq v(d_2)$ and $v(d_3) < v(d_4)$, then $v(d_1) + v(d_3) < v(d_2) + v(d_4)$ and we have our result from Definition 11 (2a).
- $v(d_1) \leq v(d_2)$ and $v(d_3) \leq v(d_4)$, then $v(d_1) + v(d_3) \leq v(d_2) + v(d_4)$ and

  1. if $\triangleleft_1 = \triangleleft_2$ and $\triangleleft_3 = \triangleleft_4$ then $\triangleleft_1 \oplus \triangleleft_3 = \triangleleft_2 \oplus \triangleleft_4$ and we have our result from Definition 11 (2b).
  2. if $\triangleleft_1 = \triangleleft_2$ and $\triangleleft_3 = <$, $\triangleleft_4 = \leq$ then $\triangleleft_1 \oplus \triangleleft_3 = <$ and $\triangleleft_2 \oplus \triangleleft_4$ is either $<$ or $\leq$ and we have our result from Definition 11 (2b).
  3. if $\triangleleft_1 = <$, $\triangleleft_2 = \leq$ and $\triangleleft_3 = \triangleleft_4$ then $\triangleleft_1 \oplus \triangleleft_3 = <$ and $\triangleleft_2 \oplus \triangleleft_4$ is either $<$ or $\leq$ and we have our result from Definition 11 (2b).
  4. if $\triangleleft_1 = \triangleleft_3 = <$ and $\triangleleft_2 = \triangleleft_4 = \leq$ then $\triangleleft_1 \oplus \triangleleft_3 = <$ and $\triangleleft_2 \oplus \triangleleft_4 = \leq$ and we have our result from Definition 11 (2b).

From Definition 11 (2a, 2b) we have that $(d_1, \triangleleft_1) + (d_3, \triangleleft_3) \leq (d_2, \triangleleft_2) + (d_4, \triangleleft_4)$ is valid for R_p. □

We can now define our data structure, namely p–PDBMs (for *precise Parametric Difference Bound Matrices*), inspired by the PDBMs of [HRSV02] themselves inspired by DBMs [Dil89]. 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 [AD94]. 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 $D_{i,0} = (d_{i,0}, \triangleleft_{i0})$ is interpreted as the constraint $frac(x_i) \triangleleft_{i0} d_{i,0}$, and $D_{0,i} = (d_{0,i}, \triangleleft_{0i})$ as the constraint $-frac(x_i) \triangleleft_{0i} d_{0,i}$. For $i \neq 0$ and $j \neq 0$, the matrix cell $D_{i,j} = (d_{i,j}, \triangleleft_{ij})$ is interpreted as $frac(x_i) - frac(x_j) \triangleleft_{ij} d_{i,j}$. Finally for all i, $D_{i,i} = (0, \leq)$.

Our p–PDBMs are partitioned into two types: open–p–PDBMs and point–p–PDBMs. A point–p–PDBM is a clock region defined by only parameters which

contains only one clock valuation; that is, it corresponds to a set of inequalities of the form $x_i = p_j$. In contrast, an open–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 of [AD94]). In particular, the initial clock region $\{0^H\}$ and any clock region $\{E_i^H\}$ where E_i is an integer for all clock x_i, is an open–p–PDBM.

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 an open–p–PDBM until the next (total) parametric update.

Definition 12 (open–p–PDBM)**.** *Let R_p be a parameter region. An* open–p–*PDBM for R_p is a pair (E, D) with $E = (E_1, \cdots, E_H)$ a vector of H integers (or ∞) 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:*

1. *For all i, $(-1, <) \leq D_{0,i} \leq (0, \leq)$ 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.rr*
3. *For all i, j, if $d_{i,j} = -d_{j,i}$ and is different from 1 then $\lhd_{ij} = \lhd_{ji} = \leq$, else $\lhd_{ij} = \lhd_{ji} = <$,*
4. *For all i, j, k, $D_{i,j} \leq D_{i,k} + D_{k,j}$ is valid for R_p (canonical form), and*
5. *(a) There is at least one i s.t. $D_{i,0} = D_{0,i} = (0, \leq)$, or*
   (b) there is at least one i s.t. $D_{i,0} = (1, <)$ and for all j s.t. $D_{0,j} = (0, \lhd_{0j})$, then we have $\lhd_{0j} = <$.

An open–p–PDBM satisfying condition 5a can be seen as a subregion of an open line segment or a corner point region of [AD94, fig. 9 example 4.4] (it can be seen as a *border* region) and one satisfying condition 5b can be seen as a subregion of an open region of [AD94, fig. 9 example 4.4] (it can be seen as a *center* region). Remark that sets of the form $\{frac(w(x)) \mid 0 \leq frac(w(x)) \leq 1\}$ are forbidden by Definition 12 (3), as in the regions of [AD94].

Let R_p be a parameter region. In the following, $p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ is the set of all possible open–p–PDBMs (E, D) for R_p.

The second type is the point–p–PDBM. It represents the unique clock valuation (for a given parameter valuation) obtained after a total parametric update in an U2P-PTA.

Definition 13 (point–p–PDBM)**.** *Let R_p be a parameter region. A* point–p–*PDBM for R_p is a pair (E, D) where D is an $(H+1)^2$ matrix where each element $D_{i,j}$ is a pair $(d_{i,j}, \leq)$ and for all $0 \leq i, j \leq H$, $d_{i,0} = frac(p_1) = -d_{0,i}$, and $d_{i,j} = frac(p_1) - frac(p_2) = -d_{j,i}$, for any $p_1, p_2 \in \mathbb{P}$. and for all $1 \leq i \leq H$, $E_i = \lfloor p_k \rfloor$ if $d_{i,0} = frac(p_k)$, for $1 \leq k \leq M$. In addition:*

1. *For all i, $(-1, <) \leq D_{0,i} \leq (0, \leq)$ and $(0, \leq) \leq D_{i,0} \leq (1, <)$ are valid for R_p,*

2. For all i, j, k, $D_{i,j} \leq D_{i,k} + D_{k,j}$ is valid for R_p (canonical form).

The fact that D is antisymmetric *i. e.,* for all i, j, $D_{i,j} = -D_{j,i}$, means that each clock is valuated to a parameter and each difference of clocks is valuated to a difference of parameters.

The set of all point–p–PDBM for R_p is denoted by $p\text{–}\mathcal{PDBM}_{\odot}(R_p)$, and the set of all p–PDBMs for R_p by $p\text{–}\mathcal{PDBM}(R_p)$ (hence $p\text{–}\mathcal{PDBM}(R_p) = p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p) \cup p\text{–}\mathcal{PDBM}_{\odot}(R_p)$).

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 $\bigwedge_{i,j\in[0,H]} frac(x_i) - frac(x_j) \triangleleft_{i,j} d_{i,j} \wedge \bigwedge_{i\in[1,H]} \lfloor x_i \rfloor = E_i$. Given a p–PDBM (E, D) and a parameter valuation v, we denote by $(E, v(D))$ the *valuated* p*–PDBM*, *i. e.,* the set of clock valuations defined by:

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

For a clock valuation w, we write $w \in (E, v(D))$ if it satisfies all constraints of $(E, v(D))$.

The following two lemmas derive from the above definitions of point–p–PDBM and p–PDBMs:

Lemma 3 (positivity of reflexivity)**.** *Let R_p be a parameter region and (E, D) be a* p*–PDBM for R_p. For all clocks i, j, $(0, \leq) \leq D_{i,j} + D_{j,i}$ is valid for R_p.*

Proof. By condition (4) in Definition 12 and Definition 13 (2), we have that $D_{i,i} \leq D_{i,j} + D_{j,i}$ is valid for R_p; the result follows from the fact that $D_{i,i} = (0, \leq)$ (again from Definition 12 and Definition 13). □

Lemma 4 (neutral element of the set of cells)**.** *Let R_p be a parameter region and (E, D) be a* p*–PDBM for R_p. For all clocks i, j, $D_{i,j} \leq D_{i,j} + D_{j,j}$ and $D_{i,j} \leq D_{i,i} + D_{i,j}$ are valid for R_p.*

Proof. Let R_p be a parameter region and (E, D) be a p–PDBM for R_p. Let $D_{i,j} = (d_{i,j}, \triangleleft_{ij})$ with $d_{i,j} \in \mathcal{PLT}$. By Definition 12 and Definition 13 for all clock i, $D_{i,i} = (0, \leq)$. We have $D_{j,i} + D_{i,i} = (d_{j,i} + 0, \triangleleft_{ij} \oplus \leq) = D_{j,i}$. Moreover from Definition 11 (2b) $D_{i,j} \leq D_{i,j}$ is valid for R_p. Hence $D_{i,j} \leq D_{i,i} + D_{i,j}$ is valid for R_p. The same way we prove $D_{i,j} \leq D_{i,j} + D_{j,j}$ is valid for R_p. □

But let us first clarify our needs graphically. 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 4.2.

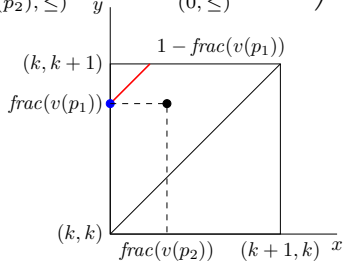


Figure 4.2: Graphical representations of p–PDBMs and [AD94] regions

(2) In contrast, an open–p–PDBM satisfying condition (5a) 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 [AD94]). In particular, the initial clock region $\{0^H\}$ and any clock region that is a single integer clock valuation is a p–PDBM. An open–p–PDBM satisfying condition 5a 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 [AD94, 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 4.2. The open line segment of [AD94, 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 4.2.

(3) An open–p–PDBM satisfying condition (5b) is a clock region which can contain several clock valuations satisfying some possibly parametric constraints (as the open region of [AD94]). An open–p–PDBM satisfying condition (5b) 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 [AD94, fig. 9 example 4.4]. After some time elapsing, and *before* any clock valuation reaches the next integer $k+1$—therefore the next open–p–PDBM satisfying condition 5a—, 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 4.2. The open region of [AD94, 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 4.2.

Remark that sets of the form $\{frac(w(x)) \mid 0 \leq frac(w(x)) \leq 1\}$ are in contradiction with Definition 12 (3) and therefore cannot be part of a p–PDBM, as in the regions of [AD94]. 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 open–p–PDBM satisfying condition 5a or 5b until the next (total) parametric update.

The differentiation made in the previous paragraph between open–p–PDBMs satisfying condition 5a and 5b is intended to give an intuition to the reader about the inclusion of p–PDBMs into [AD94] clock regions. Technical details are given in the following Section 4.4. In the following subsections Sections 4.4.1 to 4.4.5, 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.

4.4 Operations on p–PDBMs

4.4.1 Non-parametric update

To apply a non-parametric update on a p–PDBM, following classical algorithms for DBMs [BY03], we define an update operator, given in Algorithm Algorithm 1.

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

1. updating E_x to k;
2. setting the fractional part of x to 0: $D_{x,0} := D_{0,x} := (0, \leq)$;
3. 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.

Algorithm 1: $update(D, u_{np})$: for all clock x_i where u_{np} is defined, update $frac(x_i) := 0$

```
1 foreach x_i where u_np(x_i) is defined do
2 |   D_{i,0} := D_{0,i} = (0, ≤)
3 |   for j from 1 to H do
4 |   |   D_{i,j} = D_{0,j}
5 |   |   D_{j,i} = D_{j,0}
6 |   end
7 end
```

Definition 14 (update of a p–PDBM)**.** *Let u_{np} be a non-parametric update function. Given $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$, we define the* update *of (E, D), denoted*

by $(E', D') = update((E, D), u_{np})$ *as:* D' *is the result of Algorithm Algorithm 1 and for each clock* x *if* $u_{np}(x)$ *is defined* $E'_x := u_{np}(x)$, $E'_x := E_x$ *otherwise.*

Lemma 5 (stability under update)**.** *Let* R_p *be a parameter region and* $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$*. Let* u_{np} *be a non-trivial non-parametric update. Then* $update((E, D), u_{np}) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$*.*

Proof. We split this proof in two parts: the first one treats the case of point–p–PDBMs and the second one of open–p–PDBMs.

The following lemma shows that applying a *update* on any point–p–PDBM transforms it into an open–p–PDBM.

Lemma 6 ($p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ becomes $p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ after *update*)**.** *Let* R_p *be a parameter region and* $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$*. Let* u_{np} *be a non-parametric update. Then* $update((E, D), u_{np}) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$*.*

Proof. Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$. Consider $(E', D') = update((E, D), u_{np})$. After applying Algorithm 1, for all clock x_i of (E, D) where u_{np} is defined, $E'_i = u_{np}(x_i)$; moreover for all clock j, $D'_{i,j} = D_{0,j}$ and $D'_{j,i} = D_{j,0}$. First note that if x_i, x_j have been updated, $D'_{i,j} = D'_{j,i} = D'_{0,j} = D'_{j,0} = D'_{0,i} = D'_{i,0} = (0, \leq) = D_{0,0}$. For all clocks i, j, k, the following inequalities are valid for R_p:

1. (a) if x_i is updated: $D'_{i,0} = (0, \leq) = D'_{0,i}$ and therefore trivially it holds that $-1 \leq D'_{0,i} \leq 0$ and $0 \leq D'_{i,0} \leq 1$ are valid for R_p;
   (b) if x_i is not updated: $D'_{i,0} = D_{i,0}$ and therefore $-1 \leq D'_{0,i} \leq 0$ and $0 \leq D'_{i,0} \leq 1$ are valid for R_p because these constraints were already satisfied in (E, D).

2. For all x_i, x_j, if neither x_i nor x_j is updated, $D_{i,j}$ and $D_{j,i}$ are not modified so condition Definition 12 (2) still holds. If either x_i is updated, as $D'_{i,j} = D_{0,j}$ and $D'_{j,i} = D_{j,0}$ condition Definition 12 (2) still holds as it holds for $D_{0,j}$ and $D_{j,0}$ and we apply the same reasoning if x_j is updated. If both x_i, x_j are updated, condition Definition 12 (2) trivially holds.

3. For all x_i, if it is updated then $D'_{0,i} = D'_{i,0} = (0, \leq)$, hence $d_{0,i} = -d_{i,0} = 0$ and $\lhd_{0i} = \lhd_{i0} = \leq$; condition Definition 12 (3) holds. For all x_i, x_j, if neither x_i nor x_j is updated, $D'_{i,j} = D_{i,j}$ and $D'_{j,i} = D_{j,i}$ so condition Definition 12 (3) holds as it holds for $D_{i,j}$ and $D_{j,i}$. If either x_i is updated, as $D'_{i,j} = D_{0,j}$ and $D'_{j,i} = D_{j,0}$, condition Definition 12 (3) holds as it holds for $D_{0,j}$ and $D_{j,0}$. We treat the case where x_j is updated similarly. If both x_i, x_j are updated, condition Definition 12 (3) trivially holds.

4. Canonical form is preserved:

   (a) if x_i, x_j, x_k are not updated: since no clock is updated we have $D'_{i,j} = D_{i,j}$, $D'_{j,k} = D_{j,k}$ and $D'_{i,k} = D_{i,k}$ since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ from Definition 13 (2), we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore it remains valid.
   (b) if x_k is updated and x_i, x_j are not updated: $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,0}$, $D'_{i,k} = D_{i,0}$ because x_k is updated. Since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ from Definition 13 (2), we know that $D_{i,0} \leq D_{i,j} + D_{j,0}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.

(c) if x_j is updated and x_i, x_k are not updated: then $D'_{i,k} = D_{i,k}$ because neither x_i nor x_k are updated; since x_k is updated we have $D'_{j,k} = D_{0,k}$ and $D'_{i,j} = D_{i,0}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ from Definition 13 (2), we know that $D_{i,k} \leq D_{i,0} + D_{0,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.

(d) if x_j, x_k are updated and x_i is not updated: then $D'_{i,k} = D_{i,0}$ because x_k is updated; since x_j is updated we have $D'_{i,j} = D_{i,0}$ and $D'_{j,k} = D_{0,0}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ from Definition 13 (2) and Lemma 4, we know that $D_{i,0} \leq D_{i,0} + D_{0,0}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.

(e) if x_i is updated and x_j, x_k are not updated: then $D'_{i,k} = D_{0,k}$, $D'_{i,j} = D_{0,j}$ because x_i is updated; since x_j, x_k are not updated, we have $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ from Definition 13 (2), we know that $D_{0,k} \leq D_{0,j} + D_{j,k}$ is valid for R_p; therefore $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.

(f) if x_i, x_k are updated and x_j is not updated: we have $D'_{i,k} = (0, \leq) = D_{0,0}$, $D'_{i,j} = D_{0,j}$ and $D'_{j,k} = D_{j,0}$ because x_i, x_k are updated. Since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ from Definition 13 (2), we know that $D_{0,0} \leq D_{0,j} + D_{j,0}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.

(g) if x_i, x_j are updated and x_k is not updated: we have $D'_{i,k} = D_{0,k}$, $D'_{i,j} = (0 <\leq) = D_{0,0}$ and $D'_{j,k} = D_{0,k}$ because x_i, x_j are updated. Since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ from Definition 13 (2) and Lemma 4, we know that $D_{0,k} \leq D_{0,0} + D_{0,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.

(h) if x_i, x_j, x_k are updated: we have $D'_{i,k} = D_{0,0}$, $D'_{i,j} = D_{0,0}$ and $D'_{j,k} = D_{0,0}$ because x_i, x_j, x_k are updated. Since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ from Definition 13 (2) and Lemma 4, we know that $D_{0,0} \leq D_{0,0} + D_{0,0}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.

5. there is at least one clock x s.t. $D'_{x,0} = D'_{0,x} = (0, \leq)$.

Therefore, $(E', D') \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$. □

The following lemma shows that applying a *update* on any open–p–PDBM transforms it into an open–p–PDBM respecting Definition 12 (2).

Lemma 7 (stability of $p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ under *update*)**.** *Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$. Let u_{np} be a non-parametric update. Then $update((E, D), u_{np}) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$.*

Proof. Most cases are similar to the proof of Lemma 6.

The remaining cases to treat are the cases of Definition 12 (2). If i, j are different from 0, and

1. if i, j are not updated then $D'_{i,j} = D_{i,j}$ and since it is the case in (E, D), condition Definition 12 (2) holds.

2. if j is updated and i is not updated then $D'_{i,j} = D_{i,0}$ and $D'_{j,i} = D_{0,i}$ and as condition Definition 13 (1) holds for $D_{i,0}$ and $D_{0,i}$ in (E, D), condition Definition 12 (2) holds in (E', D').

3. if i is updated and j is not updated then $D'_{i,j} = D_{0,j}$ and $D'_{j,i} = D_{j,0}$ and as condition Definition 13 (1) holds for $D_{j,0}$ and $D_{0,j}$ in (E, D), condition Definition 12 (2) holds in (E', D').

4. if i, j are updated then trivially $D'_{i,j} = D'_{j,i} = (0, \leq)$ and condition Definition 12 (2) holds.

□

□

Applying a non-parametric *update* on any point–p–PDBM transforms it into an open–p–PDBM, and open–p–PDBMs are stable under *update*. It can seem a paradox that the (non-parametric) update of a point–p–PDBM becomes an open–p–PDBM; in fact, it remains geometrically speaking a point, *i. e.,* a singleton containing one clock valuation. Recall that our open–p–PDBMs include p–PDBMs geometrically corresponding to a point for each valuation. In contrast, point–p–PDBMs are also punctual (for each valuation), but are fully parametric.

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

Lemma 8 (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 w, $[w]_{u_{np}} \in update((E, v(D)), u_{np})$ iff $w \in (E, v(D))$.*

Proof. We first treat the case of the p–$\mathcal{PDBM}_{\blacksquare}(R_p)$ (the case of the p–$\mathcal{PDBM}_{\odot}(R_p)$ will be handled similarly at the end). We also prove this lemma for a singleton update (only one clock, say x_i) since updating several clocks can be done by applying several singleton updates in a 0 delay.

4.4.1.1 $(E, D) \in p$–$\mathcal{PDBM}_{\blacksquare}(R_p)$, $(\Rightarrow)$

Let R_p be a parameter region and $(E, D) \in p$–$\mathcal{PDBM}_{\blacksquare}(R_p)$. Let $v \in R_p$. Let u_{np} be a non-parametric update which updates x_i to an integer n and lets the value of other clocks unchanged. Consider $(E', D') = update((E, v(D)), u_{np})$ and suppose $w' \in (E', D')$. We want to construct a valuation $w \in (E, v(D))$ s.t. $w' = u_{np}(w)$.

Let w be a clock valuation s.t. for all clock x_j where $i \neq j$, $w(x_j) = w'(x_j)$. That means that for all $j \neq i$,

$$frac(w(x_j)) \lhd_{j0} v(d_{j,0}), \quad -frac(w(x_j)) \lhd_{0j} v(d_{0,j}) \quad \text{and} \quad \lfloor w(x_j) \rfloor = E_j$$

hold from Definition 14 since it is the case in (E', D') and these values are left untouched by the update. Moreover for all $j \neq i$, $k \neq i$,

$$frac(w(x_j)) - frac(w(x_k)) \lhd_{jk} v(d_{j,k}) \quad \text{and} \quad frac(w(x_k)) - frac(w(x_j)) \lhd_{kj} v(d_{k,j})$$

again hold from Definition 14 since it is the case in (E', D') and these values are left untouched by the update.

We want a valuation for $w(x_i)$ s.t.

$$frac(w(x_i)) \lhd_{i0} v(d_{i,0}) \quad - frac(w(x_i)) \lhd_{0i} v(d_{0,i}) \quad \text{and} \quad \lfloor w(x_i) \rfloor = E_i$$

hold, and for all $j \neq i$, $k \neq i$,

$$frac(w(x_i)) - frac(w(x_j)) \lhd_{ij} v(d_{i,j}) \quad \text{and} \quad frac(w(x_k)) - frac(w(x_i)) \lhd_{ki} v(d_{k,i}) \tag{4.1}$$

hold. Let us prove that such a valuation w exists. We set $\lfloor w(x_i) \rfloor = E_i$.

The following lemma proves transitivity of constraints on clocks with respect to constraints in a p–PDBM.

Lemma 9. *Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$. Let $v \in R_p$. Let $w \in (E, v(D))$. For all clocks i, j, k, $frac(w(x_j)) - frac(w(x_k))(\lhd_{ji} \oplus \lhd_{ik}) v(d_{j,i}) + v(d_{i,k})$.*

Proof. Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$. Let $v \in R_p$. Let $w \in (E, v(D))$.

Since $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$, for all i, j, k we have from Definition 12 (4),

$$D_{j,k} \leq D_{j,i} + D_{i,k}$$

is valid for R_p hence since $v \in R_p$, we have $v(D_{j,k}) \leq v(D_{j,i}) + v(D_{i,k})$. Precisely that is $(v(d_{j,k}), \lhd_{jk}) \leq (v(d_{j,i}), \lhd_{ji}) + (v(d_{i,k}), \lhd_{ik})$ *i. e.,*

$$(v(d_{j,k}), \lhd_{jk}) \leq (v(d_{j,i}) + v(d_{i,k}), \lhd_{ji} \oplus \lhd_{ik}).$$

For all clocks j, k satisfying constraints of (E, D),

$$frac(w(x_j)) - frac(w(x_k)) \lhd_{jk} v(d_{j,k}).$$

Then for all i, j, k, either:

- from Definition 11 (2a): $v(d_{j,k}) < v(d_{j,i}) + v(d_{i,k})$ and then, regardless of $\lhd_{jk}$ and $\lhd_{ji} \oplus \lhd_{ik}$ we have $frac(w(x_j)) - frac(w(x_k))(\lhd_{ji} \oplus \lhd_{ik}) v(d_{j,i}) + v(d_{i,k})$, or
- from Definition 11 (2b):
  - $v(d_{j,k}) \leq v(d_{j,i}) + v(d_{i,k})$ and $\lhd_{jk} = <$, $\lhd_{ji} \oplus \lhd_{ik} = \leq$ and then we have $frac(w(x_j)) - frac(w(x_k))(\lhd_{ji} \oplus \lhd_{ik}) v(d_{j,i}) + v(d_{i,k})$, or
  - $v(d_{j,k}) \leq v(d_{j,i}) + v(d_{i,k})$ and $\lhd_{jk} = \lhd_{ji} \oplus \lhd_{ik}$ and then we have $frac(w(x_j)) - frac(w(x_k))(\lhd_{ji} \oplus \lhd_{ik}) v(d_{j,i}) + v(d_{i,k})$ which completes the proof.

This completes the proof of Lemma 9. □

For all $j \neq i$ and $k \neq i$, since $v(D_{j,k}) \leq v(D_{j,i}) + v(D_{i,k})$ from Definition 12 (4), we have $frac(w(x_j)) - frac(w(x_k)) \lhd_{jk} v(d_{j,k})$ and

$$frac(w(x_j)) - frac(w(x_k))(\lhd_{ji} \oplus \lhd_{ik}) v(d_{j,i}) + v(d_{i,k})$$

holds from Lemma 9. Hence

$$frac(w(x_j)) - v(d_{j,i})(\lhd_{ji} \oplus \lhd_{ik}) frac(w(x_k)) + v(d_{i,k}) \tag{4.2}$$

holds. Note that $\lhd_{ji} \oplus \lhd_{ik}$ is either $\leq$ or $<$. Note the following trick is inspired by [HRSV02, Proof of Lemma 3.5] and [HRSV02, Proof of Lemma 3.13]. Hence

$$I = \{t \in \mathbb{R}_+ \mid frac(w(x_j)) - v(d_{j,i}) \leq t \leq frac(w(x_k)) + v(d_{i,k}) \text{ for all clocks } j, k\}$$

is a non empty set. That means that choosing a $frac(w(x_i))$ with respect to constraints (4.1), recall that they are

$$frac(w(x_j)) - frac(w(x_i)) \lhd_{ji} v(d_{j,i}) \quad \text{and} \quad frac(w(x_i)) - frac(w(x_k)) \lhd_{ik} v(d_{i,k})$$

is equivalent to choose a $frac(w(x_i))$ s.t.

$$frac(w(x_j)) - v(d_{j,i}) \lhd_{ji} frac(w(x_i)) \quad \text{and} \quad frac(w(x_i)) \lhd_{ik} frac(w(x_k)) + v(d_{i,k})$$

which is a nonempty set from formula (4.2). Finally we choose a $frac(w(x_i)) \in I$, then $w \in (E, v(D))$ and it completes the proof.

4.4.1.2 $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$, $(\Leftarrow)$

Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$. Let $v \in R_p$. Let u_{np} be a non-parametric update which updates x_i to an integer n and lets the value of other clocks unchanged. Consider $(E', D') = update((E, v(D)), u_{np})$. Now suppose $w \in (E, v(D))$ and let $w' = [w]_{u_{np}}$.

- for x_i, since u_{np} is defined, $w'(x_i) = u_{np}(x_i) = E'_{x_i}$ (*i. e.*, $frac(w'(x_i)) = 0$) by applying *update* as defined in Definition 14. By applying *update* as defined in Definition 14, $D'_{i,0} = D'_{0,i} = (0, \leq)$, hence

  $$-frac(w'(x_i)) \lhd_{0i} v(d'_{0,i}) \quad \text{and} \quad frac(w'(x_i)) \lhd_{i0} v(d'_{i,0})$$

  hold from Definition 14 and Lemma 6. Moreover we know that for all $j \neq i$

  $$-v(D'_{i,j}) = -v(D'_{0,j}) \quad \text{and} \quad v(D'_{j,i}) = v(D'_{j,0}) \tag{4.3}$$

  holds from Definition 14, and we also know that

  $$frac(w'(x_j)) - frac(w'(x_i)) = frac(w'(x_j)) \tag{4.4}$$

  since $frac(w'(x_i)) = 0$. Hence, combining (4.3) and (4.4), clearly since

  $$-frac(w'(x_j)) \lhd_{0j} v(d'_{0,j}) \quad \text{and} \quad frac(w'(x_j)) \lhd_{j0} v(d'_{j,0})$$

  hold in (E', D'),

  $$frac(w'(x_j)) - frac(w'(x_i)) \lhd_{ji} v(d'_{j,i}) \quad \text{and} \quad frac(w'(x_i)) - frac(w'(x_j)) \lhd_{ij} v(d'_{i,j})$$

  hold.

- for any two clocks x_j, x_k where u_{np} is not defined, $w(x_j) = w'(x_j)$ and $w(x_k) = w'(x_k)$. Hence

  $$-v(D'_{0,j}) \lhd_{0j} frac(w'(x_j)) \lhd_{j0} v(D'_{j,0})$$

  and

  $$-v(D'_{k,j}) \lhd_{kj} frac(w'(x_j)) - frac(w'(x_k)) \lhd_{jk} v(D'_{j,k})$$

  hold from Definition 14 and Lemma 6 since bounds remain unchanged.

Then $w' \in update((E, v(D)), u_{np})$.

This concludes the case $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$.

Let us now treat the case $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$.

4.4.1.3 $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$, $(\Rightarrow)$

Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$. Let $v \in R_p$. Let u_{np} be a non-parametric update which updates x_i to an integer n and lets the value of other clocks unchanged. Consider $(E', D') = \mathit{update}((E, v(D)), u_{np})$ and suppose $w' \in (E', D')$. We want to construct a valuation

$$w \in (E, v(D)) \quad \text{s.t.} \quad w' = u_{np}(w)$$

Let w be a clock valuation s.t. for all clock x_j where $j \neq i$, $w(x_j) = w'(x_j)$. That means for all $j \neq i$,

$$\mathit{frac}(w(x_j)) \lhd_{j0} v(d_{j,0}), \quad -\mathit{frac}(w(x_j)) \lhd_{0j} v(d_{0,j}) \quad \text{and} \quad \lfloor w(x_j) \rfloor = E_j$$

hold from Definition 14 since it is the case in (E', D') and bounds remain unchanged *i. e.,* $D_{0,j} = D'_{0,j}$ and $D_{j,0} = D'_{j,0}$. Moreover for all $k \neq i$ and $k \neq j$,

$$\mathit{frac}(w(x_j)) - \mathit{frac}(w(x_k)) \lhd_{jk} v(d_{j,k}) \quad \text{and} \quad \mathit{frac}(w(x_k)) - \mathit{frac}(w(x_j)) \lhd_{kj} v(d_{k,j})$$

also hold from Definition 14 since it is the case in (E', D') and bounds remain unchanged *i. e.,* $D_{k,j} = D'_{k,j}$ and $D_{j,k} = D'_{j,k}$.

Recall that (E, D) contains only one clock valuation for each parameter valuation $v \in R_p$.

Let $\mathit{frac}(w(x_i)) = v(d_{i,0})$ (or equivalently $\mathit{frac}(w(x_i)) = -v(d_{0,i})$ since by Definition 13 we have $(d_{i,0}, \lhd_{i0}) = (-d_{0,i}, \lhd_{0i})$). Then, as it is the case in (E, D),

$$\mathit{frac}(w(x_i)) \lhd_{i0} v(d_{i,0}), \quad -\mathit{frac}(w(x_i)) \lhd_{0i} v(d_{0,i}) \quad \text{and} \quad \lfloor w(x_i) \rfloor = E_i$$

hold, and for all $j \neq i$, $k \neq i$,

$$\mathit{frac}(w(x_i)) - \mathit{frac}(w(x_j)) \lhd_{ij} v(d_{i,j}) \quad \text{and} \quad \mathit{frac}(w(x_k)) - \mathit{frac}(w(x_i)) \lhd_{ki} v(d_{k,i})$$

hold, which completes the proof, as $w \in (E, v(D))$ and $w' = u_{np}(w)$.

4.4.1.4 $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$, $(\Leftarrow)$

This case is straightforward and similar to the case $(\Leftarrow)$ above of open–p–PDBMs. □

4.4.2 Parametric update

Given $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$ we write $\overline{\mathit{update}}((E, D), u)$ 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. Recall that a total update function which is not fully parametric (*i. e.,* an update of some clocks to parameters and some others to constants) can be encoded as a total parametric update immediately followed by a partial non-parametric update function.

4.4.3 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.

Definition 15 (clocks with the largest fractional part in a p–PDBM)**.** *Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$. A* clock with the (possibly parametric) largest fractional part *is a clock x s.t. for all $0 \leq i \leq H$, $(0, \leq) \leq D_{x,i}$ is valid for R_p.*

There is at least one clock with the (possibly parametric) largest fractional part:

Lemma 10 (existence of a clock with the largest fractional part)**.** *Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$. There is at least one clock x s.t. for all $0 \leq i \leq H$, $(0, \leq) \leq D_{x,i}$ is valid for R_p.*

Proof. Reductio ad absurdum: Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ with at least 2 clocks i, j. Suppose for all clock x_i there is another clock x_j s.t. $D_{i,j} < 0$ is valid for R_p. Let $v \in R_p$. Then $v(D_{i,j}) < 0$.

- Suppose for x_j, x_i is the clock s.t. $D_{j,i} < 0$ is valid for R_p. Then $v(D_{j,i}) < 0$. We have $v(D_{i,j}) + v(D_{j,i}) < 0$ holds, therefore $0 \leq v(D_{i,j}) + v(D_{j,i})$ does not hold, and hence $0 \leq D_{i,j} + D_{j,i}$ is not valid for R_p. Then (E, D) does not respect Lemma 3 and violates condition (4) of Definition 12. So $(E, D) \notin p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$.
- Suppose for x_j, a third clock x_k is the clock s.t. $D_{j,k} < 0$ is valid for R_p. Then $v(D_{j,k}) < 0$. Suppose we have only three clocks. Then for x_k, either x_i or x_j is the clock s.t. $D_{k,i} < 0$ is valid for R_p.
  - Assume this is x_i. Then $v(D_{k,i}) < 0$. We have $v(D_{k,i}) + v(D_{i,j}) < 0$ and $v(D_{k,j}) \leq v(D_{k,i}) + v(D_{i,j})$ by Definition 12 (4). Follows that $v(D_{k,j}) + v(D_{j,k}) < 0$ and $0 \leq D_{k,j} + D_{j,k}$ is not valid for R_p. Then (E, D) does not respect Lemma 3 and violates condition (4) of Definition 12. So $(E, D) \notin p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$.
  - Assume this is x_j. This case is similar (and simpler).

We apply the same reasoning for more than 3 clocks. Now suppose $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$. We apply the same reasoning, replacing the argument of condition (4) of Definition 12 by the fact from Definition 13 that D is antisymmetric. □

Note that several clocks may have the largest fractional parts (up to some syntactic replacements [2], in that case they satisfy the same constraints in (E, D)).

[2] Let $v \in R_p$ and suppose, we have two different syntactic expressions, such as $p, 1 - p$ that are equal once valuated *i. e.,* $v(p) = 1 - v(p)$. From Definition 10 remark that if it is for v, it is for any $v' \in R_p$. We choose one *e. g.,* $1 - v(p)$ and replace the second, $v(p)$, everywhere it appears.

For a p–PDBM (E, D), we define the set of clocks with the largest fractional part (LFP) as $\mathsf{LFP}_{R_p}(D) = \{x \in \mathbb{X} \mid 0 \leq D_{x,i} \text{ is valid for } R_p, \text{ for all } 0 \leq i \leq H\}$.

As we are able, thanks to the parameter regions, to order our parameter valuations (*i. e.,* whether one is greater or less than another one), we can define LFP from the constraints defined in the point–p–PDBM. We will define and apply successively two time-elapsing algorithms: the first one starts from a point–p–PDBM or an open–p–PDBM respecting condition Definition 12 (5a). We will prove that we obtain an open–p–PDBM respecting condition Definition 12 (5b). The second one, starts from an open–p–PDBM respecting condition Definition 12 (5b) and will define the set of constraints defining the possible clocks valuations exactly when any clock of LFP has reached its upper bound 1. We will prove that we obtain an open–p–PDBM respecting condition Definition 12 (5a). As we will obtain at each iteration of the algorithm an open–p–PDBM respecting either condition Definition 12 (5a) or (5b), this will prove we have a stable set of open–p–PDBMs. Now we explain our algorithms more precisely.

Clocks belonging to LFP are the first to reach the upper bound 1 by letting time elapse. Since LFP can contain multiple clocks and they have the same fractional part, we can consider any $x \in \mathsf{LFP}$.

Let $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$ 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: open–p–PDBM (Definition 12) satisfying condition (5a) and point–p–PDBM (Definition 13) or open–p–PDBM (Definition 12 satisfying condition (5b)).

Given an open–p–PDBM satisfying condition 5a or a point–p–PDBM (E, D) with $E_x = k$, $TE((E, D))$ described in Algorithm 9 and named $TE_<$, defines a new open–p–PDBM satisfying condition 5b by

1. setting $D_{x,0} := (1, <)$ as x is the first one that will reach $k + 1$;
2. updating the upper bound of all other clocks i, which has increased: $D_{i,0} := D_{i,x} + (1, <)$;
3. 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 [AD94] into an open region of [AD94].

Algorithm 2: $TE_<((E, D))$: set upper bound of all $frac(x_i) \in \mathsf{LFP}_{R_p}(D)$ to 1

```
pick x_i ∈ LFP_{R_p}(D)
for j from 1 to H do
    if j ∈ LFP_{R_p}(D) then
        D_{j,0} := (1, <)
    else
        D_{j,0} := D_{j,i} + (1, <)
    end
    D_{0,j} := D_{0,j} + (0, <)
end
```

$TE_<$ is applied to point–p–PDBMs and open–p–PDBMs respecting condition 5a; it sets $D_{x,0} := (1, <)$ and $D_{0,x} := D_{0,x} + (0, <)$ for all $x \in \mathsf{LFP}_{R_p}(D)$. Then, for all clocks $1 \leq j \leq H$ not in LFP sets $D_{j,0} := D_{j,i} + (1, <)$ and $D_{0,j} := D_{0,j} + (0, <)$. This gives the range of possible clock valuations before $frac(x_i)$ reaches 1. The obtained result is denoted by $TE_<((E, D))$, and it leaves E unchanged.

