Int J Softw Tools Technol Transfer (2019) 21:203–219
https://doi.org/10.1007/s10009-017-0467-0

REGULAR PAPER

What's decidable about parametric timed automata?

Étienne André[1]

Published online: 29 July 2017

Abstract Parametric timed automata (PTAs) are a powerful formalism to reason, simulate and formally verify critical real-time systems. After 25 years of research on PTAs, it is now well understood that any non-trivial problem studied is undecidable for general PTAs. We provide here a survey of decision and computation problems for PTAs. On the one hand, bounding time, bounding the number of parameters or the domain of the parameters does not (in general) lead to any decidability. On the other hand, restricting the number of clocks, the use of clocks (compared or not with the parameters), and the use of parameters (e. g., used only as upper or lower bounds) leads to decidability of some problems. We also put emphasis on open problems. We also discuss formalisms close to parametric timed automata (such as parametric hybrid automata or parametric interrupt timed automata), and we study tools dedicated to PTAs and their extensions.

Keywords Decidability · Decision problems · Parametric timed model checking · Parameter synthesis · L/U-PTAs · Hybrid automata

1 Introduction

The absence of undesired behaviors in real-time critical systems is of utmost importance in order to ensure the system safety. Model checking aims at formally verifying a model of the system against a correctness property. Timed automata (TAs) are a popular formalism to model and verify safety critical systems with timing constraints. TAs extend finite state automata with clocks, i. e., real-valued variables increasing linearly [5]. These clocks can be compared with integer constants in guards (sets of linear inequalities that must be satisfied to take a transition) and invariants (sets of linear inequalities that must be satisfied to remain in a location). TAs have been widely studied (see e. g., [8]), and several state-of-the-art model checkers (such as UPPAAL [57] or PAT [68]) support TAs as an input language.

TAs benefit from many interesting decidable properties, such as the emptiness of the accepted language, the reachability of a control state. Other problems are undecidable though, such as the universality of the accepted timed language; in addition, given a TA, building a TA recognizing the complement of the timed language of the first TA cannot be achieved in general. TAs were also studied in a robust version, i. e., when all timing guards can be enlarged or shrinked by an infinitesimal constant factor, without changing the language, the reachability of a control state, etc. (see [32,59] for surveys).

However, TAs also suffer from some limitations. First, they cannot be used to specify and verify systems incompletely specified (i. e., whose timing constants are not known yet), and hence cannot be used in early design phases. Second, verifying a system for a *set* of timing constants usually requires to enumerate all of them one by one if they are supposed to be integer valued; in addition, TAs cannot be used anymore to verify a system for a set of timing constants that are to be taken in a rational- or real-valued dense interval. Third, robustness in TAs often assumes that all guards can be enlarged or shrinked by the same small variation; consid-

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

✉ Étienne André
eandre93430@lipn13.fr

[1] Université Paris 13, LIPN, CNRS, UMR 7030, F-93430 Villetaneuse, France

ering independent variations or considering both enlarging and shrinking was not addressed.

Parametric timed automata (PTAs) overcome these limitations by allowing the use of parameters (i.e., unknown constants) in guards and invariants [7]. This increased expressive power comes at the price of the undecidability of most interesting problems—at least in the general case.

In this paper, we consider decision problems for PTAs proposed in the past 25 years. On the one hand, bounding time, bounding the number of parameters or the domain of the parameters does not (in general) lead to any decidability. On the other hand, restricting the number of clocks, the use of clocks (compared or not with the parameters), and the use of parameters (e.g., used only as upper or lower bounds) can lead to the decidability of some problems. In addition, an extension to parameters of some variants of timed automata benefit from some decidability results, such as reset-PTAs and parametric interrupt timed automata.

1.1 Related surveys

To the best of our knowledge, no survey was dedicated specifically to decision problems for PTAs. Moreover, in addition to numerous results in the past 25 years proved in various settings with different syntax and assumptions, recent results in the field in the past 3 years justify the need for a clear picture of these updated (un)decidability results. Furthermore, surveying decision problems for PTAs has important practical implications as, for undecidable decision problems, the associated synthesis problems cannot be solved exactly.

Related works include [8] that studies decidability results of timed automata. In [32,59], various problems related to the robustness in TAs are studied. Then, [48] is not a survey, but exhibits decidable subclasses of hybrid automata, an extension of timed automata where variables can have (in general) arbitrary rates. Then, [23] acts both as a survey and as a contribution paper that studies hybrid automata with "low dimensions", i.e., with few variables. Our survey is also concerned (in Sect. 4) with decidability results for PTAs with few variables (i.e., clocks and parameters).

1.2 About this manuscript

This manuscript is a revised and extended version of [10]. New results unpublished at the time of [10] were added. Moreover, Table 2 was improved, and its description and summary was significantly enhanced. In addition, two new sections were added: formalisms beyond PTAs are studied in Sect. 6, and tools and applications of PTAs are reviewed in Sect. 7.

1.3 Outline

In Sect. 2, we propose a unified syntax for PTAs, and we define the decision problems that we will consider throughout this manuscript. In Sect. 3, we recall general undecidability results for PTAs. We then study in Sect. 4 the decidability when restricting the syntax of PTAs (number of variables, syntax of the constraints, etc.). We consider specifically in Sect. 5 the subclass of PTAs where parameters must be used either always as lower bounds or always as upper bounds, namely L/U-PTAs. Formalisms beyond PTAs, including parametric versions of hybrid automata, interrupt timed automata and time Petri nets, are studied in Sect. 6. Tools supporting PTAs and known applications of PTAs are reviewed in Sect. 7. We conclude by emphasizing open problems in Sect. 8.

2 Parametric timed automata and problems

2.1 Clocks, parameters and constraints

Let $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{Q}^+$ and $\mathbb{R}^+$ denote the sets of (possibly negative) integer numbers, (nonnegative) natural numbers, nonnegative rational numbers, and nonnegative real numbers, respectively. In the following, $\mathbb{T}$ denotes the domain of time, and $\mathbb{P}$ the domain of the parameters; these domains will be instantiated with $\mathbb{N}$, $\mathbb{Q}^+$ or $\mathbb{R}^+$ subsequently. Throughout this survey, let d denote an integer constant in $\mathbb{Z}$, and d^+ denote a nonnegative constant in $\mathbb{N}$.

Let us assume a set $X = \{x_1, \ldots, x_H\}$ of *clocks*, that are $\mathbb{T}$-valued variables that evolve at the same rate. Let us assume a set $P = \{p_1, \ldots, p_M\}$ of *parameters*, i.e., unknown constants. A parameter *valuation* v is a function $v : P \to \mathbb{P}$. Throughout this survey, symbols x, x_i denote clocks, whereas p, p_i denote parameters.

A *parametric linear term* is $\sum_{1 \leq i \leq M} \alpha_i p_i + d$, with $\alpha_i \in \mathbb{Z}$; in the following *plt* will denote a parametric linear term.

A *(linear) inequality* is $x \bowtie plt$, where x is a clock, *plt* a parametric linear term, and $\bowtie \in \{<, \leq, \geq, >\}$. We give in Table 1 the conventions used throughout this survey concerning comparison operators. A *(linear) constraint* is a conjunction of linear inequalities. A *(linear) diagonal constraint* is a conjunction of either linear inequalities, or *linear diagonal inequalities* of the form $x_i - x_j \bowtie plt$.

A *simple inequality* is either $x \bowtie p$ or $x \bowtie d^+$. A *simple constraint* is a conjunction of simple inequalities.

2.2 A unified syntax for parametric timed automata

The syntax of PTAs varies a lot in the literature; we give below a definition that should include most definitions in the literature, and at least all definitions of the papers con-

Table 1 Syntax of operators in guards

Operator	Definition
$\bowtie$	$\{<, \leq, \geq, >\}$
$\leq\geq$	$\{\leq, \geq\}$
$<>$	$\{<, >\}$
$\trianglelefteq$	$\{<, \leq\}$

sidered in this survey. That is, any definition of PTAs can be obtained from the following one by adding restrictions such as removing the set of accepting locations, forbidding invariants, restricting the domain of clocks or parameters, simplifying the syntax of the guards and invariants (e. g., forbidding diagonal constraints).

Definition 1 (*Parametric timed automaton*) A *parametric timed automaton (PTA)* is a tuple $\mathsf{A} = (\Sigma, L, l_0, F, X, P, I, E)$, where:

- Σ is a finite set of actions,
- L is a finite set of locations,
- $l_0 \in L$ is the initial location,
- $F \subseteq L$ is a set of accepting (or final) locations,
- X is a set of clocks with domain $\mathbb{T} = \mathbb{R}^+$,
- P is a set of parameters with domain $\mathbb{P} = \mathbb{R}^+$,
- I is the invariant, assigning to every $l \in L$ a diagonal constraint $I(l)$, and
- E is a set of edges (l, g, a, R, l') where $l, l' \in L$ are the source and destination locations, g is a diagonal constraint which is the transition guard, $a \in \Sigma$, and $R \subseteq X$ is a set of clocks to be reset.

Given a PTA A and a parameter valuation v, the *valuation* of A with v, denoted by $v(\mathsf{A})$, is the nonparametric PTA where each occurrence of p is replaced with $v(p)$. If v assigns an integer (or rational) value to each parameter, then $v(\mathsf{A})$ is a TA. However, if some parameters are assigned to an irrational value, then $v(\mathsf{A})$ belongs to the class of TAs with irrational constants, for which the reachability of a given location is undecidable [61].

A clock is said to be a *parametric clock* if it is compared with at least one parameter in at least one guard or invariant; otherwise, it is a *nonparametric clock*. This notion is central when studying the decidability of problems for PTAs with few clocks and parameters.

