Language Emptiness of Continuous-Time Parametric Timed Automata

Nikola Beneš[1⋆], Peter Bezděk[1⋆⋆], Kim G. Larsen[2], and Jiří Srba[2]

[1] Faculty of Informatics, Masaryk University Brno, Czech Republic
[2] Department of Computer Science, Aalborg University, Denmark

Abstract. Parametric timed automata extend the standard timed automata with the possibility to use parameters in the clock guards. In general, if the parameters are real-valued, the problem of language emptiness of such automata is undecidable even for various restricted subclasses. We thus focus on the case where parameters are assumed to be integer-valued, while the time still remains continuous. On the one hand, we show that the problem remains undecidable for parametric timed automata with three clocks and one parameter. On the other hand, for the case with arbitrary many clocks where only one of these clocks is compared with (an arbitrary number of) parameters, we show that the parametric language emptiness is decidable. The undecidability result tightens the bounds of a previous result which assumed six parameters, while the decidability result extends the existing approaches that deal with discrete-time semantics only. To the best of our knowledge, this is the first positive result in the case of continuous-time and unbounded integer parameters, except for the rather simple case of single-clock automata.

1 Introduction

Timed automata [2] are a popular formalism used for modelling of real-time systems. In the classical definition, the clocks in guards are compared to fixed constants and one of the key problems, decidable in PSPACE [1], is the question of language emptiness. More than 20 years ago, Alur, Henzinger and Vardi [3] introduced a parametric variant of the language emptiness problem where clocks in timed automata can be additionally compared to a number of parameters. A clock is *nonparametric* if it is never compared with any of the parameters, otherwise the clock is *parametric*. The parametric language emptiness problem asks whether the parameters in the system can be replaced by constants so that the language of the resulting timed automaton becomes nonempty.

Unfortunately, the parametric language emptiness problem is undecidable for timed automata with three parametric clocks [3]. Yet Alur, Henzinger and Vardi

⋆ Nikola Beneš has been supported by the Czech Science Foundation grant project no. GA15-11089S.
⋆⋆ Peter Bezděk has been supported by the Czech Science Foundation grant project no. GA15-08772S.

Table 1: Decidability of the language (non)emptiness problems

	discrete time integer parameters	continuous time integer parameters	continuous time real parameters
n clocks, m parameters 1 parametric clock only	decidable [3]	**decidable**	undecidable [17]
3 clocks, 1 parameter	**undecidable**	**undecidable**	undecidable [17]
3 clocks, 6 parameters	undecidable [3]	undecidable [3]	undecidable [3]

established a positive decidability result in the case of a single parametric clock. This decidability result was recently extended by Bundala and Ouaknine [10] to the case with two parametric clocks and an arbitrary number of nonparametric clocks. Both positive results are restricted to the discrete-time semantics with only integer delays. The problem of decidability of integer parametric language emptiness in the continuous-time semantics has been open for over 20 years. The parametric language emptiness problem has two variants, which we call *reachability* (existence of a parameter valuation s.t. the language is nonempty) and *safety* (existence of a parameter valuation s.t. the language is empty).

Our main contributions, summarised in Table 1, are: (i) undecidability of the reachability and safety problems (in discrete and continuous-time semantics) for three parametric clocks, no additional nonparametric clocks and one integer parameter and (ii) decidability of the reachability and safety problems in the continuous-time semantics for one parametric clock with an arbitrary number of integer parameters and an unlimited number of additional nonparametric clocks. For reachability the problem is further decidable in NEXPTIME.

Related work. Our undecidability result holds both for discrete and continuous time semantics and it uses only a single parameter with three parametric clocks, hence strengthening the result from [3] where six parameters were necessary for the reduction. In [10] the authors established NEXPTIME-completeness of the parametric reachability problem for the case of a single parametric clock but only for the discrete-time semantics. Parametric TCTL model checking of timed automata, in the discrete-time setting, was also studied in [9,19]. Our decision procedure for one parametric clock is, to the best of our knowledge, the first one that deals with continuous-time semantics without any restriction on the usage of parameters and without bounding the range of the parameters.

Reachability for parametric timed automata was shown decidable for certain (strict) subclasses of parametric timed automata, either by bounding the range of parameters [15] or by imposing syntactic restrictions on the use of parameters as in L/U automata [8,14]. The study of parametric timed automata in continuous time with parameters ranging over the rational or real numbers showed undecidability already for one parametric clock [17], or for two parametric clocks with exclusively strict guards [12]. We thus focus solely on integer-valued parameters in this paper.

Parametric reachability problems for interrupt timed automata were investigated by Bérard, Haddad, Jovanović and Lime [11] with a number of positive

decidability results although their model is incomparable with the formalism of timed automata studied in this paper. Other approaches include the inverse method of [4] where the authors describe a procedure for deriving constrains on parameters in order to satisfy that timed automata remain time-abstract equivalent, however, the termination of the procedure is in general not guaranteed.