The time elapsing operator also operates the transformation from an open region of [AD94] to the upper open line segment or the corner-point region of [AD94], given in the algorithm Algorithm 15 as $TE_=$. Given an open–p–PDBM satisfying condition 5b (E, D) where $E_x = k$, $TE((E, D))$ defines a new open–p–PDBM satisfying condition 5a by

1. 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*;
2. updating the upper and lower bounds of all other clocks i: $D_{i,0} := D_{i,x} + (1, \leq)$ and $D_{0,i} := D_{x,i} + (-1, \leq)$;
3. 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.

Algorithm 3: $TE_=((E, D))$: set upper and lower bound of all $frac(x_i) \in \mathsf{LFP}_{R_p}(D)$ to 1

```
pick x_i ∈ LFP_{R_p}(D)
for j from 1 to H do
    if j ∈ LFP_{R_p}(D) then
        D_{j,0} := (0, ≤)
        D_{0,j} := (0, ≤)
        E_j := E_j + 1
    else
        D_{j,0} := D_{j,i} + (1, ≤)
        D_{0,j} := D_{i,j} + (−1, ≤)
    end
end
for j from 1 to H do
    D_{j,i} := D_{j,0}
    D_{i,j} := D_{0,j}
end
```

$TE_=$ is applied to open–p–PDBMs respecting condition 5b and sets $D_{x,0} := (0, \leq)$ and $D_{0,x} := (0, \leq)$ for all $x \in \mathsf{LFP}_{R_p}(D)$. Then, for all clocks $x_j \in$

$H \setminus \mathsf{LFP}_{R_p}(D)$ sets $D_{0,j} := (-1, \leq) + D_{i,j}$ and $D_{j,0} := D_{j,i} + (1, \leq)$; it gives the range of clock valuations when $frac(x)$ reaches 1, and increments E_x, for $x \in \mathsf{LFP}_{R_p}(D)$ if E_x is not greater that $K+1$. It then sets, regardless of whether $x_j \in \mathsf{LFP}_{R_p}(D)$ $D_{i,j} := D_{0,j}$ and $D_{j,i} := D_{j,0}$. Finally, for $x \in \mathsf{LFP}_{R_p}(D)$ it sets $E_x := E_x + 1$. The obtained result is denoted by $TE_{=}((E, D))$.

Definition 16 (time elapsing in a p–PDBM)**.** *Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p) \cup p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$. We define $(E', D') = TE((E, D))$ as applying either $TE_{<}$ if (E, D) respects condition 5a or $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$, or $TE_{=}$ if (E, D) respects condition 5b.*

Lemma 11 (stability under time elapsing)**.** *Let R_p be a parameter region. Let $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$. Then $TE((E, D)) \in p\text{–}\mathcal{PDBM}(R_p)$.*

Proof. We prove our lemma for the two types of open–p–PDBMs and for point–p–PDBMs, and split this proof in three lemmas.

4.4.3.1 Definition 12 type (5a) to (5b)

Lemma 12 (modification of an open–p–PDBM respecting condition 5a under $TE_{<}$)**.** *Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ respecting condition 5a, then $TE_{<}((E, D)) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ respecting condition 5b.*

Proof of Lemma 12. Suppose $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ respects condition (5a) of Definition 12, *i. e.,* we have at least an x s.t. $D_{x,0} = D_{0,x} = (0, \leq)$. Since, in R_p, we know which parameters have the largest fractional part, we can determine $\mathsf{LFP}_{R_p}(D)$ from Lemma 10. If more than one clock belong to $\mathsf{LFP}_{R_p}(D)$ then their valuations have the same fractional part. Indeed, from Definition 15 if $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$ then both $(0, \leq) \leq D_{i,j}$ and $(0, \leq) \leq D_{j,i}$ are valid for R_p, and from Definition 12 (2) we must have $D_{i,j} = D_{j,i} = (0, \leq)(\star)$.

Let $v \in R_p$. Assume $x_i \in \mathsf{LFP}_{R_p}(D)$ and $w \in (E, v(D))$, by letting time elapse, $frac(w(x_i))$ is the first that might reach 1. Moreover, for all $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $frac(w(x_j))$ cannot reach 1 before $frac(w(x_i))$. We are going to construct a new $(E', D') = TE_{<}((E, D))$, which will be an open–p–PDBM respecting condition 5b of Definition 12. While detailing the procedure of $TE_{<}$, we are going to prove that Definition 12 (1) and (2) hold for (E', D'). Further we will prove that (4) and (5b) also hold.

proof that Definition 12 (1) holds According to the definition of $TE_{<}$ (Algorithm 9), the first step is to set a new upper bound

$$D'_{i,0} = (1, <) \quad \text{for all } x_i \in \mathsf{LFP}_{R_p}(D)$$

and obviously $(0, \leq) \leq D'_{i,0} \leq (1, \leq)$ is valid for R_p. Then we set new upper bounds for all other clock $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$ by setting

$$D'_{j,0} = D_{j,i} + (1, <).$$

Indeed, $D_{j,i}$ is the constraint on the lower bound of $frac(w(x_j)) - frac(w(x_i))$ and since the upper bound of x_i has increased, this gives the new upper bound of x_j. Note that since $x_i \in \mathsf{LFP}_{R_p}(D)$, from Definition 15 and Definition 12 (2) we have that $-1 \leq D_{j,i} \leq 0$ is valid for R_p for all clock x_j. Precisely, $d_{j,i} \in$

$\{0, -p_1, p_2 - p_1, p_1 - 1 - p_2, p_1 - 1\}$ for some $p_1, p_2 \in \mathbb{P}$ where $p_2 \leq p_1$ is valid for R_p. Hence as $d_{j,i} + 1 \in \{1, 1 - p_1, p_2 + 1 - p_1, p_1 - p_2, p_1\}$, we have that $d'_{j,0} \in \mathcal{PLT}$, $\lhd_{ji'} = \lhd_{ji} \oplus < = <$ so $(0, \leq) \leq D'_{j,0} \leq (1, <)$ is valid for R_p.

Note that we cannot have $(d_{j,i}, \lhd_{ji}) = (-1, <)$ because even if $(d_{i,j}, \lhd_{ij}) = (1, <)$, since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$ we do not have have $0 \leq D_{j,i} + D_{i,j}$ is valid for R_p from Definition 12 (4) and Lemma 3.

Secondary we set for all clock x regardless of whether they are in $\mathsf{LFP}_{R_p}(D)$

$$D'_{0,x} = D_{0,x} + (0, <).$$

Since some time elapsed, lower bounds of all clocks are increased. Moreover, as $(-1, <) \leq D_{0,x} \leq (0, \leq)$ is valid for R_p from Definition 12 (1), $(-1, \leq) \leq D'_{0,x} \leq (0, \leq)$ is also valid for R_p.

Therefore, Definition 12 (1) holds.

proof that Definition 12 (2) holds Third we set for all clocks x, y regardless of whether they are in $\mathsf{LFP}_{R_p}(D)$

$$D'_{x,y} = D_{x,y}$$

so as Definition 12 (2) holds in (E, D), it still does. More intuitively since no fractional part has reached 1, constraints on differences of clocks and integer parts remain unchanged.

proof that Definition 12 (3) holds For all x_i:

- if $x_i \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,0} = (1, <)$, $D'_{0,i} = D_{0,i} + (0, <)$ hence $d'_{i,0} \neq d'_{0,i}$ and $\lhd_{i0'} \lhd_{0i'} = <$, condition Definition 12 (3) holds;
- if $x_i \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $x \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,0} = D_{i,x} + (1, <)$, $D'_{0,i} = D_{0,i} + (0, <)$ hence as $(0, \leq) \leq D'_{i,0}$ is valid for R_p and $D'_{0,i} \leq (0, \leq)$ is valid for R_p, we have $d'_{i,0} \neq d'_{0,i}$ and $\lhd_{i0'} \lhd_{0i'} = <$ and condition Definition 12 (3) holds.

For all x_i, x_j:

- if $x_i, x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $D'_{i,j} = D_{i,j}$ and $D'_{j,i} = D_{j,i}$, condition Definition 12 (3) holds as it holds for $D_{i,j}$ and $D_{j,i}$.
- if $x_i \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $x_j \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,0} = D_{i,j} + (1, <)$, $D'_{0,i} = D_{0,i} + (0, <)$ hence as $(0, \leq) \leq D'_{i,0}$ is valid for R_p and $D'_{0,i} \leq (0, \leq)$ is valid for R_p, we have $d'_{i,0} \neq d'_{0,i}$ and $\lhd_{i0'} \lhd_{0i'} = <$, condition Definition 12 (3) holds. The case $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $x_i \in \mathsf{LFP}_{R_p}(D)$ is treated similarly.
- if $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,j} = D'_{j,i} = (0, \leq)$, hence $d'_{i,j} = -d'_{j,i} = 0$ and $\lhd_{ij'} \lhd_{ji'} = \leq$ and condition Definition 12 (3) holds.

proof that Definition 12 (4) holds Now we prove that Definition 12 (4) holds, *i. e.,* for all clocks x_i, x_j, x_k, valid conditions such as $D'_{i,j} \leq D'_{i,k} + D'_{k,j}$ remain valid in R_p. Indeed, when time elapses, all clocks have the same behavior, hence the difference between two clocks does not change without an update. Precisely, for all clocks x_i, x_j, x_k, are valid for R_p:

1. if $x_i, x_j, x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: let $x \in \mathsf{LFP}_{R_p}(D)$ and

   - if i, j, k are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ from Definition 12 (4), we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if i, j are different from 0, $k = 0$, we have $D'_{i,0} = D_{i,x} + (1, <)$, $D'_{i,j} = D_{i,j}$ and $D'_{j,0} = D_{j,x} + (1, <)$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ from Definition 12 (4), we know that $D_{i,x} \leq D_{i,j} + D_{j,x}$ is valid for R_p; then $D_{i,x} + (1, <) \leq D_{i,j} + D_{j,x} + (1, <)$ is valid for R_p from Lemma 2 and therefore, $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p.
   - if i, k are different from 0, $j = 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,0} = D_{i,x} + (1, <)$ and $D'_{0,k} = D_{0,k} + (0, <)$; we claim that

     $$D_{i,k} \leq D_{i,x} + (1, <) + D_{0,k} + (0, <) \tag{4.5}$$

     is valid for R_p, which is equivalent to $D'_{i,k} \leq D'_{i,0} + D'_{0,k}$ is valid for R_p. Since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ from Definition 12 (1), we know that

     $$D_{x,0} \leq (1, <); \tag{4.6}$$

     moreover we have

     $$(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <) \tag{4.7}$$

     Since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ from Definition 12 (4), we know that $D_{x,k} \leq D_{x,0} + D_{0,k}$ is valid for R_p; combining with (4.6) and (4.7) we obtain

     $$D_{x,k} \leq (1, <) + D_{0,k} + (0, <). \tag{4.8}$$

     Now, since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ from Definition 12 (4), we know that $D_{i,k} \leq D_{i,x} + D_{x,k}$ is valid for R_p and combining with (4.8) we obtain (4.5) and therefore our result.
   - if i is different from 0, $j = k = 0$, we have $D'_{i,0} = D_{i,x} + (1, <)$; from Definition 11 (2b) we have that

     $$D_{i,x} + (1, <) \leq D_{i,x} + (1, <)$$

     is valid for R_p. Hence from Lemma 4

     $$D'_{i,0} \leq D'_{i,0} + D'_{0,0}$$

     is valid for R_p.
   - if j, k are different from 0, $i = 0$, we have $D'_{0,k} = D_{0,k} + (0, <)$, $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{0,k} \leq D_{0,j} + D_{j,k}$ is valid for R_p. Moreover we have that

     $$D_{0,k} + (0, <) = (d_{0,k}, <) \quad \text{and} \quad D_{0,j} + (0, <) + D_{j,k} = (d_{0,j} + d_{j,k}, <)$$

     so we have from Definition 11 (2b)

     $$D_{0,k} + (0, <) \leq D_{0,j} + (0, <) + D_{j,k}$$

     is valid for R_p. Hence $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.

- if j is different from 0, $i = k = 0$, we have $D'_{0,0} = (0, \leq)$, $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,0} = D_{j,x} + (1, <)$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{0,x} \leq D_{0,j} + D_{j,x}$ is valid for R_p; moreover, from Definition 11 (2b) and Lemma 2,

$$D_{0,x} + (0, <) \leq D_{0,j} + (0, <) + D_{j,x}$$

is valid for R_p. Recall that from Lemma 3 $(0, \leq) \leq D_{0,x} + D_{x,0}$ is valid for R_p and since $D_{x,0} \leq (1, <)$ from Definition 12 (1), we have

$$(0, \leq) \leq D_{0,x} + (1, <)$$

is valid for R_p. As we have $(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <)$, we obtain that

$$D_{0,x} + (1, <) \leq D_{0,j} + D_{j,x} + (1, <)$$

is valid for R_p and therefore $D'_{0,0} \leq D'_{0,j} + D'_{j,0}$ is valid for R_p.

- if k is different from 0, $i = j = 0$, we have $D'_{0,k} = D_{0,k} + (0, <)$; From Definition 11 (2b) and Lemma 2 we have that

$$D_{0,k} + (0, <) \leq D_{0,k} + (0, <)$$

is valid for R_p. Hence from Lemma 4

$$D'_{0,k} \leq D'_{0,0} + D'_{0,k}$$

is valid for R_p.

- if $i = j = k = 0$, from Definition 12 (4) and Lemma 4 we trivially have

$$D'_{0,0} \leq D'_{0,0} + D'_{0,0}$$

is valid for R_p.

2. if $x_k \in \mathsf{LFP}_{R_p}(D)$ and $x_i, x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $k \neq 0$ and

   - if i, j are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$ from Definition 12 (4), we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$.
   - if $i \neq 0$, $j = 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,0} = D_{i,k} + (1, <)$ and $D'_{0,k} = D_{0,k} + (0, <)$; we claim that $D_{i,k} \leq D_{i,k} + (1, <) + D_{0,k} + (0, <)$ is valid for R_p, *i. e.*,

$$(0, \leq) \leq (1, <) + D_{0,k} + (0, <) \tag{4.9}$$

     is valid for R_p, which is equivalent to $D'_{i,k} \leq D'_{i,0} + D'_{0,k}$ is valid for R_p. We have

$$(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <). \tag{4.10}$$

     Since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $(0, \leq) \leq D_{0,k} + D_{k,0}$ is valid for R_p and from Definition 12 (1) that $D_{k,0} \leq (1, <)$ is valid for R_p; combining with (4.9) and (4.10) we obtain our result.

- if $i = 0$, $j \neq 0$, we have $D'_{0,k} = D_{0,k} + (0, <)$, $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{0,k} \leq D_{0,j} + D_{j,k}$. Moreover we have that

  $$D_{0,k} + (0, <) = (d_{0,k}, <) \quad \text{and} \quad D_{0,j} + (0, <) + D_{j,k} = (d_{0,j} + d_{j,k}, <)$$

  so we have from Definition 11 (2b)

  $$D_{0,k} + (0, <) \leq D_{0,j} + (0, <) + D_{j,k}$$

  is valid for R_p. Hence $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.
- if $i = j = 0$, from Definition 12 (4) and Lemma 4 we trivially have

  $$D'_{0,k} \leq D'_{0,0} + D'_{0,k}$$

  is valid for R_p.

3. if $x_j \in \mathsf{LFP}_{R_p}(D)$ and $x_i, x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $j \neq 0$ and

   - if i, k are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if $i \neq 0$, $k = 0$, we have $D'_{i,0} = D_{i,j} + (1, <)$, $D'_{i,j} = D_{i,j}$ and $D'_{j,0} = (1, <)$; From Definition 11 (2b) we trivially have that $D_{i,j} + (1, <) \leq D_{i,j} + (1, <)$ is valid for R_p and therefore, $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p.
   - if $i = 0$, $k \neq 0$, we have $D'_{0,k} = D_{0,k} + (0, <)$, $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{0,k} \leq D_{0,j} + D_{j,k}$ is valid for R_p. Moreover we have that

     $$D_{0,k} + (0, <) = (d_{0,k}, <) \quad \text{and} \quad D_{0,j} + (0, <) + D_{j,k} = (d_{0,j} + d_{j,k}, <)$$

     so we have from Definition 11 (2b) and Lemma 2

     $$D_{0,k} + (0, <) \leq D_{0,j} + (0, <) + D_{j,k}$$

     is valid for R_p. Hence $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.
   - if $i = k = 0$, we have $D'_{0,0} = (0, \leq)$, $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,0} = (1, <)$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Lemma 3 we know that $(0, \leq) \leq D_{0,j} + D_{j,0}$ is valid for R_p, and since from Definition 12 (1) $D_{j,0} \leq (1, \leq)$ is valid for R_p, that means $(0, \leq) \leq D_{0,j} + (1, <)$ is valid for R_p. As we have

     $$(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <)$$

     we obtain that

     $$(0, \leq) \leq D_{0,j} + (0, <) + (1, <)$$

     is valid for R_p and therefore $D'_{0,0} \leq D'_{0,j} + D'_{j,0}$ is valid for R_p.

4. if $x_j, x_k \in \mathsf{LFP}_{R_p}(D)$ and $x_i \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $j \neq 0, k \neq 0$ and

   - if i is different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$.
   - if $i = 0$, we have $D'_{0,k} = D_{0,k} + (0, <)$, $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{0,k} \leq D_{0,j} + D_{j,k}$. Moreover we have that

     $$D_{0,k} + (0, <) = (d_{0,k}, <) \quad \text{and} \quad D_{0,j} + (0, <) + D_{j,k} = (d_{0,j} + d_{j,k}, <)$$

     so we have from Definition 11 (2b) and Lemma 2

     $$D_{0,k} + (0, <) \leq D_{0,j} + (0, <) + D_{j,k}$$

     is valid for R_p. Hence $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.

5. if $x_i \in \mathsf{LFP}_{R_p}(D)$ and $x_j, x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $i \neq 0$ and

   - if j, k are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$.
   - if $j \neq 0$, $k = 0$, we have $D'_{i,0} = (1, <)$, $D'_{i,j} = D_{i,j}$ and $D'_{j,0} = D_{j,i} + (1, <)$; from Definition 12 (4) and Lemma 3 we know that $(0, \leq) \leq D_{i,j} + D_{j,i}$ is valid for R_p. Since, from Definition 11 (2b) $(1, <) \leq (1, <)$ is valid for R_p, then from Lemma 2

     $$(1, <) \leq D_{i,j} + D_{j,i} + (1, <)$$

     is valid for R_p and therefore, $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p.
   - if $j = 0$, $k \neq 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,0} = (1, <)$ and $D'_{0,k} = D_{0,k} + (0, <)$; we claim that

     $$D_{i,k} \leq (1, <) + D_{0,k} + (0, <)$$

     is valid for R_p, which is equivalent to $D'_{i,k} \leq D'_{i,0} + D'_{0,k}$ is valid for R_p. Since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$ from Definition 12 (4), we know that $D_{i,k} \leq D_{i,0} + D_{0,k}$ is valid for R_p; moreover, from Definition 12 (1), we know that $D_{i,0} \leq (1, <)$ is valid for R_p. We have

     $$(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <)$$

     so we obtain that

     $$D_{i,k} \leq D_{i,0} + D_{0,k} \leq (1, <) + D_{0,k} = (1, <) + D_{0,k} + (0, <)$$

     is valid for R_p and therefore our result.
   - if i is different from 0, $j = k = 0$, we have $D'_{i,0} = (1, <)$, $D'_{0,0} = (0, \leq)$; from Definition 11 (2b) we have that

     $$(1, <) \leq (1, <)$$

     is valid for R_p. Hence from Lemma 4

     $$D'_{i,0} \leq D'_{i,0} + D'_{0,0}$$

     is valid for R_p.

6. if $x_i, x_k \in \mathsf{LFP}_{R_p}(D)$ and $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $i \neq 0, k \neq 0$ and

   - if $j \neq 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if $j = 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,0} = (1, <)$ and $D'_{0,k} = D_{0,k} + (0, <)$; we claim that
     $$D_{i,k} \leq (1, <) + D_{0,k} + (0, <)$$
     is valid for R_p, which is equivalent to $D'_{i,k} \leq D'_{i,0} + D'_{0,k}$ is valid for R_p. Since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$ from Definition 12 (4), we know that $D_{i,k} \leq D_{i,0} + D_{0,k}$ is valid for R_p; moreover, from Definition 12 (1), we know that $D_{i,0} \leq (1, <)$ is valid for R_p. We have
     $$(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <)$$
     so we obtain that
     $$D_{i,k} \leq D_{i,0} + D_{0,k} \leq (1, <) + D_{0,k} = (1, <) + D_{0,k} + (0, <)$$
     is valid for R_p and therefore our result.

7. if $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$ and $x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $i \neq 0, j \neq 0$ and

   - if $k \neq 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if $k = 0$, we have $D'_{i,0} = (1, <)$, $D'_{i,j} = D_{i,j} = (0, \leq)$ since both $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$ (cf.($\star$)) and $D'_{j,0} = (1, <)$; then $(1, <) \leq (0, \leq) + (1, <)$ is valid for R_p and therefore, $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p.

8. if $x_i, x_j, x_k \in \mathsf{LFP}_{R_p}(D)$: i, j, k are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.

proof that Definition 12 (5b) holds Finally, for $x_i \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,0} = (1, <)$ and for all clock j s.t. $D'_{0,j} = (0, \lhd_{0j'})$, then we have $\lhd_{0j'} = <$. Condition Definition 12 (5b) is satisfied.

We denote by (E, D') the obtained p–PDBM and $(E, D') \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$. □

4.4.3.2 Definition 12 type 5b to (5a)

Lemma 13. *Let $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$; let $x_i \in \mathsf{LFP}_{R_p}(D)$, $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$. If $(d_{i,j}, \lhd_{ij}) = (0, \lhd)$, then $\lhd = <$*

Proof. Let $x_i \in \mathsf{LFP}_{R_p}(D)$, $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$. Suppose $(d_{i,j}, \lhd_{ij}) = (0, \leq)$. From Definition 12 (2) we should have that $(d_{j,i}, \lhd_{ji}) = (0, \leq)$ so Lemma 3 is satisfied, and then $x_j \in \mathsf{LFP}_{R_p}(D)$. □

Lemma 14 (modification of an open–p–PDBM respecting condition 5b under $TE_=$)**.** *Let R_p be a parameter region and $(E,D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ respecting condition 5b, then $TE_=(E,D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ respecting condition 5a.*

Proof. Suppose $(E,D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ respects condition (5a) of Definition 12 *i. e.,* we have at least an x s.t. $D_{x,0} = (1,<)$ and for all other j s.t. $D_{0,j} = (0, \lhd_{0j})$, $\lhd_{0j} = <$. First we can determine $\mathsf{LFP}_{R_p}(D)$. Let $x \in \mathsf{LFP}_{R_p}(D)$. If more than one clock belong to $\mathsf{LFP}_{R_p}(D)$ then their valuations have the same fractional part. Indeed, from Definition 15 if $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$ then both $(0,\leq) \leq D_{i,j}$ and $(0,\leq) \leq D_{j,i}$ are valid for R_p, and from Definition 12 (2) we must have $D_{i,j} = D_{j,i} = (0,\leq)$.

Let $v \in R_p$. Let $x_i \in \mathsf{LFP}_{R_p}(D)$ and $w \in (E, v(D))$. By letting time elapse, $frac(w(x))$ is the first to actually reach 1. Moreover, for all $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $frac(w(x_j))$ cannot reach 1 before $frac(w(x_i))$. We are going to construct a new $(E', D') = TE_=((E,D))$ which is an open–p–PDBM respecting condition 5b. While detailing the procedure of $TE_=$, we are going to prove that Definition 12 (1) and (2) hold for (E', D'). Further we will prove that (4) and (5a) also hold.

proof that Definition 12 (1) holds According to the definition of $TE_=$ (Algorithm 15), the first step is to fix the value of $frac(x_i)$ to 0 by setting

$$D'_{i,0} = (0,\leq) \quad \text{and} \quad D'_{0,i} = (0,\leq) \quad \text{for all } x_i \in \mathsf{LFP}_{R_p}(D).$$

Indeed, when $frac(x_i)$ reaches 1, in the constraints expressed by $(E, v(D))$ we have to increase the integer part by 1 and set the new constraints on the fractional part to 0.

Secondary we set new upper and lower bound for all other clock $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$

$$D'_{0,j} = D_{i,j} + (-1,\leq) \quad \text{and} \quad D'_{j,0} = D_{j,i} + (1,\leq).$$

We have to force now upper and lower bounds for other clocks since we know the interval of time that elapsed when x_i reached 1.

Note that since $x_i \in \mathsf{LFP}_{R_p}(D)$, $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$ from Definition 15 we have that $(0,\leq) \leq D_{i,j} \leq (1,<)$ is valid for R_p for all clock x_j. Nonetheless, since $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, we even have $D_{i,j} \neq (0,\leq)$: suppose $(d_{i,j}, \lhd_{ij}) = (0,\leq)$: from Definition 12 (2) we should have that $(d_{j,i}, \lhd_{ji}) = (0,\leq)$ so Lemma 3 is satisfied, and then $x_j \in \mathsf{LFP}_{R_p}(D)$. The same reasoning leads to $D_{j,i} \neq (0,\leq)$.

Obviously, we have $D_{i,j} \neq (0,<)$: suppose $D_{i,j} = (0,<)$, since $x_i \in \mathsf{LFP}_{R_p}(D)$ then from Definition 15 $(0,\leq) \leq D_{i,j}$ should be valid for R_p, which is not from Definition 11 (2b).

Precisely, $d_{i,j} \in \{1, 1-p_1, p_2+1-p_1, p_1-p_2, p_1\}$ for any two $p_1, p_2 \in \mathbb{P}$ where $p_2 \leq p_1$ is valid for R_p. Hence as $-1 + d_{i,j} \in \{0, -p_1, p_2-p_1, p_1-1-p_2, p_1-1\}$, we have that $D'_{0,j} \in \mathcal{PLT}$ and $(-1,<) \leq D'_{0,j} \leq (0,\leq)$ is valid for $R_p$from Lemma 13.

Also note that since $x_i \in \mathsf{LFP}_{R_p}(D)$, from Definition 15 and Definition 12 (2) we have that $(-1,<) \leq D_{j,i} \leq (0,\leq)$ is valid for R_p for all clock x_j. Precisely, $d_{j,i} \in \{0, -p_1, p_2-p_1, p_1-1-p_2, p_1-1\}$ for some $p_1, p_2 \in \mathbb{P}$ where $p_2 \leq p_1$ is valid for R_p. Hence as $d_{j,i}+1 \in \{1, 1-p_1, p_2+1-p_1, p_1-p_2, p_1\}$, we have that $d'_{j,0} \in \mathcal{PLT}$ and $(0,\leq) \leq D'_{j,0} \leq (1,<)$ is valid for R_p.

Clearly Definition 12 (1) holds.

proof that Definition 12 (2) holds Third we set for all two clocks i, j where $x_i \in \mathsf{LFP}_{R_p}(D)$, $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$

$$D'_{i,j} = D'_{0,j} \quad \text{and} \quad D'_{j,i} = D'_{j,0},$$

for all two clocks $x_j, x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$

$$D'_{j,k} = D_{j,k}$$

and for all two clocks $x, y \in \mathsf{LFP}_{R_p}(D)$

$$D'_{x,y} = D'_{y,x} = (0, \leq).$$

Here as we have already proven above that $(-1, <) \leq D'_{0,j} \leq (0, \leq)$ and $(0, \leq) \leq D'_{0,j} \leq (1, <)$ are valid for R_p, Definition 12 (2) holds.

proof that Definition 12 (3) holds For all x_i:

- if $x_i \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,0} = (0, \leq)$, $D'_{0,i} = (0, \leq)$ hence $d'_{i,0} = -d'_{0,i}$ and $\lhd_{i0'} \lhd_{0i'} = \leq$, condition Definition 12 (3) holds;
- if $x_i \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $x \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,0} = D_{i,x} + (1, \leq)$, $D'_{0,i} = D_{x,i} + (-1, \leq)$ as condition Definition 12 (3) holds for $D_{i,x}$ and $D_{x,i}$ and $\lhd_{ij} \oplus \leq = \lhd_{ij}$, $\lhd_{ji} \oplus \leq = \lhd_{ji}$, condition Definition 12 (3) holds for $D'_{i,0}$ and $D'_{0,i}$.

For all x_i, x_j:

- if $x_i, x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $D'_{i,j} = D_{i,j}$ and $D'_{j,i} = D_{j,i}$, condition Definition 12 (3) holds as it holds for $D_{i,j}$ and $D_{j,i}$.
- if $x_i \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $x_j \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,j} = D_{i,j} + (1, \leq)$, $D'_{j,i} = D_{j,i} + (-1, \leq)$ condition Definition 12 (3) holds for $D_{i,j}$ and $D_{j,i}$ and $\lhd_{ij} \oplus \leq = \lhd_{ij}$, $\lhd_{ji} \oplus \leq = \lhd_{ji}$, condition Definition 12 (3) holds for $D'_{i,j}$ and $D'_{j,i}$. The case $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $x_i \in \mathsf{LFP}_{R_p}(D)$ is treated similarly.
- if $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,j} = D'_{j,i} = (0, \leq)$, hence $d'_{i,j} = -d'_{j,i} = 0$ and $\lhd_{ij'} \lhd_{ji'} = \leq$ and condition Definition 12 (3) holds.

proof that Definition 12 (4) holds Now we prove that Definition 12 (4) holds, *i. e.,* for all clocks x_i, x_j, x_k, valid conditions such as $D'_{i,j} \leq D'_{i,k} + D'_{k,j}$ remain valid in R_p. This is not trivial since, in this construction some clocks have been updated. Precisely, for all clocks x_i, x_j, x_k, are valid for R_p:

1. if $x_i, x_j, x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: let $x \in \mathsf{LFP}_{R_p}(D)$ and
   - if i, j, k are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if i, j are different from 0, $k = 0$, we have $D'_{i,0} = D_{i,x} + (1, \leq)$, $D'_{i,j} = D_{i,j}$ and $D'_{j,0} = D_{j,x} + (1, \leq)$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{i,x} \leq D_{i,j} + D_{j,x}$ is valid for R_p; then from Lemma 2 $D_{i,x} + (1, \leq) \leq D_{i,j} + D_{j,x} + (1, \leq)$ is valid for R_p and therefore, $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p .

- if i, k are different from 0, $j = 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,0} = D_{i,x} + (1, \leq)$ and $D'_{0,k} = D_{x,k} + (-1, \leq)$; we claim that

$$D_{i,k} \leq D_{i,x} + (1, \leq) + D_{x,k} + (-1, \leq) \tag{4.11}$$

  is valid for R_p, which is equivalent to $D'_{i,k} \leq D'_{i,0} + D'_{0,k}$ is valid for R_p. We have

$$(1, \leq) + (-1, \leq) = (1 + -1, \leq \oplus \leq) = (0, \leq) \tag{4.12}$$

  Since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$ from Definition 12 (4), we know that $D_{i,k} \leq D_{i,x} + D_{x,k}$ is valid for R_p; combining with (4.12) and since $D_{x,k} + (0, \leq) = D_{x,k}$, we obtain (4.11) and therefore our result.
- if i is different from 0, $j = k = 0$, we have $D'_{i,0} = D_{i,x} + (1, \leq)$, $D'_{j,k} = D'_{0,0} = (0, \leq)$; we have from Definition 11 (2b) that

$$D_{i,x} + (1, \leq) \leq D_{i,x} + (1, \leq)$$

  is valid for R_p. Hence Lemma 4 gives that

$$D'_{i,0} \leq D'_{i,0} + D'_{0,0}$$

  is valid for R_p.
- if j, k are different from 0, $i = 0$, we have $D'_{0,k} = D_{x,k} + (-1, \leq)$, $D'_{0,j} = D_{x,j} + (-1, \leq)$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{x,k} \leq D_{x,j} + D_{j,k}$ is valid for R_p. Moreover we have that

$$(-1, \leq) \leq (-1, \leq)$$

  is valid for R_p so we have from Definition 11 (2b) and Lemma 2

$$D_{x,k} + (-1, \leq) \leq D_{x,j} + (-1, \leq) + D_{j,k}$$

  is valid for R_p. Hence $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.
- if j is different from 0, $i = k = 0$, we have $D'_{0,j} = D_{x,j} + (-1, \leq)$ and $D'_{j,0} = D_{j,x} + (1, \leq)$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Lemma 3 we know that $(0, \leq) \leq D_{x,j} + D_{j,x}$ is valid for R_p; moreover, we have that

$$(1, \leq) + (-1, \leq) = (1 + -1, \leq \oplus \leq) = (0, \leq)$$

  and $D_{j,x} + (0, \leq) = D_{j,x}$. Then we have from Lemma 2

$$(0, \leq) \leq D_{x,j} + (-1, \leq) + D_{j,x} + (1, \leq)$$

  is valid for R_p and therefore $D'_{0,0} \leq D'_{0,j} + D'_{j,0}$ is valid for R_p.
- if k is different from 0, $i = j = 0$, we have $D'_{0,k} = D_{x,k} + (-1, \leq)$, $D'_{i,j} = D'_{0,0} = (0, \leq)$; we have from Definition 11 (2b) that

$$D_{x,k} + (-1, \leq) \leq D_{x,k} + (-1, \leq)$$

  is valid for R_p. Hence, as $D_{x,k} + (-1, \leq) + (0, \leq) = D_{x,k} + (-1, \leq)$ we have

$$D'_{0,k} \leq D'_{0,0} + D'_{0,k}$$

  is valid for R_p.

- if $i = j = k = 0$, we trivially have from Definition 12 (4) and Lemma 4

$$D'_{0,0} \leq D'_{0,0} + D'_{0,0}$$

is valid for R_p.

2. if $x_k \in \mathsf{LFP}_{R_p}(D)$ and $x_i, x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $k \neq 0$ and

- if i, j are different from 0, we have $D'_{i,k} = D'_{i,0} = D_{i,k} + (1, \leq)$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D'_{j,0} = D_{j,k} + (1, \leq)$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; moreover, since we have $(1, \leq) \leq (1, \leq)$ is valid for R_p then from Lemma 2

$$D_{i,k} + (1, \leq) \leq D_{i,j} + D_{j,k} + (1, \leq)$$

is valid for R_p, therefore we have $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
- if $i \neq 0$, $j = 0$, we have $D'_{i,k} = D'_{i,0} = D_{i,k} + (1, \leq)$, $D'_{i,0} = D_{i,k} + (1, \leq)$ and $D'_{0,k} = (0, \leq)$; clearly

$$(1, \leq) \leq (1, \leq) + (0, \leq)$$

and

$$D_{i,k} \leq D_{i,k}$$

are valid for R_p, then from Lemma 2 we obtain $D'_{i,k} \leq D'_{i,0} + D'_{0,k}$ is valid for R_p.
- if $i = 0$, $j \neq 0$, we have $D'_{0,k} = (0, \leq)$, $D'_{0,j} = D_{k,j} + (-1, \leq)$ and $D'_{j,k} = D'_{j,0} = D_{j,k} + (1, \leq)$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Lemma 3 we know that $(0, \leq) \leq D_{k,j} + D_{j,k}$ is valid for R_p. Moreover we have that

$$(1, \leq) + (-1, \leq) = (1 + -1, \leq \oplus \leq) = (0, \leq)$$

so we have from Lemma 2

$$(0, \leq) \leq D_{k,j} + D_{j,k} + (0, \leq)$$

is valid for R_p. Hence $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.
- if $i = j = 0$, we trivially have from Definition 12 (4) and Lemma 4

$$D'_{0,k} \leq D'_{0,0} + D'_{0,k}$$

is valid for R_p.

3. if $x_j \in \mathsf{LFP}_{R_p}(D)$ and $x_i, x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $j \neq 0$ and

- if i, k are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D'_{i,0} = D_{i,j} + (1, \leq)$ and $D'_{j,k} = D'_{0,k} = D_{j,k} + (-1, \leq)$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; moreover, since we have

$$(1, \leq) + (-1, \leq) = (1 + (-1), \leq \oplus \leq) = (0, \leq)$$

then as $D_{i,j} + D_{j,k} + (0, \leq) = D_{i,j} + D_{j,k}$, clearly $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.

- if $i \neq 0$, $k = 0$, we have $D'_{i,0} = D_{i,j} + (1, \leq)$, $D'_{i,j} = D'_{i,0} = D_{i,j} + (1, \leq)$ and $D'_{j,0} = (0, \leq)$; From Definition 11 (2b) we trivially have that $D_{i,j} + (1, \leq) \leq D_{i,j} + (1, \leq)$ is valid for R_p and therefore, $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p.
- if $i = 0$, $k \neq 0$, we have $D'_{0,k} = D_{j,k} + (-1, \leq)$, $D'_{0,j} = (0, \leq)$ and $D'_{j,k} = D'_{0,k} = D_{j,k} + (-1, \leq)$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{0,k} \leq D_{0,j} + D_{j,k}$ is valid for R_p. From Definition 11 (2b) we trivially have that $D_{j,k} + (-1, \leq) \leq D_{j,k} + (-1, \leq)$ is valid for R_p. As $(-1, \leq) + (0, \leq) = (-1, \leq)$, we have $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.
- if $i = k = 0$, we have $D'_{0,j} = (0, \leq)$ and $D'_{j,0} = (0, \leq)$; As we have

$$(0, \leq) + (0, \leq) = (0, \leq)$$

  we clearly have that $D'_{0,0} \leq D'_{0,j} + D'_{j,0}$ is valid for R_p.