Example 1 Consider the coffee machine in Fig. 1, modeled using a PTA with 4 locations, 2 clocks (x_1 and x_2) and 3 parameters (p_1, p_2, p_3). Both clocks x_1 and x_2 are parametric clocks. No diagonal constraints are used (in fact all constraints are simple constraints). The machine can initially idle for an arbitrarily long time. Then, whenever the user presses the (unique) button (action press), the PTA enters location "add sugar", resetting both clocks. The machine can remain in this location as long as the invariant ($x_2 \leq p_2$) is satisfied; there, the user can add a dose of sugar by pressing the button (action press), provided the guard ($x_1 \geq p_1$) 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 p_1. Then, p_2 time units after the machine left the idle mode, a cup is delivered (action cup), and the coffee is being prepared; eventually, 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.

2.2.1 Semantics

The semantics of a PTA A can be defined as the union over all parameter valuations v of the semantics of $v(\mathsf{A})$. In the following, given $\delta \in \mathbb{R}^+$, $w + \delta$ denotes the valuation such that $(w+\delta)(x) = w(x)+\delta$, for all $x \in X$. Given $R \subseteq X$, we define the *reset* of a clock valuation w, denoted by $[w]_R$, as the valuation resetting the clocks in R, and keeping the other clocks unchanged. Given a rational parameter valuation v, $v(C)$ denotes the constraint over X obtained by replacing each parameter p in C with $v(p)$. Likewise, given a clock valuation w, $w(v(C))$ denotes the expression obtained by replacing each clock x in $v(C)$ with $w(x)$. We use the notation

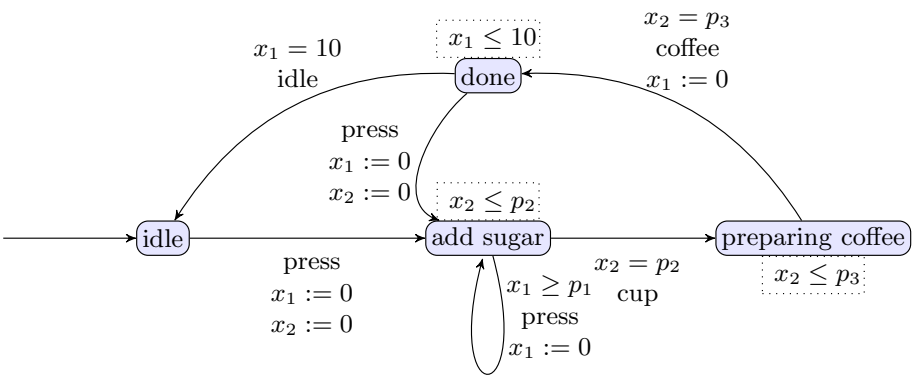

Fig. 1 A coffee machine modeled using a PTA

$w|v \models C$ to indicate that $w(v(C))$ evaluates to true. We write $\mathbf{0}$ for the clock valuation that assigns 0 to all clocks.

Definition 2 (*Concrete semantics of a TA*) Given a PTA $\mathsf{A} = (\Sigma, L, l_0, F, X, P, I, E)$, and a rational parameter valuation v, the concrete semantics of $v(\mathsf{A})$ is given by the timed transition system $(S, s_0, \rightarrow)$, with

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

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

Given a TA $v(\mathsf{A})$ with concrete semantics $(S, s_0, \rightarrow)$, we refer to the states of S as the *concrete states* of $v(\mathsf{A})$. A concrete run of $v(\mathsf{A})$ is an alternating sequence of concrete states of $v(\mathsf{A})$ and edges starting from the initial concrete state s_0 of the form $s_0 \overset{e_0}{\mapsto} s_1 \overset{e_1}{\mapsto} \cdots \overset{e_{m-1}}{\mapsto} s_m$, such that for all $i = 0, \ldots, m-1$, $e_i \in E$, $s_{i+1} \in S$, and $(s_i, e_i, s_{i+1}) \in \mapsto$. Given a concrete state $s = (l, w)$, we say that s is reachable (or that $v(\mathsf{A})$ reaches s) if s belongs to a concrete run of $v(\mathsf{A})$. By extension, we say that l is reachable in $v(\mathsf{A})$. A run is *maximal* if it is either infinite, or cannot be extended by any discrete transition (possibly after some delay transition).

A finite run $s_0 \overset{e_0}{\mapsto} s_1 \overset{e_1}{\mapsto} \cdots \overset{e_{m-1}}{\mapsto} (l_m, w_m)$ is *accepting* if $l_m \in F$.

The accepted timed language is the set of timed words (alternating sequences of actions and time elapsing) associated with an accepting run, i. e., a run ending in a location of F (or, in some works, passing infinitely often by a location in F). Note that some works make a difference between finite and infinite runs. The untimed language of a TA is the timed language projected onto the actions. The set of traces (or trace set) is the set of accepting runs projected onto the locations and actions, i. e., a set of alternating locations and actions. This is a nonstandard definition of traces (compared to e. g., [45]), but we keep this term as it is used in, e. g., [11,21].

A *symbolic semantics* is also defined for PTAs in [11,52,53] as a parametric zone graph, where a symbolic state is made of a discrete part (the current location) and a symbolic, continuous part (a set of diagonal constraints, i. e., $x_i - x_j \bowtie plt$, sometimes allowing disjunctions).

2.2.2 Simple PTAs

We define *simple PTAs* as the subclass of PTAs where guards and invariants are simple constraints. We propose this class to show that, even in this restricted situation, all non-trivial problems are undecidable (Sect. 3).

2.2.3 Variants of the PTA syntax

PTAs were first defined in [7] using a set of accepting locations. This is similar to timed automata [5]. *Timed safety automata* were introduced later in [50] by removing the final locations, but adding invariants to locations; many subsequent papers then refer to timed safety automata as simply "timed automata". When timed automata with accepting locations are equipped with Büchi conditions (to be accepting, an infinite timed word must pass infinitely often through at least one of the accepting locations), they are referred to as timed Büchi automata. It was shown that the timed expressive power of timed safety automata is strictly less than that of timed Büchi automata [49].

The syntax of PTAs differs in most of the papers in the literature. Concerning guards and invariants, in [7] (resp. [61]), guards (resp. guards and invariants) are conjunctions of inequalities of the form $x \bowtie p$. In [33,52], guards are conjunctions of inequalities of the form $x_i - x_j \trianglelefteq plt \cup \{\infty\}$; in [52] invariants have the same form as guards (invariants are not considered in [33]). In [11], any linear constraint over $X \cup P$ is allowed in guards and invariants. In [42], guards and invariants are all open, i. e., of the form $x <> p$ or $x <> d^+$. In [17,53], guards and invariants are conjunctions of inequalities of the form $x \bowtie plt$; in addition, in [53] invariants can only bound clocks from above (i. e., $x \trianglelefteq plt$). In [26], guards are conjunctions of inequalities of the form $x \bowtie p$ and invariants can only bound clocks from above (i. e., $x \trianglelefteq p$). In [21], guards and invariants are conjunctions of inequalities of the form $x \bowtie p + d$, $x \bowtie d^+$ or $p \bowtie d$ (although the proofs of undecidability only need inequalities of the form $x \bowtie p$ or $x \bowtie d^+$). In [15,16], guards and invariants are conjunction of simple inequalities.

A set of accepting locations is considered in [7,17,26,33], but only [33] is interested in infinite accepting runs, i. e., runs that pass infinitely often by an accepting location; hence, this latter work considers what could be referred to as parametric timed Büchi automata. In contrast, [11,16,21,42,52,53,61] consider parametric timed safety automata (i. e., without accepting locations).

Remark 1 The restriction that the invariants can only bound clocks from above (i. e., $x \trianglelefteq plt$) is not a real restriction: in timed automata, invariant that bound clocks from below (i. e., $d \trianglelefteq x$) can be moved to all incoming edges. The same applies to PTAs. In other words, papers defining PTAs requir-

ing invariants to use only invariants with clocks bounded from above are equivalent to PTAs with no restrictions at all on the invariants.

2.2.4 Expressiveness

A comparison of the expressiveness of these different syntactic models remains to be done. Whereas it is likely that allowing constraints of the form $x \bowtie plt$ may be simulated using constraints of the form $x \bowtie p \mid d^+$ (perhaps adding additional locations, clocks and parameters), the expressiveness may differ when adding a set of accepting locations. In fact, the expressiveness of a PTA was not even defined, until we recently proposed two first possible definitions [17]: the expressiveness of a PTA A (with accepting locations) is either the union over all parameter valuations of the accepted untimed words ("untimed language of A"), or the union over all parameter valuations of pairs made of an accepted untimed word and the associated valuation ("constrained untimed language of A"). Then, several subclasses of PTAs are compared w.r.t. these two definitions.

However, no comparison of the syntax used in guards and invariants was proposed. A challenging future work would be to show that a PTA with constraints of the form $x \bowtie plt$ can be for example translated into an equivalent PTA with constraints of the form $x \bowtie p \mid d^+$ at the cost of n additional clocks and/or parameters.

2.3 Decision and computation problems

Following the presentation in [53], given a class of decision problems $\mathcal{P}$ (reachability, unavoidability, etc.), let us define the $\mathcal{P}$-emptiness, the $\mathcal{P}$-universality and the $\mathcal{P}$-finiteness.

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

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

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

In this survey, we mainly focus on reachability and unavoidability properties and call them EF and AF, respectively.[1] For example, given a PTA A and a subset G of its locations[2], EF-emptiness asks: "is the set of parameter valuations v such that at least one location of G is reachable in $v(\mathsf{A})$ empty?" And AF-universality asks: "are all parameter valuations v such that any location in G is unavoidable in $v(\mathsf{A})$?" We will also mention the EG property that checks whether there exists a maximal run along which the locations remain in G, and the AG property that checks whether the locations remain in G for all runs.[3]

Additionally, we will consider the language (resp. trace) preservation (emptiness) problem [21]: given a PTA A and a parameter valuation v, does there exist another valuation $v' \neq v$ such that the untimed languages (resp. sets of traces) of $v(\mathsf{A})$ and $v'(\mathsf{A})$ are the same?

We finally define the following computation problem:

$\mathcal{P}$-synthesis problem:
INPUT: A PTA A and an instance ϕ of $\mathcal{P}$
PROBLEM: Compute the parameter valuations such that $v(\mathsf{A})$ satisfies ϕ.

For example, given a PTA A and a subset G of its locations, EF-synthesis consists in synthesizing parameter valuations v such that at least one location of G is reachable in $v(\mathsf{A})$ from the initial state.

Example 2 Let us exemplify some decision and computation problems for the PTA in Fig. 1. Assume the unique target location is "done", i.e., $G = \{\text{done}\}$. EF-emptiness asks whether the set of parameter valuations that can reach location "done" for some run is empty; this is false (e.g., $p_1 = 1$, $p_2 = 2$, $p_3 = 3$ can reach "done"). EF-universality asks whether all parameter valuations can reach location "done" for some run; this is false (no parameter valuation such that $p_2 > p_3$ can reach "done"). AF-emptiness asks whether the set of parameter valuations that can reach location "done" for all runs is empty; this is false (e.g., $p_1 = 1$, $p_2 = 2$, $p_3 = 3$ cannot avoid "done"). EF-synthesis consists in synthesizing all valuations for which a run reaches location "done"; the resulting set of valuations is $0 \leq p_2 \leq p_3 \leq 10 \wedge p_1 \geq 0$.

3 Almost everything is undecidable for simple PTAs

In this entire section, we consider simple PTAs without restriction on the number of clocks and parameters. In that situation, all non-trivial problems studied in the literature are undecidable, with the exception of the membership problem (that asks whether the language of a valuated PTA is

[1] The names EF, AF, EG, AG were first used for PTAs in [53], and come from the CTL syntax.

[2] In general, it can be handful to set $G = F$; but as not all definitions of PTAs in the literature have accepting locations, we use here the set G to denote goal locations.

[3] Note that EF-, AF-, EG-, and AG-emptiness are equivalent to AG-, EG-, AF-, EF-universality, respectively.

empty)—which is rather a problem for TAs. By non-trivial, we mean requiring a semantic analysis, and not, e. g., a sole analysis of the syntax of the PTA (e. g., "is the number of clocks even", or any problem defined in Sect. 2.3 by setting $G = L$).

We also show that bounding time (Sect. 3.3) or bounding the parameter domain for rational-valued parameters (Sect. 3.4) preserves the undecidability. However, we will show in Sect. 4 that bounding the number of clocks and/or parameters brings decidability.

All proofs of undecidability reduce from either the halting problem, or the boundedness problem, of a 2-counter machine, both known to be undecidable [62].

3.1 Decidability of the membership

In [7], the membership problem for PTAs is defined as follows: given a PTA A and a parameter valuation v, is the language of $v(\mathsf{A})$ empty? The membership problem is not strictly speaking a problem for PTAs, but rather for TAs, since it considers a valuated PTA. As a consequence, the decidability of this problem only relies on known results for TAs.

On the one hand, the membership problem is decidable (and PSPACE-complete) for PTAs over discrete time ($\mathbb{T} = \mathbb{N}$ and $\mathbb{P} = \mathbb{N}$), over dense time with integer-valued parameters ($\mathbb{T} = \mathbb{R}^+$ and $\mathbb{P} = \mathbb{N}$), and over dense time with rational-valued parameters ($\mathbb{T} = \mathbb{R}^+$ and $\mathbb{P} = \mathbb{Q}$) [5].

On the other hand, the membership problem becomes undecidable with real-valued (in fact irrational) parameters. Indeed, the reachability of a location in a TA with irrational constants is undecidable [61]. The idea is to encode a 2-counter machine using 2 clocks x_1 and x_2 (plus an additional third clock), where the value c_i of counter i is encoded using $x_i = c_i \times \tau$, for $i \in \{1, 2\}$, with τ the irrational constant (the value $\sqrt{2}$ is suggested for τ).

3.2 General undecidable problems

3.2.1 EF-emptiness

The seminal paper on PTAs [7] showed that the EF-emptiness problem is undecidable for PTAs, both over discrete time, and over dense time (real-valued clocks and real-valued parameters). The proof consists in reducing from the halting problem of a 2-counter machine. The idea of the encoding of the 2-counter machine is to use parameters (the value of which can be arbitrarily large) to encode the maximum value of the counters. Although not explicitly stated in [7], the proof of undecidability also works for real-valued clocks with integer-valued parameters.

3.2.2 AF-emptiness

In [53], it is proved that the AF-emptiness is undecidable for L/U-PTAs (a subclass of PTAs, see Sect. 5) with 3 clocks and 4 integer-valued parameters, and hence for PTAs as well. Again, the proof of undecidability consists in reducing from the halting problem of a 2-counter machine. Another proof is provided in [16] that uses 3 clocks and only 2 rational-valued parameters.

3.2.3 AG-emptiness

In [16], it is proved that the AG-emptiness problem is undecidable with 3 clocks and 2 rational-valued parameters.

3.2.4 EG-emptiness

In [13], it is proved that the EG-emptiness problem is undecidable with 4 clocks and 3 parameters.

Remark 2 For all three previous problems (AF-emptiness, AG-emptiness and EG-emptiness), the result is in fact proved for a subclass of PTAs—namely L/U-PTAs for AF-emptiness and EG-emptiness, and bounded integer-points PTAs (see Sect. 6.3) for AG-emptiness—so the number of clocks and parameters needed for the encoding is certainly not minimal for general PTAs, and might therefore be reduced using smarter constructions.

Remark 3 Note that the undecidability of all of these problems rules out the possibility to perform exact parametric model checking of CTL-like properties on PTAs.

3.2.5 Language and trace preservation problems

Both the language preservation and the trace preservation problems are undecidable for simple PTAs [21]. The *continuous* (or robust) versions of those problems additionally require that the language (resp. set of traces) is preserved under any intermediary valuation of the form $\lambda \cdot v + (1-\lambda) \cdot v'$, for $\lambda \in [0, 1]$ (with the classical definition of addition and scalar multiplication).

The language preservation problems and its continuous version are undecidable for a PTA with at least 4 parametric clocks [21].

The trace preservation and its continuous version are undecidable too; the proof of this result comes with three flavors:

1. the first proof involves diagonal constraints (i. e., of the form $x_i - x_j \bowtie plt$, which goes beyond the syntax of simple PTAs), but only a fixed number of parametric clocks [21];
2. the second proof does not involve diagonal constraints. It involves a bounded number of locations (but with an

unbounded number of transitions) and an unbounded number of parametric clocks; by unbounded we mean not constant but depending on the size of the counter machine [21];
3. the third proof uses a bounded number of clocks and parameters, and an unbounded number of locations [14].

The need for an unbounded number of clocks in the first two versions of this proof comes from the fact that the proof encodes the 2-counter machine with a fixed number of locations (to reduce easily from language preservation to trace preservation), which thus requires to encode each location with a different clock. Note that the first two versions of the proof are, to the best of our knowledge, the only attempt to model a 2-counter machine using PTAs with a constant number of locations (at the cost of an unbounded number of clocks).

3.3 Bounding time

Bounded time model checking consists in checking a property *within a bounded time domain*. That is, we assume a predefined time bound (say T), and we only consider the system behavior in the time interval $[0, T]$. Undecidable problems might become decidable in this situation, or be of a lower complexity. For example, the language inclusion for timed automata becomes decidable over bounded time [64], although it is undecidable in general. In addition, time-bounded reachability becomes decidable for a special subclass of hybrid automata with monotonic (either nonnegative or non-positive) rates [34], although it is undecidable in general.

In contrast, the EF-emptiness problem remains undecidable for (general) PTAs over bounded, dense time [54, Theorem 3.4].

This said, we emphasize that (quite trivially) model checking *discrete-time* PTAs over bounded time would become decidable; the same is likely to hold for *dense-time* PTAs with *integer-valued* parameters over bounded time. (This remains to be shown formally though.)

3.4 Bounding the parameter domain

Bounding the parameter domain consists in setting a minimal and a maximal (non-infinite) bound on the possible parameter valuations of a PTA.

3.4.1 Decidability for integer-valued parameters

For integer parameters, any problem for a PTA over a bounded parameter domain is decidable iff the corresponding problem is decidable for a TA. In fact, the $\mathcal{P}$-emptiness problem for PTAs with bounded integer is PSPACE-complete for any class of problems $\mathcal{P}$ that is PSPACE-complete for TAs [53]. Indeed, it suffices to enumerate all parameter valuations, of which there is a finite number. As a consequence, EF-, AF-, EG-, AG-emptiness are all decidable, and so are language and trace preservation. More generally, the whole TCTL model checking, including reachability and unavoidability, is PSPACE-complete [2], and therefore the corresponding emptiness problems are PSPACE-complete for PTAs with bounded integer parameters.

In [53], a symbolic method is proposed to compute EF- and AF-synthesis; experiments showed that this symbolic computation is faster than an exhaustive enumeration (using UPPAAL).

3.4.2 Undecidability for rational-valued parameters

For rational-valued parameters, the EF-emptiness problem is undecidable for a single parameter in $[1, 2]$ [61]. EG-emptiness [13], AF- and AG-emptiness [16], as well as language and trace preservation [21] are also undecidable for one or two rational-valued bounded parameter(s) (typically bounded by $[0, 1]$).

4 Bounding the numbers of clocks and parameters

4.1 EF-emptiness

Since the seminal paper on PTAs [7], the decidability of the EF-emptiness problem was studied in various settings, by bounding the number of parametric clocks, of nonparametric clocks, and of parameters. The syntax was also restrained.