4. if $x_j, x_k \in \mathsf{LFP}_{R_p}(D)$ and $x_i \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $j \neq 0, k \neq 0$ and

   - if i is different from 0, we have $D'_{i,k} = D'_{i,0} = D_{i,k} + (-1, \leq)$, $D'_{i,j} = D'_{i,0} = D_{i,k} + (-1, \leq)$ and $D'_{j,k} = (0, \leq)$; we have that $(-1, \leq) + (0, \leq) = (-1, \leq)$ and

$$D_{i,k} + (-1, \leq) \leq D_{i,k} + (-1, \leq)$$

     holds from Definition 11 (2b). Therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$.
   - if $i = 0$, we have $D'_{0,k} = (0, \leq)$, $D'_{0,j} = (0, \leq)$ and $D'_{j,k} = (0, \leq)$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ from Definition 12 (4), we know that $D_{0,k} \leq D_{0,j} + D_{j,k}$. As we have

$$(0, \leq) + (0, \leq) = (0, \leq)$$

     we clearly have that $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.

5. if $x_i \in \mathsf{LFP}_{R_p}(D)$ and $x_j, x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $i \neq 0$ and

   - if j, k are different from 0, we have $D'_{i,k} = D'_{0,k} = D_{i,k} + (-1, \leq)$, $D'_{i,j} = D'_{0,j} = D_{i,j} + (-1, \leq)$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Definition 12 (4) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; moreover, since we have

$$(-1, \leq) \leq (-1, \leq)$$

     is valid for R_p then from Lemma 2 we have $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if $j \neq 0$, $k = 0$, we have $D'_{i,0} = (0, \leq)$, $D'_{i,j} = D'_{0,j} = D_{i,j} + (-1, \leq)$ and $D'_{j,0} = D_{j,i} + (1, \leq)$; from Lemma 3 we know that $(0, \leq) \leq D_{i,j} + D_{j,i}$ is valid for R_p. Moreover, we have

$$(1, \leq) + (-1, \leq) = (1 + (-1), \leq \oplus \leq) = (0, \leq)$$

     then

$$(0, \leq) \leq D_{i,j} + D_{j,i} + (0, \leq)$$

     is valid for R_p and therefore, $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p.

- if $j = 0$, $k \neq 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,0} = (1, \leq)$ and $D'_{0,k} = D_{i,k} + (-1, \leq)$; we have that
$$(1, \leq) + (-1, \leq) = (1 + (-1), \leq \oplus \leq) = (0, \leq)$$
and from Definition 11 (2b) that
$$D_{i,k} \leq D_{i,k} + (0, \leq)$$
is valid for R_p, which gives us our result.
- if i is different from 0, $j = k = 0$, we have $D'_{i,0} = (0, \leq)$, $D'_{j,k} = D'_{0,0} = (0, \leq)$; we have from Definition 11 (2b) that
$$(0, \leq) \leq (0, \leq)$$
is valid for R_p. Hence
$$D'_{i,0} \leq D'_{i,0} + D'_{0,0}$$
is valid for R_p.

6. if $x_i, x_k \in \mathsf{LFP}_{R_p}(D)$ and $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $i \neq 0, k \neq 0$ and
   - if $j \neq 0$, we have $D'_{i,k} = (0, \leq)$, $D'_{i,j} = D'_{0,j} = D_{i,j} + (-1, \leq)$ and $D'_{j,k} = D'_{j,0} = D_{j,i} + (1, \leq)$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\blacksquare}(R_p)$, from Lemma 3 we know that $(0, \leq) \leq D_{i,j} + D_{j,i}$ is valid for R_p; we have
   $$(1, \leq) + (-1, \leq) = (1 + (-1), \leq \oplus \leq) = (0, \leq)$$
   and therefore from Lemma 2, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if $j = 0$, we have $D'_{i,k} = (0, \leq)$, $D'_{i,0} = (0, \leq)$ and $D'_{0,k} = (0, \leq)$; we have that $(0, \leq) + (0, \leq) = (0, \leq)$ and from Definition 11 (2b)
   $$(0, \leq) \leq (0, \leq)$$
   is valid for R_p. Therefore we obtain our result.

7. if $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$ and $x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $i \neq 0, j \neq 0$ and
   - if $k \neq 0$, we have $D'_{i,k} = D'_{0,k} = D_{i,k} + (-1, \leq)$, $D'_{i,j} = (0, \leq)$ and $D'_{j,k} = D'_{0,k} = D_{i,k} + (-1, \leq)$; we have that
   $$D_{i,k} \leq D_{i,k}$$
   is valid for R_p and from Lemma 2
   $$(-1, \leq) \leq (-1, \leq)$$
   is valid for R_p. Therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if $k = 0$, we have $D'_{i,0} = (0, \leq)$, $D'_{i,j} = (0, \leq)$ and $D'_{j,0} = (0, \leq)$; we have that $(0, \leq) + (0, \leq) = (0, \leq)$ and from Definition 11 (2b)
   $$(0, \leq) \leq (0, \leq)$$
   is valid for R_p: therefore $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p.

8. if $x_i, x_j, x_k \in \mathsf{LFP}_{R_p}(D)$: i, j, k are different from 0, we have $D'_{i,k} = (0, \leq)$, $D'_{i,j} = (0, \leq)$ and $D'_{j,k} = (0, \leq)$; we have that $(0, \leq) + (0, \leq) = (0, \leq)$ and from Definition 11 (2b)
$$(0, \leq) \leq (0, \leq)$$
is valid for R_p: therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.

proof that Definition 12 (5a) holds Finally, there is at least one clock $x_i \in \mathsf{LFP}_{R_p}(D)$ s.t. $D_{0,i} = D_{i,0} = (0, \leq)$. Hence condition Definition 12 (5a) holds.

Finally, we set $E'_i = E_i + 1$ if $x_i \in \mathsf{LFP}_{R_p}(D)$ and $E'_j = E_j$ if $x_j \in \mathbb{X} \backslash \mathsf{LFP}_{R_p}(D)$ We denote by (E, D') the obtained p–PDBM and $(E', D') \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$. □

4.4.3.3 Definition 13 to Definition 12 type (5a)

Lemma 15 ($p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ becomes $p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ after $TE_<$)**.** *Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$, then $TE_<\big((E, D)\big) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ respecting condition 5b.*

Proof. Suppose $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$. Since, in R_p, we know which parameters have the largest fractional part, we can determine $\mathsf{LFP}_{R_p}(D)$ from Lemma 10. If more than one clock belong to $\mathsf{LFP}_{R_p}(D)$ then their valuations have the same fractional part.

Indeed, from Definition 15 if $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$ then both $(0, \leq) \leq D_{i,j}$ and $(0, \leq) \leq D_{j,i}$ are valid for R_p, and from Definition 12 (2) we must have $D_{i,j} = D_{j,i} = (0, \leq)$.

Let $v \in R_p$. Let $x_i \in \mathsf{LFP}_{R_p}(D)$ and $w \in (E, v(D))$. By letting time elapse, $frac(w(x_i))$ is the first that might reach 1. Moreover, for all $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $frac(w(x_j))$ cannot reach 1 before $frac(w(x_i))$. We are going to construct a new $(E', D') = TE_<(E, D))$ which is an open–p–PDBM respecting condition 5b. While detailing the procedure of $TE_<$, we are going to prove that Definition 12 (1) and (2) hold for (E', D'). Further we will prove that (4) and (5b) also hold.

proof that Definition 12 (1) holds According to the definition of $TE_<$ (Algorithm 9), the first step is to set a new upper bound

$$D'_{i,0} = (1, <) \quad \text{for all } x_i \in \mathsf{LFP}_{R_p}(D)$$

and obviously $(0, \leq) \leq D'_{i,0} \leq (1, <)$ is valid for R_p. Then we set new upper bounds for all other clock $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$ by setting

$$D'_{j,0} = D_{j,i} + (1, <).$$

Indeed, $D_{j,i}$ is the constraint on the lower bound of $w(x_j) - w(x_i)$ and since the upper bound of x_i has increased, this gives the new upper bound of x_j. Note that since $x_i \in \mathsf{LFP}_{R_p}(D)$, from Definition 15 we have for all clock x_j that $(-1, <) \leq D_{j,i} \leq (0, \leq)$ is valid for R_p. Precisely, $d_{j,i} \in \{0, -p_1, p_2 - p_1, p_1 - 1 - p_2, p_1 - 1\}$ for some $p_1, p_2 \in \mathbb{P}$ where $p_2 \leq p_1$ is valid for R_p. Hence as $d_{j,i} + 1 \in \{1, 1 - p_1, p_2 + 1 - p_1, p_1 - p_2, p_1\}$, we have that $d'_{j,0} \in \mathcal{PLT}$, $\lhd_{j0'} = \lhd_{j0'} \oplus < = <$ and $(0, \leq) \leq D'_{j,0} \leq (1, <)$ is valid for R_p. Note that we cannot have $(d_{j,i}, \lhd_{ji}) = (-1, <)$ because even if $(d_{i,j}, \lhd_{ij}) = (1, <)$, since $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ we do not have have $0 \leq D_{j,i} + D_{i,j}$ is valid for R_p from Definition 12 (4) and Lemma 3.

Secondary we set for all clock x regardless of whether they are in $\mathsf{LFP}_{R_p}(D)$

$$D'_{0,x} = D_{0,x} + (0, <).$$

Since some time elapsed, lower bounds of all clocks are increased. Moreover, from Definition 13 (1) as $(-1, <) \leq D_{0,x} \leq (0, \leq)$ is valid for R_p, $(-1, <) \leq D'_{0,x} \leq (0, \leq)$ is also valid for R_p.

proof that Definition 12 (2) holds Third we set for all clocks x, y regardless of whether they are in $\mathsf{LFP}_{R_p}(D)$

$$D'_{x,y} = D_{x,y}$$

since no fractional part has reached 1, constraints on differences of clocks and integer parts remain unchanged. As it is the case in (E, D), Definition 12 (2) holds.

proof that Definition 12 (3) holds For all x_i:

- if $x_i \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,0} = (1, <)$, $D'_{0,i} = D_{0,i} + (0, <)$ hence $d'_{i,0} \neq d'_{0,i}$ and $\lhd_{i0'} \lhd_{0i'} = <$, condition Definition 12 (3) holds;
- if $x_i \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $x \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,0} = D_{i,x} + (1, <)$, $D'_{0,i} = D_{0,i} + (0, <)$ hence as $(0, \leq) \leq D'_{i,0}$ is valid for R_p and $D'_{0,i} \leq (0, \leq)$ is valid for R_p, we have $d'_{i,0} \neq d'_{0,i}$ and $\lhd_{i0'} \lhd_{0i'} = <$ and condition Definition 12 (3) holds.

For all x_i, x_j:

- if $x_i, x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $D'_{i,j} = D_{i,j}$ and $D'_{j,i} = D_{j,i}$, condition Definition 12 (3) holds as it holds for $D_{i,j}$ and $D_{j,i}$.
- if $x_i \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $x_j \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,0} = D_{i,j} + (1, <)$, $D'_{0,i} = D_{0,i} + (0, <)$ hence as $(0, \leq) \leq D'_{i,0}$ is valid for R_p and $D'_{0,i} \leq (0, \leq)$ is valid for R_p, we have $d'_{i,0} \neq d'_{0,i}$ and $\lhd_{i0'} \lhd_{0i'} = <$, condition Definition 12 (3) holds. The case $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $x_i \in \mathsf{LFP}_{R_p}(D)$ is treated similarly.
- if $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,j} = D'_{j,i} = (0, \leq)$, hence $d'_{i,j} = -d'_{j,i} = 0$ and $\lhd_{ij'} \lhd_{ji'} = \leq$ and condition Definition 12 (3) holds.

proof that Definition 12 (4) holds Now we prove that Definition 12 (4) holds, *i. e.,* for all clocks x_i, x_j, x_k valid conditions such as $D'_{i,j} \leq D'_{i,k} + D'_{k,j}$ remain valid in R_p. Indeed, when time elapses, all clocks have the same behavior, hence the difference between two clocks does not change without an update. Precisely, for all clocks x_i, x_j, x_k, are valid for R_p:

1. if $x_i, x_j, x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: let $x \in \mathsf{LFP}_{R_p}(D)$ and
   - if i, j, k are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ from Definition 13 (2), we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if i, j are different from 0, $k = 0$, we have $D'_{i,0} = D_{i,x} + (1, <)$, $D'_{i,j} = D_{i,j}$ and $D'_{j,0} = D_{j,x} + (1, <)$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{i,x} \leq D_{i,j} + D_{j,x}$ is valid for R_p; then from Lemma 2 $D_{i,x} + (1, <) \leq D_{i,j} + D_{j,x} + (1, <)$ is valid for R_p and therefore, $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p.
   - if i, k are different from 0, $j = 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,0} = D_{i,x} + (1, <)$ and $D'_{0,k} = D_{0,k} + (0, <)$; we claim that

$$D_{i,k} \leq D_{i,x} + (1, <) + D_{0,k} + (0, <) \tag{4.13}$$

is valid for R_p, which is equivalent to $D'_{i,k} \leq D'_{i,0} + D'_{0,k}$ is valid for R_p. Since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (1) we know that

$$D_{x,0} \leq (1, <) \tag{4.14}$$

is valid for R_p; moreover we have

$$(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <). \tag{4.15}$$

Since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{x,k} \leq D_{x,0} + D_{0,k}$ is valid for R_p; combining with (4.14) and (4.15) we obtain $D_{x,k} \leq (1, <) + D_{0,k} + (0, <)$ is valid for R_p. As $D_{i,x} \leq D_{i,x}$ is valid for R_p, using Lemma 2 we obtain

$$D_{i,x} + D_{x,k} \leq D_{i,x} + (1, <) + D_{0,k} + (0, <) \tag{4.16}$$

is valid for R_p. Now, since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{i,k} \leq D_{i,x} + D_{x,k}$ is valid for R_p and combining with (4.16) we obtain (4.13) and therefore our result.

- if i is different from 0, $j = k = 0$, we have $D'_{i,0} = D_{i,x} + (1, <)$, $D'_{j,k} = D'_{0,0} = (0, \leq)$; we have from Definition 11 (2b) that

$$D_{i,x} + (1, <) \leq D_{i,x} + (1, <)$$

is valid for R_p. Hence

$$D'_{i,0} \leq D'_{i,0} + D'_{0,0}$$

is valid for R_p.

- if j, k are different from 0, $i = 0$, we have $D'_{0,k} = D_{0,k} + (0, <)$, $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{0,k} \leq D_{0,j} + D_{j,k}$ is valid for R_p. Moreover we have that

$$D_{0,k} + (0, <) = (d_{0,k}, <) \quad \text{and} \quad D_{0,j} + (0, <) + D_{j,k} = (d_{0,j} + d_{j,k}, <)$$

so we have from Lemma 2

$$D_{0,k} + (0, <) \leq D_{0,j} + (0, <) + D_{j,k}$$

is valid for R_p. Hence $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.

- if j is different from 0, $i = k = 0$, we have $D'_{i,k} = D'_{0,0} = (0, \leq)$, $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,0} = D_{j,x} + (1, <)$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{0,x} \leq D_{0,j} + D_{j,x}$ is valid for R_p; moreover from Lemma 2,

$$D_{0,x} + (0, <) \leq D_{0,j} + (0, <) + D_{j,x}$$

is valid for R_p. As we have

$$(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <)$$

we obtain from Lemma 2 that

$$D_{0,x} + (1, <) \leq D_{0,j} + D_{j,x} + (1, <)$$

is valid for R_p. Recall that from Lemma 3 $(0, \leq) \leq D_{0,x} + D_{x,0}$ is valid for R_p. Since from Definition 13 (1) $D_{x,0} \leq (1, <)$ is valid for R_p, we have $(0, \leq) \leq D_{0,x} + (1, <)$ is valid for R_p. Therefore $D'_{0,0} \leq D'_{0,j} + D'_{j,0}$ is valid for R_p.

- if k is different from 0, $i = j = 0$, we have $D'_{i,k} = D'_{j,k} = D'_{0,k} = D_{0,k} + (0, <)$, $D'_{i,j} = D'_{0,0} = (0, \leq)$; we have from Definition 11 (2b) that

  $$D_{0,k} + (0, <) \leq D_{0,k} + (0, <)$$

  is valid for R_p. Hence from Lemma 4

  $$D'_{0,k} \leq D'_{0,0} + D'_{0,k}$$

  is valid for R_p.
- if $i = j = k = 0$, we trivially have

  $$D'_{0,0} \leq D'_{0,0} + D'_{0,0}$$

  is valid for R_p.

2. if $x_k \in \mathsf{LFP}_{R_p}(D)$ and $x_i, x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $k \neq 0$ and

   - if i, j are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if $i \neq 0$, $j = 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,0} = D_{i,k} + (1, <)$ and $D'_{0,k} = D_{0,k} + (0, <)$; we claim that $D_{i,k} \leq D_{i,k} + (1, <) + D_{0,k} + (0, <)$, *i. e.*,

     $$0 \leq (1, <) + D_{0,k} + (0, <) \tag{4.17}$$

     is valid for R_p, which is from Lemma 2 equivalent to $D'_{i,k} \leq D'_{i,0} + D'_{0,k}$ is valid for R_p. We have

     $$(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <). \tag{4.18}$$

     Since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $0 \leq D_{0,k} + D_{k,0}$ is valid for R_p and from Definition 13 (1) that $D_{k,0} \leq (1, <)$ is valid for R_p; combining with (4.18) we obtain (4.17) and therefore our result.
   - if $i = 0$, $j \neq 0$, we have $D'_{0,k} = D_{0,k} + (0, <)$, $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{0,k} \leq D_{0,j} + D_{j,k}$ is valid for R_p. Moreover we have that $(0, <) \leq (0, <)$ is valid for R_p so we have from Lemma 2

     $$D_{0,k} + (0, <) \leq D_{0,j} + (0, <) + D_{j,k}$$

     is valid for R_p. Hence $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.

- if $i = j = 0$, from Definition 13 (2) we trivially have

$$D'_{0,k} \leq D'_{0,0} + D'_{0,k}$$

is valid for R_p.

3. if $x_j \in \mathsf{LFP}_{R_p}(D)$ and $x_i, x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $j \neq 0$ and

- if i, k are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
- if $i \neq 0$, $k = 0$, we have $D'_{i,0} = D_{i,j} + (1, <)$, $D'_{i,j} = D_{i,j}$ and $D'_{j,0} = (1, <)$; from Definition 11 (2b) we trivially have that $D_{i,j} + (1, <) \leq D_{i,j} + (1, <)$ is valid for R_p and therefore, $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p.
- if $i = 0$, $k \neq 0$, we have $D'_{0,k} = D_{0,k} + (0, <)$, $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{0,k} \leq D_{0,j} + D_{j,k}$ is valid for R_p. Moreover we have that $(0, <) \leq (0, <)$ is valid for R_p so we have

$$D_{0,k} + (0, <) \leq D_{0,j} + (0, <) + D_{j,k}$$

holds from Definition 11 (2b). Hence $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.
- if $i = k = 0$, we have $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,0} = (1, <)$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Lemma 3 we know that $0 \leq D_{0,j} + D_{j,0}$ is valid for R_p, from Definition 13 (1) we know that $D_{j,0} \leq 1$ is valid for R_p which means $0 \leq D_{0,j} + (1, <)$ is valid for R_p. As we have

$$(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <)$$

we obtain that

$$(0, \leq) \leq D_{0,j} + (0, <) + (1, <)$$

is valid for R_p and therefore $D'_{0,0} \leq D'_{0,j} + D'_{j,0}$ is valid for R_p.

4. if $x_j, x_k \in \mathsf{LFP}_{R_p}(D)$ and $x_i \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $j \neq 0, k \neq 0$ and

- if i is different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
- if $i = 0$, we have $D'_{0,k} = D_{0,k} + (0, <)$, $D'_{0,j} = D_{0,j} + (0, <)$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{-}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{0,k} \leq D_{0,j} + D_{j,k}$ is valid for R_p. Moreover we have that $(0, <) \leq (0, <)$ is valid for R_p so we have from Lemma 2

$$D_{0,k} + (0, <) \leq D_{0,j} + (0, <) + D_{j,k}$$

is valid for R_p. Hence $D'_{0,k} \leq D'_{0,j} + D'_{j,k}$ is valid for R_p.

5. if $x_i \in \mathsf{LFP}_{R_p}(D)$ and $x_j, x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $i \neq 0$ and

   - if j, k are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if $j \neq 0$, $k = 0$, we have $D'_{i,0} = (1, <)$, $D'_{i,j} = D_{i,j}$ and $D'_{j,0} = D_{j,i} + (1, <)$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$, from Lemma 3 we know that $0 \leq D_{i,j} + D_{j,i}$. Then from Lemma 2

     $$(1, <) \leq D_{i,j} + D_{j,i} + (1, <)$$

     is valid for R_p and therefore, $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p.
   - if $j = 0$, $k \neq 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,0} = (1, <)$ and $D'_{0,k} = D_{0,k} + (0, <)$; we claim that

     $$D_{i,k} \leq (1, <) + D_{0,k} + (0, <)$$

     is valid for R_p, which is equivalent to $D'_{i,k} \leq D'_{i,0} + D'_{0,k}$ is valid for R_p. Since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ from Definition 13 (2), we know that $D_{i,k} \leq D_{i,0} + D_{0,k}$ is valid for R_p; moreover, from Definition 13 (1), we know that $D_{i,0} \leq (1, <)$ is valid for R_p. We have

     $$(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <)$$

     We obtain that

     $$D_{i,k} \leq D_{i,0} + D_{0,k} \leq (1, <) + D_{0,k} = (1, <) + D_{0,k} + (0, <)$$

     is valid for R_p and therefore our result.
   - if i is different from 0, $j = k = 0$, we have $D'_{i,0} = (1, <)$, $D'_{j,k} = D'_{0,0} = (0, \leq)$; from Definition 11 (2b) we have that

     $$(1, <) \leq (1, <)$$

     is valid for R_p. Hence

     $$D'_{i,0} \leq D'_{i,0} + D'_{0,0}$$

     is valid for R_p.

6. if $x_i, x_k \in \mathsf{LFP}_{R_p}(D)$ and $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $i \neq 0, k \neq 0$ and

   - if $j \neq 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if $j = 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,0} = (1, <)$ and $D'_{0,k} = D_{0,k} + (0, <)$; we claim that

     $$D_{i,k} \leq (1, <) + D_{0,k} + (0, <)$$

     is valid for R_p, which is equivalent to $D'_{i,k} \leq D'_{i,0} + D'_{0,k}$ is valid for R_p. Since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$ from Definition 13 (2), we know

that $D_{i,k} \leq D_{i,0} + D_{0,k}$ is valid for R_p; moreover, from Definition 13 (1), we know that $D_{i,0} \leq (1, <)$ is valid for R_p. We have

$$(1, <) + (0, <) = (1 + 0, < \oplus <) = (1, <)$$

We obtain that

$$D_{i,k} \leq D_{i,0} + D_{0,k} \leq (1, <) + D_{0,k} = (1, <) + D_{0,k} + (0, <)$$

is valid for R_p and therefore our result.

7. if $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$ and $x_k \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: $i \neq 0, j \neq 0$ and
   - if $k \neq 0$, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2), we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.
   - if $k = 0$, since both $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$ we have $D'_{i,j} = D_{i,j} = (0, \leq)$, $D'_{i,0} = (1, <)$ and $D'_{j,0} = (1, <)$; trivially $(1, <) \leq (0, \leq) + (1, <)$ is valid for R_p and therefore, $D'_{i,0} \leq D'_{i,j} + D'_{j,0}$ is valid for R_p.
8. if $x_i, x_j, x_k \in \mathsf{LFP}_{R_p}(D)$: i, j, k are different from 0, we have $D'_{i,k} = D_{i,k}$, $D'_{i,j} = D_{i,j}$ and $D'_{j,k} = D_{j,k}$; since $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$, from Definition 13 (2) we know that $D_{i,k} \leq D_{i,j} + D_{j,k}$ is valid for R_p; therefore, $D'_{i,k} \leq D'_{i,j} + D'_{j,k}$ is valid for R_p.

proof that Definition 12 (5b) holds Finally, for $x_i \in \mathsf{LFP}_{R_p}(D)$, $D'_{i,0} = (1, <)$ and for all clock j s.t. $D'_{0,j} = (0, \lhd)$, then we have $\lhd = <$. Condition Definition 12 (5b) is satisfied.

We set $E' = E$ and denote by (E, D') the obtained p–PDBM, which is $(E, D') \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$.

□

□

Note that, by Lemma 11 (E', D') is a p–PDBM. open–p–PDBMs are stable under $TE_<$ and $TE_=$, switching the condition they respect (5a, 5b). Applying $TE_<$ on a point–p–PDBM transforms it into an open–p–PDBM.

The following proposition proves that time elapsing behaves as we expect.

Proposition 3 (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. Note that this proof is inspired by [HRSV02, Proof of Lemma 3.13]

Lemma 16. *Let $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$. If (E, D) satisfies condition Definition 12 (5b) it has been obtained after applying Algorithm 9 on another* open–p*–PDBM satisfying condition Definition 12 (5a) or a* point–p*–PDBM.*

Let $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$. If (E, D) satisfies condition Definition 12 (5a) it has been obtained after applying Algorithm 15 on another open–p*–PDBM satisfying condition Definition 12 (5b) or after a non-parametric update applied on another* open–p*–PDBM or a* point–p*–PDBM.*

Proof. Let $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ and suppose (E, D) satisfies condition Definition 12 (5b). Since for all y, if $d_{0,y} = 0$ we have $\lhd_{0y} = <$, from Lemma 6 and Lemma 7 it cannot be the result of a non-parametric update where there is at least a clock x update and $D_{x,0} = D_{0,x} = (0, \leq)$. From Lemma 14 it cannot be the result of Algorithm 15, as there must be at least a clock x s.t. $D_{x,0} = D_{0,x} = (0, \leq)$. Then it is the result either from Lemma 12 of Algorithm 9 applied on an open–p–PDBM satisfying condition Definition 12 (5a), or from Lemma 15 of Algorithm 9 applied on a point–p–PDBM.

Let $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ and suppose (E, D) satisfies condition Definition 12 (5a). Since there is at least a clock y s.t. $D_{y,0} = D_{0,y} = (0, \leq)$, from Lemma 12 and Lemma 15 it cannot be the result of Algorithm 9, as for all x, if $d_{0,x} = 0$ we must have $\lhd_{ox} = <$. Then it is the result of either from Lemma 14 of Algorithm 15 applied on an open–p–PDBM satisfying condition Definition 12 (5b) or from Lemma 6 and Lemma 7 of Algorithm 1 applied on an open–p–PDBM or a point–p–PDBM. □

Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}(R_p)$. We have to consider two different cases: $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$ and $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$.

Lemma 17. *Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$. Let $v \in R_p$. There is $w' \in TE((E, v(D)))$ iff there is $w \in (E, v(D))$ and a delay δ s.t. $w' = w + \delta$.*

Proof. Let R_p a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}_{\blacksquare}(R_p)$. Let $v \in R_p$.

$\Longrightarrow$ **open–p–PDBM respecting Definition 12 (5a)**

Let $v \in R_p$. Consider $(E', D') = TE((E, D))$ respecting condition Definition 12 (5a), *i. e.,* suppose there is x_i s.t. $D'_{i,0} = -D'_{0,i} = (0, \leq)$. Let $w' \in (E', v(D'))$, for this x_i we have $w'(x_i) = 0$. We need to find a value δ s.t. $w' - \delta \in (E, v(D))$ which is equivalent to prove for all x_i, x_j

$$frac(w'(x_j)) - \delta - frac(w'(x_i)) + \delta \lhd_{ji} v(d_{j,i})$$

and

$$frac(w'(x_i)) - \delta - frac(w'(x_j)) + \delta \lhd_{ij} v(d_{i,j})$$

and

$$-frac(w'(x_j)) + \delta \lhd_{0j} v(d_{0,j}) \quad \text{and} \quad frac(w'(x_j)) - \delta \lhd_{j0} v(d_{j,0}).$$

In this proof we are going to define a δ which is different from 0, and give it an upper bound in order to show that constraints in (E, D) are satisfied while going backward of δ units of time from w'.

First we will prove that for all clock j, its constraints of lower bound $D_{0,j}$ and upper bound $D_{j,0}$ are satisfied. Second we will prove that for all i, bounds on their difference $D_{i,j}$ and $D_{j,i}$ are also satisfied.

We want to show that we have to go a little backward in time from w' to ensure the upper bounds $D_{j,0}$ of (E, D) hold. For this purpose, we are going to prove that for all x_j

$$D_{j,0} \leq D'_{j,0}$$

is valid for R_p. Intuitively this means upper bounds of clocks in (E', D') are greater than in (E, D), which is consistent as time is elapsing.

As (E', D') respects Definition 12 (5a) and precisely $(E', D') = TE_{=}((E, D))$, we know (E, D) is respecting condition Definition 12 (5b) from Lemma 12. As $frac(w'(x_i)) = 0$ it was in (E, D) a clock with the largest fractional part, *i. e.,* $x_i \in \mathsf{LFP}_{R_p}(D)$ and $D_{i,0} = (1, <)$.

By definition of $TE_{<}$ (cf. Algorithm 9), in (E, D) which is the open–p–PDBM obtained after the application of $TE_{<}$ on another p–PDBM (see Lemma 16), for each $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $D_{j,0} = D_{j,i} + (1, <)$ and for all $x_j \in \mathbb{X}$, we have $D_{j,0}$ is of the form $(d_{j,0}, <)$ for some $d_{j,0}$.

By definition of $TE_{=}$ applied to (E, D) (cf. Algorithm 15), in (E', D'), for each $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $D'_{j,0} = D_{j,i} + (1, \leq)$, *i. e.,* $d_{j,0} = d'_{j,0}$. Hence by Definition 11 (2b) and as $\lhd_{j0'}$ is either $\leq$ or $<$, we have

$$(d_{j,0}, <) = D_{j,0} \leq D'_{j,0} = (d_{j,0}, \lhd_{j0'})$$

is valid for R_p. Next we define the largest amount of time so that all upper bounds of (E, D) are satisfied.

We claim that for all x_j, $frac(w'(x_j)) - v(d_{j,0}) \leq 0$. Indeed, remark that by applying Algorithm 9 then Algorithm 15, constraints on upper bounds of clocks in (E, D) and (E', D') differ only by their $\lhd$. As for $i \in \mathsf{LFP}_{R_p}(D)$ and $j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$ it we have $D_{j,0} = D_{j,i} + (1, <)$ in (E, D) and $D'_{j,0} = D_{j,i} + (1, \leq)$ in (E', D'), so $d_{j,0} = d'_{j,0}$. Since for any x, its fractional part is less or equal to its upper bound in D and therefore in D', any difference between a fractional part and its upper bound is either negative or null. For all x, since $frac(w'(x)) \lhd_{x0'} v(d'_{x,0})$ we have $frac(w'(x)) - v(d'_{x,0}) \lhd_{x0'} 0$. Since $v(d'_{x,0}) = v(d_{x,0})$, $frac(w'(x)) - v(d_{x,0}) \lhd_{x0'} 0$, therefore we have our result.

Now we claim that we have to go at least an $\epsilon > 0$ backward in time to ensure all bounds of (E, D) are met. Let $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$. As

$$frac(w'(x_j)) \lhd_{j0'} v(d_{j,0})$$

we have

- either $\lhd_{j0'} = <$ and we already have $frac(w'(x_j)) < v(d_{j,0})$,
- or $\lhd_{j0'} = \leq$ and for any $\epsilon > 0$ we have $frac(w'(x_j)) - \epsilon < v(d_{j,0})$.

It is also true for each $x_i \in \mathsf{LFP}_{R_p}(D)$: after applying $TE_{<}$ recall that we have $D_{i,0} = (1, <)$. We can take $\epsilon > 0$ and define $frac(w(x_i)) = 1 - \epsilon$, so we have $frac(w(x_i)) < v(d_{i,0})$.

Now that we know we have to go a little backward in time (at least an $\epsilon > 0$) so upper bounds of (E, D) are satisfied, we are going to give an upper bound to ϵ so that all lower bounds $D_{0,j}$ of (E, D) are also satisfied.

Let

$$t_1 = \min_{x \in \mathbb{X}} \{frac(w'(x)) + v(d_{0,x})\}$$

We want to prove that $t_1 > 0$.

Let us prove that for all x_j, $D'_{0,j} \leq D_{0,j}$ is valid for R_p. Recall that for $x_i \in \mathsf{LFP}_{R_p}(D)$, we have that $D_{i,0} = (1, <)$. Moreover, from Definition 12 (4) $D_{i,j} \leq D_{i,0} + D_{0,j}$ is valid for R_p, then we have

$$D_{i,j} \leq (1, <) + D_{0,j}$$

is valid for R_p. Recall that after applying Algorithm 15, $D'_{0,j} = D_{i,j} + (-1, \leq)$. By Definition 11 (2b) we have $(-1, \leq) \leq (-1, \leq)$. We invoke Lemma 2 which gives

$$D_{i,j} + (-1, \leq) \leq (1, <) + D_{0,j} + (-1, \leq) = D_{0,j} + (0, <) \text{ is valid for } R_p. \tag{4.19}$$

As, from Definition 11 (2b) we have $D_{0,j} + (0, <) \leq D_{0,j}$ is valid for R_p, we infer (4.19) and it gives

$$D'_{0,j} \leq D_{0,j} \text{ is valid for } R_p.$$

Since $w' \in (E', v(D'))$ we have $-frac(w'(x_j)) \lhd_{0j'} v(d'_{0,j})$,

$$0 \lhd_{0j'} frac(w'(x_j)) + v(d'_{0,j}).$$

Then we have that

$$0 \lhd_{0j'} frac(w'(x_j)) + v(d'_{0,j}) \leq frac(w'(x_j)) + v(d_{0,j})$$

where ,

- either from Definition 11 (2a) $d'_{0,j} < d_{0,j}$;
- or from Definition 11 (2b), $d'_{0,j} \leq d_{0,j}$ and then $\lhd_{0j'} = \lhd_{0j} = <$. Indeed as $D'_{0,j} \leq D_{0,j}$ is valid for R_p, and since (E, D) is the open–p–PDBM obtained after the application of $TE_<$ (cf. Algorithm 9) on another p–PDBM (see Lemma 16), we have $\lhd_{0j} = <$.

To conclude we have that for all x_j either

$$0 \lhd_{0j'} frac(w'(x_j)) + v(d'_{0,j}) < frac(w'(x_j)) + v(d_{0,j})$$

or

$$0 < frac(w'(x_j)) + v(d'_{0,j}) \leq frac(w'(x_j)) + v(d_{0,j}).$$

As t_1 is by definition the minimum value of an expression $frac(w'(x_j)) + v(d_{0,j})$ for a given x_j, which as we just proved are all strictly positive, we have that for all x_j

$$0 < t_1 \leq frac(w'(x_j)) + v(d_{0,j}).$$

We proved that $t_1 > 0$, so we can set $\delta = \frac{t_1}{2}$ (therefore $\delta > 0$).

More intuitively δ is the value right in the middle of the least and the largest amount of time s.t. we can go backward in time from w' and respect all constraints defined in $(E, v(D))$.

Now we are going to prove that for any clock x_j, its constraints on lower and upper bounds are satisfied, *i. e.,*

$$-v(d_{0,j}) \lhd_{0j} frac(w'(x_j)) - \delta \lhd_{j0} v(d_{j,0}).$$

First as $\delta < t_1$, we have

$$-frac(w'(x_j))+\delta < -frac(w'(x_j))+t_1 \leq -frac(w'(x_j))+frac(w'(x_j))+v(d_{0,j}) = v(d_{0,j})$$

which is $-v(d_{0,j}) < frac(w'(x_j)) - \delta$. Since (E, D) is the open–p–PDBM obtained after the application of $TE_<$ (cf. Algorithm 15) on another p–PDBM (see Lemma 16), we have $\lhd_{0j} = <$ so $-v(d_{0,j}) \lhd_{0j} frac(w'(x_j)) - \delta$. Secondary as $0 < \delta$, we have

$$frac(w'(x_j))-\delta < frac(w)'(x_j)-0 \leq frac(w'(x_j))-frac(w'(x_j))+v(d_{j,0}) = v(d_{j,0})$$

which is $frac(w'(x_j)) - \delta < v(d_{j,0})$. Since (E, D) is the open–p–PDBM obtained after the application of $TE_<$ (cf. Algorithm 15) on another p–PDBM (see Lemma 16), we have $\lhd_{j0} = <$ so $frac(w'(x_j)) - \delta \lhd_{j0} v(d_{j,0})$

Now we prove that constraints defined in (E, D) on differences of clocks are also satisfied by going back of δ units of time from w'.

Recall that in (E', D') we have for all clock x_j,

$$D'_{j,i} = D'_{j,0} = D_{j,i} + 1 \quad \text{and} \quad D'_{i,j} = D'_{0,j} = -1 + D_{i,j}.$$

In addition by definition of $TE_=$, for $x_i \in \mathsf{LFP}_{R_p}(D)$, $E_{x_i} = E'_{x_i} - 1$ and for $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$, $E_{x_j} = E'_{x_j}$.

We already treated the case whether i or j are 0, now suppose i, j are both different from 0.