We summarize these results in Table 2.[4] We only keep in Table 2 the best known results as of the current state of the art. For example, the decidability of the EF-emptiness problem over dense time with 1 parametric clock and arbitrarily many nonparametric clocks and integer-valued parameters as proved in [7] with a non-elementary complexity does not appear in Table 2 as it is subsumed by Beneš et al. [26] with an NEXPTIME complexity and a more permissive syntax (use of invariants).

The open question of the syntax expressiveness requires to consider a multi-dimensional table: we need to consider not only the number of clocks and parameters, but also the syntax allowed in guards and invariants. For example, for the undecidability over discrete time, [26] improves the number of parameters when compared to [7] (6 instead of 1), but requires both strict and non-strict inequalities, whereas [7] uses only equalities in their construction; it is therefore unclear whether the result of [7] is really subsumed by Beneš

[4] This table is partially inspired by a similar table in [42], improved by adding more dimensions, and of course more recent results.

Table 2 Decidability of the EF-emptiness problem for general PTAs

$\mathbb{T}$	$\mathbb{P}$	Guards	Invariants	P-clocks	NP-clocks	Params	Decidability	Main ref.
$\mathbb{N}$	$\mathbb{N}$	$x \bowtie p\|d$		1	0	fixed	(at most) PTIME	[61] (consequence)
$\mathbb{N}$	$\mathbb{N}$	$x \bowtie p\|d$		1	0	any	(at most) NP-complete	[61] (consequence)
$\mathbb{N}$	$\mathbb{N}$	$x \bowtie p\|d$		1	any	any	NEXPTIME-complete	[37,26] (consequence)
$\mathbb{N}$	$\mathbb{N}$	$x \leq\geq p\|d^+$		2	any	1	PSPACE$^{\text{NEXP}}$-hard	[37]
$\mathbb{N}$	$\mathbb{N}$	any		2	any	> 1	open	
$\mathbb{N}$	$\mathbb{N}$	$x \bowtie p\|d$	None	3	0	1	undecidable	[26]
$\mathbb{N}$	$\mathbb{N}$	$x = p\|d$	None	3	0	6	undecidable	[7]
$\mathbb{N}$	$\mathbb{N}$	$x <> p$		any	any	any	open	
$\mathbb{N}$	$\mathbb{N}$ bounded	$x \bowtie plt$		any	any	any	(at most) PSPACE-complete	[54] (consequence)
$\mathbb{R}^+$	$\mathbb{N}$	$x \bowtie p\|d$		1	0	fixed	(at most) PTIME	[61] (consequence)
$\mathbb{R}^+$	$\mathbb{N}$	$x \bowtie p\|d$		1	0	any	(at most) NP-complete	[61] (consequence)
$\mathbb{R}^+$	$\mathbb{N}$	$x \bowtie p\|d$		1	any	any	NEXPTIME	[26]
$\mathbb{R}^+$	$\mathbb{N}$	any		2	any	any	open	
$\mathbb{R}^+$	$\mathbb{N}$	$x \bowtie p\|d$	None	3	0	1	undecidable	[26]
$\mathbb{R}^+$	$\mathbb{N}$	$x = p\|d$	None	3	0	6	undecidable	[7] (consequence)
$\mathbb{R}^+$	$\mathbb{N}$	$x <> p$		any	any	any	open	
$\mathbb{R}^+$	$\mathbb{N}$ bounded	$x \bowtie plt$		any	any	any	PSPACE-complete	[54]
$\mathbb{R}^+$	$\mathbb{Q}^+$	$x \bowtie p\|d$		1	0	fixed	PTIME	[61]
$\mathbb{R}^+$	$\mathbb{Q}^+$	$x \bowtie p\|d$		1	0	any	NP-complete	[61]
$\mathbb{R}^+$	$\mathbb{Q}^+$	any		1	1 or 2	any	open	
$\mathbb{R}^+$	$\mathbb{Q}^+_{[1;2]}$	$x \bowtie p\|d$		1	3	1	undecidable	[61]
$\mathbb{R}^+$	$\mathbb{Q}^+$	any		2	0 or 1	any	open	
$\mathbb{R}^+$	$\mathbb{Q}^+_{[1;2]}$	$x \bowtie p\|d$		2	2	1	undecidable	[61] (consequence)
$\mathbb{R}^+$	$\mathbb{Q}^+_{[1;2]}$	$x \bowtie p\|d$		3	0	1	undecidable	[61]
$\mathbb{R}^+$	$\mathbb{R}^+$	$x = p\|d$	None	3	0	6	undecidable	[7]
$\mathbb{R}^+$	$\mathbb{Q}^+$	$x <> p$		< 2	3	2	open	
$\mathbb{R}^+$	$\mathbb{Q}^+$	$x <> p$		2	< 3	2	open	
$\mathbb{R}^+$	$\mathbb{Q}^+$	$x <> p$		2	3	< 2	open	
$\mathbb{Q}^+/\mathbb{R}^+$	$\mathbb{Q}^+/\mathbb{R}^+$	$x <> p$		2	3	2	undecidable	[42]

et al. [26]. However, following Remark 1, we considered that the works requiring invariants to contain only clocks bounded from above impose in fact no constraint on the invariants form (as the clocks bounded from below can be moved to the incoming guards).

"Consequence" indicates a result originally proved for a less expressive or a more expressive setting; "at most" in the complexity column indicates in the latter case that the complexity is necessarily lower or equal to that of the more expressive setting. For example, [61] proved that the single clock case is PTIME over dense time with a fixed number of rational-valued parameters, and therefore the corresponding problem cannot be harder over discrete time (with integer-valued parameters).

In the following, we extract the most important results out of Table 2. The decidability is clearly impacted by the number of parametric clocks, and we therefore reason by the number of parametric clocks.

4.1.1 Main results: 1 parametric clock

First, let us consider PTAs with a single parametric clock: The EF-emptiness problem is (at most) NP-complete over discrete and dense time with no nonparametric clock and arbitrarily many parameters [61].

It is decidable and NEXPTIME-complete over discrete time with arbitrarily many nonparametric clocks [26]. Over dense time with arbitrarily many nonparametric clocks and integer-valued parameters, it is NEXPTIME [26].

It is undecidable with three nonparametric clocks [61] over dense time with rational-valued parameters; note that this problem is decidable over discrete time [7,26,37] and over dense time with integer-valued parameters [26], which exhibits a difference between dense and discrete time [61], as well as between integer- and rational-valued parameters over dense time.

4.1.2 Main results: 2 parametric clocks

Second, let us consider PTAs with two parametric clocks: the EF-emptiness problem is decidable over discrete time with arbitrarily many nonparametric clocks and a single parameter and is PSPACE$^{\text{NEXP}}$-hard [37].

Over dense time with rational-valued parameters, the case with 2 parametric clocks and 2 nonparametric clocks is undecidable: [61] gives a proof of undecidability with 1 parametric

clock and 3 nonparametric clocks: comparing one of the nonparametric clocks with a parameter in an additional location (e. g., after the halting location) does not impact the proof and turns a nonparametric clock into a parametric one.

Any other case with two parametric clocks remains open.

4.1.3 Main results: other undecidability

The EF-emptiness problem is undecidable in all settings with three (or more) parametric clocks.

Finally, using only strict inequalities, the EF-emptiness problem is undecidable over dense time for two parametric clocks, three nonparametric clocks and two parameters [42]; this situation was not considered over discrete time.

4.1.4 Open cases

The main open case is the "two parametric clocks" case. The decidability is open for 2 parametric clocks with:

- over discrete time: arbitrarily many nonparametric clocks and more than one parameter;
- over dense time with integer-valued parameters: arbitrarily many nonparametric clocks and parameters;
- over dense time with rational-valued parameters: 0 or 1 nonparametric clock and any number of parameters.

In addition, the decidability remains open over dense time with rational-valued parameters for 1 nonparametric clock, 1 or 2 nonparametric clocks and arbitrarily many parameters.

Finally, the decidability using only strict inequalities remains open for cases not considered by Doyen [42]: less clocks and parameters, or with integer-valued parameters (both over dense and discrete time).

4.2 Language and trace preservation

Let us first recall the definition of determinism from [21]. We say that a PTA is *deterministic* if, for any $l \in L$, for any $a \in \Sigma$, there exists at most one edge $(l, g, a, R, l') \in E$, for some g, R, l'. (Note that it differs from a rather common definition of determinism for TAs, that allows two or more outgoing transitions with the same action label provided that the corresponding guards are pairwise disjoint.)

The language and trace preservation problems are decidable for deterministic PTAs with a single (parametric) clock and with linear parameter constraints allowed in guards and invariants, i. e., of the form $x \bowtie plt$ or $plt \bowtie 0$ [21]. A procedure to compute parameter valuations with the same trace set as a given valuation is proposed in [21] (close to the "inverse method" [11]), that is complete for deterministic PTAs, and terminates in the case of a single clock.

4.3 Parametric model checking

Parametric model checking was addressed in different settings: verifying a nonparametric model against a parametric formula, or a parametric model against a nonparametric formula, or a parametric model against a parametric formula.

4.3.1 Nonparametric model/parametric formula

In [6], an extension of LTL with parameters in the formula ("PLTL") is studied. When only parametric "always" modalities are allowed of the form "$\leq p$", checking emptiness of the valuation set is PSPACE-complete. The solution to the synthesis problem is doubly exponential in the number of parameters. However, when allowing equality in PLTL, the emptiness problem becomes undecidable [6].

4.3.2 Parametric model/nonparametric formula

In [65], it is shown that model-checking PTAs with the (nonparametric) logic MTL [56] is undecidable, even with a single clock and a single parameter and even when the PTAs is deterministic. This negative result comes in contrast to the decidability of the EF-emptiness problem for one-clock PTAs and to the decidability of MTL model checking for (nonparametric) timed automata in the pointwise semantics over finite timed words [63]. Note that the proof of undecidability of [65] requires the parameters to be rational-valued (integer-valued parameters are not sufficient—and this latter case can hence be considered as open).

4.3.3 Parametric model/parametric formula

Model checking a PTA over discrete time with a single parametric clock against a PTCTL formula (a parametric version of TCTL) is decidable, provided the formula does not use equality constraints; otherwise the problem becomes undecidable [36].

4.4 Other problems: open

Other problems are open. However, two constructions were recently proposed for the one parametric clock case, that may help solve most problems in this particular setting. First, in [21], we show that the parametric zone graph is finite for a single (parametric) clock and arbitrarily many rational-valued parameters over dense time. This implies that all problems that reason on the zone graph can be decided. This includes in particular EF-, EG-, AF and AG-emptiness, as well as the language and trace preservation problems.

Second, in [26], an abstraction is proposed for one parametric clock and arbitrarily many nonparametric clocks and integer-valued parameters over dense time. Although this

remains to be shown formally, this abstraction (based on the elimination of the nonparametric clocks followed by a corner-point abstraction on the subsequent region graph) apparently preserves enough elements of the region graph to be used to solve all aforementioned problems.

In both cases, the synthesis seems also to be feasible.

5 The (quite) disappointing class of L/U-PTAs

Lower-bound/upper-bound parametric timed automata (L/U-PTAs), proposed in [52], restrict the use of parameters in the model. A parameter is said to be an *upper-bound parameter* if, whenever it is compared with a clock, it is necessarily compared as an upper bound, i.e., it only appears in inequalities of the form $x \trianglelefteq p$. Conversely, a parameter is a *lower-bound parameter* if it is only compared with clocks as a lower bound, i.e., of the form $p \trianglelefteq x$.

An L/U-PTA is a PTA where the set of parameters is partitioned into upper-bound parameters and lower-bound parameters. In [33], two additional subclasses are introduced: L-PTAs (resp. U-PTAs) are PTAs with only lower-bound (resp. upper-bound) parameters.

Example 3 Consider again the coffee machine in Fig. 1, modeled using a PTA A. This PTA is not an L/U-PTA; indeed, in the guard $x_2 = p_2$ (resp. $x_2 = p_3$), p_2 (resp. p_3) is compared with clocks both as a lower bound and as an upper bound. (Recall that $=$ stands for $\leq$ and $\geq$.)

However, if one replaces $x_2 = p_2$ with $x_2 \leq p_2$ and one replaces $x_2 = p_3$ with $x_2 \leq p_3$, then A becomes an L/U-PTA with lower-bound parameter p_1 and upper-bound parameters $\{p_2, p_3\}$. Note that equalities are not forbidden in L/U-PTAs (e.g., $x_1 = 10$), but only equalities involving parameters.

Several case studies fit into the class of L/U-PTAs: the root contention protocol, the bounded retransmission protocol and the Fischer mutual exclusion protocol are all modeled with L/U-PTAs in [52]; in [52,55], both the Fischer mutual exclusion protocol and a producer–consumer are verified using L/U-PTAs. Interestingly, the two case studies of the seminal paper on PTAs [7] (viz., a toy railroad crossing model and a model of Fischer mutual exclusion protocol) are also L/U-PTAs, although the concept of L/U-PTAs had not been proposed yet at that time. In addition, most models of asynchronous circuits with bi-bounded delays (i.e., where each delay between the change of an input signal and the change of the corresponding output is a parametric interval) can be modeled using L/U-PTAs.

L/U-PTAs were first known for their decidability results (Sect. 5.1); then, new undecidability results (Sects. 5.2 and 5.3) rendered this class less interesting. The most disappointing aspect of L/U-PTAs is the impossibility to perform exact synthesis even when the associated decision problems are decidable. We review these results in the remainder of this section.

5.1 A main decidability result

The first (and main) positive result for L/U-PTAs is the decidability of the EF-emptiness problem [52]. L/U-PTAs benefit from the following interesting monotonicity property: increasing the value of an upper-bound parameter or decreasing the value of a lower-bound parameter necessarily relaxes the guards and invariants, and hence can only add behaviors. Hence, checking the EF-emptiness of an L/U-PTA can be achieved by replacing all lower-bound parameters with 0, and all upper-bound parameters with ∞; this yields a nonparametric TA, for which emptiness is PSPACE [5]. This procedure is not only sound but also complete.

Further decidability results are exhibited in [33], for infinite runs acceptance properties, i.e., where a location is met infinitely often (a problem to which we refer hereafter as Büchi). Note that, in contrast to [52] where the parameters are valued with nonnegative reals, the results in [33] consider integer-valued parameters (though time is dense, i.e., clocks are real-valued). It is shown in [33] that Büchi-emptiness, Büchi-universality, and Büchi-finiteness are PSPACE-complete. Remark that the decidability of the Büchi-finiteness is due to the fact that the parameters are integer-valued; in short, a sufficient bound is computed on the parameters, and then valuations smaller or equal to this bound are enumerated, which would not be feasible for real- or rational-valued parameters.

Oddly, the decidability of EF-universality was never shown for L/U-PTAs. On the one hand, EF-emptiness is decidable for L/U-PTAs with rational-valued parameters [52]. On the other hand, Büchi-universality is decidable for L/U-PTAs with integer-valued parameters [33], and this result extends in a very straightforward manner to EF-universality for L/U-PTAs with integer-valued parameters. Let us first extend Büchi-universality to rational-valued parameters. The result mainly consists in a reasoning dual to [13, Lemma 2].

Proposition 1 *The Büchi-universality problem is PSPACE-complete for L/U-PTAs with rational-valued parameters.*

Proof We aim at proving that, given an L/U-PTA A and a subset of its locations G, the problem of the universality of the set of parameter valuations v such that $v(\mathsf{A})$ has a run passing infinitely often through G is PSPACE-complete. Let us prove that the set of rational valuations satisfying the property is not universal iff the set of integer valuations doing so is not universal.

$\Leftarrow$ Considering that integer valuations are also rational valuations, the result trivially holds.

⇒ Assume there exists a rational-valued parameter valuation v for which $v(\mathsf{A})$ contains *no* infinite run passing infinitely often through locations of G. Let v' be the integer parameter valuation obtained from v as follows:

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

That is, v' is more restrictive than p, and less guards will be enabled, and therefore less behaviors will be possible in $v(\mathsf{A})$. Formally, from the well-known monotonicity property of L/U-PTAs (recalled in e. g., [13, Lemma 1]), if $v(\mathsf{A})$ yields no infinite run passing infinitely often through locations of G, then neither does $v'(\mathsf{A})$.

Now, in [33, Theorem 8], it is proved that the problem of the universality of the set of integer parameter valuations for which there exists an infinite run passing infinitely often through G is PSPACE-complete. This concludes the proof. □

This result extends trivially to EF-universality (by adding self-loops with no guard on all accepting locations).

Corollary 1 *The EF-universality problem is PSPACE-complete for L/U-PTAs with rational-valued parameters.*

5.2 Undecidability results

The first undecidability results for L/U-PTAs are shown in [33]: the *constrained* Büchi-emptiness problem and constrained Büchi-universality problem are undecidable for L/U-PTAs. By constrained, it is meant that some parameters of the L/U-PTA can be constrained by an initial linear constraint, e. g., $p_1 \leq 2 \times p_2 + p_3$. Indeed, using linear constraints, one can constrain an upper-bound parameter to be equal to a lower-bound parameter, and hence build a 2-counter machine using an L/U-PTA. However, when no upper-bound parameter is compared to a lower-bound parameter (i. e., when no initial linear inequality contains both an upper-bound and a lower-bound parameter), these two problems retrieve decidability [33]. The exact decidability frontier may not be found yet: the case where a lower-bound parameter is constrained to be less than or equal to an upper-bound parameter fits in none of the considered cases.

A second negative result is shown in [53]: the AF-emptiness problem is undecidable for L/U-PTAs. This is achieved by a reduction from a 2-counter machine where a lower-bound parameter is equal to an upper-bound parameter iff AF holds. This restricts again the use of L/U-PTAs, as AF is essential to show that all possible runs of a system eventually reach a (good) state.

Then, in [21], it is shown that the language preservation problem is undecidable for L/U-PTAs. Again, this is achieved by a reduction from a 2-counter machine where a lower-bound parameter is equal to an upper-bound parameter iff the language is preserved.

5.3 A frontier between decidability and undecidability

The EG-emptiness problem stands at the frontier between decidability and undecidability [13]. Recall that the EG-emptiness problem is false if there exists at least one parameter valuation for which a maximal run remains entirely within some predefined set G of locations. That is, either this run is an infinite run, and therefore contains a cycle (remaining within G); or this run is a finite run (remaining within G), and therefore ends with a deadlock, i. e., ends with a state from which no discrete transition can be taken, even after letting some time elapse.

On the one hand, deciding whether there exists a valuation in an L/U-PTA yielding a cycle is decidable (and PSPACE-complete). On the other hand, deciding whether there exists a valuation in an L/U-PTA yielding a deadlock is undecidable. (These two problems, not studied in this survey, are without surprise shown to be undecidable for general PTAs.)

The EG-emptiness problem stands in between decidability and undecidability: while this problem is decidable for L/U-PTAs with a bounded parameter domain with closed bounds, it becomes undecidable if either the assumption of boundedness or of closed bounds is lifted.

5.4 Model-checking L/U-PTAs

In [33], a parametric extension of the dense-time linear temporal logic $\mathsf{MITL}_{0,\infty}$ (denoted "$\mathsf{PMITL}_{0,\infty}$") is proposed; when parameters are used only as lower or upper bound in the formula (to which we refer as L/U-$\mathsf{PMITL}_{0,\infty}$), satisfiability and model checking are PSPACE-complete; this is obtained by translating the formula into an L/U-PTA and checking an infinite acceptance property.