- if $x_i, x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: let $x \in \mathsf{LFP}_{R_p}(D)$ and recall that after applying Algorithm 15, $D'_{i,j} = D_{i,j}$, $D'_{j,i} = D_{j,i}$; we have that $frac(w'(x_j)) - frac(w'(x_i)) \lhd_{ij'} d'_{j,i} = d_{j,i}$, and therefore $frac(w'(x_j)) - \delta - frac(w'(x_i)) + \delta \lhd_{ji} d_{j,i}$.
  We also have that $frac(w'(x_i)) - frac(w'(x_j)) \lhd_{ij'} d'_{i,j} = d_{i,j}$, therefore $frac(w'(x_i)) - \delta - frac(w'(x_j)) + \delta \lhd_{ij} d_{i,j}$;
- if $x_i \in \mathsf{LFP}_{R_p}(D)$ and $x_j \in \mathbb{X} \setminus \mathsf{LFP}_{R_p}(D)$: recall that after applying Algorithm 15, $D'_{j,0} = D_{j,i} + (1, \leq)$, and $D'_{0,j} = D_{i,j} + (-1, \leq)$. Observe that as we added $\leq$ which is the neutral element of the addition $\oplus$ between two operators $\lhd$, we have $\lhd_{j0'} = \lhd_{ji}$ and $\lhd_{0j'} = \lhd_{ij}$. Note that as $x_i \in \mathsf{LFP}_{R_p}(D)$, in (E', D') we have $D'_{0,i} = (0, \leq) = D'_{i,0}$ which means $frac(w'(x_i)) = 0$. Going backward in time of δ units of time from $w'(x_i)$ means that $frac(w(x_i)) = 1 - \delta$.
  We have that

  $$frac(w'(x_j)) \lhd_{j0'} v(d'_{j,0}) = v(d_{j,i}) + 1$$

  hence $frac(w'(x_j)) - 1 \lhd_{ji} v(d_{j,i})$ which is equivalent to

  $$frac(w'(x_j)) - \delta - 1 + \delta \lhd_{ji} v(d_{j,i}).$$

  The same way we have

  $$-frac(w'(x_j)) \lhd_{0j'} v(d'_{0,j}) = v(d_{i,j}) - 1$$

  hence $1 - frac(w'(x_j)) \lhd_{ij} v(d_{i,j})$ which is equivalent to

  $$1 - \delta - frac(w'(x_j)) + \delta \lhd_{ij} v(d_{i,j}).$$

To conclude, we define for all x_j s.t. $D'_{0,j} \neq (0, \leq)$ and $D'_{j,0} \neq (0, \leq)$

$$w(x_j) = w'(x_j) - \delta$$

and for all x_i s.t. $D'_{0,i} = (0, \leq) = D'_{i,0}$

$$w(x_i) = (w'(x_i) - 1) + 1 - \delta$$

and clearly, $w \in (E, v(D))$.

$\Longrightarrow$ open–p–**PDBM respecting Definition 12 (5b)**

Let $v \in R_p$. Consider $(E', D') = TE((E, D))$ respecting condition Definition 12 (5b), *i. e.,* suppose there is at least an x_i s.t. $D'_{i,0} = (1, <)$ and for all j s.t. $D_{0,j} = (0, \lhd_{0j})$, then we have $\lhd_{0j} = <$. Let $w' \in (E', v(D'))$.

We need to find a value δ s.t. $w' - \delta \in (E, v(D))$ which is equivalent to prove for all x_i, x_j

$$frac(w'(x_j)) - \delta - frac(w'(x_i)) + \delta \lhd_{ji} v(d_{j,i})$$

and

$$frac(w'(x_i)) - \delta - frac(w'(x_j)) + \delta \lhd_{ij} v(d_{i,j})$$

and

$$-frac(w'(x_j)) + \delta \lhd_{0j} v(d_{0,j}) \quad \text{and} \quad frac(w'(x_j)) - \delta \lhd_{j0} v(d_{j,0}).$$

As done previously we are going to define a δ which is different from 0 so we satisfy condition Definition 12 (5a), and show that constraints in (E, D) are satisfied while going backward of δ units of time from w'.

We define the largest and the least amount of time so that all upper bounds of (E, D) are satisfied. Let

$$t_0 = \max_{x \in \mathbb{X}} \{0, frac(w'(x)) - v(d_{x,0})\}$$

and

$$t_1 = \min_{x \in \mathbb{X}} \{frac(w'(x)) + v(d_{0,x})\}.$$

We want to prove that $t_0 = t_1 > 0$. For this purpose, let us first show that for all i, j we have $frac(w'(x_j)) - v(d'_{j,0}) \leq frac(w'(x_i)) + v(d'_{0,i})$, which is $t_0 \leq t_1$.

First note that for all i, j

$$frac(w'(x_j)) - frac(w'(x_i)) \lhd_{ji'} v(d'_{j,i}).$$

By applying $TE_<$ (Algorithm 9) to (E, D), we have that $D'_{j,i} = D_{j,i}$, *i. e.,* $(d_{i,j}, \lhd_{ij}) = (d'_{i,j}, \lhd_{ij'})$, and from Definition 12 (4) we have that $D_{j,i} \leq D_{j,0} + D_{0,i}$ is valid for R_p.

Hence, we have from Definition 11 (2b) that either $v(d_{j,i}) < v(d_{j,0}) + v(d_{0,i})$ or $v(d_{j,i}) \leq v(d_{j,0}) + v(d_{0,i})$ and $\lhd_{ji} = \lhd_{j0} \oplus \lhd_{0i}$ or $\lhd_{ji} = <$ and $\lhd_{j0} \oplus \lhd_{0i} = \leq$.

We can then write that

$$frac(w'(x_j)) - frac(w'(x_i))(\lhd_{j0} \oplus \lhd_{0i}) v(d_{j,0}) + v(d_{0,i})$$

which is equivalent to

$$frac(w'(x_j)) - v(d_{j,0})(\lhd_{j0} \oplus \lhd_{0i}) frac(w'(x_i)) + v(d_{0,i})$$

so we obtain our result, as $(\lhd_{j0} \oplus \lhd_{0i})$ is either $\leq$ or $<$.

Now, recall that (E, D) respects condition Definition 12 (5a) so we have at least an x s.t. $D_{x,0} = D_{0,x} = (0, \leq)$.

For this clock x we have that $frac(w'(x)) = frac(w'(x)) - v(d_{x,0}) \leq t_0$ and that $t_1 \leq frac(w'(x)) + v(d_{0,x}) = frac(w'(x))$.

Hence $t_0 = t_1 = frac(w'(x))$.

As $\lhd_{x0} = \leq$, we have $(\lhd_{x0} \oplus \lhd_{0i}) = \lhd_{0i}$ and $(\lhd_{j0} \oplus \lhd_{0x}) = \lhd_{j0}$, which gives

$$frac(w'(x)) = frac(w'(x)) - v(d_{x,0}) \lhd_{0i} frac(w'(x_i)) + v(d_{0,i})$$

and

$$frac(w'(x_j)) - v(d_{j,0}) \lhd_{j0} frac(w'(x)) + v(d_{0,x}) = frac(w'(x)).$$

Moreover in (E', D') we have that $frac(w'(x)) \lhd_{0x'} v(d'_{0,x})$. Since (E', D') respects condition Definition 12 (5b), if $D'_{0,x} = (0, \lhd_{0x'})$ then $\lhd_{0x'} = <$. Hence $0 < frac(w'(x))$ and

$$0 < t_0 = t_1.$$

Let $\delta = t_0 = t_1$. More intuitively δ is the value right in the middle of the least and the largest amount of time s.t. we can go backward in time from w' and respect all constraints defined in $(E, v(D))$.

First we have

$$-frac(w'(x_j)) + \delta \leq -frac(w'(x_j)) + t_1 \lhd_{j0} - frac(w'(x_j)) + frac(w'(x_j)) + v(d_{0,j}) = v(d_{0,j})$$

which is $-v(d_{0,j}) \lhd_{j0} frac(w'(x_j)) - \delta$.

Secondary we have

$$frac(w'(x_j)) - \delta \leq frac(w)'(x_j) - t_0 \lhd_{0j} frac(w'(x_j)) - frac(w'(x_j)) + v(d_{j,0}) = v(d_{j,0})$$

which is $frac(w'(x_j)) - \delta \lhd_{0j} v(d_{j,0})$.

Now we prove that constraints defined in (E, D) on differences of clocks are also satisfied by going back of δ units of time from w'

Recall that in (E', D') from the definition of Algorithm 9 we have for all clocks x_i, x_j,

$$D'_{j,i} = D_{j,i} \quad \text{and} \quad D'_{i,j} = D_{i,j}.$$

Since we already treated the case whether i or j are 0, now suppose i, j are both different from 0. We have that $frac(w'(x_j)) - frac(w'(x_i)) \lhd_{ji'} v(d'_{j,i}) = v(d_{j,i})$, and therefore as $\lhd_{ji'} = \lhd_{ji}$,

$$frac(w'(x_j)) - \delta - frac(w'(x_i)) + \delta \lhd_{ji} v(d_{j,i}).$$

We also have that $frac(w'(x_i)) - frac(w'(x_j)) \lhd_{ij'} v(d'_{i,j}) = v(d_{i,j})$, therefore as $\lhd_{ij'} = \lhd_{ij}$,

$$frac(w'(x_i)) - \delta - frac(w'(x_j)) + \delta \lhd_{ij} v(d_{i,j}).$$

To conclude, we define for all x_j

$$w(x_j) = w'(x_j) - \delta$$

and clearly, $w \in (E, v(D))$.

Conversely, let $w \in (E, v(D))$,

$\Longleftarrow$ open–p–**PDBM respecting Definition 12 (5b)**

Suppose in (E, D) there is at least an x_i s.t. $D_{i,0} = (1, <)$ and for all j s.t. $D_{0,j} = (0, \lhd)$, we have $\lhd = <$. Let x_i be such a clock and $v \in R_p$.

Now consider $(E', D') = TE((E, D))$. We need to find a value δ s.t. $w + \delta \in (E', v(D'))$. which is equivalent to prove for all x_i, x_j

$$frac(w(x_j)) + \delta - frac(w(x_i)) - \delta \lhd_{ji'} v(d'_{j,i})$$

and

$$frac(w(x_i)) + \delta - frac(w(x_j)) - \delta \lhd_{ij'} v(d'_{i,j})$$

and

$$-frac(w(x_j)) - \delta \lhd_{0j'} v(d'_{0,j}) \quad \text{and} \quad frac(w(x_j)) + \delta \lhd_{j0'} v(d'_{j,0}).$$

As done previously we are going to define a δ which is different from 0, and show that constraints in (E, D) are satisfied while going forward of δ units of time from w.

Recall that $x_i \in \mathsf{LFP}_{R_p}(D)$ and let $\delta = 1 - frac(w(x_i)$ which we will prove is the exact amount of time so that all upper bounds of (E', D') are satisfied. Let

$$t_0 = \max_{x \in \mathbb{X}} \{-frac(w(x)) - frac(v(d'_{0,x}))\}$$

and

$$t_1 = \min_{x \in \mathbb{X}} \{frac(v(d'_{x,0})) - frac(w(x))\}.$$

Recall that since (E, D) respects condition Definition 12 (5b), for all j s.t. $D_{0,j} = (0, \lhd_{0j})$, we have $\lhd_{0j} = <$. Hence as $-frac(w(x_i) < v(d_{0,j})$, $frac(w(x_i)) \neq 0$. Using the same reasoning as before, we are going to prove that $t_0 \leq \delta \leq t_1$.

First we will prove that $t_0 \leq \delta$. Consider $x_i \in \mathsf{LFP}_{R_p}(D)$. For all clock x_j, since $w \in (E, v(D))$ we have $frac(w(x_i)) - frac(w(x_j)) \lhd_{ij} frac(v(d_{i,j}))$.

From Algorithm 15 applied to (E, D) and since $x_i \in \mathsf{LFP}_{R_p}(D)$ we obtain in (E', D') that $D'_{0,j} = D_{i,j} + (-1, \leq)$. Clearly we have $\lhd_{0j'} = \lhd_{ij} \oplus \leq = \lhd_{ij}$. It gives that

$$frac(w(x_i)) - frac(w(x_j)) - 1 (\lhd_{ij} \oplus \leq) frac(v(d_{i,j})) - 1$$

which is equivalent to $frac(w(x_i)) - frac(w(x_j)) - 1 \lhd_{0j'} frac(v(d'_{0,j}))$ which is equivalent to

$$frac(w(x_i)) - 1 \lhd_{0j'} frac(v(d'_{0,j})) + frac(w(x_j)).$$

This gives us our first result.

Second we will prove that $\delta \leq t_1$. Consider $x_i \in \mathsf{LFP}_{R_p}(D)$. For all clock x_j, from Definition 12 (4) we have $frac(w(x_j)) - frac(w(x_i)) \lhd_{ji} frac(v(d_{j,i}))$. We have

$$frac(w(x_j)) - frac(w(x_i)) + 1 \lhd_{ji} frac(v(d_{j,i})) + 1.$$

From Algorithm 15 applied to (E, D) and since $x_i \in \mathsf{LFP}_{R_p}(D)$ we obtain in (E', D') that $D'_{j,0} = D_{j,i} + (1, \leq)$. Clearly we have $\lhd_{j0'} = \lhd_{ji} \oplus \leq = \lhd_{ji}$. Then we can write that $frac(w(x_j)) - frac(w(x_i)) + 1 \lhd_{j0'} frac(v(d'_{j,0}))$ which is equivalent to

$$1 - frac(w(x_i)) \lhd_{j0'} frac(v(d'_{j,0})) - frac(w(x_j)).$$

This gives us our second result.

Now for all clock x_j, we obtain two results. First we have

$$-frac(w(x_j)) - \delta \lhd_{0j'} -frac(w(x_j)) - t_1 \leq -frac(w(x_j)) + frac(w(x_j)) + v(d'_{0,j}) = v(d'_{0,j})$$

which is $-v(d'_{0,j}) \lhd_{0j'} frac(w(x_j)) + \delta$.

Secondary we have

$$frac(w(x_j)) + \delta \lhd_{j0'} frac(w(x_j)) + t_0 \leq frac(w(x_j)) - frac(w(x_j)) + v(d'_{j,0}) = v(d'_{j,0})$$

which is $frac(w(x_j)) + \delta \lhd_{j0'} v(d'_{j,0})$.

Since we already treated the case whether i or j are 0, now suppose i, j are both different from 0.

Note that if both $x_i, x_j \in \mathsf{LFP}_{R_p}(D)$, as $frac(w(x_i)) = frac(w(x_j))$, $D_{i,j} = D'_{i,j} = (0, \leq)$ and $D_{j,i} = D'_{j,i} = (0, \leq)$ from Definition 15. Hence $frac(w(x_i)) + \delta - frac(w(x_j)) - \delta \lhd_{ij'} frac(v(d'_{i,j}))$ and $frac(w(x_j)) + \delta - frac(w(x_j)) - \delta \lhd_{ji'} frac(v(d'_{j,i}))$.

The same way, if both $x_i, x_j \notin \mathsf{LFP}_{R_p}(D)$ we have $D_{i,j} = D'_{i,j}$ and $D_{j,i} = D'_{j,i}$ and again our result. If either x_i or x_j is in $\mathsf{LFP}_{R_p}(D)$, the case is similar to $D'_{0,j}$ or $D'_{i,0}$.

Finally, we define $w' = w + \delta$ and $w' \in (E', v(D'))$.

$\Longleftarrow$ open–p–**PDBM respecting Definition 12 (5a)**

Suppose in (E, D) there is at least an x_j s.t. $D_{j,0} = D_{0,j} = (0, \leq)$ Let $v \in R_p$, and $x_i \in \mathsf{LFP}_{R_p}(D)$.

Now consider $(E', D') = TE((E, D))$. We need to find a value δ s.t. $w + \delta \in (E', v(D'))$. which is equivalent to prove for all x_i, x_j

$$frac(w(x_j)) + \delta - frac(w(x_i)) - \delta \lhd_{ji'} v(d'_{j,i})$$

and

$$frac(w(x_i)) + \delta - frac(w(x_j)) - \delta \lhd_{ij'} v(d'_{i,j})$$

and

$$-frac(w(x_j)) - \delta \lhd_{0j'} v(d'_{0,j}) \quad \text{and} \quad frac(w(x_j)) + \delta \lhd_{j0'} v(d'_{j,0}).$$

As done previously we are going to define a δ which is different from 0, and show that constraints in (E, D) are satisfied while going forward of δ units of time from w.

Let

$$t_0 = \max_{x \in \mathbb{X}} \{0, -frac(w(x)) - frac(v(d'_{0,x}))\}$$

and

$$t_1 = \min_{x \in \mathbb{X}} \{frac(v(d'_{x,0})) - frac(w(x))\}.$$

We want to prove that $t_0 \leq t_1$. For this purpose, we are going to prove for all clocks i, j that $-frac(w(x_j)) - v(d'_{j,0}) \leq v(d'_{0,i}) - frac(w(x_i))$.

First note that

$$frac(w(x_j)) - frac(w(x_i)) \lhd_{ji} v(d_{j,i})$$

By definition of $TE_<$ applied to (E, D), we have that $D'_{j,i} = D_{j,i}$, and from Definition 12 (4) we have that $D'_{j,i} \leq D'_{j,0} + D'_{0,i}$.

Hence, we have from Definition 11 (2b) that either $d'_{j,i} < d'_{j,0} + d'_{0,i}$ or $d'_{j,i} = d'_{j,0} + d'_{0,i}$ and $\lhd_{ji'} = \lhd_{j0'} \oplus \lhd_{0i'}$ or $\lhd_{ji'} = <$ and $\lhd_{j0'} \oplus \lhd_{0i'} = \leq$.

We can then write that

$$frac(w(x_j)) - frac(w(x_i))(\lhd_{j0'} \oplus \lhd_{0i'}) v(d'_{j,0}) + v(d'_{0,i})$$

which is equivalent to

$$-frac(w(x_i)) - v(d'_{0,i})(\lhd_{j0'} \oplus \lhd_{0i'}) v(d'_{j,0}) - frac(w(x_j))$$

Now we prove that $t_0 = 0$. Clearly from Definition 12 for any clock i we have that $-frac(w(x_i)) \lhd_{0i} v(d_{0,i})$ which is equivalent to $-frac(w(x_i)) - v(d_{0,i}) \lhd_{0i} 0$.

Hence if as (E, D) there is at least an x_j s.t. $D_{j,0} = D_{0,j} = (0, \leq)$, for this clock j we have $-frac(w(x_j)) - v(d_{0,j}) = 0$.

By definition of $TE_<$ applied to (E, D), we have that $D'_{0,i} = D_{0,i} + (0, <)$. In order to respect the constraint $-frac(w(x_i)) - \delta \lhd_{0i'} v(d'_{0,i})$ which is, as $\lhd_{0i'} = <$, $-frac(w(x_i)) - \delta < v(d'_{0,i})$ and especially for j where $v(d'_{0,j}) = 0$ we have to find a $\delta > 0$.

In order to find an upper bound for δ, we are going to prove that $t_1 > 0$. From Definition 12 (4) we have in (E, D) that for any clocks i, j $D_{j,0} \leq D_{j,i} + D_{i,0}$. Let $x_i \in \mathsf{LFP}_{R_p}(D)$. From Definition 12 (1), we have that $D_{i,0} \leq (1, <)$. This gives that $D_{j,i} + D_{i,0} \leq D_{j,i} + (1, <)$.

By definition of $TE_<$ applied to (E, D), we have that $D'_{j,0} = D_{j,i} + (1, <)$. Hence we have $D_{j,0} \leq D'_{j,0}$.

Now as $frac(w(x_i)) \lhd_{i0} v(d_{i,0})$ we can write $frac(w(x_i)) \lhd_{i0'} v(d'_{i,0})$ and then $0 \lhd_{i0'} v(d'_{i,0}) - frac(w(x_i))$ where $\lhd_{i0'} = <$, which prove our result.

We define $\delta = \frac{t_1}{2}$, therefore $t_0 < \delta < t_1$. Now for all clock x_j, we obtain two results. First we have

$$-frac(w(x_j)) - \delta < -frac(w(x_j)) - t_1 \lhd_{0j'} -frac(w(x_j)) + frac(w(x_j)) + v(d'_{0,j}) = v(d'_{0,j})$$

which is $-v(d'_{0,j}) \lhd_{0j} frac(w(x_j)) + \delta$ as $\lhd_{0j'} = <$.

Secondary we have

$$frac(w(x_j))+\delta < frac(w(x_j))+t_0 \lhd_{j0'} frac(w(x_j))-frac(w(x_j))+v(d'_{j,0}) = v(d'_{j,0})$$

which is $frac(w(x_j)) + \delta \lhd_{j0} v(d'_{j,0})$ as $\lhd_{0j'} = <$.

Now we prove that constraints defined in (E', D') on differences of clocks are also satisfied by going forward of δ units of time from w

Recall that in (E', D') from the definition of Algorithm 9 we have for all clock x_j,

$$D'_{j,i} = D_{j,i} \quad \text{and} \quad D'_{i,j} = D_{i,j}.$$

Since we already treated the case whether i or j are 0, now suppose i, j are both different from 0. We have that $frac(w(x_j)) - frac(w(x_i)) \lhd_{ji} v(d_{j,i}) = v(d'_{j,i})$, and therefore as $\lhd_{ji'} = \lhd_{ji}$,

$$frac(w(x_j)) + \delta - frac(w(x_i)) - \delta \lhd_{ji'} v(d'_{j,i}).$$

We also have that $frac(w(x_i)) - frac(w(x_j)) \lhd_{ij} v(d_{i,j}) = v(d'_{i,j})$, therefore as $\lhd_{ij'} = \lhd_{ij}$,

$$frac(w(x_i)) + \delta - frac(w(x_j)) - \delta \lhd_{ij'} v(d'_{i,j}).$$

Finally, we define $w' = w + \delta$ and $w' \in (E', v(D'))$.

□

Lemma 18. *Let R_p be a parameter region and $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$. Let $v \in R_p$. There is $w' \in TE((E, v(D)))$ iff there is $w \in (E, v(D))$ and a delay δ s.t. $w' = w + \delta$.*

Proof. $\Longleftarrow$ $p\text{–}\mathcal{PDBM}_{\odot}(R_p)$

Let $v \in R_p$. Consider $(E', D') = TE((E, D))$ respecting condition Definition 12 (5b), *i. e.,* suppose there is at least an x_i s.t. $D'_{i,0} = (1, <)$ and for all j s.t. $D_{0,j} = (0, \lhd_{0j})$, then we have $\lhd_{0j} = <$. Let $w' \in (E', v(D'))$.

We need to find a value δ s.t. $w' - \delta \in (E, v(D))$ which is equivalent to prove for all x_i, x_j

$$frac(w'(x_j)) - \delta - frac(w'(x_i)) + \delta \lhd_{ji} v(d_{j,i})$$

and

$$frac(w'(x_i)) - \delta - frac(w'(x_j)) + \delta \lhd_{ij} v(d_{i,j})$$

and

$$-frac(w'(x_j)) + \delta \lhd_{0j} v(d_{0,j}) \quad \text{and} \quad frac(w'(x_j)) - \delta \lhd_{j0} v(d_{j,0}).$$

As done previously we are going to define a δ which is different from 0, and show that constraints in (E, D) are satisfied while going backward of δ units of time from w'.

We define the largest and the least amount of time so that all upper bounds of (E, D) are satisfied. Let

$$t_0 = \max_{x \in \mathbb{X}} \{0, frac(w'(x)) - v(d_{x,0})\}$$

and

$$t_1 = \min_{x \in \mathbb{X}} \{frac(w'(x)) + v(d_{0,x})\}.$$

We want to prove that $t_0 = t_1 > 0$. For this purpose, let us first show that for all i, j we have $frac(w'(x_j)) - v(d'_{j,0}) \leq frac(w'(x_i)) + v(d'_{0,i})$, which is $t_0 \leq t_1$.

First note that for all i, j

$$frac(w'(x_j)) - frac(w'(x_i)) \lhd_{ji'} v(d'_{j,i}).$$

By applying $TE_<$ (Algorithm 9) to (E, D), we have that $D'_{j,i} = D_{j,i}$, *i. e.*, $(d_{i,j}, \lhd_{ij}) = (d'_{i,j}, \lhd_{ij'})$, and from Definition 13 (2) we have that $D_{j,i} \leq D_{j,0} + D_{0,i}$ is valid for R_p.

Hence, we have from Definition 11 (2b) that either $v(d_{j,i}) < v(d_{j,0}) + v(d_{0,i})$ or $v(d_{j,i}) \leq v(d_{j,0}) + v(d_{0,i})$ and $\lhd_{ji} = \lhd_{j0} \oplus \lhd_{0i}$ or $\lhd_{ji} = <$ and $\lhd_{j0} \oplus \lhd_{0i} = \leq$.

We can then write that

$$frac(w'(x_j)) - frac(w'(x_i))(\lhd_{j0} \oplus \lhd_{0i}) v(d_{j,0}) + v(d_{0,i})$$

which is equivalent to

$$frac(w'(x_j)) - v(d_{j,0})(\lhd_{j0} \oplus \lhd_{0i}) frac(w'(x_i)) + v(d_{0,i})$$

so we obtain our result, as $(\lhd_{j0} \oplus \lhd_{0i})$ is either $\leq$ or $<$.

Now, recall that in (E, D) for all x we have $d_{0,x} = -d_{x,0}$ and $\lhd_{0x} = \lhd_{x0}$.

For any clock x we have that $frac(w'(x)) - v(d_{x,0}) \leq t_0$ and that $t_1 \leq frac(w'(x)) + v(d_{0,x}) = frac(w'(x)) - v(d_{x,0})$.

Hence $t_0 = t_1$.

As for all x, $\lhd_{x0} = \leq$, we have for all i, j that $(\lhd_{x0} \oplus \lhd_{0i}) = \lhd_{0i}$ and $(\lhd_{j0} \oplus \lhd_{0x}) = \lhd_{j0}$, which gives

$$t_1 \lhd_{0i} frac(w'(x_i)) + v(d_{0,i})$$

and

$$frac(w'(x_j)) - v(d_{j,0}) \lhd_{j0} t_0.$$

Moreover in (E', D') we have that $frac(w'(x)) \lhd_{0x'} v(d'_{0,x})$. From Lemma 16, (E', D') is obtained after applying Algorithm 9 and therefore $\lhd_{0x'} = <$. Hence $0 < frac(w'(x))$ and

$$0 < t_0 = t_1.$$

Let $\delta = t_0 = t_1$. More intuitively δ is the value right in the middle of the least and the largest amount of time s.t. we can go backward in time from w' and respect all constraints defined in $(E, v(D))$.

First we have

$$-frac(w'(x_j))+\delta \leq -frac(w'(x_j))+t_1 \lhd_{j0} -frac(w'(x_j))+frac(w'(x_j))+v(d_{0,j}) = v(d_{0,j})$$

which is $-v(d_{0,j}) \lhd_{j0} frac(w'(x_j)) - \delta$.

Secondary we have

$$frac(w'(x_j))-\delta \leq frac(w)'(x_j)-t_0 \lhd_{0j} frac(w'(x_j))-frac(w'(x_j))+v(d_{j,0}) = v(d_{j,0})$$

which is $frac(w'(x_j)) - \delta \lhd_{0j} v(d_{j,0})$.

Now we prove that constraints defined in (E, D) on differences of clocks are also satisfied by going back of δ units of time from w'

Recall that in (E', D') from the definition of Algorithm 9 we have for all clocks x_i, x_j,

$$D'_{j,i} = D_{j,i} \quad \text{and} \quad D'_{i,j} = D_{i,j}.$$

Since we already treated the case whether i or j are 0, now suppose i, j are both different from 0. We have that $frac(w'(x_j))-frac(w'(x_i)) \lhd_{ji'} v(d'_{j,i}) = v(d_{j,i})$, and therefore as $\lhd_{ji'} = \lhd_{ji}$,

$$frac(w'(x_j)) - \delta - frac(w'(x_i)) + \delta \lhd_{ji} v(d_{j,i}).$$

We also have that $frac(w'(x_i))-frac(w'(x_j)) \lhd_{ij'} v(d'_{i,j}) = v(d_{i,j})$, therefore as $\lhd_{ij'} = \lhd_{ij}$,

$$frac(w'(x_i)) - \delta - frac(w'(x_j)) + \delta \lhd_{ij} v(d_{i,j}).$$

To conclude, we define for all x_j

$$w(x_j) = w'(x_j) - \delta$$

and clearly, $w \in (E, v(D))$.

$\Longrightarrow$ $p\text{–}\mathcal{PDBM}_{\odot}(R_p)$

Assume in $(E, D) \in p\text{–}\mathcal{PDBM}_{\odot}(R_p)$. Let $v \in R_p$, and $x_i \in \mathsf{LFP}_{R_p}(D)$.

Now consider $(E', D') = TE((E, D))$. We need to find a value δ s.t. $w + \delta \in (E', v(D'))$. which is equivalent to prove for all x_i, x_j

$$frac(w(x_j)) + \delta - frac(w(x_i)) - \delta \lhd_{ji'} v(d'_{j,i})$$

and

$$frac(w(x_i)) + \delta - frac(w(x_j)) - \delta \lhd_{ij'} v(d'_{i,j})$$

and

$$-frac(w(x_j)) - \delta \lhd_{0j'} v(d'_{0,j}) \quad \text{and} \quad frac(w(x_j)) + \delta \lhd_{j0'} v(d'_{j,0}).$$

As done previously we are going to define a δ which is different from 0, and show that constraints in (E, D) are satisfied while going forward of δ units of time from w.

Let

$$t_0 = \max_{x \in \mathbb{X}} \{0, -frac(w(x)) - frac(v(d'_{0,x}))\}$$

and

$$t_1 = \min_{x \in \mathbb{X}} \{frac(v(d'_{x,0})) - frac(w(x))\}.$$

We prove that $t_1 \leq t_0$.

for any clock i we have that $D_{i,0} = (frac(p), \leq)$ and $D_{i,0} = (-frac(p), \leq)$ *i. e.,* $d_{0,i} = -d_{i,0}$ for some p, hence $-frac(w(x_i)) - v(d_{0,i}) = -frac(w(x_i)) + v(d_{i,0})$.

By definition of $TE_<$ applied to (E, D), we have that $D'_{0,i} = D_{0,i} + (0, <)$. In order to respect the constraint $-frac(w(x_i)) - \delta \lhd_{0i'} v(d'_{0,i})$ which is, as $\lhd_{0i'} = <$, $-frac(w(x_i)) - \delta < v(d'_{0,i})$, we have to find a $\delta > 0$.

In order to find an upper bound for δ, we are going to prove that $t_1 > 0$. From Definition 13 (2) we have in (E, D) that for any clocks i, j $D_{j,0} \leq D_{j,i} + D_{i,0}$. Let $x_i \in \mathsf{LFP}_{R_p}(D)$. From Definition 13 (1), we have that $D_{i,0} \leq (1, <)$. This gives that $D_{j,i} + D_{i,0} \leq D_{j,i} + (1, <)$.

By definition of $TE_<$ applied to (E, D), we have that $D'_{j,0} = D_{j,i} + (1, <)$. Hence we have $D_{j,0} \leq D'_{j,0}$.

Now as $frac(w(x_i)) \lhd_{i0} v(d_{i,0})$ we can write $frac(w(x_i)) \lhd_{i0'} v(d'_{i,0})$ and then $0 \lhd_{i0'} v(d'_{i,0}) - frac(w(x_i))$ where $\lhd_{i0'} = <$, which prove our result.

We define $\delta = \frac{t_1}{2}$, therefore $t_0 < \delta < t_1$. Now for all clock x_j, we obtain two results. First we have

$$-frac(w(x_j)) - \delta < -frac(w(x_j)) - t_1 \lhd_{0j'} -frac(w(x_j)) + frac(w(x_j)) + v(d'_{0,j}) = v(d'_{0,j})$$

which is $-v(d'_{0,j}) \lhd_{0j} frac(w(x_j)) + \delta$ as $\lhd_{0j'} = <$.

Secondary we have

$$frac(w(x_j)) + \delta < frac(w(x_j)) + t_0 \lhd_{j0'} frac(w(x_j)) - frac(w(x_j)) + v(d'_{j,0}) = v(d'_{j,0})$$

which is $frac(w(x_j)) + \delta \lhd_{j0} v(d'_{j,0})$ as $\lhd_{0j'} = <$.

Now we prove that constraints defined in (E', D') on differences of clocks are also satisfied by going forward of δ units of time from w

Recall that in (E', D') from the definition of Algorithm 9 we have for all clock x_j,

$$D'_{j,i} = D_{j,i} \quad \text{and} \quad D'_{i,j} = D_{i,j}.$$

Since we already treated the case whether i or j are 0, now suppose i, j are both different from 0. We have that $frac(w(x_j)) - frac(w(x_i)) \lhd_{ji} v(d_{j,i}) = v(d'_{j,i})$, and therefore as $\lhd_{ji'} = \lhd_{ji}$,

$$frac(w(x_j)) + \delta - frac(w(x_i)) - \delta \lhd_{ji'} v(d'_{j,i}).$$

We also have that $frac(w(x_i)) - frac(w(x_j)) \lhd_{ij} v(d_{i,j}) = v(d'_{i,j})$, therefore as $\lhd_{ij'} = \lhd_{ij}$,

$$frac(w(x_i)) + \delta - frac(w(x_j)) - \delta \lhd_{ij'} v(d'_{i,j}).$$

Finally, we define $w' = w + \delta$ and $w' \in (E', v(D'))$.

□

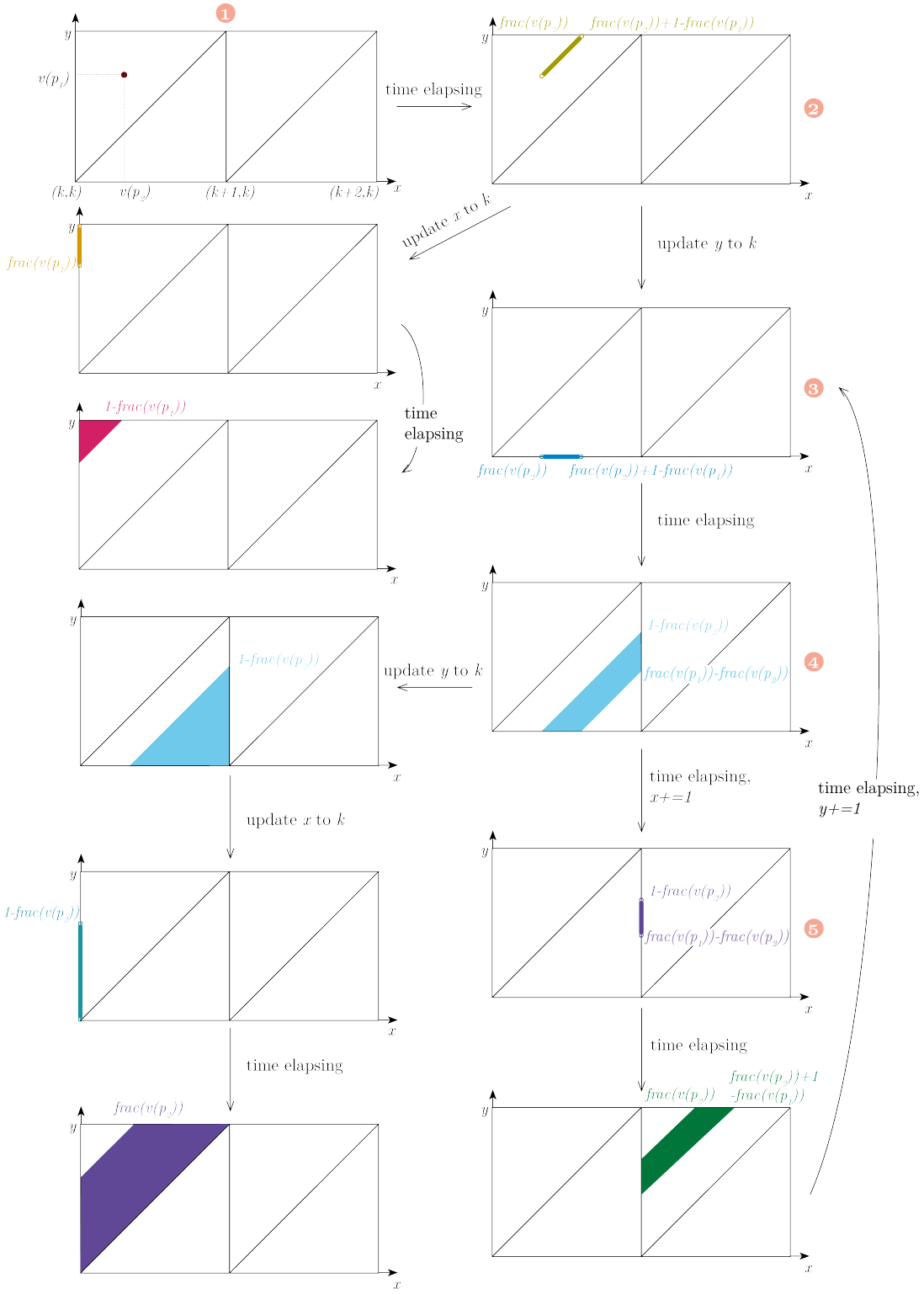


Figure 4.3: Representation of p–PDBMs in two dimensions with two clocks x, y, two parameters p_1, p_2 and v s.t. $\lfloor v(p_1) \rfloor = \lfloor v(p_2) \rfloor$ and $frac(v(p_1)) > frac(v(p_2))$. □

Running example: Figure 4.3 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 $y \in$ LFP reaches the next integer gives the open–p–PDBM satisfying

condition 5b (figure 2)

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

After an update of y to k prior to reaching $k+1$, the open–p–PDBM satisfying condition 5a obtained is (figure 3)

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

Time elapsing before $x \in \mathsf{LFP}$ reaches the next integer gives the open–p–PDBM satisfying condition 5b (figure 4)

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

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

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

4.4.4 Non-parametric guard

From [AD94, 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 [AD94, 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, the following lemma is straightforward

Lemma 19. *Let (E, D) be a p–PDBM for R_p and $v \in R_p$. Let g be a non-parametric guard. If $v \in guard_\forall(g, E, D)$, then for all $v' \in R_p$, $v' \in guard_\forall(g, E, D)$.*

4.4.5 Parametric guard

As for the previous result, using a projection on parameters *i. e.,* eliminating clocks, 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, when eliminating clocks in both a p–PDBM (E, D) and a parametric guard, 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. $w \models v(g)$.[3] Again, as any two $v, v' \in R_p$ satisfy the same constraints, the following lemma is straightforward

Lemma 20. *Let (E, D) be a* p*–PDBM for R_p and $v \in R_p$. Let g be a parametric guard. 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.