Then, in [41], an extension of MITL allowing parametric linear expressions in bounds is proposed (yielding PMITL). Two sets of (integer-valued) parameter valuations are considered: (1) the set of valuations for which a PMITL formula is satisfiable, i. e., for which there exists a timed sequence (possibly belonging to a given L/U-PTA) satisfying it, and (2) the set of valuations for which a PMITL formula is valid, i. e., for which all timed sequences (possibly belonging to a given L/U-PTA) satisfy it. Under some assumptions, the emptiness and universality of the valuation set for which a PMITL property is satisfiable or valid (possibly w.r.t. a given L/U-PTA) are decidable, and EXPSPACE-complete. Essential assumptions for decidability include the fact that

Table 3 Decision problems for L/U-PTAs over dense time

Problem	$\mathbb{P}$	Complexity	Main ref.
EF-emptiness	$\mathbb{Q}^+$	PSPACE-complete	[52]
AG-emptiness	$\mathbb{Q}^+$	PSPACE-complete	Corollary 1
AF-emptiness	$\mathbb{Q}^+$	Undecidable	[53]
Cycle-existence-emptiness	$\mathbb{Q}^+$	Decidable	[13]
Deadlock-existence-emptiness	$\mathbb{Q}^+$	Undecidable	[13]
EG-emptiness (closed bounded)	$\mathbb{Q}^+$	Decidable	[13]
EG-emptiness (general)	$\mathbb{Q}^+$	Undecidable	[13]
Büchi-emptiness	$\mathbb{Q}^+$	PSPACE-complete	[13]
Büchi-universality	$\mathbb{Q}^+$	PSPACE-complete	Proposition 1
Büchi-finiteness	$\mathbb{N}$	PSPACE-complete	[33]
Constrained Büchi-emptiness	$\mathbb{N}$	Undecidable	[33]
Constrained Büchi-universality	$\mathbb{N}$	Undecidable	[33]
L/U-constrained Büchi-emptiness	$\mathbb{N}$	PSPACE-complete	[33]
L/U-constrained Büchi-universality	$\mathbb{N}$	PSPACE-complete	[33]
Language preservation	$\mathbb{N}$	Undecidable	[21]
Language preservation	$\mathbb{Q}^+$	Undecidable	[21]
L/U-$\mathsf{PMITL}_{0,\infty}$-emptiness	$\mathbb{N}$	PSPACE-complete	[33]
L/U-$\mathsf{PMITL}_{0,\infty}$-universality	$\mathbb{N}$	PSPACE-complete	[33]
PMITL model-checking	$\mathbb{N}$	EXPSPACE-complete	[41]

parameters should be used with the same polarity (positive or negative coefficient, as lower or upper bound in the intervals) within the entire PMITL formula, and each interval can only use parameters in one of the endpoints. Additional assumptions include that no interval of the PMITL formula should be punctual (nor empty), and linear parametric expressions are only used in right endpoints of the intervals (single parameters can still be used as left endpoints). In addition, two fragments of PMITL are showed to be in PSPACE, including one that allows for expressing parameterized response ("if an event occurs, then another event shall occur within some possibly parametric time interval").

5.5 Summary of decidability problems for L/U-PTAs

We summarize in Table 3 decision problems for L/U-PTAs. Cases not considered in the literature are not depicted.

5.6 Intractability of the synthesis

The most disappointing result concerning L/U-PTAs is shown in [53]: despite decidability of the underlying decision problem (EF-emptiness), the solution to the EF-synthesis problem for L/U-PTAs cannot be represented using a formalism for which the emptiness of the intersection with equality constraints is decidable. The proof relies on the undecidability of the constrained emptiness problem of [33]. A very annoying consequence is that such a solution cannot be represented as a finite union of polyhedra (since the emptiness of the intersection with equality constraints is decidable).

5.7 Two open classes: L-PTAs and U-PTAs

L-PTAs and U-PTAs (introduced in [33]) are very open classes, in the sense that to the best of our knowledge, no result known to be decidable for L-PTAs (or U-PTAs) was shown undecidable for L/U-PTAs (and is hence either decidable or open). Conversely, and even stronger, no result known to be undecidable for L/U-PTAs was shown decidable for L-PTAs (or U-PTAs)—and remains open.

To summarize, the EG-emptiness, AG-emptiness and AF-emptiness problems, as well as the language and trace preservation problems, are all undecidable for (general) L/U-PTAs, but remain open for L-PTAs and U-PTAs.

5.7.1 Synthesis

The synthesis for L-PTAs and U-PTAs did not receive much attention, with the exception of integer-valued parameters: in that case, it is possible to synthesize the solution to the Büchi-synthesis problem in the form of a union of linear constraints doubly exponential in the number of parameters [33]. The authors note that it remains open whether one can construct a linear constraint with a single exponential blow-up. This result does not extend in a straightforward manner to

rational-valued parameters, as the technique in [33] (for U-PTAs) requires the computation of a sufficient upper bound, and then an exhaustive enumeration of parameters below this bound.

6 Beyond parametric timed automata

6.1 Parametric hybrid automata

Hybrid automata [3,4,46] are an extension of timed automata where clocks (called continuous variables) can have an arbitrary rate (i. e., non-necessarily equal to 1).

The reachability of a location in linear hybrid automata is undecidable, although semi-algorithms were proposed [3]. Interestingly, the simple extension of timed automata to stopwatch automata (where the elapsing of some clocks may be stopped in selected locations) yields a formalism as expressive as linear hybrid automata [38], and for which reachability is undecidable too.

First, remark that parameters can be encoded naturally in the general class of hybrid automata, provided diagonal constraints are allowed (of the form $v_i - v_j \bowtie c$, with v_i, v_j variables and c a constant): a parameter is a variable that is not initialized (its initial value is arbitrary), the rate of which is always 0 (therefore constant), and that is never reset along any transition. However, the undecidability results for linear hybrid automata rule out the possibility of exhibiting any decidability results for (general) parametric linear hybrid automata.

Second, several subclasses of linear hybrid automata were defined in the literature and were shown to enjoy some decidable results (e. g., [23,31,34,35,48]). However, obviously, any such subclass at least expressive as timed automata (such as [34,48]) would necessarily lead to undecidability when adding parameters. This is not the case of some of subclasses of linear hybrid automata, which are incomparable (at least from a syntactic point of view) with timed automata (e. g., [31,35]), or restrict the use and the number of variables [23]. We believe studying parametric extensions of these formalisms represent an interesting direction of research.

6.2 Parametric interrupt timed automata

Interrupt timed automata (ITAs) are a subclass of hybrid systems where clock variables only have a rate of 0 (stopped) or 1 (processing): in fact, ITAs define levels such that, at each level, exactly one clock is active (rate 1), while clocks of lower levels are stopped (rate 0) [31]. In addition, guards can only involve clocks from the current level and the lower levels. Clock updates allow the use of linear expressions involving clocks from lower levels. The model is well suited to define real-time systems with multiple tasks running on a single processor and subject to interruption (where a lower-priority task can be preempted by a higher-priority task). A main positive result for ITAs is that reachability is in NEXPTIME (and in PTIME when the number of clocks is fixed). Interrupt timed automata and timed automata are incomparable in terms of timed language.

In [29], ITAs are extended with parameters, which yields parametric ITAs (PITAs). When parameters are combined with clock values in linear expressions as additive coefficients, the reachability in PITAs reduces to the same problem in nonparametric ITAs and is therefore decidable (with an upper bound of 2EXPTIME on the complexity, due to the reduction). When parameters are combined with clock values in linear expressions as both additive and multiplicative coefficients, the reachability in PITAs remains decidable, with an upper bound of 2EXPSPACE on the complexity. This significantly increases the expressiveness of ITAs and allows to model clock drifts.

Finally, ITAs are extended to polynomial ITAs (PolITAs) in [30], where polynomial expressions on clocks are allowed in guards. Reachability remains decidable, and parameters can be used (without harming the complexity) in polynomials.

6.3 Integer-point parametric timed automata

Integer-point parametric timed automata (IP-PTAs) were introduced in [16] as a subclass of PTAs in which each state in the parametric zone graph (a construction with location and symbolic convex constraints over $X \cup P$) contains an integer point. The main positive result for IP-PTAs with bounded (rational-valued) parameters is the decidability of the EF-emptiness problem. However, the AF-emptiness and AG-emptiness problems are both undecidable.

A more disappointing result is the undecidability of the membership problem, i. e., it is undecidable whether a PTA is an IP-PTA. In addition, synthesis is proved to be intractable (as for L/U-PTAs).

However, a sufficient syntactic condition for the membership of IP-PTAs is the class of *reset-PTAs* [16]: whenever a clock is compared to a parameter in a transition guard (resp. in location invariant), then all clocks must be reset on that transition (resp. along all transitions going out from that location). As a consequence, the EF-emptiness problem is decidable for bounded reset-PTAs too. In addition, we conjecture that the parametric zone graph of reset-PTAs should be finite, which would allow to prove the decidability of the AF, AG and EG-emptiness problems. This remains to be shown formally.

6.4 Other formalisms

6.4.1 Time Petri nets

In parallel to timed automata, many works in the literature were dedicated to time Petri nets [60], which are an extension of Petri nets where transitions are labeled with a firing interval, which represents the duration between the time when the transition becomes enabled (enough tokens are present in the incoming places) and the time it can actually fire. Time Petri nets and timed automata were compared in, e. g., [27,28,66].

In [69], time Petri nets are extended with rational-valued parameters in firing intervals. Using translations between time Petri nets and timed automata [27,39], it is shown that the emptiness and reachability problems are undecidable for bounded parametric time Petri nets, and turn decidable when parameters are only used as lower bounds or upper bounds, in the spirit of L/U-PTAs. Then, semi-algorithms are defined for the parametric model checking of a subset of parametric TCTL formulas applied on parametric time Petri nets extended with inhibitor arcs (which play a similar role as stopwatches in timed automata). The tool ROMÉO implements these algorithms.