4.5 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 17 (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:*

1. $L' = L \times p\text{–}\mathcal{PDBM}(R_p)$
2. $L'_0 = (l_0, (E_0, D_0))$
3. $\zeta' = \{\big((l, (E, D)), a, (l', (E', D')\big) \in L' \times \Sigma \times L' \mid$ *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* $(E', D') = update(E'', D'', u_{np})$ *is an* open*–*p*–PDBM, 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* $(E', D') = \overline{update}(E'', D'', u)$ *is a* point*–*p*–PDBM.*$\}$

Let R_p be a parameter region, $\mathcal{A}$ be an R-U2P-PTA and $\mathcal{R}(\mathcal{A}) = (L', \Sigma, L'_0, \zeta')$ its parametric region automaton over R_p. 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 that for all i we have $\big((l_i, (E_i, D_i)), a_i, (l_{i+1}, (E_{i+1}, D_{i+1}))\big) \in \zeta'$, which we also write $(l_i, (E_i, D_i)) \xrightarrow{e_i} (l_{i+1}, (E_{i+1}, D_{i+1}))$. Note that we label our transitions with the edges of the R-U2P-PTA.

[3] 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 $v \in p\text{-}guard_\exists(g, E, D)$.

4.6 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 4. *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 $0 \leq i \leq f$, $w_i \in (E_i, v(D_i))$.*

Proof. $\Leftarrow$ By induction on the length of the run.

Let $v \in R_p$. As the basis for the induction, in the initial location $(l_0, \{0\}^H)$ the only valuation is reachable by an empty run of $v(\mathcal{A})$. Moreover $\{0\}^H \in (E_0, v(D_0))$ the initial p–PDBM containing only 0. Therefore the initial location $(l_0, (E_0, v(D_0)))$ is reachable by an empty run of $\mathcal{R}(\mathcal{A})$.

For the induction step, suppose for all v, there is run in $v(\mathcal{A})$ of length $f-1$ we have our result.

Let $v \in R_p$ and $\rho = (l_0, w_0) \xrightarrow{e_0} \cdots \xrightarrow{e_{f-2}} (l_{f-1}, w_{f-1}) \xrightarrow{e_{f-1}} (l_f, w_f)$ be a run of $v(\mathcal{A})$ of length f. By induction hypothesis, there is a run $\sigma = (l_0, (E_0, D_0)) \xrightarrow{e_0} \cdots \xrightarrow{e_{f-2}} (l_{f-1}, (E_{f-1}, D_{f-1}))$ in $\mathcal{R}(\mathcal{A})$ and for all $0 \leq i \leq f-1$, $w_i \in (E_i, v(D_i))$.

Consider e_{f-1}. By Definition 17 of the parametric region automaton, it is also in its set of edges ζ'. Three cases show up:

- If $e_{f-1} = \langle l_{f-1}, a, g, u_{np}, l_f \rangle$ contains no parametric guard nor parametric update. Using Definition 4 there is a delay δ (possibly 0) s.t. $(l_{f-1}, w_{f-1}) \overset{\delta}{\mapsto} (l_{f-1} w'_{f-1}) \overset{e_{f-1}}{\mapsto} (l_f, w_f)$ where $w'_{f-1} \models g$ and $w_f = [w'_{f-1}]_{u_{np}}$. As $w_{f-1} \in (E_{f-1}, v(D_{f-1}))$ there is $(E'_{f-1}, D'_{f-1}) \in \mathsf{Succ}((E_{f-1}, D_{f-1}))$ s.t. from Proposition 3 we have $w'_{f-1} \in (E'_{f-1}, v(D'_{f-1}))$. As $w'_{f-1} \models g$ by construction of our p–PDBMs (see Section 4.4.4) any other clock valuation belonging to $(E'_{f-1}, v(D'_{f-1}))$ satisfies g. Therefore $v \in \mathit{guard}_\forall(g, E'_{f-1}, D'_{f-1})$ and from Lemma 19, $R_p \subseteq \mathit{guard}_\forall(g, E'_{f-1}, D'_{f-1})$. Now, as $w_f = [w'_{f-1}]_{u_{np}}$ consider the open–p–PDBM $(E_f, D_f) = \mathit{update}((E'_{f-1}, D'_{f-1}), u_{np})$; from Lemma 8 we have $w_f \in (E_f, v(D_f))$. Finally there is an edge $(l_{f-1}, (E_{f-1}, D_{f-1})) \xrightarrow{e_{f-1}} (l_f, (E_f, D_f))$.
- If $e_{f-1} = \langle l_{f-1}, a, g, u, l_f \rangle$ contains a parametric guard and a parametric update. Using Definition 4 there is a delay δ (possibly 0) s.t. $(l_{f-1}, w_{f-1}) \overset{\delta}{\mapsto} (l_{f-1}, w'_{f-1}) \overset{e_{f-1}}{\mapsto} (l_f, w_f)$ where $w'_{f-1} \models v(g)$ and $w_f = [w'_{f-1}]_{v(u)}$. As $w_{f-1} \in (E_{f-1}, v(D_{f-1}))$ there is $(E'_{f-1}, D'_{f-1}) \in \mathsf{Succ}((E_{f-1}, D_{f-1}))$ s.t. from Proposition 3 we have $w'_{f-1} \in (E'_{f-1}, v(D'_{f-1}))$. As $w'_{f-1} \models v(g)$, $v \in \mathit{p\text{-}guard}_\exists(g, E'_{f-1}, D'_{f-1})$ and from Lemma 20, $R_p \subseteq \mathit{p\text{-}guard}_\exists(g, E'_{f-1}, D'_{f-1})$. Now, as $w_f = [w'_{f-1}]_{v(u)}$ consider the point–p–PDBM $(E_f, D_f) = \overline{\mathit{update}}((E'_{f-1}, D'_{f-1}), u)$; $(E_f, v(D_f))$ contains only one clock valuation, precisely defined by the fully parametric update $v(u)$ so we have $w_f \in (E_f, v(D_f))$. Finally there is an edge $(l_{f-1}, (E_{f-1}, D_{f-1})) \xrightarrow{e_{f-1}} (l_f, (E_f, D_f))$.

– The case where e_{f-1} contains a non parametric guard and a parametric update is similar to the previous one.

Finally, there is a run $\sigma' = \sigma \xrightarrow{e_{f-1}} (l_f, (E_f, D_f))$ of length f in $\mathcal{R}(\mathcal{A})$ s.t. for all $0 \leq i \leq f$, $w_i \in (E_i, v(D_i))$.

$\Rightarrow$ By induction on the length of the run.

Let $v \in R_p$. As the basis for the induction, the initial location $(l_0, (E_0, v(D_0)))$ is reachable by an empty run of $\mathcal{R}(\mathcal{A})$. Moreover, as $\{0\}^H \in (E_0, v(D_0))$, the initial location $(l_0, \{0\}^H)$ is reachable by an empty run of $v(\mathcal{A})$.

For the induction step, suppose it is true for all run in $\mathcal{R}(\mathcal{A})$ of length $f-1$.

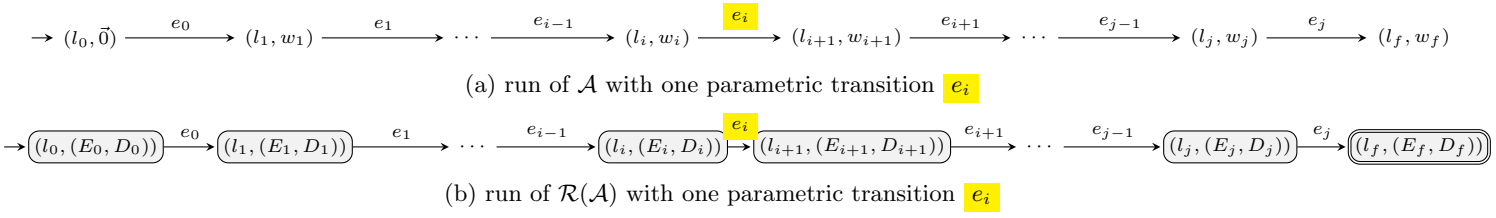
Let $v \in R_p$ and $\sigma = (l_0, (E_0, D_0)) \xrightarrow{e_0} \cdots \xrightarrow{e_{f-2}} (l_{f-1}, (E_{f-1}, D_{f-1})) \xrightarrow{e_{f-1}} (l_f, (E_f, D_f))$ be a run of $\mathcal{R}(\mathcal{A})$ of length f. Consider e_{f-1}. By Definition 17 of the parametric region automaton, it is also in the set of edges ζ of $\mathcal{A}$. Two cases show up:

– If $e_{f-1} = \langle l_{f-1}, a, g, u_{np}, l_f \rangle$ contains no parametric guard nor parametric update. By induction hypothesis, there is a run $\rho = (l_0, w_0) \xrightarrow{e_0} \cdots \xrightarrow{e_{f-2}} (l_{f-1}, w_{f-1})$ of $v(\mathcal{A})$ of length $f-1$ s.t. for all $0 \leq i \leq f-1$, $w_i \in (E_i, v(D_i))$. Using Definition 17 there is $(E'_{f-1}, D'_{f-1}) \in \mathsf{Succ}((E_{f-1}, D_{f-1}))$, $R_p \subseteq \mathit{guard}_\forall(g, E'_{f-1}, D'_{f-1})$ and $(E_f, D_f) = \mathit{update}((E'_{f-1}, D'_{f-1}), u_{np})$. From Proposition 3 we have $w'_{f-1} \in (E'_{f-1}, v(D'_{f-1}))$ and a delay δ s.t. $w'_{f-1} = w_{f-1} + \delta$. As $R_p \subseteq \mathit{guard}_\forall(g, E'_{f-1}, D'_{f-1})$ from Lemma 19 we have $v \in \mathit{guard}_\forall(g, E'_{f-1}, D'_{f-1})$ and $w'_{f-1} \models g$. Moreover, since $(E_f, D_f) = \mathit{update}((E'_{f-1}, D'_{f-1}), u_{np})$, we define $w_f = [w'_{f-1}]_{u_{np}}$ and therefore from Lemma 8, $w_f \in (E_f, v(D_f))$. Finally there is an edge $(l_{f-1}, w_{f-1}) \xrightarrow{e_{f-1}} (l_f, w_f)$ and a run $\rho' = \rho \xrightarrow{e_{f-1}} (l_f, w_f)$ in $v(\mathcal{A})$ of length f s.t. for all $0 \leq i \leq f$, $w_i \in (E_i, v(D_i))$.

– If $e_{f-1} = \langle l_{f-1}, a, g, u, l_f \rangle$ contains a parametric guard and a parametric update. Using Definition 17 there is $(E'_{f-1}, D'_{f-1}) \in \mathsf{Succ}((E_{f-1}, D_{f-1}))$, $R_p \subseteq p\text{-}\mathit{guard}_\exists(g, E'_{f-1}, D'_{f-1})$ and $(E_f, D_f) = \overline{\mathit{update}}((E'_{f-1}, D'_{f-1}), u)$. From Lemma 20 we can take $w'_{f-1} \in (E'_{f-1}, v(D'_{f-1}))$ s.t. $w'_{f-1} \models v(g)$. Let $w_f = [w'_{f-1}]_{v(u)}$. Clearly, $(E_f, D_f) = \overline{\mathit{update}}((E'_{f-1}, D'_{f-1}), u)$ is a point–p–PDBM; as $(E_f, v(D_f))$ contains only one clock valuation precisely defined by the fully parametric update $v(u)$, we have $w_f \in (E_f, v(D_f))$. From Proposition 3 as $w'_{f-1} \in (E'_{f-1}, v(D'_{f-1}))$ there is a delay δ and a $w_{f-1} \in (E_{f-1}, v(D_{f-1}))$ s.t. $w'_{f-1} = w_{f-1} + \delta$. Using the induction hypothesis, there is a run $\rho = (l_0, w_0) \xrightarrow{e_0} \cdots \xrightarrow{e_{f-2}} (l_{f-1}, w_{f-1})$ of $v(\mathcal{A})$ of length $f-1$ s.t. for all $0 \leq i \leq f-1$, $w_i \in (E_i, v(D_i))$. Finally there is an edge $(l_{f-1}, w_{f-1}) \xrightarrow{e_{f-1}} (l_f, w_f)$ and a run $\rho' = \rho \xrightarrow{e_{f-1}} (l_f, w_f)$ in $v(\mathcal{A})$ of length f s.t. for all $0 \leq i \leq f$, $w_i \in (E_i, v(D_i))$.

– The case where e_{f-1} contains a non parametric guard and a parametric update is similar to the previous one.

□



(a) run of $\mathcal{A}$ with one parametric transition e_i

(b) run of $\mathcal{R}(\mathcal{A})$ with one parametric transition e_i

Figure 4.4: A run in an R-U2P-PTA $\mathcal{A}$ (above) and its equivalent run in $\mathcal{R}(\mathcal{A})$ (below)

Example 3. *Consider Figure 4.4. Let $\mathcal{A}$ be an R-U2P-PTA, R_p a parameter region and $v \in R_p$. Suppose there is a run in $\mathcal{A}$, starting from the initial location $(l_0, \vec{0})$ reaching a goal location (l_f, w_f). Along this run, all edges are non-parametric transitions but $e_i = \langle l_i, g, a_i, u, l_{i+1} \rangle$. That is, u is a total parametric update, and g is a possibly parametric guard.*

The first part of this run, from $(l_0, \vec{0})$ to (l_i, w_i) is bisimulated by $\mathcal{R}_L(\mathcal{A})_0$, which is the local region automaton starting from $(l_0, (E_0, D_0))$ where (E_0, D_0) is the p*–PDBM of the initial clock region $\{\vec{0}\}$, and ends in $(l_i, (E_i, D_i))$. The second part of this run, from (l_{i+1}, w_{i+1}) to (l_f, w_f) is bisimulated by $\mathcal{R}_L(\mathcal{A})_1$, which is the local region automaton starting from $(l_{i+1}, (E_{i+1}, D_{i+1}))$ where (E_{i+1}, D_{i+1}) is a* point–p*–PDBM, and can reach $(l_f, (E_f, D_f))$ and further ends in $(l_s, (E_s, D_s))$.*

These runs in $\mathcal{R}_L(\mathcal{A})_0$ and $\mathcal{R}_L(\mathcal{A})_1$ contain only non-parametric transitions, and as there is an edge in $\mathcal{A}$ from (l_i, w_i) to (l_{i+1}, w_{i+1}), we have to bisimulate this run in $\mathcal{R}(\mathcal{A})$: this is the run starting from $(\mathcal{R}_L(\mathcal{A})_0, (l, (E_i, D_i)))$ and ending in $(\mathcal{R}_L(\mathcal{A})_1, (l_s, (E_s, D_s)))$ containing the parametric transition e_i, where $\overline{update}((E_i, D_i), u)$ gives (E_{i+1}, D_{i+1}).

From Proposition 4, we deduce that 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 4. *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 $v'(\mathcal{A})$ such that for all i, $(w_i, v) \eqsim (w'_i, v')$.*

Proof. Let $v \in R_p$ and ρ a run of $v(\mathcal{A})$ reaching (l_i, w_i). From Proposition 4, there is a run σ in $\mathcal{R}(\mathcal{A})$ s.t. each clock valuation at a location in ρ is in the p–PDBM at the same location in σ. Still from Proposition 4, for all $v' \in R_p$ there is a run ρ' in $v'(\mathcal{A})$ reaching (l_i, w'_i) s.t. each clock valuation at a location in ρ' is in the p–PDBM at the same location in σ (note that possibly $v = v'$). Therefore, we have for all $0 \leq j \leq i$ that $(w_i, v) \eqsim (w'_i, v')$ and the expected result.

□

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 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 [AD94, BDFP04], (integer) values exceeding the largest constant used in the guards or the parameter bounds are equivalent.

To test EF-emptiness given an R-U2P-PTA $\mathcal{A}$ and a goal location l, we first enumerate all parameter regions (which are a finite number), and apply for each R_p the following process: we pick $v \in R_p$ (*e. g.,* using a linear programming algorithm [Kar84]). Then, we consider $v(\mathcal{A})$ which is an updatable timed automaton and test the reachability of l in $v(\mathcal{A})$ [BDFP04]. Then EF-emptiness is false if and only if there is v and a run in $v(\mathcal{A})$ reaching l.

Theorem 5. *The EF-emptiness problem is PSPACE-complete for bounded R-U2P-PTAs.*

Proof. Since a TA is a special case of R-U2P-PTA we have the PSPACE-hardness [AD94]. Now, let G be a set of goal locations of $\mathcal{A}$. We build a non-deterministic Turing machine that:

1. takes $\mathcal{A}$, G and K as input
2. non-deterministically "guesses" a parameter region R_p
3. takes $v \in R_p$ and writes it to the tape
4. overwrites on the tape each parameter p by $v(p)$, giving the updatable TA $v(\mathcal{A})$
5. solves reachability in $v(\mathcal{A})$ for G
6. accepts iff the result of the previous step is "yes".

The machine accepts iff there is an integer valuation v bounded by K and a run in $v(\mathcal{A})$ reaching a location $l \in G$.

The size of the input is $|\mathcal{A}| + |G| + |K|$, using $|.|$ to denote the size in bits of the different objects. Moreover, the number of parameter regions is bounded (M is the number of parameters in $\mathcal{A}$) by $\left(M! \times 2^M \times \prod_{p \in \mathbb{P}}(2M + 2)\right) \times \left(2 \times (2 + M(3^{\frac{M-1}{2}} + 4))^3\right)$ since they are constructed as the clock regions of [AD94], the second part being the maximal number of constraints in a parameter region. Picking v at step *iii)* uses a PSPACE linear programming algorithm (*e. g.,* [Kar84]). Storing the valuation at step *iv)* uses at most $M \times |K|$ additional bits, which is polynomial w.r.t. the size of the input. Step *v)* also needs polynomial space from [BDFP04]. So globally this non-deterministic machine runs in polynomial space. Finally, by Savitch's theorem we have PSPACE = NPSPACE [Sav70], and the expected result.

□

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 6. *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.*

Proof. The procedure to obtain synthesis is as follows. We assume an R-U2P-PTA $\mathcal{A}$ and a goal location l.

1. enumerate all parameter regions (of which there is a finite number)
2. for each R_p, pick a parameter valuation we pick $v \in R_p$ (*e. g.,* using a linear programming algorithm [Kar84])
3. test the reachability of l in the updatable timed automaton $v(\mathcal{A})$, which is decidable [BDFP04]
4. if l is reachable in $v(\mathcal{A})$, add R_p to the list of synthesized regions

We finally return the union of all regions R_p that reach l.

The correctness immediately comes from Theorems 4 and 5.

□

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 $v(p) > K$.*

4.7 Case study

We implemented EFsynth for R-U2P-PTAs in IMITATOR, a parametric model checker for (extensions of) PTAs [AFKS12].

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 4.1 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 (blockSolution_x if $x = max$ or blockSolution_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"* ($EF(\text{reward}_y)$-synthesis), considered as a bug state in the automaton.

We run this R-U2P-PTA using IMITATOR [AFKS12][4]. 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;

1. $p_1 \geq p_2$: 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;

2. $pv_1 > v$: 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.

4.8 Conclusion and perspectives

Our class of 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 are the first "subclass" of PTAs to allow exact synthesis of bounded *rational*-valued parameters.

In terms of future works, beyond reachability emptiness, we aim at studying unavoidability-emptiness and language preservation emptiness ("given a reference parameter valuation, does there exist another parameter valuation with the same untimed language"), as well as their synthesis.

Finally, we would like to investigate whether our parametric updates can be applied to decidable hybrid extensions of TAs [HKPV98, BDG+13].

In the last two chapters we have studied parametric updates in TAs and PTAs, trying to find subclasses of PTAs for which classical TCTL formulae are decidable. In the next chapter, we will consider TCTL itself and study U-PTAs and L/U-PTAs without invariants.

[4]Experiments were conducted with IMITATOR 2.10.4 "Butter Jellyfish" on a 2.4 GHz Intel Core i5 processor with 2 GiB memory. Computation time is less than 1 second. Sources, binaries, models and results are available at imitator.fr/static/FORTE19/

Chapter 5

TCTL model checking lower/upper-bound parametric timed automata without invariants

5.1 Introduction

Recall that, EF-emptiness is undecidable even for a single bounded parameter [Mil00], even for a single rational-valued or integer-valued parameter [BBLS15], even with only one clock compared to parameters [Mil00], or with strict constraints only [Doy07] (see [And19] for a survey). In contrast, decidability is ensured in some restrictive settings such as over discrete time with a single parametric clock (*i. e.,* compared to parameters in at least one guard) [AHV93], or over discrete or dense time with one parametric clock and arbitrarily many non-parametric clocks [BO14, BBLS15], or over discrete time with two parametric clocks and a single parameter [BO14]. But the practical power of these restrictive settings remains unclear.

5.1.1 Motivation

In order to overcome these disappointing results, lower-bound/upper-bound parametric timed automata (L/U-PTAs) are introduced as a subclass of PTAs where each parameter either always appears as an upper bound when compared to a clock, or always as a lower bound [HRSV02]. L/U-PTAs enjoy mixed decidability results: while the EF-emptiness problem and the EF-universality problem ("Can we reach a given location, regardless of what valuations we give to the parameters?") are decidable, AF-emptiness ("is the set of valuations for which all runs eventually reach a given location empty?") is undecidable [JLR15]; as for EG-emptiness ("is the set of valuations for which one infinite or finite maximal run always remains in a given set of locations empty?"), it is decidable only when the parameter domain is bounded with closed bounds [AL17].

U-PTAs are L/U-PTAs with only upper-bound parameters [BL09], and are

Class	U-PTAs without invariants	integer-valued L/U-PTAs without invariant	L/U-PTAs	bounded PTAs	PTAs
EF	[HRSV02]	[HRSV02]	[HRSV02]	*[Mil00]*	*[AHV93, Mil00]*
AF	open	**Theorem 8**	*[JLR15]*	*[ALR16a]*	*[JLR15]*
EG	open	**Theorem 8**	*[AL17]*	*[AL17]*	*[AL17]*
AG	[HRSV02]	[HRSV02]	[HRSV02]	*[ALR16a]*	*[ALR16a]*
flat TCTL	open	**Theorem 8**	*[JLR15]*	*[Mil00]*	*[AHV93]*
TCTL	***Theorem 6***	***Theorem 6***	*[JLR15]*	*[Mil00]*	*[AHV93]*

Table 5.1: Decidability of the emptiness problems for PTAs and subclasses

TAs' simplest parametric extension; since their introduction, no problem was ever shown undecidable for U-PTAs, when decidable for TAs, and all their known decidability results only came from the decidability for the larger class of L/U-PTAs. [ALR16b] showed that, in terms of union of untimed words, U-PTAs are not more expressive than TAs. A natural question is to investigate whether their expressiveness is anyhow beyond that of TAs, or whether the parametric emptiness version of all problems decidable for TAs remains decidable for U-PTAs.

Note that in [JLR13], the authors claim that AF-emptiness is undecidable for U-PTAs but the original unpublished proof had a fatal flaw, which is why the result was weakened to L/U-PTAs in [JLR15]. The result for U-PTAs is therefore still open.

5.1.2 Contribution

Our first contribution is to show that the TCTL-emptiness problem ("given a TCTL formula, is the set of valuations v for which $v(\mathcal{A}) \models \varphi$ empty?") is undecidable for U-PTAs. This result comes in contrast with the fact that investigated flat TCTL formulas (namely EF, AG)—formulas that cannot be obtained by restraining another TCTL formula—are known to be decidable for U-PTAs, while others (EG and AF) are open. Our proof relies on the reduction of the halting problem of a 2-counter machine to the emptiness of the $\mathsf{EGAF}_{=0}$ formula.

Our second contribution is that EG-emptiness is PSPACE-complete for (unbounded) integer-valued L/U-PTAs without invariants. Let us stress that EG-emptiness is undecidable for classical unbounded integer-valued L/U-PTAs with invariants [AL17], which draws a more accurate border between decidability and undecidability results regarding L/U-PTAs. Moreover, we show that EG-universality (also known as AF-emptiness) is PSPACE-complete for (unbounded) integer-valued L/U-PTAs without invariants, despite being undecidable for classical (rational- or integer-valued) L/U-PTAs with invariants [JLR15]. These results highlight the power invariants confer upon the expressiveness of L/U-PTAs. We deduce from all this that flat TCTL emptiness and universality is also decidable for integer-valued L/U-PTAs without invariants, which also makes the decidability frontier more precise with respect to nesting of TCTL formulas.

We give a summary of the known decidability results in Table 5.1, with our

contributions in bold. We give from left to right the (un)decidability for U-PTAs, L/U-PTAs with integer-valued parameters without invariants, L/U-PTAs (the undecidability results also hold for integer-valued parameters), bounded PTAs (*i. e.,* with a bounded parameter domain), and PTAs. We review the emptiness of TCTL subformulas (EF, AF, EG, AG), flat TCTL and full TCTL. Decidability is given in green, whereas undecidability is given in italic red. As U-PTAs can be seen as the simplest parametric extension of TAs, our undecidability result moves the undecidability frontier closer to TAs, and confirms that timed automata (while enjoying many decidability results) are a formalism very close to the undecidability frontier.

5.1.3 Outline

Sections 5.2 and 5.3 show that TCTL-emptiness is undecidable for U-PTAs and bounded U-PTAs, respectively. Section 5.4 consists of the decidability results for integer-valued L/U-PTAs without invariants. Section 5.5 concludes the section and proposes some perspectives.

5.1.4 Additional notations

Given $U \subseteq \mathbb{X}$, we define the *reset* of a valuation w, denoted by $[w]_U$, as follows: $[w]_U(x) = 0$ if $x \in U$, and $[w]_U(x) = w(x)$ otherwise.

An upper-bound (resp. lower-bound) parameter p is such that, whenever it appears in a constraint $x \bowtie p + d$ with $d \in \mathbb{N}$ then necessarily $\bowtie \in \{\leq, <\}$ (resp. $\bowtie \in \{\geq, >\}$). A parameter *valuation* v is a function $v : \mathbb{P} \to \mathbb{Q}_+$. An *integer* parameter *valuation* v is a function $v : \mathbb{P} \to \mathbb{N}$. A clock is *parametric* if it is compared at least once to a parameter, and *non-parametric* otherwise.

A u-guard g (resp. an l-guard g) is a conjunction of inequalities of the form $x \bowtie d$, or $x \triangleleft p + d$ with p an upper-bound parameter (resp. $p + d \triangleleft x$ with p a lower-bound parameter) and $d \in \mathbb{N}$.

5.1.5 Lower/Upper-bound parametric timed automata

Let AP be a set of atomic propositions. Let us recall L/U-PTAs [BL09], in which we added a label function:

Definition 18 (L/U-PTA)**.** *An L/U-PTA $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, L, \mathbf{L}, l_0, \mathbb{X}, \mathbb{P}, \zeta)$, where:*

1. *Σ is a finite set of actions,*
2. *L is a finite set of locations,*
3. *$\mathbf{L}$ is a label function $\mathbf{L} : L \to 2^{AP}$,*
4. *$l_0 \in L$ is the initial location,*
5. *$\mathbb{X}$ is a finite set of clocks,*
6. *$\mathbb{P}$ is a finite set of parameters partitioned into lower-bound parameters and upper-bound parameters*

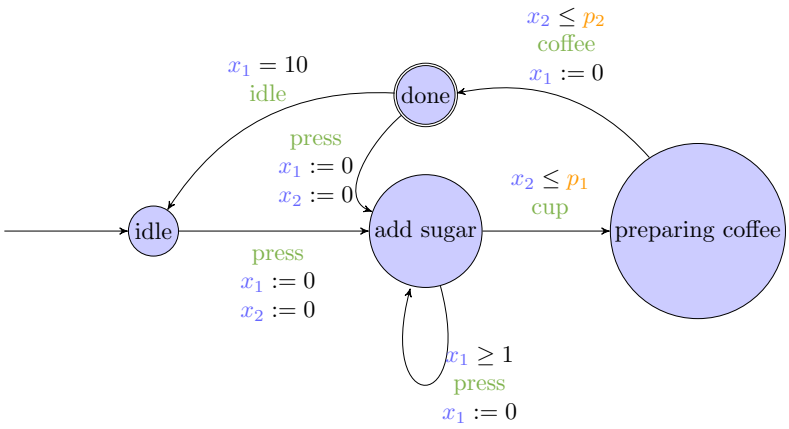


Figure 5.1: A coffee machine modeled using a U-PTA

7. *ζ is a finite set of edges $e = (l, g, a, U, l')$ where $l, l' \in L$ are the source and target locations, $a \in \Sigma$, $U \subseteq \mathbb{X}$ is a set of clocks to be reset, and g is a conjunction of a* u*-guard and an* l*-guard.*

Unlike the classical definition of [HRSV02], we consider L/U-PTAs without invariants. We define a U-PTA [BL09] as an L/U-PTA where in each edge, g is a u-guard. An example of U-PTA is given in Figure 5.1.

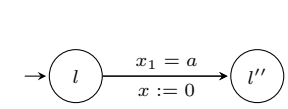
Example 4. *Consider the coffee machine in Figure 5.1, modeled using a U-PTA with 4 locations, 2 clocks (x_1 and x_2) and 2 parameters (p_1, p_2). The only accepting location (with a double border) is* done*. Only x_2 is a parametric clock, i. e., compared to a parameter.*

The machine can initially idle for an arbitrarily long time. Then, whenever the user presses the (unique) button (action press*), the U-PTA enters location "add sugar", resetting both clocks. There, the user can add a dose of sugar by pressing the button (action* press*), provided the guard ($x_1 \geq 1$) is satisfied, which resets x_1. That is, the user cannot press twice the button (and hence add two doses of sugar) in a time less than* 1*. Then, at most p_1 time units after the machine left the idle mode, a cup is delivered (action* cup*), and the coffee is being prepared; eventually, at most p_2 time units after the machine left the idle mode, the coffee (action* coffee*) is delivered. Then, after 10 time units, the machine returns to the idle mode—unless a user again requests a coffee by pressing the button.*

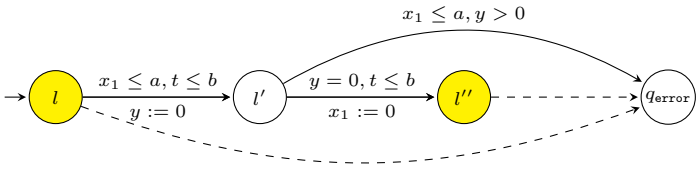
Note that an L/U-PTA where we replace all guards are made of conjunctions of inequalities of the form $x \bowtie p$, or $x \bowtie d$, with $d \in \mathbb{N}$, becomes a PTA as defined in [AHV93].

Given a state $s = (l, w)$, we say that s is reachable if s appears in a run of $v(\mathcal{A})$. By extension, we say that a label lb is reachable in $v(\mathcal{A})$ if there exists a state (l, w) that is reachable such that $lb \in \mathbf{L}(l)$. Given a set of locations $G \subseteq L$, we say that a run stays in G if all of its states (l, w) are such that $l \in G$.

A *maximal* run is a run that is either infinite (*i. e.,* contains an infinite number of discrete transitions), or that cannot be extended by a discrete transition. A



(a) Gadget fragment of [BBLS15]



(b) Modified gadget of [BBLS15] enforcing $\mathsf{EGAF}_{=0}\heartsuit$

Figure 5.2: A gadget fragment and its modification into a U-PTA

maximal run is deadlocked if it is finite, *i. e.,* contains a finite number of discrete transitions. By extension, we say that a TA is deadlocked if it contains at least one deadlocked run.

In this chapter, we address the following problems:

TCTL-emptiness problem:
INPUT: an L/U-PTA $\mathcal{A}$ and a TCTL formula φ
PROBLEM: is the set of valuations v such that $v(\mathcal{A}) \models \varphi$ empty?

TCTL-universality problem:
INPUT: an L/U-PTA $\mathcal{A}$ and a TCTL formula φ
PROBLEM: are all valuations v such that $v(\mathcal{A}) \models \varphi$?

More specifically, we will address in Section 5.4 the EG-emptiness (resp. EG-universality problem) *i. e.,* whether, given an L/U-PTA $\mathcal{A}$ and a subset of its locations G, the set of parameter valuations for which there is a run in $v(\mathcal{A})$ that stays in G is empty (resp. universal).

5.2 Undecidability of TCTL emptiness for U-PTAs

We exhibit here a formula that shows that TCTL emptiness is undecidable for U-PTAs.

Theorem 6. *The $\mathsf{EGAF}_{=0}$-emptiness problem is undecidable for U-PTAs.*

Proof. We reduce from the halting problem for two-counter machines, which is undecidable [Min67]. Recall that a two-counter machine is a finite state machine with two integer-valued counters $\mathsf{c}_1, \mathsf{c}_2$. Two different instructions (presented for c_1 and identical for c_2) are considered:

1. when in state q_i, increment c_1 and go to q_j;
2. when in state q_i, if $\mathsf{c}_1 = 0$ go to q_k, otherwise decrement c_1 and go to q_j.

We assume w.l.o.g. that the machine halts iff it reaches a special state q_{halt}.

We define a U-PTA that, under some conditions, will encode the machine, and for which $\mathsf{EGAF}_{=0}\heartsuit$-emptiness holds iff the machine does not halt (for some $\heartsuit \in AP$). Our U-PTA $\mathcal{A}$ uses two (possibly integer-valued) parameters a, b, and five clocks *i. e.,* a single non-parametric clock y and four parametric clocks x_1, x_2, z, t. We also omit the transition labels as they are not relevant for

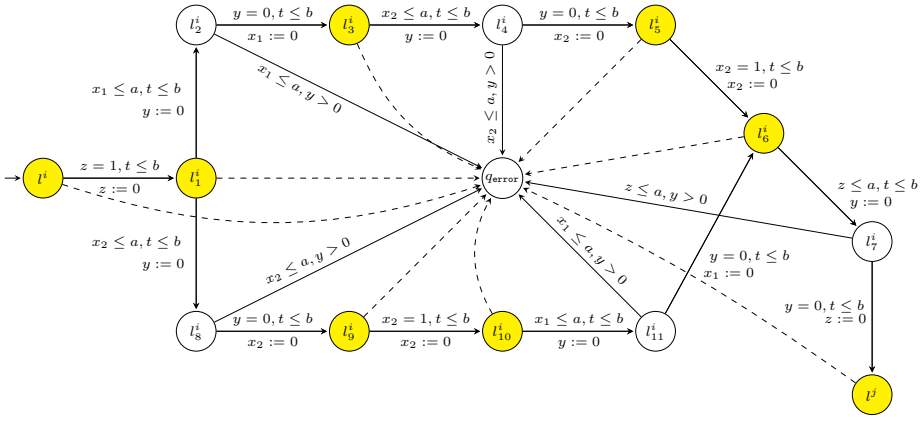


Figure 5.3: increment gadget

the emptiness problem. Each state q_i of the two-counter machine is encoded by a location l^i of $\mathcal{A}$. Each increment (resp. decrement) instruction of the two-counter machine is encoded into a U-PTA fragment depicted in Figures 5.3 and 5.4, respectively.

Our encoding is inspired by [BBLS15] and is such that when in l^i with $w(z) = 0$ then $w(x_1)$ (resp. $w(x_2)$) represents the value of the counter c_1 (resp. c_2). However, as U-PTAs disallow constraints of the form $x = a$, we need to considerably modify the encoding. Each of our locations has exactly one label: ♡ for the locations already present in [BBLS15] (depicted in yellow in our figures), and ♠ for the newly introduced locations (depicted in white in our figures). In [BBLS15], the gadgets encoding the two-counter machine instructions use edges of the form of Figure 5.2a. To define a proper U-PTA, we replace each of these edges by a special construction given in Figure 5.2b using only inequalities of the form $x \leq a$. Our goal is to show that a run will exactly encode the two-counter machine if all guards $x \leq a$ are in fact taken when the clock valuation is exactly equal to a. Those runs are further denoted by $\rho_{\heartsuit}$. Consider the transformed version given in Figure 5.2b: due to the $\leq$, runs exist that take the guard "too early" (*i. e.,* before $x_1 = a$). Those are denoted by $\rho_{\spadesuit}$. But, in that case, observe that in l', one can either take the transition to l'' in 0-time, or spend some time in l' and then (with guard $y > 0$) go to $q_{\texttt{error}}$. Therefore on this gadget, $\mathsf{EGAF}_{=0}\heartsuit$ is true at l' iff the guard $x_1 \leq a$ from l to l' is taken at the very last moment. Note that $\mathsf{EGAF}_{=0}\heartsuit$ is trivially true in l and l'' as both locations are labeled with ♡. (Also note that there are plenty of runs from l to $q_{\texttt{error}}$ that do not encode properly the machine; they will be discarded in our reasoning later.)

We also assume a condition $t \leq b$ on all guarded transitions, where t is a clock never reset. As presented in Figure 5.2b, there are transitions without guard (dashed) from l, l'' (labeled with ♡) to $q_{\texttt{error}}$. This is done to enforce the violation of $\mathsf{EGAF}_{=0}\heartsuit$ whenever $t = b$: indeed, while $t < b$ a run can either go to $q_{\texttt{error}}$ from a location labeled with ♡, or not, but as $t = b$ every run is forced to go to $q_{\texttt{error}}$, making $\mathsf{EGAF}_{=0}\heartsuit$ false.

The gadgets presented in Figures 5.3 and 5.4 provide an encoding to respectively increase and decrease the values of the counters of the two-counter

machine.

Increment We give the increment gadget for c_1 in Figure 5.3 (the gadget for c_2 is symmetric). Let v be a valuation, and assume we are in configuration (l^i, w), where $w(z) = 0$. First note that if $w(x_1) \geq v(a)$, there is no execution ending in l^j due to the delay of one time unit on the transition from l^i to l^i_1, and the guard $x_1 \leq a$ tested in both the upper and the lower branch in the automaton. The same reasoning is relevant for $w(x_2)$.

Assume $w(x_1), w(x_2) < v(a)$. Two cases show up: $w(x_1) \leq w(x_2)$ and $w(x_1) > w(x_2)$, which explains why we need two paths in Figure 5.3. First, if $w(x_1) \leq w(x_2)$, we can perform several executions with different time delays, but those are bounded. In the following, we write w as the tuple $(w(x_1), w(x_2), w(z), w(y))$, omitting t.

From l^i, we prove that there is a unique run that reaches l^j without violating our property. It is the one that takes each transition with a u-guard $x \leq a$ at the exact moment $w(x) = v(a)$ which we describe in the following.

From (l^i, w), the unique delay to pass the transition is 1, hence we arrive in the configuration $(l^i_1, (w(x_1) + 1, w(x_2) + 1, w(y) + 1, 0))$. Here, the largest delay to pass the transition is $v(a) - w(x_1) - 1$ so a configuration we possibly obtain is $(l^i_2, (d_1, d_2, d_3, 0))$ with $(d_1, d_2, d_3) \leq (v(a), w(x_2) - w(x_1) + v(a), v(a) - w(x_1) - 1)$. If $(d_1, d_2, d_3) < (v(a), w(x_2) - w(x_1) + v(a), v(a) - w(x_1) - 1)$ then the guard $y > 0$ in the transition to $q_{\texttt{error}}$ is verified, hence our property $\mathsf{EGAF}_{=0}\heartsuit$ is violated. We remove all these runs and keep the only run that ends in the exact configuration $(l^i_2, (v(a), w(x_2) - w(x_1) + v(a), v(a) - w(x_1) - 1, 0))$. As $y = 0$ holds the next configuration is $(l^i_3, (0, w(x_2) - w(x_1) + v(a), v(a) - w(x_1) - 1, 0))$. The largest delay to pass the next transition is $w(x_1) - w(x_2)$, so a configuration we possibly obtain is $(l^i_4, (d_1, d_2, d_3, 0))$ with $(d_1, d_2, d_3) \leq (w(x_1) - w(x_2), v(a), v(a) - w(x_2) - 1)$. If $(d_1, d_2, d_3) < (w(x_1) - w(x_2), v(a), v(a) - w(x_2) - 1)$ then the guard $y > 0$ in the transition to $q_{\texttt{error}}$ is verified, hence our property $\mathsf{EGAF}_{=0}\heartsuit$ is violated. We remove all these runs and keep the only run that ends in the exact configuration $(l^i_4, (w(x_1) - w(x_2), v(a), v(a) - w(x_2) - 1, 0)$. As $y = 0$ holds the next configuration is $(l^i_5, (w(x_1) - w(x_2), 0, v(a) - w(x_2) - 1, 0)$. Now the unique delay to pass the transition is 1, hence as we reset x_2 we arrive in the configuration $(l^i_6, (w(x_1) - w(x_2) + 1, 0, v(a) - w(x_2), 1)$. The largest delay to pass the next transition is $w(x_2)$, so a configuration we possibly obtain is $(l^i_7, (d_1, d_2, d_3, 0))$ with $(d_1, d_2, d_3) \leq (w(x_1) + 1, w(x_2), v(a))$. If $(d_1, d_2, d_3) < (w(x_1) + 1, w(x_2), v(a))$ then the guard $y > 0$ in the transition to $q_{\texttt{error}}$ is verified, hence our property $\mathsf{EGAF}_{=0}\heartsuit$ is violated. We remove all these runs and keep the only run that ends in the exact configuration $(l^i_7, (w(x_1) + 1, w(x_2), v(a), 0))$. As $y = 0$ holds the next configuration is $(l^j, (w(x_1) + 1, w(x_2), 0, 0))$, and as $w(z) = 0$, $w(x_1)$ represents the exact value of the counter c_1 increased by 1.

In its shorter form, this run is: $(l^i, w) \xrightarrow{1} (l^i_1, (w(x_1) + 1, w(x_2) + 1, w(y) + 1, 0)) \xrightarrow{v(a) - w(x_1) - 1} (l^i_2, (v(a), w(x_2) - w(x_1) + v(a), v(a) - w(x_1) - 1, 0)) \xrightarrow{0} (l^i_3, (0, w(x_2) - w(x_1) + v(a), v(a) - w(x_1) - 1, 0)) \xrightarrow{w(x_1) - w(x_2)} (l^i_4, (w(x_1) - w(x_2), v(a), v(a) - w(x_2) - 1, 0)) \xrightarrow{0} (l^i_5, (w(x_1) - w(x_2), 0, v(a) - w(x_2) -$

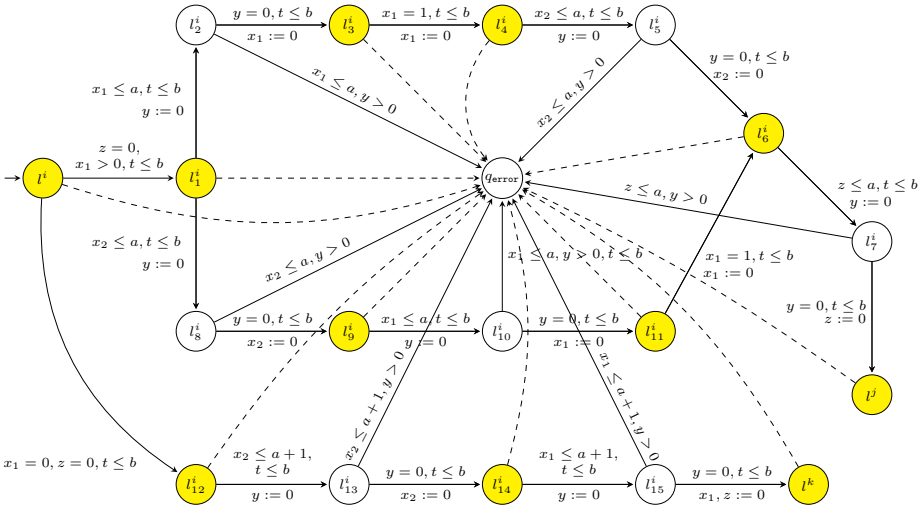


Figure 5.4: decrement gadget

$1, 0)) \xrightarrow{1} (l^i_6, (w(x_1){-}w(x_2){+}1, 0, v(a){-}w(x_2), 1)) \xrightarrow{w(x_2)} (l^i_7, (w(x_1){+}1, w(x_2), v(a), 0)) \xrightarrow{0}$
$(l^j, (w(x_1) + 1, w(x_2), 0, 0))$.

Second, if $w(x_1) > w(x_2)$ we take the lower branch of the automaton and apply the same reasoning.

Decrement and 0-test The decrement and 0-test gadget is similar: we reuse the reasoning of [BBLS15], and apply the same modifications as in Figure 5.2b. Note that the 0-test gadget has been completely rewritten from [BBLS15] to ensure a time elapsing of at least $a + 1$ time units when the guards are taken at the last moment.

We give the decrement gadget in Figure 5.4. Assume we are in a configuration (l^i, w) where $w(z) = 0$ and suppose $w(x_1) > 0$. We can enter the configuration $(l_1, (w(x_1), w(x_2), 0, w(y)))$ as the guard $z = 0$ ensures no time has elapsed.

Two cases show up: $w(x_1) \leq w(x_2)$ and $w(x_1) > w(x_2)$.

First, if $w(x_1) \leq w(x_2)$, we can perform several executions with different time delays, but those are bounded. From l^i, there is a unique run that reaches l^j without violating our property. It is the one that takes each transition with a u-guard $x \leq a$ at the exact moment $w(x) = v(a)$:

$(l^i, (w(x_1), w(x_2), 0, w(y)) \xrightarrow{0} (l^i_1, (w(x_1), w(x_2), 0, w(y)) \xrightarrow{v(a)-w(x_1)} (l^i_2, (v(a), w(x_2){+}$
$v(a){-}w(x_1), v(a){-}w(x_1), 0)) \xrightarrow{0} (l^i_3, (0, w(x_2){+}v(a){-}w(x_1), v(a){-}w(x_1), 0)) \xrightarrow{1}$
$(l^i_4, (0, w(x_2) + v(a) - w(x_1) + 1, v(a) - w(x_1) + 1, 1)) \xrightarrow{w(x_1)-w(x_2)-1} (l^i_5, (w(x_1) -$
$w(x_2){-}1, v(a), v(a){-}w(x_2), 0)) \xrightarrow{0} (l^i_6, (w(x_1){-}w(x_2){-}1, 0, v(a){-}w(x_2), 0)) \xrightarrow{w(x_2)}$
$(l^i_7, (w(x_1) - 1, w(x_2), v(a), 0)) \xrightarrow{0} (l^j, (w(x_1) - 1, w(x_2), 0, 0))$.

From (l^i, w), the unique delay to pass the transition is 0. From $(l^i_1, (w(x_1), w(x_2), 0, w(y))$, the largest delay to pass the next transition is $v(a) - w(x_1)$, so a configuration

we possibly obtain is $(l_2^i, (d_1, d_2, d_3, 0))$ with $(d_1, d_2, d_3) \leq (v(a), w(x_2) + v(a) - w(x_1), v(a) - w(x_1))$. If $(d_1, d_2, d_3) < (v(a), w(x_2) + v(a) - w(x_1), v(a) - w(x_1))$, then the guard $y > 0$ in the transition to $q_{\texttt{error}}$ is verified, hence our property $\mathsf{EGAF}_{=0}\heartsuit$ is violated. We remove all these runs and keep the only run that ends in the exact configuration $(l_2^i, (v(a), w(x_2)+v(a)-w(x_1), v(a)-w(x_1), 0))$. As $y = 0$ holds the next configuration is $(l_3^i, (0, w(x_2) + v(a) - w(x_1), v(a) - w(x_1), 0))$. Now the unique delay to pass the transition is 1, hence as we reset x_1 we arrive in the configuration $(l_4^i, (0, w(x_2) + v(a) - w(x_1) + 1, v(a) - w(x_1) + 1, 1))$. The largest delay to pass the next transition is $w(x_1) - w(x_2) - 1$, so a configuration we possibly obtain is $(l_5^i, (d_1, d_2, d_3, 0))$, with $(d_1, d_2, d_3) \leq (w(x_1) - w(x_2) - 1, v(a), v(a) - w(x_2))$. If $(d_1, d_2, d_3) < (w(x_1) - w(x_2) - 1, v(a), v(a) - w(x_2))$, then the guard $y > 0$ in the transition to $q_{\texttt{error}}$ is verified, hence our property $\mathsf{EGAF}_{=0}\heartsuit$ is violated. We remove all these runs and keep the only run that ends in the exact configuration $(l_5^i, (w(x_1)-w(x_2)-1, v(a), v(a)-w(x_2), 0))$. As $y = 0$ holds the next configuration is $(l_6^i, (w(x_1) - w(x_2) - 1, 0, v(a) - w(x_2), 0))$. The largest delay to pass the next transition is $w(x_2)$, so a configuration we possibly obtain is $(l_7^i, (d_1, d_2, d_3, 0))$, with $(d_1, d_2, d_3) \leq (w(x_1) - 1, w(x_2), v(a))$. If $(d_1, d_2, d_3) < (w(x_1) - 1, w(x_2), v(a))$, then the guard $y > 0$ in the transition to $q_{\texttt{error}}$ is verified, hence our property $\mathsf{EGAF}_{=0}\heartsuit$ is violated. We remove all these runs and keep the only run that ends in the exact configuration $(l_7^i, (w(x_1) - 1, w(x_2), v(a), 0))$. As $y = 0$ holds the next configuration is $(l^j, (w(x_1) - 1, w(x_2), 0, 0))$, and as $w(z) = 0$, $w(x_1)$ represents the exact value of the counter c_1 decreased by 1.

Secondary, if $w(x_1) > w(x_2)$ we take the lower branch of the automaton and apply the same reasoning.

Now assume we are in a configuration (l^i, w) where $w(z) = 0$ and suppose $w(x_1) = 0$. We have to reach location l^k, and ensure this is done in a $a+1$ time unit delay. We can enter the configuration $(l_{13}^i, (0, w(x_2), 0, w(y))$ as the guard $x_1 = 0, z = 0$ ensures no time has elapsed. The largest delay to pass the next transition is $v(a) + 1 - w(x_2)$, so a configuration we possibly obtain is $(l_{14}^i, (d_1, d_2, d_3, 0))$, with $(d_1, d_2, d_3) \leq (v(a)+1-w(x_2), v(a), v(a)+1-w(x_2))$. If $(d_1, d_2, d_3) < (v(a) + 1 - w(x_2), v(a), v(a) + 1 - w(x_2))$, then the guard $y > 0$ in the transition to $q_{\texttt{error}}$ is verified, hence our property $\mathsf{EGAF}_{=0}\heartsuit$ is violated. We remove all these runs and keep the only run that ends in the exact configuration $(l_{14}^i, (v(a) + 1 - w(x_2), v(a), v(a) + 1 - w(x_2), 0))$. As $y = 0$ holds the next configuration is $l_{15}^i, (v(a) + 1 - w(x_2), 0, v(a) + 1 - w(x_2), 0))$. The largest delay to pass the next transition is $w(x_2)$, so a configuration we possibly obtain is $(l_{16}^i, (d_1, d_2, d_3, 0))$, with $(d_1, d_2, d_3) \leq (v(a) + 1, w(x_2), v(a) + 1)$. If $(d_1, d_2, d_3) < (v(a) + 1, w(x_2), v(a) + 1)$, then the guard $y > 0$ in the transition to $q_{\texttt{error}}$ is verified, hence our property $\mathsf{EGAF}_{=0}\heartsuit$ is violated. We remove all these runs and keep the only run that ends in the exact configuration $(l_{16}^i, (v(a) + 1, w(x_2), v(a) + 1, 0))$. As $y = 0$ holds the next configuration is $(l^k, (0, w(x_2), 0, 0))$, and as $w(z) = 0$, $w(x_1)$ still represents the value of the counter $\mathsf{c}_1 = 0$.

Simulating the 2-counter machine Now, consider the runs $\rho_{\spadesuit}$ that take a u-guard $x \leq a$ "too early". At this moment, since after a small amount of time

we have $x \leq a$ and $y > 0$ are true, there is a run that eventually reaches $q_{\texttt{error}}$ and can never leave it; hence $\mathsf{EGAF}_{=0}\heartsuit$ does not hold for these runs. The same way, the runs $\rho_{\spadesuit}$ that take an unguarded transition to $q_{\texttt{error}}$ (whether or not $t \leq b$ is true) are stuck in a location labeled by $\spadesuit$; hence $\mathsf{EGAF}_{=0}\heartsuit$ does not hold for these runs. In the following, we do not consider these runs anymore.

Now, let us consider the runs $\rho_{\heartsuit}$ that take each u-guard at the very last moment, which is exactly when a clock $w(x) = v(a)$.

- If the two-counter machine halts then, there exist parameter valuations v (typically $v(a)$ larger than the maximum value of the counters during the computation and $v(b)$ larger than the duration of the corresponding run in $\mathcal{A}$), for which there is a (unique) run in the constructed U-PTA simulating correctly the machine, reaching $q_{\texttt{halt}}$ and staying there forever, so $\mathsf{EGAF}_{=0}\heartsuit$ holds for these valuations: hence $\mathsf{EGAF}_{=0}\heartsuit$-emptiness is false.
- Conversely, if the two-counter machine does not halt, then for any valuation, all runs either end in $q_{\texttt{error}}$ (either because they took an unguarded transition to $q_{\texttt{error}}$ or because they blocked due to the guard $t \leq b$—each gadget takes at least one time unit, so we can combine at most $v(b)$ gadgets—and again reached $q_{\texttt{error}}$); hence there is no parameter valuation for which $\mathsf{EGAF}_{=0}\heartsuit$ holds. Then $\mathsf{EGAF}_{=0}\heartsuit$-emptiness is true.

Therefore $\mathsf{EGAF}_{=0}\heartsuit$-emptiness is true iff the two-counter machine does not halt. □

□

Remark 2 (CTL)**.** *We may wonder if the* timed *aspect of TCTL is responsible for the undecidability. In fact, it is not, and we could modify the proof to show that CTL itself leads to undecidability. The idea is that we remove the unguarded transitions in both the increment and the decrement and 0-test gadgets, label each location of* $L \setminus \{q_{\texttt{error}}\}$ *with* $\heartsuit$*, and add an unguarded self-loop on* $q_{\texttt{halt}}$*. We claim that* EGAX*-emptiness is undecidable: we show that* $\mathsf{EGAX}\heartsuit$ *holds for a unique run of a U-PTA that simulates a two-counter machine, with a similar reasoning.*

5.3 Undecidability for bounded U-PTAs

We now show that undecidability remains even when the parameter domain is bounded. Note that, if we were addressing the full class of PTAs, showing an undecidability result for bounded PTAs automatically extends to the full class of PTAs, as we can simulate any bounded PTA by an unbounded PTA (see, *e. g.,* [ALR16b, Fig. 3]). This is not the case for U-PTAs: indeed, in [ALR16b], we showed that bounded (L/)U-PTAs are incomparable with (L/)U-PTAs; that is, it is impossible to simulate a bounded U-PTA using a U-PTA (*e. g.,* by using a gadget that enforces parameters to be bounded), due to the nature of guards, preventing us to artificially bound a parameter both from above and from below (in fact, for U-PTAs, bounding from below is possible, but not from above). Therefore, we must study both problems. Finally note that the EG-emptiness is decidable for bounded L/U-PTAs but undecidable for L/U-PTAs [AL17], which motivates further the need to investigate both versions.

Theorem 7. *The* $\mathsf{EGAF}_{=0}$*-emptiness is undecidable for bounded U-PTAs.*

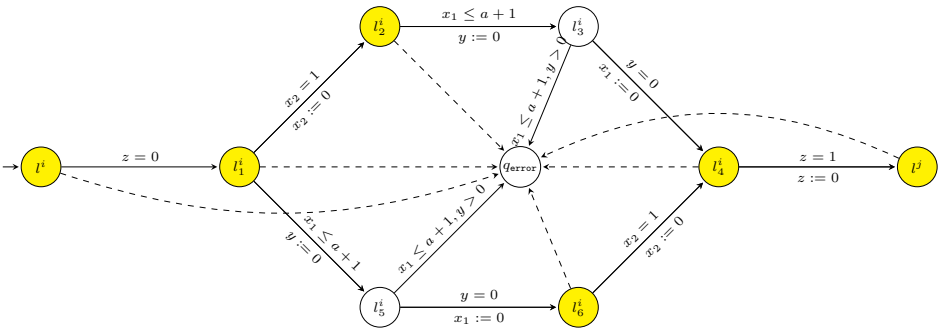


Figure 5.5: increment gadget

We reduce this time from the boundedness problem for two-counter machines (*i. e.,* whether the value of the counters remains bounded along the execution), which is undecidable [KC10].

We define a U-PTA that, under some conditions, will encode the machine, and for which $\mathsf{EGAF}_{=0}\heartsuit$-emptiness holds iff the counters in the machine remain bounded. The idea is as follows: we reuse a different encoding (originally from [ALR16a]), and apply the same modifications as we did in the proof of Theorem 6.

Our U-PTA $\mathcal{A}$ uses one parameter a, and four clocks *i. e.,* a single non-parametric clock y and three parametric clocks x_1, x_2, z. Each state q_i of the two-counter machine is encoded by a location l^i of $\mathcal{A}$. Each increment instruction of the two-counter machine is encoded into a U-PTA fragment depicted in Figure 5.5; the decrement instruction is a modification of the one in [ALR16a] using the same modifications as the increment gadget, and is depicted in Figure 5.6.

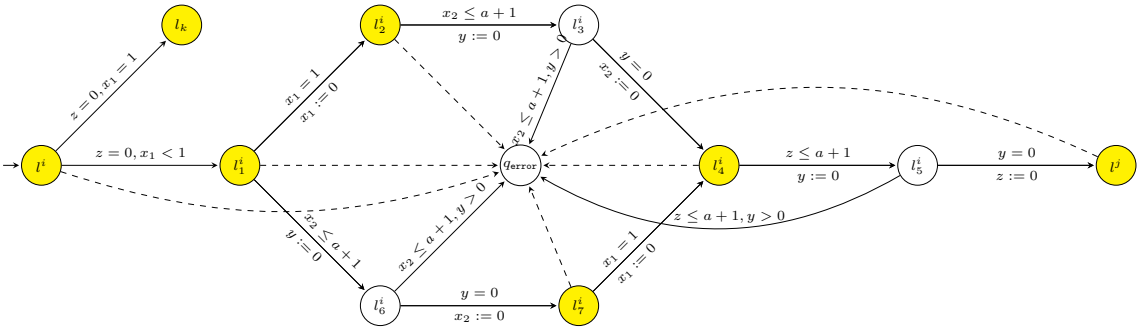


Figure 5.6: decrement gadget

Given v, our encoding is such that when in l^i with $w(z) = 0$ then $w(x_1)$ (resp. $w(x_2)$) represents the value of the counter c_1 (resp. c_2) encoded by $1 - v(a)\mathsf{c}_1$ (resp. $1 - v(a)\mathsf{c}_2$). Each of our locations has exactly one label: $\heartsuit$ for the locations already present in [ALR16a] (depicted in yellow in our figures), and $\spadesuit$ for the newly introduced locations (depicted in white in our figures).

We assume $a \in [0,1]$. The initial encoding when $w(z) = 0$ is $w(x_1) = 1 - v(a)\mathsf{c}_1, w(x_2) = 1 - v(a)\mathsf{c}_2, w(y) = 0$. Suppose $w(x_2) \leq w(x_1)$. From l^i, we prove that there is a unique run, going through the upper branch of the gadget, that reaches l^j without violating our property. It is the one that takes each transition with a u-guard $x \leq a+1$ at the exact moment $w(x) = v(a) + 1$:

$(l^i, w) \xrightarrow{0} (l^i_1, (1 - v(a)\mathsf{c}_1, 1 - v(a)\mathsf{c}_2, 0, 0)) \xrightarrow{v(a)\mathsf{c}_2} (l^i_2, (1 - v(a)\mathsf{c}_1 + v(a)\mathsf{c}_2, 0, v(a)\mathsf{c}_2, v(a)\mathsf{c}_2)) \xrightarrow{v(a)-v(a)\mathsf{c}_2+v(a)\mathsf{c}_1} (l^i_3, (v(a)+1, v(a)-v(a)\mathsf{c}_2+v(a)\mathsf{c}_1, v(a)+v(a)\mathsf{c}_1, 0) \xrightarrow{0} (l^i_4, (0, v(a) - v(a)\mathsf{c}_2 + v(a)\mathsf{c}_1, v(a) + v(a)\mathsf{c}_1, 0) \xrightarrow{1-v(a)-v(a)\mathsf{c}_1} (l^j, (1 - v(a)(\mathsf{c}_1 + 1), 1 - v(a)\mathsf{c}_2, 0, 1 - v(a)(\mathsf{c}_1 + 1))$.

The case were $w(x_2) \leq w(x_1)$ is similar, taking the lower branch of the gadget.

Now, let us consider the runs $\rho_\heartsuit$ that take each u-guard at the very last moment, which is exactly when a clock $w(x) = v(a) + 1$. (For the same reason as in the proof of Theorem 6, other runs violate the property anyway.)

- If the counters of the two-counter machine remain bounded then,
  - either the two-counter machine halts (by reaching q_{halt}) and there exist parameter valuations v (typically $v(a)$ small enough to encode the required value of the counters during the computation), for which there is a (unique) run in the constructed U-PTA simulating correctly the machine, reaching $q_{\texttt{halt}}$ and staying there forever, so $\mathsf{EGAF}_{=0}\heartsuit$ holds for these valuations: hence $\mathsf{EGAF}_{=0}\heartsuit$-emptiness is false;
  - or the two-counter machine loops forever (and never reaches q_{halt}) with bounded values of the counters, and again there exist parameter valuations v (again small enough to encode the maximal value of the counters) for which there is an infinite (unique) run in the U-PTA simulating correctly the machine. As this run is infinite, we infinitely often visit the decrement and/or the increment gadget(s), so $\mathsf{EGAF}_{=0}\heartsuit$ holds for these valuations: hence $\mathsf{EGAF}_{=0}\heartsuit$-emptiness is again false.
- Conversely, if the counters of the two-counter machine are unbounded, then for any valuation, all runs either end in $q_{\texttt{error}}$, either because they took an unguarded transition to $q_{\texttt{error}}$ or because they blocked due to the guard $x \leq a+1$ —indeed when in l^i_6, we have $w(z) = v(a)(\mathsf{c}_1 + 1)$ so if c_1 is unbounded, after a sufficient number of steps we cannot pass the guard $z = 1$— and again reached $q_{\texttt{error}}$. Hence there is no parameter valuation for which $\mathsf{EGAF}_{=0}\heartsuit$ holds. Then $\mathsf{EGAF}_{=0}\heartsuit$-emptiness is true.

Using the same reasoning as in the proof of Theorem 6 and [ALR16a], we conclude that $\mathsf{EGAF}_{=0}\heartsuit$-emptiness is true iff the values of the counters of the two-counter machine are unbounded.

5.4 Decidability of flat-TCTL for L/U-PTAs without invariants

In this section, we prove that the EG-emptiness and universality problems are decidable for L/U-PTAs without invariants and with integer-valued parameters. Recall that for L/U-PTAs in their classical form with invariants (even over integer-valued parameters), these same problems are undecidable [AL17]. L/U-PTAs enjoy a well-known monotonicity property recalled in the following lemma (that corresponds to a reformulation of [HRSV02, Prop 4.2]), stating that increasing upper-bound parameters or decreasing lower-bound parameters can only add behaviors. As our definition of L/U-PTAs does not involve invariants, our model is a subclass of L/U-PTAs as defined in [HRSV02, BL09]. Therefore, it holds for our definition of L/U-PTAs.

Lemma 21 (monotonicity)**.** *Let $\mathcal{A}$ be an L/U-PTA without invariant and v be a parameter valuation. Let v' be a valuation such that for each upper-bound parameter p^+, $v'(p^+) \geq v(p^+)$ and for each lower-bound parameter p^-, $v'(p^-) \leq v(p^-)$. Then any run of $v(\mathcal{A})$ is a run of $v'(\mathcal{A})$.*

We will see that EG-emptiness can be reduced to the following two problems. The first one is *cycle-existence* [AL17]: given a TA $v(\mathcal{A})$, is there at least one run of $v(\mathcal{A})$ with an infinite number of discrete transitions? Before introducing the second problem, we need to have a closer look at deadlocks: recall that a state is deadlocked when no discrete transition can be taken, even after elapsing some time. As we do not have invariants, it will be either a state with no outgoing edge, or a state in which each outgoing transition contains at least one constraint on any clock x of the form $x \triangleleft k$, where k is a constant, or $x \triangleleft p^+$, where p^+ is a parameter. Indeed, for any parameter valuation, it suffices to wait enough time until all such guards are disabled—and the state becomes deadlocked. Note that with invariants, like in the L/U-PTAs of [HRSV02], this would not be sufficient: a state containing an invariant $x \triangleleft k$ and a transition containing a constraint $x \triangleleft k$ is not a deadlocked state, as the transition is forced to be taken. Formally, given an L/U-PTA[1] $\mathcal{A} = (\Sigma, L, \mathbf{L}, l_0, \mathbb{X}, \mathbb{P}, \zeta)$, we define $L_D(\mathcal{A}) := \{l \in L \mid$ for all edges $(l, g, a, U, l') \in \zeta$, g contains at least one constraint on a clock x of the form $x \triangleleft k$, where $k \in \mathbb{N}$, or $x \triangleleft p^+$, where $p^+ \in \mathbb{P}\}$.[2]

Now, the second problem we need to distinguish is *deadlock-existence*: given a TA $v(\mathcal{A})$, is there at least one run of $v(\mathcal{A})$ that is deadlocked, *i. e.,* has no discrete successor (possibly after some delay)? As mentioned above, unlike the L/U-PTAs of [HRSV02], given an L/U-PTA $\mathcal{A}$, detecting deadlocks is equivalent in our L/U-PTAs without invariants to the *reachability* problem of a given location of $L_D(\mathcal{A})$. Let $v_{0/\infty}$ be the parameter valuation s.t. for each lower-bound parameter p^-, $v_{0/\infty}(p^-) = 0$ and for each upper-bound parameter p^+, $v_{0/\infty}(p^+) = \infty$.

Recall that EG G holds if either there is an infinite run staying in G, or there is a finite deadlocked run staying in G.

[1] Throughout this section, we do not use the labeling function $\mathbf{L}$.

[2] Observe that this definition also includes the locations with syntactically no outgoing edge at all.

Lemma 22. *Let $\mathcal{A}$ be an L/U-PTA without invariant. There is a deadlock in $v(\mathcal{A})$ for some parameter valuation v iff there is $l \in L_D(\mathcal{A})$ reachable in $v_{0/\infty}(\mathcal{A})$.*

Proof. $\Rightarrow$ Suppose $v(\mathcal{A})$ is deadlocked. There is a run in $v(\mathcal{A})$ ending in a state (l, w) with no possible outgoing transition. That means for all edges $(l, g, a, U, l') \in \zeta$, guard $v(g)$ is not satisfied by $w + d$, for all $d \geq 0$. In particular, let M be the maximal constant appearing in the guards of $v_{0/\infty}(\mathcal{A})$ plus one, then g is not satisfied for $w + M$. Yet, for that clock valuation, for sure, all simple constraints of the form $k \triangleleft x$ are satisfied, so this means that g must contain at least one constraint on a clock x of the form $x \triangleleft k$, where $k \in \mathbb{N}$ and $k < w(x) + M$, or $x \triangleleft p^+$, where $p^+ \in \mathbb{P}$ and $v(p^+) < w(x) + M$. Therefore, $l \in L_D(\mathcal{A})$.

Moreover as constraints in $v(\mathcal{A})$ are stronger than those in $v_{0/\infty}(\mathcal{A})$ (*i. e.,* for each lower-bound parameter p^-, $v_{0/\infty}(p^-) \leq v(p^-)$ and for each upper-bound parameter p^+, $v(p^+) \leq v_{0/\infty}(p^+)$), from Lemma 21 l is reachable along a run of $v_{0/\infty}(\mathcal{A})$.

$\Leftarrow$ Conversely, let $l \in L_D(\mathcal{A})$ and suppose there is a run of $v_{0/\infty}(\mathcal{A})$ reaching (l, w), for some clock valuation w. Let v be the parameter valuation, defined as in the proof of [HRSV02, Prop 4.4], such that (l, w) is also reachable in $v(\mathcal{A})$. That valuation assigns a finite value to upper bound parameters that we denote by μ.

Let $e = (l, g, a, U, l') \in \zeta$. For each constraint of the form $x \triangleleft k$ with $k \in \mathbb{N}$ in g, define $d_1 = \max(0, \max_x(k - w(x))) + 1$. Then, for all clocks x and for all $d \geq d_1$, $w(x) + d \triangleleft k$ is false. Similarly, for each constraint of the form $x \triangleleft p^+$ with p^+ an upper-bound parameter in g, define $d_2 = \max(0, \max_x(\mu - w(x))) + 1$. Then, for all clocks x and for all $d \geq d_2$, $w(x) + d \triangleleft v(p^+)$ is false. Let $d_0 = \max(d_1, d_2)$ then, by construction $(l, w + d_0)$ is a deadlocked state in $v(\mathcal{A})$. □

□

Consider now a TA without invariants $\mathcal{A}$, and a subset G of its locations. We build a TA $G^+(\mathcal{A})$ as follows: first remove all locations not in G and remove all transitions to and from those removed locations. Second, add self-loops to all locations in $L_D(\mathcal{A})$, with a guard that is true, and no reset.

Lemma 23. $\mathsf{EG}(G)$ *holds if and only if there exists an infinite run in $G^+(\mathcal{A})$.*

Proof. $\Rightarrow$ Suppose $\mathsf{EG}(G)$ holds. Then there is a maximal path in $\mathcal{A}$ that stays in G. If that path is infinite then, by construction it is still possible in $G^+(\mathcal{A})$. Otherwise, it is finite and therefore it is a deadlock. From Lemma 22, this means that some location in $G \cap L_D(\mathcal{A})$ is reachable in $\mathcal{A}$, by always staying in G. Consequently that location is still reachable in $G^+(\mathcal{A})$ and since it belongs to $L_D(\mathcal{A})$, it has a self-loop in $G^+(\mathcal{A})$, which implies that there is an infinite run there.

$\Leftarrow$ In the other direction, suppose that there is an infinite run in $G^+(\mathcal{A})$. Either the corresponding infinite path never uses any of the added self-loops and therefore it is possible as is in $\mathcal{A}$, which implies $\mathsf{EG}(G)$, or it goes through $L_D(\mathcal{A})$ at least once. The latter means that some location in

$L_D(\mathcal{A})$ is reachable in $\mathcal{A}$ by staying in G, and by Lemma 22, this implies that there exists a finite maximal path in $\mathcal{A}$, and finally that we have $\mathsf{EG}(G)$ in $\mathcal{A}$. □

□

Corollary 7. *The* EG*-emptiness and* EG*-universality problems are PSPACE-complete for integer-valued L/U-PTAs without invariants.*

Proof. PSPACE-hardness comes from the fact that an L/U-PTA that does not use parameters in guards is a TA and EG is PSPACE-hard for TAs [AD94].

Let $\mathcal{A}$ be an L/U-PTA and G a subset of its locations. Remark that the construction of Lemma 23 is independant of the constants in the guards, and hence can be done in the same way for a PTA, giving another PTA $G^+(\mathcal{A})$ such that, for all parameter valuations v, $G^+(v(\mathcal{A})) = v(G^+(\mathcal{A}))$. By Lemma 23, EG-emptiness (resp. EG-universality) then reduces to the emptiness (resp. universality) of the set of parameter valuations v such that $v(G^+(\mathcal{A}))$ has an infinite accepting path. We conclude by recalling that the latter problem can be solved in PSPACE for both emptiness and universality [BL09]. □

□

This result is important as it is the first non-trivial subclass of PTAs for which EG-universality (equivalent by negation to AF-emptiness) is decidable.

We already had the same complexity for EF-emptiness and EF-universality [HRSV02], and by negation we can get the other flat formulas of TCTL, both for universality and emptiness (*e. g.,* AF-emptiness is "not EG-universality"). It is also easy to see that all those results would hold for flat formulas using the "until" operator. Therefore we have:

Theorem 8. *Flat-TCTL-emptiness and flat-TCTL-universality are PSPACE-complete for integer-valued L/U-PTAs without invariant.*

Remark 3. *These results come without Flat-TCTL-synthesis. Indeed, suppose we can compute the set of parameters s.t. a Flat-TCTL formula is satisfied by an integer-valued L/U-PTAs without invariant, say* EF*, and check for the emptiness of its intersection with a set of equality constraints. Consider an integer-valued PTA $\mathcal{A}$ without invariants. For each parameter p of $\mathcal{A}$ that is used both as an upper-bound and as a lower-bound, syntactically replace its occurrences as an upper-bound (resp. lower-bound) by a new parameter p^+ (resp. p^-). We obtain an integer-valued L/U-PTAs without invariant $\mathcal{A}'$. By hypothesis, let S be the solution set of parameters valuations to the* EF*-synthesis problem for $\mathcal{A}'$. Let S' be the set of equality constraints $p^+ = p^-$. Therefore we can decide whether $S \cap S' = \emptyset$ and the* EF*-emptiness problem is decidable for integer-valued PTAs without invariants, in contradiction with the results of [BBLS15].*

5.5 Conclusion and perspectives

In this chapter, we solved the open problem of the nested TCTL-emptiness for U-PTAs, that implies the undecidability of the whole TCTL-emptiness problem for this subclass of L/U-PTAs. Note that our proof holds even for integer-valued parameters, and even without invariants. This is a reminder that the border between undecidability and decidability problems for L/U-PTAs and its

subclasses is quite thin. We used a reduction from a U-PTA to a two-counter machine using several gadgets to prove that a precise TCTL-emptiness problem is undecidable. Unlike PTAs and bounded PTAs, U-PTAs and bounded U-PTAs are incomparable, hence we had to verify whether the same reasoning was applicable when the parameter domain is bounded. For this purpose, we used another construction to reduce to a bounded U-PTA from a two-counter machine to prove that the same TCTL-emptiness problem is also undecidable.

Moreover, we proved that EG-emptiness and universality are PSPACE-complete for (unbounded) integer-valued L/U-PTAs without invariants. This result is particularly interesting as it was undecidable with invariants [AL17]. Using existing results, we have that flat TCTL-emptiness and universality are decidable for this class, and therefore for integer-valued U-PTAs without invariants, which contrasts with our undecidability result and shows that we are there again at the frontier of decidability.

Future work This work opens new perspectives: where exactly the undecidability starts (in particular whether EG and AF are decidable for U-PTAs with invariants or real-valued parameters, which remains open, see Table 5.1), whether our proofs in Sections 5.2 and 5.3 can be extended over bounded time, and whether the same results hold for L-PTAs (lower-bound PTAs).

Also, extending our decidability result in Theorem 8 while keeping decidability will be an interesting challenge.

Chapters 3 to 5 mainly focused on the theoretical study of formalisms related to TAs and PTAs. After studying PTAs themselves, in Chapter 6 we will find how an extension of PTAs can be successfully applied to the domain of security, especially in modeling attack and fault scenarios of organizations or infrastructures.

Chapter 6

Parametric analyses of attack-fault trees

This work is the result of a visit at Universiteit Twente, The Netherlands, in the Formal Methods and Tool team. This visit was partially supported by the PHC Van Gogh project PAMPAS.