6.4.2 Stateful timed CSP

In [19], the process algebra stateful timed CSP [67] (itself an extension of Hoare's communicating sequential processes [51]) is extended with parameters in syntactic constructs such as `Wait`, `Deadline` or `Within`, yielding PSTCSP. Without surprise (as the expressiveness of PSTCSP is very close to that of PTAs), the emptiness of the valuation set for which a configuration is reachable is undecidable. Although most of the (timed) syntactic constructs allowed are not necessary for the proof of undecidability, the `Wait` construct (used to test an exact amount of time, similar to equality in timed automata), is extensively used. Decidability for subsets of the syntax without the `Wait` construct was not studied. PSTCSP is implemented in a tool PSyHCoS [20] implementing some parameter synthesis algorithms.

7 Tools and applications

7.1 Tools

The first tool to support modeling and verification using parametric timed automata was HYTECH [47]. In fact, HYTECH supports linear hybrid automata (including clocks, parameters, stopwatches and general continuous variables); it can compute the state space and perform operations (such as intersection, convex hull, difference) between sets of symbolic states. Therefore, it can be used to perform parametric model checking using reachability checking [1]. HYTECH is not maintained anymore, but can still be found online in the form of a standalone binary for Linux.[5]

In [52], an extension of UPPAAL implementing parametric difference bound matrices (PDBMs) and hence allowing for verification using PTAs is mentioned. However, this tool does not seem to be available anywhere online.

ROMÉO [58] primarily supports parametric time Petri nets (extended with stopwatches), a formalism shown to be close to PTAs in terms of expressiveness [27,69]. ROMÉO supports the use of parametric linear expressions in the time intervals of the transitions and allows to add linear constraints on the parameters to restrict their domain. ROMÉO also implements an original algorithm for integer parameter synthesis using a symbolic (continuous) representation [53]. In addition, ROMÉO provides a simulator and an integrated model checker supporting a subset of the TCTL syntax (including EF-synthesis and AF-synthesis). ROMÉO is mainly written in C++ and makes use of the Parma Polyhedra Library [25].

IMITATOR [12] is a software tool for parametric verification and robustness analysis of PTAs augmented with integer variables and stopwatches. Parameters can be used both in the model and in the properties. Verification capabilities include EF-synthesis, deadlock-freeness-synthesis [9], non-Zeno model checking [22], and trace preservation synthesis. IMITATOR is fully written in OCaml and makes use of the Parma Polyhedra Library [25]. It also features distributed capabilities to run over a cluster.

7.2 Applications

The formalism of PTAs has been used to model and verify various case studies featuring real-time constraints and parameters.

Beyond the usual academic examples (such as variants of train controllers [7,52]), PTAs were also used to successfully specify and verify numerous interesting case studies such as the root contention protocol [52], Philip's bounded retransmission protocol [52], a 4-phase handshake protocol [55], the alternating bit protocol [53], an asynchronous circuit commercialized by ST-Microelectronics [40], (non-preemptive) schedulability problems [53], a distributed prospective architecture for the flight control system of the next generation of spacecrafts designed at ASTRIUM Space Transportation [44], an unmanned aerial video system by Thales,[6] and even analysis of music scores [43].

[5] https://embedded.eecs.berkeley.edu/research/hytech/.

[6] http://www.imitator.fr/static/FMTV15/.

8 Open questions and perspectives

8.1 Syntax and expressiveness

A first perspective is to compare the expressiveness of the various syntaxes of guards and invariants for general PTAs used in the literature. The definitions of expressiveness recently proposed in [16] could be reused for that purpose, using either untimed or timed languages. Comparing the expressiveness of the syntaxes in the literature would reduce the number of dimensions for the various decidability results of the EF-emptiness problem studied in Table 2.

8.2 Open decision problems

A main open problem is the decidability of PTAs with two parametric clocks, that was only studied with a single integer-parameter [37]. Studying further the EG-, AF- and AG-emptiness problems for few clocks and parameters (as it was quite extensively done for EF-emptiness) remains to be done too, although the practical interest may be somehow debatable.

In addition, with the exception of [21], all proofs of undecidability in the literature use a bounded number of clocks and parameters, but an unbounded number of locations. Exhibiting a minimal number of locations (at the possible cost of an unbounded number of variables) may be of theoretical interest.

More interesting (and promising) are the two open classes of L-PTAs and U-PTAs. These classes are non-trivial and relate to the robust analysis of TAs: most robustness problems (see [32,59]) consider an enlargement of all guards by (usually) the same constant factor, whereas U-PTAs allow to enlarge or decrease *some* of the upper-bound guards by a possibly different rational-valued parameter, which gives an orthogonal definition of robustness. The language preservation problem remains open for U-PTAs [21] (except in the case of a single integer-valued parameter where it becomes decidable), and the question of the synthesis is also challenging.

8.3 Hidden decidable subclasses?

Despite many undecidability problems, PTAs were often used to model and verify various case studies (see Sect. 7). This can be seen as a paradox considering the numerous undecidability results PTAs suffer from. In fact, as the aforementioned analyses terminate almost always with an exact result, it is challenging to understand why, and perhaps to exhibit further classes for which the problems considered in this survey become decidable.

8.4 Hybrid systems with parameters

Some subclasses of linear hybrid automata are incomparable with timed automata (e. g., [23,35]), and parametric extensions could be studied. Recall that the class of interrupt timed automata benefits from decidability results even when extended with parameters [30].

8.5 Synthesis

Whereas decision problems (considered in this document) were much studied, little interest has been dedicated to the synthesis of parameters, which should, however, be a main practical challenge. Despite undecidability (in general [7]) or intractability (for L/U-PTAs [53]), semi-algorithms or approximated procedures could be devised; SMT-based techniques [55], or the integer hull approximation [15,53] can serve as a basis for future works. Also note that two recent orthogonal works aimed at performing synthesis in a compositional manner [18,24].

Also, combining nonparametric analysis (e. g., with the efficient model checker UPPAAL) with parametric analysis, so as to find perhaps not all valuations, but at least some of them, is certainly a promising direction of research.

Acknowledgements This manuscript benefited from discussions with Didier Lime, Nicolas Markey, and Olivier H. Roux, as well as from the useful comments and suggestions of all three anonymous reviewers.

References

1. Aceto, L., Bouyer, P., Burgueño, A., Larsen, K.G.: The power of reachability testing for timed automata. In: Arvind, V., Ramanujam, R. (eds.) FSTTCS. LNCS, vol. 1530, pp. 245–256. Springer, New York (1998)
2. Alur, R., Courcoubetis, C., Dill, D.L.: Model-checking in dense real-time. Inf. Comput. **104**(1), 2–34 (1993)
3. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theor. Comput. Sci. **138**(1), 3–34 (1995)
4. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.H.: Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) Hybrid Systems 1992. LNCS, vol. 736, pp. 209–229. Springer, New York (1993)
5. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994)
6. Alur, R., Etessami, K., La Torre, S., Peled, D.: Parametric temporal logic for "model measuring". ACM Trans. Comput. Logic **2**(3), 388–407 (2001)
7. Alur, R., Henzinger, T.A., Vardi, M.Y.: Parametric real-time reasoning. In: STOC, pp. 592–601. ACM (1993)
8. Alur, R., Madhusudan, P.: Decision problems for timed automata: a survey. In: Bernardo, M., Corradini, F. (eds.) Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13–18,

2004, Revised Lectures. LNCS, vol. 3185, pp. 1–24. Springer, New York (2004)
9. André, É.: Parametric deadlock-freeness checking timed automata. In: Sampaio, A.C.A., Wang, F. (eds.) ICTAC. LNCS, vol. 9965, pp. 469–478. Springer, New York (2016)
10. André, É.: What's decidable about parametric timed automata? In: Artho, C., Ölveczky, P.C. (eds.) FTSCS. Communications in Computer and Information Science, vol. 596, pp. 1–17. Springer, New York (2016)
11. André, É., Chatain, T., Encrenaz, E., Fribourg, L.: An inverse method for parametric timed automata. IJFCS **20**(5), 819–836 (2009)
12. André, É., Fribourg, L., Kühne, U., Soulat, R.: IMITATOR 2.5: a tool for analyzing robustness in scheduling problems. In: Giannakopoulou, D., Méry, D. (eds.) FM. LNCS, vol. 7436, pp. 33–36. Springer, New York (2012)
13. André, É., Lime, D.: Liveness in L/U-parametric timed automata. In: Legay, A., Schneider, K. (eds.) ACSD, pp. 9–18. IEEE, New York (2017)
14. André, É., Lime, D., Markey, N.: Language preservation problems in parametric timed automata (journal version). Technical report (2016), submitted
15. André, É., Lime, D., Roux, O.H.: Integer-complete synthesis for bounded parametric timed automata. In: Bojańczyk, M., Lasota, S., Potapov, I. (eds.) RP. LNCS, vol. 9058. Springer, New York (2015)
16. André, É., Lime, D., Roux, O.H.: Decision problems for parametric timed automata. In: Ogata, K., Lawford, M., Liu, S. (eds.) ICFEM (2016), to appear
17. André, É., Lime, D., Roux, O.H.: On the expressiveness of parametric timed automata. In: Fränzle, M., Markey, N. (eds.) FORMATS (2016), to appear
18. André, É., Lin, S.W.: Learning-based compositional parameter synthesis for event-recording automata. In: Bouajjani, A., Alexandra, S. (eds.) FORTE. LNCS, vol. 10321, pp. 17–32. Springer, New York (2017)
19. André, É., Liu, Y., Sun, J., Dong, J.S.: Parameter synthesis for hierarchical concurrent real-time systems. In: Perseil, I., Pouzet, M., Breitman, K. (eds.) ICECCS, pp. 253–262. IEEE Computer Society, Silver Spring (2012)
20. André, É., Liu, Y., Sun, J., Dong, J.S., Lin, S.W.: PSyHCoS: parameter synthesis for hierarchical concurrent real-time systems. In: Sharygina, N., Veith, H. (eds.) CAV. LNCS, vol. 8044, pp. 984–989. Springer, New York (2013)
21. André, É., Markey, N.: Language preservation problems in parametric timed automata. In: Sankaranarayanan, S., Vicario, E. (eds.) FORMATS. LNCS, vol. 9268, pp. 27–43. Springer, New York (2015)
22. André, É., Nguyen, H.G., Petrucci, L., Sun, J.: Parametric model checking timed automata under non-Zenoness assumption. In: Barrett, C., Kahsai, T. (eds.) NFM. Lecture Notes in Computer Science, vol. 10227, pp. 35–51. Springer, New York (2017)
23. Asarin, E., Mysore, V., Pnueli, A., Schneider, G.: Low dimensional hybrid systems—decidable, undecidable, don't know. Inf. Comput. **211**, 138–159 (2012)
24. Aştefănoaei, L., Bensalem, S., Bozga, M., Cheng, C., Ruess, H.: Compositional parameter synthesis. In: Fitzgerald, J.S., Heitmeyer, C.L., Gnesi, S., Philippou, A. (eds.) Proceedings of the 21st International Symposium on Formal Methods (FM 2016). Lecture Notes in Computer Science, vol. 9995, pp. 60–68 (2016)
25. Bagnara, R., Hill, P.M., Zaffanella, E.: The Parma Polyhedra Library: toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Sci. Comput. Program. **72**(1–2), 3–21 (2008)
26. Beneš, N., Bezděk, P., Larsen, K.G., Srba, J.: Language emptiness of continuous-time parametric timed automata. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP, Part II. LNCS, vol. 9135, pp. 69–81. Springer, New York (2015)
27. Bérard, B., Cassez, F., Haddad, S., Lime, D., Roux, O.H.: Comparison of the expressiveness of timed automata and time Petri nets. In: Pettersson, P., Yi, W. (eds.) FORMATS. LNCS, vol. 3829, pp. 211–225. Springer, New York (2005)
28. Bérard, B., Cassez, F., Haddad, S., Lime, D., Roux, O.H.: The expressive power of time Petri nets. Theor. Comput. Sci. **474**, 1–20 (2013)
29. Bérard, B., Haddad, S., Jovanovic, A., Lime, D.: Interrupt timed automata with auxiliary clocks and parameters. Fundamenta Informormatica **143**(3–4), 235–259 (2016)
30. Bérard, B., Haddad, S., Picaronny, C., Din, M.S.E., Sassolas, M.: Polynomial interrupt timed automata. In: Bojanczyk, M., Lasota, S., Potapov, I. (eds.) RP. LNCS, vol. 9328, pp. 20–32. Springer, New York (2015)
31. Bérard, B., Haddad, S., Sassolas, M.: Interrupt timed automata: verification and expressiveness. Form. Methods Syst. Des. **40**(1), 41–87 (2012)
32. Bouyer, P., Markey, N., Sankur, O.: Robustness in timed automata. In: Abdulla, P.A., Potapov, I. (eds.) RP. LNCS, vol. 8169, pp. 1–18. Springer (2013), invited paper
33. Bozzelli, L., La Torre, S.: Decision problems for lower/upper bound parametric timed automata. Form. Methods Syst. Des. **35**(2), 121–151 (2009)
34. Brihaye, T., Doyen, L., Geeraerts, G., Ouaknine, J., Raskin, J., Worrell, J.: Time-bounded reachability for monotonic hybrid automata: complexity and fixed points. In: Hung, D.V., Ogawa, M. (eds.) ATVA. LNCS, vol. 8172, pp. 55–70. Springer, New York (2013)
35. Brihaye, T., Michaux, C., Rivière, C., Troestler, C.: On O-minimal hybrid systems. In: Alur, R., Pappas, G.J. (eds.) HSCC. LNCS, vol. 2993, pp. 219–233. Springer, New York (2004)
36. Bruyère, V., Raskin, J.F.: Real-time model-checking: parameters everywhere. Log. Methods Comput. Sci. **3**(1:7), 1–30 (2007)
37. Bundala, D., Ouaknine, J.: Advances in parametric real-time reasoning. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS. LNCS, vol. 8634, pp. 123–134. Springer, New York (2014)
38. Cassez, F., Larsen, K.G.: The impressive power of stopwatches. In: Palamidessi, C. (ed.) CONCUR. LNCS, vol. 1877, pp. 138–152. Springer, New York (2000)
39. Cassez, F., Roux, O.H.: Structural translation from time Petri nets to timed automata. J. Syst. Softw. **79**(10), 1456–1468 (2006)
40. Chevallier, R., Encrenaz-Tiphène, E., Fribourg, L., Xu, W.: Timed verification of the generic architecture of a memory circuit using parametric timed automata. Form. Methods Syst. Des. **34**(1), 59–81 (2009)
41. Di Giampaolo, B., La Torre, S., Napoli, M.: Parametric metric interval temporal logic. Theor. Comput. Sci. **564**, 131–148 (2015)
42. Doyen, L.: Robust parametric reachability for timed automata. Inf. Process. Lett. **102**(5), 208–213 (2007)
43. Fanchon, L., Jacquemard, F.: Formal timing analysis of mixed music scores. In: International Computer Music Conference (2013)
44. Fribourg, L., Lesens, D., Moro, P., Soulat, R.: Robustness analysis for scheduling problems using the inverse method. TIME, pp. 73–80. IEEE Computer Society Press, Silver Spring (2012)
45. van Glabbeek, R.J.: The linear time-branching time spectrum (extended abstract). In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR. LNCS, vol. 458, pp. 278–297. Springer, New York (1990)
46. Henzinger, T.A.: The theory of hybrid automata. In: Vardi, M.Y., Clarke, E.M. (eds.) LICS. pp. 278–292. IEEE Computer Society, Silver Spring (1996)
47. Henzinger, T.A., Ho, P.H., Wong-Toi, H.: HyTech: a model checker for hybrid systems. Softw. Tools Technol. Transf. **1**, 110–122 (1997)

48. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? J. Comput. Syst. Sci. **57**(1), 94–124 (1998)
49. Henzinger, T.A., Kopke, P.W., Wong-Toi, H.: The expressive power of clocks. In: Fülöp, Z., Gécseg, F. (eds.) ICALP. LNCS, vol. 944, pp. 417–428. Springer, New York (1995)
50. Henzinger, T.A., Nicollin, X., Sifakis, J., Yovine, S.: Symbolic model checking for real-time systems. Inf. Comput. **111**(2), 193–244 (1994)
51. Hoare, C.: Communicating sequential processes. Commun. ACM **21**, 666–677 (1978)
52. Hune, T., Romijn, J., Stoelinga, M., Vaandrager, F.W.: Linear parametric model checking of timed automata. JLAP **52–53**, 183–220 (2002)
53. Jovanović, A., Lime, D., Roux, O.H.: Integer parameter synthesis for timed automata. IEEE Trans. Softw. Eng. **41**(5), 445–461 (2015)
54. Jovanović, A.: Parametric verification of timed systems. Ph.D. thesis, École Centrale Nantes, France (2013)
55. Knapik, M., Penczek, W.: Bounded model checking for parametric timed automata. ToPNoC **5**, 141–159 (2012)
56. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Syst. **2**(4), 255–299 (1990)
57. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a nutshell. Int. J. Softw. Tools Technol. Transf. **1**(1–2), 134–152 (1997)
58. Lime, D., Roux, O.H., Seidner, C., Traonouez, L.M.: Romeo: a parametric model-checker for Petri nets with stopwatches. In: Kowalewski, S., Philippou, A. (eds.) TACAS. LNCS, vol. 5505, pp. 54–57. Springer, New York (2009)
59. Markey, N.: Robustness in real-time systems. In: Bate, I., Passerone, R. (eds.) SIES, pp. 28–34. IEEE Computer Society Press (2011)
60. Merlin, P.M.: A study of the recoverability of computing systems. Ph.D. thesis, University of California, Irvine, CA, USA (1974)
61. Miller, J.S.: Decidability and complexity results for timed automata and semi-linear hybrid automata. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC. LNCS, vol. 1790, pp. 296–309. Springer, New York (2000)
62. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice-Hall Inc, Englewood Cliffs, NJ (1967)
63. Ouaknine, J., Worrell, J.: On the decidability and complexity of metric temporal logic over finite words. Log. Methods Comput. Sci. **3**(1), 1–27 (2007)
64. Ouaknine, J., Worrell, J.: Towards a theory of time-bounded verification. In: Abramsky, S., Gavoille, C., Kirchner, C., auf der Heide, F.M., Spirakis, P.G. (eds.) ICALP Part II. Lecture Notes in Computer Science, vol. 6199, pp. 22–37. Springer, New York (2010)
65. Quaas, K.: MTL-model checking of one-clock parametric timed automata is undecidable. In: André, É., Frehse, G. (eds.) SynCoP. EPTCS, vol. 145, pp. 5–17 Open Publishing Association, Waterloo, Australia (2014)
66. Srba, J.: Comparing the expressiveness of timed automata and timed extensions of Petri nets. In: Cassez, F., Jard, C. (eds.) FORMATS. LNCS, vol. 5215, pp. 15–32. Springer, New York (2008)
67. Sun, J., Liu, Y., Dong, J.S., Liu, Y., Shi, L., André, É.: Modeling and verifying hierarchical real-time systems using stateful timed CSP. ACM Trans. Softw. Eng. Methodol. **22**(1), 1–29 (2013)
68. Sun, J., Liu, Y., Dong, J.S., Pang, J.: PAT: Towards flexible verification under fairness. In: Bouajjani, A., Maler, O. (eds.) CAV. LNCS, vol. 5643, pp. 709–714. Springer, New York (2009)
69. Traonouez, L.M., Lime, D., Roux, O.H.: Parametric model-checking of stopwatch Petri nets. J. Univers. Comput. Sci. **15**(17), 3273–3304 (2009)