6.1 Introduction

In the past few years, the range of security breaches in the security of organizations has become larger and larger. The process of unifying them by determining relations and consequences between separated events has become more difficult: how to relate the presence of solid oxygen in a helium tank in SpaceX rocket Falcon 9 to its the explosion during firing tests? What is the cost for the attacker and the damages caused to SpaceX manufacturing plants? One of the tools available to help structure risk assessments and security analyses is *attack trees*, recommended, *e. g.,* by NATO Research and Technology Organisation (RTO) [NAT08] and OWASP (Open Web Application Security Project) [MGK+13]. *Attack trees* [SSSW98] were formalized in [KMRS10] as a popular and convenient formalism for security analysis (see [KPS14] for a survey) and are inspired by *fault trees* [FMC09, RS15] a well-known formalism used in safety engineering. Bottom-up computation for a single parameter (*e. g.,* cost, probability or time of an attack), can be performed directly on attack trees [BKMS12]. Attack trees and fault trees are quite similar but differ on their gates and/or goals [BKMS12, KMRS14]. Both are constructed with leaves that model component and attack step failures or successes that propagate through the system via gates. While fault trees focus on safety properties, attack trees considerate skills, resources and risk appetite possessed by an attacker performing actions. *Attack-fault trees* (AFTs) [KS17] combine safety properties from fault trees and security conditions from attack trees; therefore gates of both fault trees and attack trees are used in this formalism.

Quantitative analysis of AFTs with multiple quantitative annotations on AFTs like cost, time, failure probabilities—which can functionally be dependent on each other—evaluates risks and helps to determine the most risky scenarios and therefore to select the most effective counter-measures.

6.1.1 Contribution

In this chapter, we study a more abstract version of the security problem, and we propose an approach to *synthesize* times and costs necessary to individual actions in order to perform a successful attack or individual failures causing the failure of the entire system. The global attack time and cost can then be expressed as a combination of the parametric unit costs. To this end, we propose a formalization of attack-fault trees using an *ad-hoc* extension of parametric timed automata called *parametric weighted timed automata* (PWTAs). PWTAs can be seen as a generalization of parametric timed automata (PTAs) [AHV93] and weighted/priced automata [BFH+01, ALP04] with only costs on transitions.

We implement our framework within the tool ATTop presented in [KSR+18], allowing to define AFTs in the Galileo format, and provide an automated translation into the IMITATOR input format [AFKS12].

As a proof of concept, we apply our framework to an attack tree of [KS17] and an original attack-fault tree. With the help of the parametric timed model checker IMITATOR, we are able to synthesize constraints in several dimensions; further we discuss induced possible attack and fault scenarios.

This enlarges the scope of quantitative analysis for AFTs by parameterizing multiple annotations on the AFT *at once* such as time, cost and damages and then compute for instance the optimal combination of parameter values for the attack to fail quickly while keeping damages to the system low.

6.1.2 Related work

Attack tree analysis has been studied through lattice theory [KMRS10], timed automata [KRS15, KS17, KSR+18], I/O-IMCs [KGS15, AGKS15], Bayesian networks [GIM15], Petri nets [DMCR06], stochastic games [ANP16, HKKS16], etc. UPPAAL has been used for model transformations in [SYR+17] and in [HV06] UML sequence diagrams are manually transformed into timed automata models. [KS17] especially tackles the problem of multiple complex risk metrics and attacker profiles, in a probabilistic and timed formalism that can be computed and analyzed using stochastical model-checking [RS14] and Uppaal SMC [DLL+15]. AFTs are modeled in the Galileo format and translated with the tool ATTop [KSR+18] into stochastic timed automata [DLL+11].

However, synthesis of multidimensional parameters (time, cost for the attacker, damages for the organization...) at once for fully timed systems is not treated in the previously cited works, and these works require testing one by one a set of possible attribute values for an AFT.

Besides, attack-defense trees are one of the most well-studied extensions of attack trees and new analysis methods are still developed [KMRS14, KW18].

In a completely different area, asynchronous hardware circuits' gates were translated into (parametric) timed automata in [CEFX09]; our translation of AFTs gates into PWTAs synchronized using parallel composition shares some similarities with that approach.

6.1.3 Outline

We recall attack-fault trees in Section 6.2. We then introduce the formalism of *parametric weighted timed automata* in Section 6.3. Our translation from AFTs

to PWTAs is given in Section 6.4. Then, we describe our implementation in Section 6.5 and report on experiments in Section 6.6. We conclude by discussing future works.

6.2 Attack-fault Trees

Attack-fault trees (AFTs) model how a safety or security goal can be refined into smaller sub-goals, represented as *gates*, until no further refinement is possible, represented as *leaves*. The leaves of the tree model are either basic component failures (BCF) or basic attack steps (BAS). Since subtrees can be shared in the literature (see *e. g.,* [KS17]), AFTs are actually directed acyclic graphs, rather than trees. In this work, we consider only trees without shared gates or leaves. Safety is compromised with the failure of a BCF, *i. e.,* without any outside spark action. Security is compromised when an outside attacker causes the activation of a BAS. Following the terminology of [KS17], in this work we write that a gate or a leaf is *disrupted* if the output is true *i. e.,* it succeeds, and *fails* otherwise. A success event (disruption) models the fact that a component (gate or leaf) is compromised *i. e.,* the attack is successful or the component fails. In contrast, a fail event models the robustness of the component against an attacker through a BAS, or a BCF.

6.2.1 AFT leaves

AFT leaves are equipped with an execution time and a rich cost structure that includes the cost incurred by an attacker and the damage inflicted on the organization. In contrast to [RS15, KS17] where BCF and BAS are equipped with probability distributions, we consider both BCF and BAS as parametric time-dependent events. This allows us to compute a range of cost values, damages values and time intervals at once in order to perform operations such as optimum time values for a counter-measure while keeping damage to the organization low, and cost for the attacker high.

6.2.2 AFT gates

In order to model complex scenarios with multiple leaves, BCF and BAS have to be composed. For this purpose, logical gates are used that output either the propagation of a disruption, or not. Gates take as an input either leaves or outputs from gates in their subtrees. Logical gates used in AFTs are taken from both dynamic fault trees and attack trees: AND, PAND, SAND, OR, SOR, FDEP, SPARE, VOT(k/n), depicted in Figure 6.1. These gates are the translatable ones in ATTop [KSR+18] from the Galileo format. We also added the XOR gate to improve our modeling capabilities.

AND gate propagates a disruption (*i. e.,*, it synchronizes a success event [KS17]) if all of its children are disrupted, regardless of the order of disruption. Children are activated initially by the AND. Children of a SAND gate are activated sequentially from left to right. After the success (disruption) of the leftmost child, the second left most child is activated, and so on until the disruption of rightmost child. If *all* children are disrupted, the SAND gate is disrupted. However, if any child fails (to be disrupted), the SAND gate directly fails. SAND gate is a

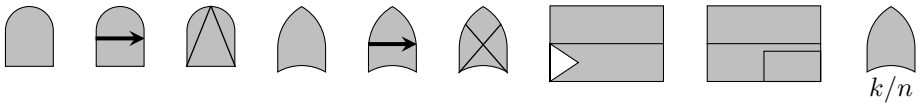


Figure 6.1: From left to right: AND, SAND, PAND, OR, SOR, XOR, FDEP, SPARE, VOT(k/n) gates

specific gate of attack trees. Compared with SAND gate, all children of a PAND gate are activated initially when the PAND gate is activated. The rest of the execution is similar to a SAND gate, and propagates a disruption if all children are disrupted from left to right (which in contrast is not mandatory for an AND gate), otherwise the PAND gate fails.

OR gate propagates a disruption if at least one of its children is disrupted. Children are activated initially by the OR gate. Similarly to a SAND gate, children of a SOR gate are activated sequentially after the termination of the previous one and from left to right. It propagates a disruption *when* one of its children is disrupted, otherwise if all children fail the SOR gate fails. XOR gate propagates a disruption if one of its children is disrupted and the other one fails.

FDEP (functional dependency) gate consists of a trigger event and several dependent events, and is a specific gate of fault trees. When the trigger event occurs, all its dependent BCF events are disrupted (*i. e.,* the failure of the power supply automatically deactivate the alarm and security cameras, therefore the BCFs are successful).

SPARE gate is similar to SAND, but is a specific gate for fault events while SAND gate is used for attack events. SPARE gate consists of one primary BCF and several secondary BCF which are activated sequentially. If the primary BCF is disrupted (*i. e.,* the component fails), a secondary becomes primary. If no BCFs are left (they all are disrupted), SPARE gate propagates a disruption.

VOT(k/n) gate is similar to OR gate and consists of $n \in \mathbb{N}$ children initially activated. VOT(k/n) gate is disrupted when k of its n children are disrupted.

6.3 Parametric weighted timed automata

First we introduce a slighlty different notation than for PTAs that will be used in this chapter.

Given $U \subseteq \mathbb{X}$, we define the *reset* of a valuation w, denoted by $[w]_U$, as follows: $[w]_U(x) = 0$ if $x \in U$, and $[w]_U(x) = w(x)$ otherwise.

We assume a set $\mathbb{TP} = \{p_1, \ldots, p_J\}$ of *timing parameters*. A *timing parameter valuation* tv is a function $tv : \mathbb{TP} \to \mathbb{Q}_+$. A *guard* g is a constraint over $\mathbb{X} \cup \mathbb{TP}$ defined by a conjunction of inequalities of the form $x \bowtie d$, or $x \bowtie p$ with $x \in \mathbb{X}$, $d \in \mathbb{N}$ and $p \in \mathbb{TP}$. Given g, we write $w \models tv(g)$ if the expression obtained by replacing each x with $w(x)$ and each p with $tv(p)$ in g evaluates to true.

We assume a set $\mathbb{W} = \{w_1, \ldots, w_M\}$ of *weights*. A *weight valuation* μ is a function $\mu : \mathbb{W} \to \mathbb{Q}$. We write $\vec{0}_{\mathbb{W}}$ for the weight valuation assigning 0 to all weights. We assume a set $\mathbb{WP} = \{q_1, \ldots, q_N\}$ of *weight parameters*, *i. e.,* unknown weight constants. A *weight parameter valuation* wv is a function $wv : \mathbb{WP} \to \mathbb{Q}$.[1]

[1]Observe that, in contrast to timing parameters that should be non-negative (which is

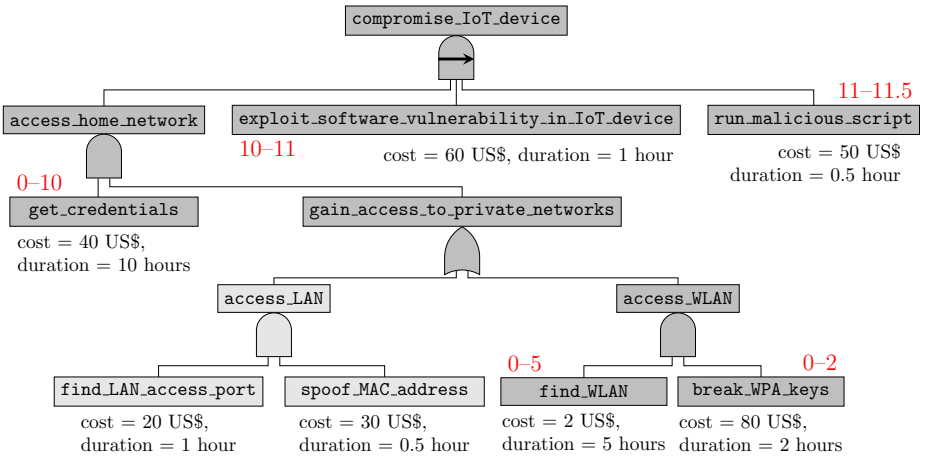


Figure 6.2: Attack Tree modeling the compromise of an IoT device from [SYR+17]. Leaves are equipped with the cost and time required to execute the corresponding step. The parts of the tree attacked in a successful attack are indicated by a darker color, with start and end times for the steps in this attack denoted in red.

A linear arithmetic expression over $\mathbb{W} \cup \mathbb{WP}$ is $\sum_i a_i w_i + \sum_j b_j q_j + c$, where $w_i \in \mathbb{W}$, $q_j \in \mathbb{WP}$ and $a_i, b_j, c \in \mathbb{Q}$. Let $\mathcal{LA}(\mathbb{W} \cup \mathbb{WP})$ denote the set of arithmetic expressions over $\mathbb{W}$ and $\mathbb{WP}$. A parametric weight update is a partial function $\alpha : \mathbb{W} \nrightarrow \mathcal{LA}(\mathbb{W} \cup \mathbb{WP})$. That is, we can assign a weight to an arithmetic expression of parametric weights and other weight values, and rational constants. Given a weight valuation μ, a parametric weight update α and a weight parameter valuation wv, we need an evaluation function $eval_{wv}(\alpha, \mu)$ returning a weight valuation, and defined as follows:

$$eval_{wv}(\alpha, \mu)(w) = \begin{cases} \mu(w) & \text{if } \alpha(w) \text{ is undefined} \\ \mu(wv(\alpha(w))) & \text{otherwise} \end{cases}$$

where $\mu(wv(\alpha(w)))$ denotes the replacement within the linear arithmetic expression $\alpha(w)$ of all occurrences of a weight parameter q_i by $wv(q_i)$, and of a weight variable w_j with its current value $\mu(w_j)$. Observe that this replacement gives a rational constant, therefore $eval_{wv}(\alpha, \mu)$ is indeed a weight valuation $\mathbb{W} \to \mathbb{Q}$. That is, $eval_{wv}(\alpha, \mu)$ computes the new (non-parametric) weight valuation obtained after applying to μ the partial function α valuated with wv.

We extend further PTA with (discrete) rational-valued *weight parameters*, giving birth to parametric weighted timed automata (PWTA).

Definition 19. *A parametric weighted timed automaton (PWTA) $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, L, l_0, F, \mathbb{X}, \mathbb{TP}, \mathbb{W}, \mathbb{WP}, I, \zeta)$, where:*

1. *Σ is a finite set of synchronization actions,*
2. *L is a finite set of locations,*

usual for *parametric timed automata*), our weight parameters may be negative.

3. $l_0 \in L$ *is the initial location,*

4. $F \subseteq L$ *is the set of accepting locations,*

5. $\mathbb{X}$ *is a finite set of clocks,*

6. $\mathbb{TP}$ *is a finite set of timing parameters,*

7. $\mathbb{W}$ *is a finite set of weights,*

8. $\mathbb{WP}$ *is a finite set of weight parameters,*

9. I *is the invariant, assigning to every* $l \in L$ *a guard* $I(l)$*,*

10. ζ *is a finite set of edges* $e = (l, g, a, U, \alpha, l')$ *where* $l, l' \in L$ *are the source and target locations,* g *is a guard,* $a \in \Sigma$*,* $U \subseteq \mathbb{X}$ *is a set of clocks to be reset, and* $\alpha : \mathbb{W} \nrightarrow \mathcal{LA}(\mathbb{W} \cup \mathbb{WP})$ *is a parametric weight update.*

Given a timing parameter valuation tv and a weight parameter valuation wv, we denote by $tv|wv(\mathcal{A})$ the non-parametric structure where all occurrences of a timing parameter p_i have been replaced by $tv(p_i)$, and all occurrences of a weight parameter q_j have been replaced by $wv(q_j)$. The resulting structure can be seen as an extension of a parametric weighted/priced timed automaton [BFH+01, ALP04] with only rational weights on edges.[2] However, our structure goes beyond a simple parametric extensions of weighted/priced timed automata, for two reasons:

1. we allow multiple weights;

2. we allow to not only *increment* weight values over a path, but also perform more complex operations on that weight, notably incrementing it *with another weight value*, which is clearly not possible in [BFH+01, ALP04].

Note that, if we restrict our parametric weight update function to expressions of the form $\alpha(w_i) = w_i + z$, where z is either a weight parameter or a rational constant, then our formalism is exactly the parametric extension of (the discrete "switch" weight part of) [BFH+01, ALP04].[3]

In addition, our formalism shares some similarities with the statically parametric timed automata of [Wan00], where timed automata are extended with parameters that can only be used in guards, but not compared to clocks. In contrast, our weight parameters can only be used in updates, and not in guards; in addition, we also feature the timing parameters of [AHV93] that can be compared to clocks.

Example 5. *In the PWTA in Figure 6.3, we have the following elements:* $L = \{l_1, l_2, l_3\}$*,* $l_0 = l_1$ *(also the unique element of* F*),* $\mathbb{X} = \{x, y\}$*, and* $\Sigma = \{\text{press}, \text{prepare}, \text{serve}\}$*, with the set* $\mathbb{TP} = \{p_1, p_2\}$ *and weights* $\mathbb{W} = \{w\}$*,* $\mathbb{WP} = \{q\}$*. There are four edges:*

[2] In [ALP04] cost is defined as the sum of each discrete cost on transitions (*switch* cost) plus the time spent in a location multiplied by an integer rate (*duration* cost), resulting in a rational value. Here, we omit the duration costs.

[3] Technically, as weighted/priced timed automata use integer constants, a rescaling of the constants is necessary: by multiplying all constants in $tv|wv(\mathcal{A})$ by the least common multiple of their denominators, we obtain an equivalent (integer-valued) weighted/priced timed automata.

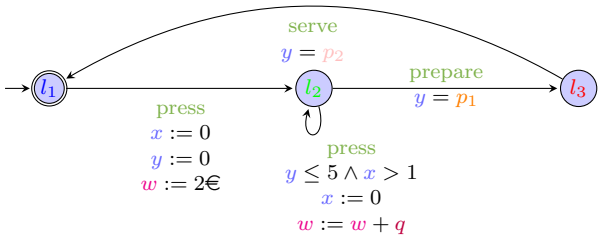


Figure 6.3: A PWTA modeling a coffee machine

- $e_1 = \langle l_1, g, a, U, l_2 \rangle$ *where* U *sets both* x, y *to* 0, α *is* $w = 2€$,
- $e_2 = \langle l_2, g, a, U, l_2 \rangle$ *where* g *is* $y \leq 5 \wedge x > 1$ *and* U *sets* x *to* 0, α *is* $w := w + q$,
- $e_3 = \langle l_2, g, a, U, l_3 \rangle$ *where* g *is* $y = p_1$ *and*
- $e_4 = \langle l_3, g, a, U, l_1 \rangle$ *where* g *is* $y = p_2$.

Let us now define the concrete semantics of PWTA as the union over all timing parameter and weight parameter valuations.

Definition 20 (Semantics of a valuated PWTA)**.** *Given a PWTA* $\mathcal{A} = (\Sigma, L, l_0, F, \mathbb{X}, \mathbb{TP}, \mathbb{W}, \mathbb{WP}, I, \zeta)$, *a timing parameter valuation* tv, *and a weight parameter valuation* wv, *the semantics of* $tv|wv(\mathcal{A})$ *is given by the TTS* $(S, s_0, \to)$, *with*

- $S = \{(l, w, \mu) \in L \times \mathbb{R}_+^H \times \mathbb{Q}^M \mid w \models tv(I(l))\}$,
- $s_0 = (l_0, \vec{0}, \vec{0}_{\mathbb{W}})$,
- $\to$ *consists of the discrete and (continuous) delay transition relations:*
  1. *discrete transitions:* $(l, w, \mu) \overset{e}{\mapsto} (l', w', \mu')$, *if* $(l, w, \mu), (l', w', \mu') \in S$, *and there exists* $e = (l, g, a, U, \alpha, l') \in \zeta$, *such that* $w \models tv(g)$, $w' = [w]_U$, *and* $\mu' = eval_{wv}(\alpha, \mu)(w)$;
  2. *delay transitions:* $(l, w, \mu) \overset{d}{\mapsto} (l, w + d, \mu)$, *with* $d \in \mathbb{R}_+$, *if* $\forall d' \in [0, d], (l, w + d', \mu) \in S$.

That is, a state is a triple made of the current location, the current (non-parametric) clock valuation, and the current (non-parametric) weight valuation. The clock valuations evolve naturally as in timed automata, while the current weight evolves according to the weight update function.

Moreover we write $(l, w, \mu) \xrightarrow{(e,d)} (l', w', \mu')$ for a combination of a delay and discrete transition if $\exists w'' : (l, w, \mu) \overset{d}{\mapsto} (l, w'', \mu) \overset{e}{\mapsto} (l', w', \mu')$. Given $tv|wv(\mathcal{A})$ with concrete semantics $(S, s_0, \to)$, we refer to the states of S as the *concrete states* of $tv|wv(\mathcal{A})$. A *run* of $tv|wv(\mathcal{A})$ is an alternating sequence of concrete states of $tv|wv(\mathcal{A})$ and pairs of edges and delays starting from the initial state s_0 of the form $s_0, (e_0, d_0), s_1, \cdots$ with $i = 0, 1, \ldots, e_i \in \zeta$, $d_i \in \mathbb{R}_+$ and $(s_i, e_i, s_{i+1}) \in \to$.

Example 6. *A concrete execution of the PWTA* $tv|wv(\mathcal{A})$ *of Example 5 with* $w = 2€$, $wv(q) = 0.5€$, $tv(p_1) = 5$ *and* $tv(p_2) = 8$ *is*

$(l_1, (0,0), (0)) \xrightarrow{(\text{press},2)} (l_2, (0,0), (2)) \xrightarrow{(\text{press},1.5)} (l_2, (0,1.5), (2.5)) \xrightarrow{(\text{press},1)} (l_2, (0,2.5), (3)) \xrightarrow{(\text{prepare},2.5)} (l_3, (2.5,5), (3)) \xrightarrow{(\text{serve},3)} (l_1, (5.5,8), (2.5))$.

Note that no coffee can be served if $tv(p_1) = 8$ *and* $tv(p_2) = 5$.

Remark 4. *Despite the name of* weights *(justified by our context of measuring costs and damages), our parametric weights are in fact sufficiently expressive to encode parametric (rational-valued) data.*

6.4 Translation of AFTs to PTAs

6.4.1 Overview of the translation

We will model an attack-fault tree using a network of PWTAs that will synchronize along actions (using the usual composition semantics). Each gate and each leaf (*i. e.,* BAS or BCF) will be modeled as a PWTA. Leaves PWTA have a duration and a weight, while gates PWTA store the weight value of their children to forward it to their parents. Therefore, each gate PWTA maintains its own weight, and its value will be added to that of their parents in case of success (thanks to the parametric weight update).

All gates and leaves PWTAs initially synchronize their start action—referred as *activation* in this work—, and end with either a success or fail synchronization action. After gates synchronize their start action, they synchronize the start action of their children.

Intuitively, the process is top-bottom-top: the top level gate PWTA activates its children, which themselves activate their children (if any), and so on until the leaves PWTAs at the bottom of the attack-falt tree. Once a leaf PWTA terminates, it synchronizes either its success or fail action. In case of success, the leaf PWTA forwards its weight value to its parent, where this value is stored. When its parent gate PWTA terminates, the gate PWTA synchronizes either a success or a fail action. In case of success, the gate PWTA forwards its weight value to its parent, and so on until the top-level gate PWTA terminates.

If the top-level PWTA terminates in its success location, the attack is successful. We apply the reachability synthesis algorithm of PTAs on the success location in the top-level PWTA, that is, we synthesize all valuations for which this location is reachable: this gives us the success conditions of an attack. The set of constraints on time and weight (such as cost for the attacker, damages for the organization) that allowed this attack to be successful are output by this analysis.

As a running example, we consider the attack tree in Figure 6.2 taken from [SYR+17].

6.4.2 Translation of leaves

A BAS/BCF is modeled as a PWTA with clocks and weights (see Figure 6.4). Note that in real life while a BAS needs to success so the attack is possibly successful, a BCF needs to fail in order to propagate a disruption (as in basic

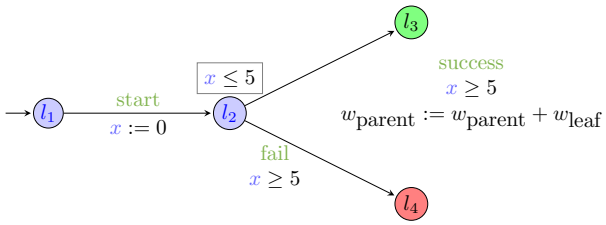


Figure 6.4: PWTA translation of leaf that can reach the success location in exactly 5 units of time, and of weight w_{leaf}

component *failure*). However we consider in our models that both BAS and BCF need to reach the success location.

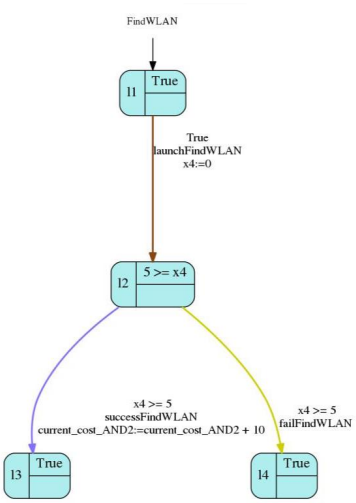


Figure 6.5: BAS translation of `find_WLAN`

There is two paths in a BAS/BCF PWTA, one that reaches the success location and one that reaches the fail location. In case of success, its weight is forwarded to and stored in its parent gate.

Example 7. *The translation of leaf* `find_WLAN` *of Figure 6.2 is given in Figure 6.5. To express the leaf* `find_WLAN` *we use four locations, one clock* $x4$*. The first step is to activate the basic attack step using the synchronization action* launchFindWLAN*. Once activated, and at most five units of time after (modeled by the invariant* $5 \geq x4$ *and the guard* $x4 \geq 5$*) it can either success with the action* successFindWLAN *or fail with the action* failFindWLAN*. If the success state is reached, the weight of its parent gate is increased by its own weight* 10*.*

6.4.3 Translation of gates

Concrete translations of SAND, AND, OR gates are given in Table 6.1 (yellow locations denote urgency: time cannot elapse). We describe them and give examples in the following. Other gates are similar.

AND recall that an AND gate is disrupted if all of its children are disrupted. It activates all of its children then waits for their disruptions regardless of the order of the successes. At any moment if one fails, the AND gate fails. If the success action is synchronized, its parent weight w_{parent} is updated: the weight w_{AND} carried by the AND gate is added to w_{parent}.

Example 8. *We give in Figure 6.6b the PWTA corresponding to the AND gate* `access_home_network` *of Figure 6.2. When all children are activated in the PWTA of Figure 6.6b, there are four paths leading to the fail state, while only two (success of the two children in any order) leading to the success state.* startAND3 *launches the AND gate* `access_home_network`*. Both children, the BAS* `get_credential` *and the OR gate* `gain_access_to_private_networks` *are*

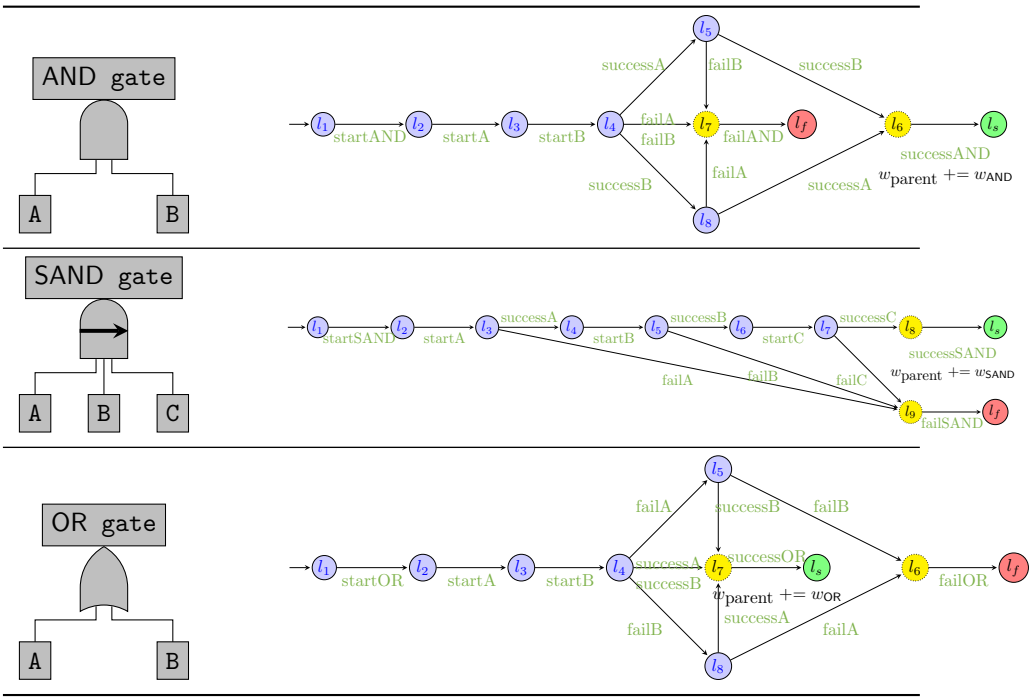


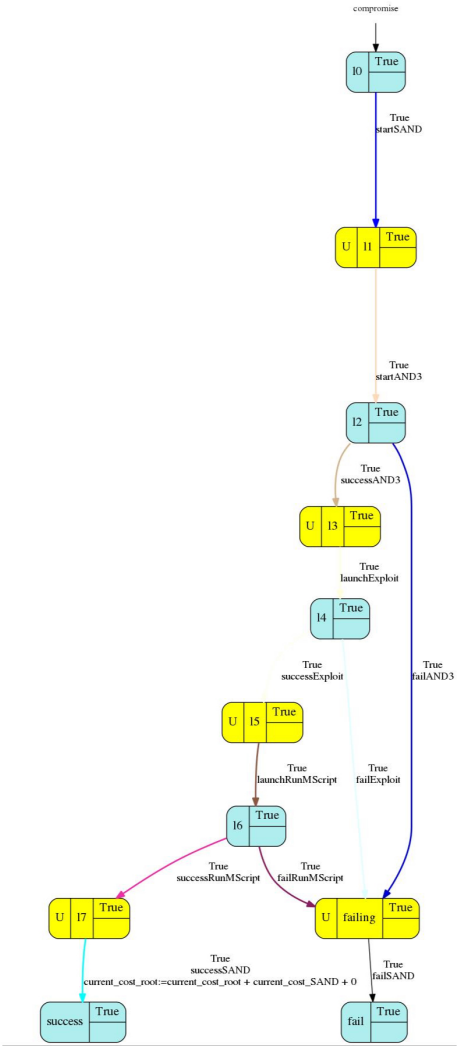
Table 6.1: Translation rules of AND, SAND and OR gates to PWTA

activated with the synchronization of the actions launchGetCred *and* startOR*. Unlike the SAND gate, AND gate waits for any of its child to synchronize a success action. If* successGetCred *is synchronized, it then will wait for* successOR *to go to the location success. If* failGetCred *is synchronized, the automaton will go to the location failing. When waiting for the action* successOR*, if* failOR *is synchronized the automaton will also go to the location failing. The other possibility (*successOR *then* successGetCred*) is similar. When in location failing it synchronizes the action* failAND3*, while if going to the location success it will synchronize the action* successAND3*. If the success state is reached, the weight of its parent gate is increased by its own weight.*

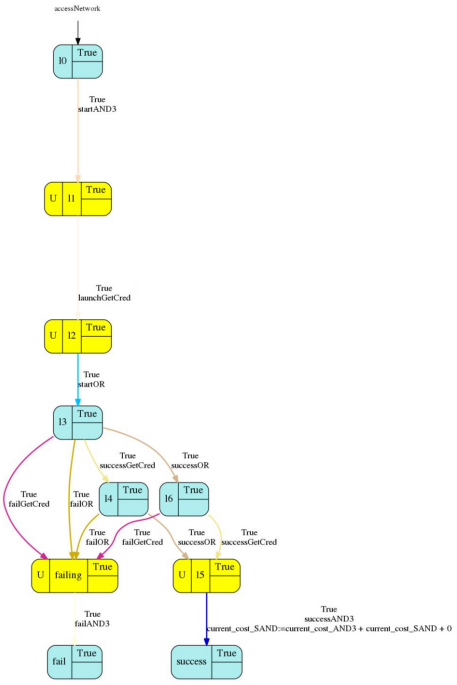
SAND recall that a SAND gate is disrupted if all its children from left to right are disrupted sequentially from left to right. It activates its leftmost child then waits for its success or failure, then activates its second leftmost child and so on. If the rightmost child succeeds, the SAND gate is disrupted. If one child fails, the SAND gate fails. For a SAND gate modeled as a PWTA with n children, there is only one path leading to the success state, while there are n paths leading to the fail state (one from each child). If the success action is synchronized, its parent weight w_{parent} is updated: the weight w_{SAND} carried by the SAND gate is added to w_{parent}.

Example 9. *The top event of the attack tree in Figure 6.2 is a SAND gate. We give the PWTA corresponding to this SAND in Figure 6.6a. It synchronizes the action* startSAND*. Then it activates its leftmost child* `access_home_network` *with the action* startAND3*, which is an AND gate. If the action* successAND3

is synchronized, its second leftmost child is activated with the action launchExploit. *If the action* successExploit *is synchronized, its third and last child is activated with the action* launchRunMScript. *If the action* successRunMScript *is synchronized, the action* successSAND *is synchronized. At any moment, if one of its children fail and an action* failAND3, failExploit *or* failRunMscript *is synchronized the automaton goes to the location failing where the action* failSAND *is synchronized. If the success state is reached, the weight of its parent gate is increased by its own weight.*



(a) Translation of the top-level SAND gate



(b) Translation of the AND gate `access_home_network`

Figure 6.6: SAND and AND gate

OR OR gate initially activates all of its children. OR gate is disrupted if at least one of its children is disrupted, and fails if all of its children fail. Therefore in the case of two children, one child can fail and the OR gate still propagates a disruption if the other one succeeds right after. However, if one child succeeds no need to wait for the second one and the success action of the OR gate is synchronized.

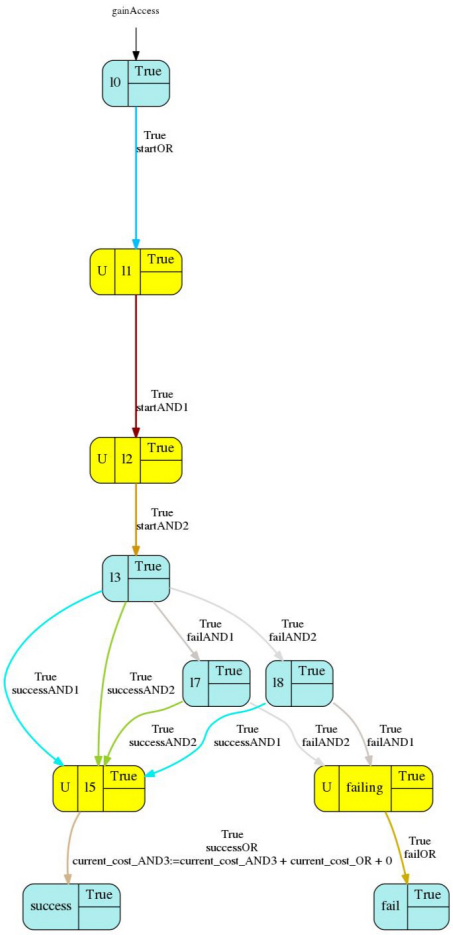


Figure 6.7: Translation of the OR gate `gain_access_to_private_networks`

If the success action is synchronized, its parent weight w_{parent} is updated: the weight w_{OR} carried by the OR gate is added to w_{parent}.

Example 10. *The PWTA translating the only OR of Figure 6.2 (given in Figure 6.7) activates all of its children, then waits for one to succeed, regardless of the order. Afterwards, whatever happens leads to the success state. If one child fails, then the other has to succeed, otherwise the OR fails. Therefore there is six possible paths to the success state, while there is two paths to the fail state (failure of both children in any order).* startOR *launches the OR gate* `gain_access_to_private_networks` *which activates its two children using the actions* startAND1 *and* startAND2 *which activates the AND* `access_LAN` *and AND* `access_WLAN`*. Only one action* successAND1 *or* successAN2 *is needed to be synchronized so the automaton goes to the location success regardless of which action is synchronized afterwards. Then it synchronizes the action* successOR*. If at first the action* failAND1 *(resp.* failAND2*) is synchronized, then* successAND2 *(resp.* successAND1*) has to be synchronized afterwards in order to reach the location success. Otherwise, if* failAND2 *(resp.* failAND1*) is synchronized, the automaton will go to the location failing and then synchronize the action* failOR*. If the success state is reached, the weight of its parent gate is increased by its own weight.*

6.4.4 Top-level automaton

Finally, we need to create an automaton that will activate the first top-event gate of the AFT. We call it rootTA. This PWTA is the one that starts the chain reaction by activating the top-event PWTA gate, which at its turn will activate

its own children and so on. It waits for the success or fail action of this PWTA gate. In case of success, its weight has been updated with the total weight value of the execution forwarded by the top-event gate PWTA.

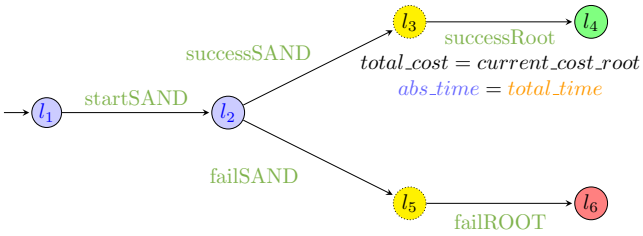


Figure 6.8: The rootTA

This bottom-to-top addition stores in the weight *current_cost_root* the total weight of the attack. The rootTA also stores the total time spent since the first activation of the top-event PWTA (using an extra clock and parameter).

Example 11. *We give in Figure 6.8 the top-level PWTA for the AFT in Figure 6.2. It is very similar to a leaf PWTA. It activates the top-level gate PWTA, then waits for its success or fail action. If the success action is synchronized, its weight has been updated to the total weight value of the execution and is checked against an additional parameter* total_cost *so IMITATOR outputs this* current_cost_root *value. Likewise, the clock abs_time which is never reset since the activation of rootTA is checked against a timing parameter total_time. Therefore IMITATOR outputs the total time of execution.*

6.5 Implementation of the translation

6.5.1 IMITATOR

IMITATOR [AFKS12] is a parametric model checker taking as input an extension of networks parametric timed automata extended with synchronization, stopwatches and discrete variables. IMITATOR supports global (shared) discrete rational-valued variables, that can be either *concrete* (in which case they are syntactic sugar for an unbounded number of locations), or *symbolic*, in which case they can be updated to or compared with parameters. While IMITATOR technically considers a single type of parameters (where symbolic variables can be compared or even updated to *timing* parameters), our weight parameters are never compared to timing parameters, and this setting can be considered as a subclass of the IMITATOR expressiveness.

IMITATOR implements several synthesis algorithms, notably reachability synthesis (EFsynth), that attempts to synthesize all parameter valuations for which a given location is reachable—which is the algorithm we use here. IMITATOR relies on the symbolic semantics of parametric timed automata (see *e. g.,* [JLR15]), where symbolic states are made of a discrete location, and a constraint over the clocks and parameters. The weight parameters are added to this symbolic semantics in a straightforward manner, with symbolic states enriched with linear constraints over weight parameters.

Note that, while parametric timed automata are highly undecidable (see [And19] for a survey), and while our parametric extension adds a new layer of complexity, all analyses terminate with an exact result (sound and complete) because our models are *acyclic*: our AFTs are trees, and their translation yields structurally acyclic PWTAs. As a consequence, the symbolic semantics of these PWTAs can be represented as a finite structure, and the analysis is guaranteed to terminate.

6.5.2 Translation from AFTs to PWTAs

I implemented the translation from AFTs to PWTAs (around 700 lines of code) within the framework of ATTop [SYR+17]. The existing software ATTop can take as input a Galileo formatted file. This format is pretty easy to use and to understand. The code in Figure 6.9 expresses an attack-fault tree of one OR gate named `A`, with two children `B` and `C`. The BAS `B` takes between 50 and 100 units of time to terminate, and costs \$50 to the attacker. The BAS `C` takes between 30 and 70 units of time to terminate, and costs \$30 to the attacker.

```
1 toplevel "A";
2 "A" or "B" "C";
3 "B" mintime=50 maxtime=100 cost=50;
4 "C" mintime=30 maxtime=70 cost=30;
```

Figure 6.9: Example of Galileo attack-tree

ATTop takes as an input a Galileo file and parses it to represent it as an attack-fault tree meta-model (ATMM) (see [SYR+17, Section 3], and Figure 6.10 for a screenshot of the tool).

Then, different translations are available: one quite interesting is the translation into an UPPAAL file, for instance a network of stochastic timed automata [KS17]. ATTop takes the ATMM and translates it in its UPPAAL

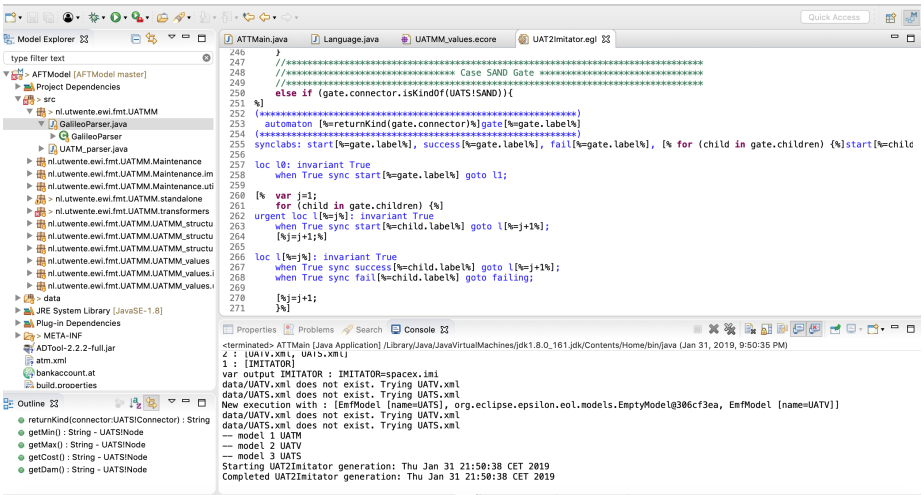


Figure 6.10: Screenshot of the tool ATTop after the translation of the SpaceX AFT

meta-model, then serializes it into an UPPAAL formatted file. In our approach we directly translate the representation of the ATMM into an IMITATOR formatted file, using the Epsilon Generation Language (EGL) [RPKP08]. This translation is a very efficient way to obtain AFTs modeled using PWTAs: designing manually a PWTA model from an AFT is very tedious to achieve, while defining an AFT within the Galileo syntax is simple.

Once the PWTA obtained, we synthesize using IMITATOR all parameter valuations for which the success location of the rootTA can be reached (using EFsynth). These sets of parameter values will help us to determine attack and fault scenarios in the following section.

6.6 Case studies

As a proof of concept, we apply our approach to an attack tree from the literature and an original attack-fault tree. Experiments were conducted with IMITATOR 2.10.4 "Butter Jellyfish",[4] on a 2,4 GHz Intel Core i5 processor with 2 GiB of RAM. Computation times of parameter values ranges from 1 to 9 seconds with four parameters.

6.6.1 Compromising an IoT device

We apply our approach to the AFT depicted in Figure 6.2 taken from [SYR+17]. We choose to parametrize the cost of finding a LAN access point ($CostFindLAN\_AP$) and the maximum amount of time to break WPA keys ($tMax\_Break$) of the AFT. This configuration will describe which attack (WLAN or LAN) is smarter for the attacker, depending on their resources: finding a LAN access point can be difficult depending on the infrastructure security and perhaps social engineering is needed. However, if the attacker does not have enough resources but a large amount of time (s)he can spend time trying to break WPA keys. IMITATOR computes several constraints on these parameters such that the attack is successful.

Different constraints are possible representing possible time and weight values s.t. an attack is possible. This is represented as a disjunction of conjunctions of constraints on parameters. For instance it can be a quick but very costly attack, or a long but cheap one; therefore different attack and fault scenarios appear. The conjunction of constraints

$$2 * tMax\_Break \geq 23 \wedge CostFindLAN\_AP \geq 0$$
$$\wedge CostFindLAN\_AP + 180 = total\_cost \wedge 2 * total\_time = 23$$

represents an attack that is very expensive for the attacker: indeed, the total cost of the attack is at least \$180 and fully depends on the cost of finding a LAN access point. However, the time spent on the attack is negligible and fixed (11.5h).

[4] Sources, binaries, models and results are available at https://www.imitator.fr/static/ACSD19PAT/

In opposition, the constraint

$$2 * tMax\_Break + 3 \geq 2 * total\_time$$
$$\wedge CostFindLAN\_AP \geq 0$$
$$\wedge 2 * total\_time \geq 23 \wedge total\_cost = 232$$

shows a an attack that will last at least 11.5h—that is, the attacker does not exactly knows when (s)he will break the WPA keys depending for instance of her/his computation power—but with a fixed cost of $232..

Contrarily to our initial intuition, the cost of this second attack can be high above the first one, as breaking the WPA keys is quite costly ($80) in opposition with finding a LAN access point. A smart attacker could choose, regardless of their time and resources the first attack through LAN access point.

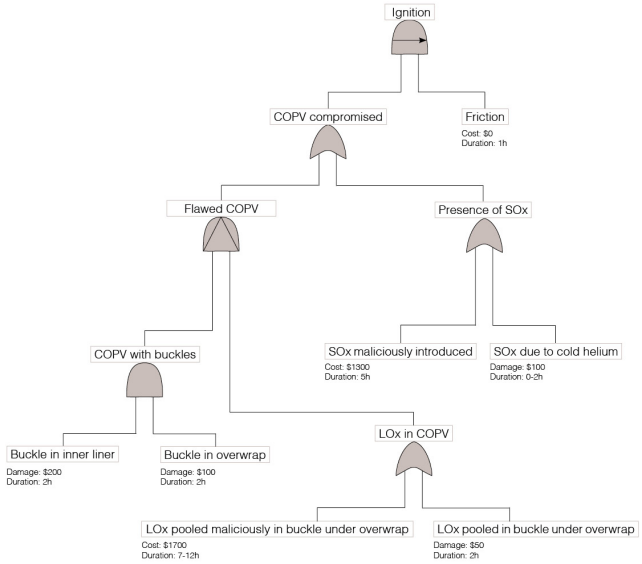


Figure 6.11: AFT of SpaceX rocket explosion

6.6.2 SpaceX rocket Falcon 9 explosion

Our second case study is an adaptation of the anomaly investigation that followed the explosion of SpaceX rocket Falcon 9 in september 2016[5]. The AFT in Figure 6.11 depicts the different configurations that can eventually end up with the explosion. The objective of this case-study is to show that the explosion is more likely to be accidental, due to the expensiveness of the BAS for the attacker who could attempt a sabotage.

The rocket carries a helium tank with three composite overwrapped pressure vessels (COPVs) inside. One COPV possibly had a manufacturing defect and buckles in its liner and the carbon overwrap (AND gate). Afterwards (PAND

[5]SpaceX anomaly update, https://www.spacex.com/news/2016/09/01/anomaly-updates

gate) liquid oxygen (LOx) can pool in these buckles and become trapped when pressurized under the carbon overwrap, resulting in a flawed COPV. An other possibility is the presence of solid oxygen (SOx) either due to the loading temperature of helium or placed here intentionally by an attacker (OR gate).

These two configurations result in a compromised COPV. When the COPV is compromised, a friction due to take-off tests can start the rocket ignition (SAND gate).

BCFs have a duration representing the time taken until the component failure. Damage is the cost for the organization for having built a defective component, or the cost induced when the component has failed. BASs have a cost for the attacker to perform the attack, and a duration for the attack to be successful.

We choose to parametrize the damages induced to the manufacturing facility by $damage\_BuckleInInnerLiner$, and the cost of pooling solid oxygen near the COPV, $cost\_SOXmaliciouslyIntroduced$.

The constraint

$$\begin{gathered} 13 \geq total\_time \geq 8 \\ \wedge cost\_SOXmaliciouslyIntroduced \geq 0 \\ \wedge total\_damages \geq 100 \\ \wedge damage\_BuckleInInnerLiner + 100 = total\_damages \\ \wedge total\_cost = 1700 \end{gathered}$$

represents the attack using the malicious introduction of LOx between the inner liner and the carbon overwrap of the COPV. Clearly this attack is very costly ($1700) and assumes the presence of these buckles. It is highly prejudicial to SpaceX as the company may want to investigate the manufacturing facility that produces COPV components.

The other attack, represented by the constraint

$$\begin{gathered} total\_time = 6 \\ \wedge cost\_SOXmaliciouslyIntroduced \geq 0 \\ \wedge total\_damages = 0 \\ \wedge damage\_BuckleInInnerLiner \geq 0 \\ \wedge total\_cost = cost\_SOXmaliciouslyIntroduced \end{gathered}$$

shows that the cost of the attack is equal to the cost of introducing SOx near the COPV. The higher is the parameter $cost\_SOXmaliciouslyIntroduced$, the higher is the cost of the attack. We may assume this cost is high enough as SpaceX surely secured its launch complex. Otherwise, an efficient counter-measure would be to find means to increase this cost for the attacker.

The constraint

$$\begin{gathered} cost\_SOXmaliciouslyIntroduced \geq 0 \\ \wedge total\_damage \geq 150 \\ \wedge damage\_BuckleInInnerLiner + 150 \geq total\_damage \\ \wedge total\_time = 3 \wedge total\_cost = 0 \end{gathered}$$

represents the fact that buckles in the inner liner and in the carbon overwrap of the COPV, and then LOx pooled under the overwrap, lead to a complete failure

of the system, *i. e.,* the rocket explodes. In this scenario, there is in all likelihood no attacker. However, the damages for the manufacturing facility can be huge if it is flawed: SpaceX should probably investigate in their manufacturing facilities in order to prevent the production of other flawed components.

Finally, the constraint

$$\begin{gathered} cost\_SOXmaliciouslyIntroduced \geq 0 \\ \wedge total\_damage = 100 \\ \wedge damage\_BuckleInInnerLiner \geq 0 \\ \wedge 3 \geq total\_time \geq 1 \wedge total\_cost = 0 \end{gathered}$$

shows that the explosion can be provoked by the presence of SOx due to cold helium. This case is possible without any attacker or component failure and is therefore fully accidental. No damages are caused to SpaceX (excepted the cost of the unusable rocket) or its suppliers.

These scenarios indicate that the rocket explosion is more likely to be accidental, as the cost in both scenarios where there is an attacker is very high. However, the worst case indicates that SpaceX should investigate their production lines to prevent other flawed components, as well as the presence of an attacker.

6.7 Conclusion

We addressed the problem of formalizing attack-fault trees in a more abstract framework allowing to cope with parametric timings, costs and damages. We defined and implemented a translation from attack-fault trees to PWTAs (a new extension of PTAs) that can be analyzed using the IMITATOR model-checker. This translation allows us to define easily an AFT using the Galileo syntax, and obtain as an output this AFT modeled with PWTAs. Using IMITATOR, we synthesize all parameter values such that there is a successful attack and/or a system failure. Finally, obtaining a disjunction of convex sets of parameter values allows us to define different attack and fault scenarios. Therefore it helps selecting the most plausible scenario and the most efficient counter-measures.

Future works In this work, we only considered three parameters: timing, cost and damage parameters. However, it is trivial to split these parameters into more precise ones, such as human damages (health and insurance) and material damages caused by the attacker or the failure of the system: an attack can be cheap for the attacker but inflict many kind of damages to the organization, as in our SpaceX case study. Thanks to the vector of weights defined in our PWTAs, this would be immediate to consider in our framework and implementation.

Moreover, extending our framework to attack-defense trees [KMRS14, GHL+16] is also on our agenda.

Finally, adding probabilities in order to create probabilistic parametric attack-fault trees will be an interesting and challenging future work. Indeed, in our SpaceX rocket case study adding probabilities to the manufacturing defects of the COPV on top of the damages inflicted to the company would strengthen considerably our formalism.

Chapter 7

Conclusion

In this thesis we mainly focused on developing new extensions and restrictions of a widely studied formalism, Timed Automata extended with parameters. We tried to determine whether classic TCTL properties are decidable for these extensions of PTAs and also if we can efficiently compute the set of parameters such that a given TCTL property evaluates to true. After this theoretical study, we also applied an extension of PTAs to cyber-security by demonstrating its usefulness in modeling attack and fault scenarios of IT infrastructures.

7.1 Summary

In Chapter 3 we defined a new extension of TAs, U2P-TA, which are TAs where parameters are allowed in updates. We proved that the EF, AF-emptiness and universality problems are undecidable when the parameters take rational values.

Nonetheless, when we consider only integer-valued parameters, we discovered that these problems become decidable, and even PSPACE-complete. Moreover, and unlike classical PTAs, EF, AF-emptiness and universality problems are decidable with unbounded integer values. We also can perform the synthesis of parameters for these problems. This result sharpens the thin border between decidability and undecidability results [ALR18b].

Following this lead and trying to investigate deeper the parametric updates, in Chapter 4 we focused on PTAs and also allowed parameters in updates. Our second extension, U2P-PTA allows parameters in guards and in updates. Its syntactic restriction R-U2P-PTA forces the update function to be *total*—*i. e.,* each clock is updated to a parameter or a constant—each time a clock is compared to a parameter in a guard. For R-U2P-PTA, we proved that EF-emptiness is PSPACE-complete for bounded rational-valued parameters in guards, but unbounded in parametric updates. Moreover we can compute EF-synthesis [ALR19].

In Chapter 5, we investigated L/U-PTAs and U-PTAs, without invariants. For L/U-PTAs without invariants we proved that EG-emptiness and -universality are PSPACE-complete for integer-valued parameters, completing

the exploration of decidability questions of flat TCTL for L/U-PTAs without invariants. This result is interesting as EG-emptiness is undecidable for classical L/U-PTAs [AL17]. We also answered an open question that is, non-flat TCTL is undecidable for U-PTAs without invariants by exhibiting a formula for which the emptiness problem is undecidable: it is the first time a problem decidable for TAs that is undecidable for U-PTAs is found. We proved this for bounded and unbounded integer valued parameters [ALR18a].

Finally in Chapter 6, we studied a parametric extension of attack-fault trees. We defined and implemented a translation from attack-fault trees to PWTAs that can be analyzed using the IMITATOR model-checker. This translation allows us to define easily an AFT using the Galileo syntax, and obtain as an output this AFT modeled with PWTAs using the tool ATTop. We can augment this model with parameters such as time and cost and we perform, using IMITATOR, the synthesis of all parameter values such that there is a successful attack and/or a system failure [ALRS19].

Therefore, this thesis can be summarized the following way:

- introduction of parametric updates in TAs and PTAs, which leads to several subclasses of PTAs for which, in some cases, the EF-emptiness problem is *decidable*;
- study of TCTL-emptiness decidability for L/U-PTAs and U-PTAs without invariants, sharpening the thin border between decidability and undecidability results.
- definition of a parametric extension of attack-fault trees, implementation of the translation from attack-fault trees in the Galileo syntax into PWTAs that can be analysed in IMITATOR.

7.2 Perspectives

As a follow-up to this work we will investigate hybrid systems when parameters are allowed. Indeed, hybrid systems [HKPV98, LPY99, BMRT04] represent a powerful formalism to model physical systems. Adding parameters along with strongly restrictions on the syntax, as done with initialised rectangular automata in [HKPV98] seems to be an interesting investigation: our intuition being, inspired by [HKPV98] where they transform an initialised rectangular automaton into a TA, to transform a parametric hybrid system into a R-U2P-PTA.

In [BDG+13], time-bounded reachability is proved decidable for a subclass of hybrid automata with monotonic (either non-negative or non-positive) rates: parametric timed automata can fit into this framework: clocks and parameters all have non-negative rates (1 for clocks, and 0 for parameters). To "initialize" parameters, one can initialize them to 0, let time elapse for an arbitrary amount of time (for each parameter), and then set their rate to 0 (while resetting all clocks). However, to compare clocks and parameters together in a hybrid automaton, one needs diagonal constraints—that are not allowed in [BDG+13]. As we showed that our undecidability results hold over bounded-time with a single parameter,

one can revisit the result of [BDG+13] as follows: allowing a single variable (our parameter) in diagonal constraints, with only one location with a non-zero rate for this variable (the initialization location for this parameter) renders the decidable problem of [BDG+13] undecidable.

Investigating O-minimal hybrid systems, where rates of variables, guards and updates are defined in a O-minimal structure, seemed also to be a nice opportunity. O-minimal structure allows one to express an infinite number of elements in a finite union of possibly infinite sets. Unfortunately, it happened that diagonal constraints where not allowed in [BMRT04], going back to the similar case discussed above.

Besides, we would like to investigate L-PTAs where parameters can be used only as lower bounds, and figure out whether the same properties we proved for U-PTAs are still satisfied. It is interesting also to study these subclasses of PTAs over bounded time.

Another research opportunity lies in security: we would like to study attack defense trees in which, besides probabilities we allow parameters to model cost constraints. Studying also parameterized extensions of non-interferent timed systems [GMR07, BCLR15], which are systems that communicate possibly important data through channels that are more or less safe, is an interesting future work.

Bibliography

[ABPvdP19] Étienne André, Vincent Bloemen, Laure Petrucci, and Jaco van de Pol. Minimal-time synthesis for parametric timed automata. In Tomáš Vojnar and Lijun Zhang, editors, *TACAS, Part II*, volume 11428 of *LNCS*, pages 211–228. Springer, April 2019.

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

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

[AFKS12] Étienne André, Laurent Fribourg, Ulrich Kühne, and Romain Soulat. IMITATOR 2.5: A tool for analyzing robustness in scheduling problems. In Dimitra Giannakopoulou and Dominique Méry, editors, *FM*, volume 7436 of *LNCS*, pages 33–36. Springer, August 2012.

[AGKS15] Florian Arnold, Dennis Guck, Rajesh Kumar, and Mariëlle Stoelinga. Sequential and parallel attack tree modelling. In Floor Koornneef and Coen van Gulijk, editors, *SAFECOMP*, volume 9338 of *LNCS*, pages 291–299. Springer, September 2015.

[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. ACM, May 1993.

[AK12] Étienne André and Ulrich Kühne. Parametric analysis of hybrid systems using HyMITATOR. In Franco Mazzanti and Gianluca Trentanni, editors, *iFM posters*, pages 16–19. CNR and ISTI, June 2012.

[AL17] Étienne André and Didier Lime. Liveness in L/U-parametric timed automata. In Alex Legay and Klaus Schneider, editors, *ACSD*, pages 9–18. IEEE, June 2017.

[ALP04] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. *Theoretical Computer Science*, 318(3):297–322, 2004.

[ALR16a] Étienne André, Didier Lime, and Olivier H. Roux. Decision problems for parametric timed automata. In Kazuhiro Ogata, Mark Lawford, and Shaoying Liu, editors, *ICFEM*, volume 10009 of *LNCS*, pages 400–416. Springer, November 2016.

[ALR16b] Étienne André, Didier Lime, and Olivier H. Roux. On the expressiveness of parametric timed automata. In Martin Fränzle and Nicolas Markey, editors, *FORMATS*, volume 9984 of *LNCS*, pages 19–34. Springer, August 2016.

[ALR18a] 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 *LNCS*, pages 37–52. Springer, September 2018.

[ALR18b] Étienne André, Didier Lime, and Mathias Ramparison. Timed automata with parametric updates. In *ACSD*, pages 21–29. IEEE Computer Society, June 2018.

[ALR19] Étienne André, Didier Lime, and Mathias Ramparison. Parametric updates in parametric timed automata. In Jorge A. Pérez and Nobuko Yoshida, editors, *FORTE*, volume 11535 of *LNCS*, pages 39–56. Springer, June 2019.

[ALRS19] Étienne André, Didier Lime, Mathias Ramparison, and Mariëlle Stoelinga. Parametric analyses of attack-fault trees. In *ACSD*, 2019. To appear.

[ALS+13] Étienne André, Yang Liu, Jun Sun, Jin Song Dong, and Shang-Wei Lin. PSyHCoS: Parameter synthesis for hierarchical concurrent real-time systems. In Natasha Sharygina and Helmut Veith, editors, *CAV*, volume 8044 of *LNCS*, pages 984–989. Springer, July 2013.

[ALSD14] Étienne André, Yang Liu, Jun Sun, and Jin Song Dong. Parameter synthesis for hierarchical concurrent real-time systems. *Real-Time Systems*, 50(5-6):620–679, 2014.

[And17] Étienne André. A unified formalism for monoprocessor schedulability analysis under uncertainty. In Laure Petrucci, Cristina Seceleanu, and Ana Cavalcanti, editors, *FMICS-AVoCS*, volume 10471 of *LNCS*, pages 100–115. Springer, September 2017.

[And19] Étienne André. What's decidable about parametric timed automata? *International Journal on Software Tools for Technology Transfer*, 21(2):203–2019, 2019.

[ANP16] Zaruhi Aslanyan, Flemming Nielson, and David Parker. Quantitative verification and synthesis of attack-defence scenarios. In *CSF*, pages 105–119. IEEE Computer Society, June 2016.

[BBLS15] Nikola Benes, Peter Bezdek, Kim Guldstrand Larsen, and Jirí 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 *LNCS*, pages 69–81. Springer, July 2015.

[BC13] Sandie Balaguer and Thomas Chatain. Avoiding shared clocks in networks of timed automata. *Logical Methods in Computer Science*, 9(4), 2013.

[BCLR15] Gilles Benattar, Franck Cassez, Didier Lime, and Olivier H. Roux. Control and synthesis of non-interferent timed systems. *International Journal of Control*, 88(2):217–236, 2015.

[BDFP04] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Updatable timed automata. *Theoretical Computer Science*, 321(2-3):291–345, 2004.

[BDG+13] Thomas Brihaye, Laurent Doyen, Gilles Geeraerts, Joël Ouaknine, Jean-François Raskin, and James Worrell. Time-bounded reachability for monotonic hybrid automata: Complexity and fixed points. In Dang Van Hung and Mizuhito Ogawa, editors, *ATVA*, volume 8172 of *LNCS*, pages 55–70. Springer, October 2013.

[BFH+01] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, and Frits W. Vaandrager. Minimum-cost reachability for priced timed automata. In Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli, editors, *HSCC*, volume 2034 of *LNCS*, pages 147–161. Springer, March 2001.

[BHJL13] Béatrice Bérard, Serge Haddad, Aleksandra Jovanovic, and Didier Lime. Parametric interrupt timed automata. In Parosh Aziz Abdulla and Igor Potapov, editors, *RP*, volume 8169 of *LNCS*, pages 59–69. Springer, September 2013.

[BHJL16] Béatrice Bérard, Serge Haddad, Aleksandra Jovanovic, and Didier Lime. Interrupt timed automata with auxiliary clocks and parameters. *Fundamenta Informaticae*, 143(3-4):235–259, 2016.

[BHP+15] Béatrice Bérard, Serge Haddad, Claudine Picaronny, Mohab Safey El Din, and Mathieu Sassolas. Polynomial interrupt timed automata. In Mikolaj Bojanczyk, Slawomir Lasota, and Igor Potapov, editors, *RP*, volume 9328 of *LNCS*, pages 20–32. Springer, September 2015.

[BKMS12] Alessandra Bagnato, Barbara Kordy, Per H:=akon Meland, and Patrick Schweitzer. Attribute decoration of attack-defense trees. *International Journal of Secure Software Engineering*, 3(2):1–35, 2012.

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

[BLL+95] Johan Bengtsson, Kim G. Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. UPPAAL — a Tool Suite for Automatic Verification of Real–Time Systems. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Proc. of Workshop on Verification and Control of Hybrid Systems III*, volume 1066 of *LNCS*, pages 232–243. Springer, October 1995.

[BMRS19] Damien Busatto-Gaston, Benjamin Monmege, Pierre-Alain Reynier, and Ocan Sankur. Robust controller synthesis in timed büchi automata: A symbolic approach. In Isil Dillig and Serdar Tasiran, editors, *CAV*, volume 11561 of *LNCS*, pages 572–590. Springer, July 2019.

[BMRT04] Thomas Brihaye, Christian Michaux, Cédric Rivière, and Christophe Troestler. On o-minimal hybrid systems. In Rajeev Alur and George J. Pappas, editors, *HSCC*, volume 2993 of *LNCS*, pages 219–233. Springer, March 2004.

[BMS13] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robustness in timed automata. In Parosh Aziz Abdulla and Igor Potapov, editors, *RP*, volume 8169 of *LNCS*, pages 1–18. Springer, September 2013.

[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 *LNCS*, pages 123–134. Springer, August 2014.

[BY03] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *ACPN*, volume 3098 of *LNCS*, pages 87–124. Springer, September 2003.

[CCGR00] Alessandro Cimatti, Edmund M. Clarke, Fausto Giunchiglia, and Marco Roveri. NUSMV: A new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.

[CEFX09] Rémy Chevallier, Emmanuelle Encrenaz-Tiphène, Laurent Fribourg, and Weiwen Xu. Timed verification of the generic architecture of a memory circuit using parametric timed automata. *Formal Methods in System Design*, 34(1):59–81, 2009.

[CEHM04] Ricardo Corin, Sandro Etalle, Pieter H. Hartel, and Angelika Mader. Timed model checking of security protocols. In Vijayalakshmi Atluri, Michael Backes, David A. Basin, and Michael Waidner, editors, *FMSE*, pages 23–32. ACM, October 2004.

[CL00] Franck Cassez and Kim Guldstrand Larsen. The impressive power of stopwatches. In Catuscia Palamidessi, editor, *CONCUR*, volume 1877 of *LNCS*, pages 138–152. Springer, August 2000.

[Cou12] Patrick Cousot. Formal verification by abstract interpretation. In Alwyn Goodloe and Suzette Person, editors, *NFM*, volume 7226 of *LNCS*, pages 3–7. Springer, April 2012.

[Dil89] David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In Joseph Sifakis, editor, *CAV*, volume 407 of *LNCS*, pages 197–212. Springer, June 1989.

[DLL+11] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikucionis, Danny Bøgsted Poulsen, Jonas van Vliet, and Zheng Wang. Statistical model checking for networks of priced timed automata. In Uli Fahrenberg and Stavros Tripakis, editors, *FORMATS*, volume 6919 of *LNCS*, pages 80–96. Springer, September 2011.

[DLL+15] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikucionis, and Danny Bøgsted Poulsen. Uppaal SMC tutorial. *STTT*, 17(4):397–415, 2015.

[DMCR06] G.C. Dalton, Robert Mills, John Colombi, and R.A. Raines. Analyzing attack trees using generalized stochastic Petri nets. In *2006 IEEE Information Assurance Workshop*, pages 116 – 123, June 2006.

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

[FK11] Laurent Fribourg and Ulrich Kühne. Parametric verification and test coverage for hybrid automata using the inverse method. In Giorgio Delzanno and Igor Potapov, editors, *RP*, volume 6945 of *LNCS*, pages 191–204. Springer, September 2011.

[FKPY07] Elena Fersman, Pavel Krcál, Paul Pettersson, and Wang Yi. Task automata: Schedulability, decidability and undecidability. *Information and Computation*, 205(8):1149–1172, 2007.

[FMC09] Igor Nai Fovino, Marcelo Masera, and Alessio De Cian. Integrating cyber attacks within fault trees. *Reliability Engineering & System Safety*, 94(9):1394–1402, 2009.

[Fre08] Goran Frehse. Phaver: algorithmic verification of hybrid systems past hytech. *STTT*, 10(3):263–279, 2008.

[GHL+16] Olga Gadyatskaya, René Rydhof Hansen, Kim Guldstrand Larsen, Axel Legay, Mads Chr. Olesen, and Danny Bøgsted Poulsen. Modelling attack-defense trees using timed automata. In Martin Fränzle and Nicolas Markey, editors, *FORMATS*, volume 9884 of *LNCS*, pages 35–50. Springer, August 2016.

[GIM15] Marco Gribaudo, Mauro Iacono, and Stefano Marrone. Exploiting bayesian networks for the analysis of combined attack trees. *Electronic Notes in Theoretical Computer Science*, 310:91–111, 2015.

[GLMR05] Guillaume Gardey, Didier Lime, Morgan Magnin, and Olivier H. Roux. Romeo: A tool for analyzing time petri nets. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *LNCS*, pages 418–423. Springer, July 2005.

[GMR07] Guillaume Gardey, John Mullins, and Olivier H. Roux. Non-interference control synthesis for security timed automata. *Electronic Notes in Theoretical Computer Science*, 180(1):35–53, 2007.

[HH14] Gérard P. Huet and Hugo Herbelin. 30 years of research and development around coq. In Suresh Jagannathan and Peter Sewell, editors, *POPL*, pages 249–250. ACM, January 2014.

[HKKS16] Holger Hermanns, Julia Krämer, Jan Krcál, and Mariëlle Stoelinga. The value of attack-defence diagrams. In Frank Piessens and Luca Viganò, editors, *POST*, volume 9635 of *LNCS*, pages 163–185. Springer, April 2016.

[HKPV98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, 1998.

[Hoa78] C. A. R. Hoare. Communicating sequential processes. *Communications of the Association for Computing Machinery (ACM)*, 21(8):666–677, 1978.

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

[HSTW16] Frédéric Herbreteau, B. Srivathsan, Thanh-Tung Tran, and Igor Walukiewicz. Why liveness for timed automata is hard, and what we can do about it. In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *FSTTCS*, volume 65 of *LIPIcs*, pages 48:1–48:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, December 2016.

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

[HV06] Martijn Hendriks and Marcel Verhoef. Timed automata based analysis of embedded system architectures. In *IPDPS*. IEEE, April 2006.

[JLR13] Aleksandra Jovanovic, Didier Lime, and Olivier H. Roux. Integer parameter synthesis for timed automata. In Nir Piterman and Scott A. Smolka, editors, *TACAS*, volume 7795 of *LNCS*, pages 401–415. Springer, March 2013.

[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.

[JP07] Gizela Jakubowska and Wojciech Penczek. Modelling and checking timed authentication of security protocols. *Fundamenta Informaticae*, 79(3-4):363–378, 2007.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984.

[KC10] E. V. Kuzmin and D. J. Chalyy. Decidability of boundedness problems for minsky counter machines. *Automatic Control and Computer Sciences*, 44(7):387–397, 2010.

[KGS15] Rajesh Kumar, Dennis Guck, and Mariëlle Stoelinga. Time dependent analysis with dynamic counter measure trees. *CoRR*, abs/1510.00050, 2015.

[KMRS10] Barbara Kordy, Sjouke Mauw, Sasa Radomirovic, and Patrick Schweitzer. Foundations of attack-defense trees. In Pierpaolo Degano, Sandro Etalle, and Joshua D. Guttman, editors, *FAST*, volume 6561 of *LNCS*, pages 80–95. Springer, September 2010.

[KMRS14] Barbara Kordy, Sjouke Mauw, Sasa Radomirovic, and Patrick Schweitzer. Attack-defense trees. *Journal of Logic and Computation*, 24(1):55–87, 2014.

[KP12a] Michal Knapik and Wojciech Penczek. Bounded model checking for parametric timed automata. *Transactions on Petri Nets and Other Models of Concurrency*, 5:141–159, 2012.

[KP12b] Michal Knapik and Wojciech Penczek. Smt-based parameter synthesis for L/U automata. In Lawrence Cabac, Michael Duvigneau, and Daniel Moldt, editors, *PNSE*, volume 851 of *CEUR Workshop Proceedings*, pages 77–92. CEUR-WS.org, June 2012.

[KP14] Michal Knapik and Wojciech Penczek. Parameter synthesis for timed kripke structures. *Fundamenta Informaticae*, 133(2-3):211–226, 2014.

[KPS14] Barbara Kordy, Ludovic Pièvre-Cambacédès, and Patrick Schweitzer. Dag-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, 13-14:1–38, 2014.

[KRS15] Rajesh Kumar, Enno Ruijters, and Mariëlle Stoelinga. Quantitative attack tree analysis via priced timed automata. In *FORMATS*, volume 9268 of *LNCS*, pages 156–171. Springer, September 2015.

[KS17] Rajesh Kumar and Mariëlle Stoelinga. Quantitative security and safety analysis with attack-fault trees. In *HASE*, pages 25–32. IEEE Computer Society, January 2017.

[KSR+18] Rajesh Kumar, Stefano Schivo, Enno Ruijters, Bugra Mehmet Yildiz, David Huistra, Jacco Brandt, Arend Rensink, and Mariëlle Stoelinga. Effective analysis of attack trees: A model-driven approach. In *FASE*, volume 10802 of *LNCS*, pages 56–73. Springer, April 2018.

[KW18] Barbara Kordy and Wojciech Widel. On quantitative analysis of attack-defense trees with repeated labels. In Lujo Bauer and Ralf Küsters, editors, *POST*, volume 10804 of *LNCS*, pages 325–346. Springer, April 2018.

[LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *STTT*, 1(1-2):134–152, 1997.

[LPY99] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. A new class of decidable hybrid systems. In *HSCC*, volume 1569 of *LNCS*, pages 137–151. Springer, March 1999.

[MGK+13] Marco Morana, Tobias Gondrom, Eoin Keary, Andy Lewis, Stephanie Tan, and Colin Watson. OWASP. CISO AppSec guide: Criteria for managing application security risks. Technical report, OWASP, 2013.

[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 *LNCS*, pages 296–309. Springer, March 2000.

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

[NAT08] NATO Research and Technology Organisation (RTO). Improving common security risk analysis. Technical Report AC/323(ISP-049)TP/193, , North Atlantic Treaty Organisation, University of California, Berkeley, 2008.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[NWY99] Christer Norström, Anders Wall, and Wang Yi. Timed automata as task models for event-driven systems. In *RTCSA*, pages 182–189. IEEE Computer Society, December 1999.

[QSW17] Karin Quaas, Mahsa Shirmohammadi, and James Worrell. Revisiting reachability in timed automata. In *LICS*, pages 1–12. IEEE Computer Society, June 2017.

[Qua14] Karin Quaas. MTL-model checking of one-clock parametric timed automata is undecidable. In Étienne André and Goran Frehse, editors, *SynCoP*, volume 145 of *EPTCS*, pages 5–17, April 2014.

[RPKP08] Louis M. Rose, Richard F. Paige, Dimitrios S. Kolovos, and Fiona Polack. The epsilon generation language. In Ina Schieferdecker and Alan Hartman, editors, *ECMDA-FA*, volume 5095 of *LNCS*, pages 1–16. Springer, June 2008.

[RS14] Anne Remke and Mariëlle Stoelinga, editors. *Stochastic Model Checking, ROCKS*, volume 8453 of *LNCS*. Springer, October 2014.

[RS15] Enno Ruijters and Mariëlle Stoelinga. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, 15:29–62, 2015.

[San15] Ocan Sankur. Symbolic quantitative robustness analysis of timed automata. In Christel Baier and Cesare Tinelli, editors, *TACAS*, volume 9035 of *LNCS*, pages 484–498. Springer, April 2015.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[SBM14] Ocan Sankur, Patricia Bouyer, and Nicolas Markey. Shrinking timed automata. *Inf. Comput.*, 234:107–132, 2014.

[SLDP09] Jun Sun, Yang Liu, Jin Song Dong, and Jun Pang. PAT: towards flexible verification under fairness. In Ahmed Bouajjani and Oded Maler, editors, *CAV*, volume 5643 of *LNCS*, pages 709–714. Springer, 2009.

[SSSW98] C. Salter, O. S. Saydjari, B. Schneier, and J. Wallner. Toward a secure system engineering methodology. In *NSPW*, pages 2–10, September 1998.

[SYR+17] Stefano Schivo, Bugra M. Yildiz, Enno Ruijters, Christopher Gerking, Rajesh Kumar, Stefan Dziwok, Arend Rensink, and Mariëlle Stoelinga. How to efficiently build a front-end tool for UPPAAL: A model-driven approach. In *SETTA*, volume 10606 of *LNCS*, pages 319–336. Springer, October 2017.

[TLR09] Louis-Marie Traonouez, Didier Lime, and Olivier H. Roux. Parametric model-checking of stopwatch petri nets. *Journal of Universal Computer Science*, 15(17):3273–3304, 2009.

[VP99] Irina Virbitskaite and E. Pokozy. Parametric behaviour analysis for time petri nets. In Victor E. Malyshkin, editor, *PaCT*, volume 1662 of *LNCS*, pages 134–140. Springer, September 1999.

[Wan00] Farn Wang. Parametric analysis of computer systems. *Formal Methods in System Design*, 17(1):39–60, 2000.