2 Definitions

We shall now introduce parametric timed automata, the studied problems and give an example of a parametric system for alarm sensor coordination.

Let $\mathbb{N}_0$ denote the set of nonnegative integers and $\mathbb{R}_{\geq 0}$ the set of nonnegative real numbers. Let $\mathcal{C}$ be a finite set of *clocks* and let $\mathcal{P}$ be a finite set of *parameters*. A *simple clock constraint* is an expression of the form $x \bowtie c$ where $x \in \mathcal{C}$, $c \in \mathbb{N}_0 \cup \mathcal{P}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. A *guard* is a conjunction of simple clock constraints, we denote the set of all guards by $\mathcal{G}$. A conjunction of simple clock constraints that contain only upper bounds on clocks, i.e. $\bowtie \in \{<, \leq\}$, is called an *invariant* and the set of all invariants is denoted by $\mathcal{I}$.

A *clock valuation* is a function $\nu : \mathcal{C} \to \mathbb{R}_{\geq 0}$ that assigns to each clock its nonnegative real-time age and *parameter valuation* is a function $\gamma : \mathcal{P} \to \mathbb{N}_0$ that assigns to each parameter its nonnegative integer value. Given a clock valuation ν, a parameter valuation γ and a guard (or invariant) $g \in \mathcal{G}$, we write $\nu, \gamma \models g$ if the guard expression g, after the substitution of all clocks $x \in \mathcal{C}$ with $\nu(x)$ and all parameters $p \in \mathcal{P}$ with $\gamma(p)$, is true. By ν_0 we denote the initial clock valuation where $\nu_0(x) = 0$ for all $x \in \mathcal{C}$. For a clock valuation ν and a delay $d \in \mathbb{R}_{\geq 0}$, we define the clock valuation $\nu + d$ by $(\nu + d)(x) = \nu(x) + d$ for all $x \in \mathcal{C}$.

Definition 1 (Parametric Timed Automaton). *A* parametric timed automaton (PTA) *over the set of clocks $\mathcal{C}$ and parameters $\mathcal{P}$ is a tuple $A = (\Sigma, L, \ell_0, F, I, \to)$ where Σ is a finite* input alphabet*, L is a finite set of* locations*, $\ell_0 \in L$ is the* initial location*, $F \subseteq L$ is the set of* final (accepting) locations*, $I : L \to \mathcal{I}$ is an* invariant function *assigning invariants to locations, and $\to \subseteq L \times \mathcal{G} \times \Sigma \times 2^{\mathcal{C}} \times L$ is the set of transitions, written as $\ell \xrightarrow{g,a,R} \ell'$ whenever $(\ell, g, a, R, \ell') \in \to$.*

For the rest of this section, let $A = (\Sigma, L, \ell_0, F, I, \to)$ be a fixed PTA. We say that a clock $x \in \mathcal{C}$ is a *parametric clock* in A if there is a simple clock constraint of the form $x \bowtie p$ with $p \in \mathcal{P}$ that appears in a guard or an invariant of A. Otherwise, if the clock x is never compared to any parameter, we call it a *nonparametric clock*.

A configuration of A is a pair (ℓ, ν) where $\ell \in L$ is the current location and ν is the current clock valuation. For every parameter valuation γ we define the corresponding timed transition system $T_\gamma(A)$ where states are all configurations (ℓ, ν) of A that satisfy the location invariants, i.e. $\nu, \gamma \models I(\ell)$, and the transition relation is defined as follows:

- $(\ell, \nu) \xrightarrow{d} (\ell, \nu + d)$ where $d \in \mathbb{R}_{\geq 0}$ if $\nu + d, \gamma \models I(\ell)$;

– $(\ell, \nu) \xrightarrow{a} (\ell', \nu')$ where $a \in \Sigma$ if there is a transition $\ell \xrightarrow{g,a,R} \ell'$ in A such that $\nu, \gamma \models g$ and $\nu', \gamma \models I(\ell')$ where for all $x \in \mathcal{C}$ we define $\nu'(x) = 0$ if $x \in R$ and $\nu'(x) = \nu(x)$ otherwise.

A *timed language* of A under a parameter valuation γ, denoted by $L_\gamma(A)$, is the collection of all accepted *timed words* of the form $(a_0, d_0)(a_1, d_1) \ldots (a_n, d_n) \in (\Sigma \times \mathbb{R}_{\geq 0})^*$ such that in the transition system $T_\gamma(A)$ there is a computation $(\ell_0, \nu_0) \xrightarrow{d_0} (\ell'_0, \nu'_0) \xrightarrow{a_0} (\ell_1, \nu_1) \xrightarrow{d_1} \cdots \xrightarrow{a_{n-1}} (\ell_n, \nu_n) \xrightarrow{d_n} (\ell'_n, \nu'_n) \xrightarrow{a_n} (\ell_{n+1}, \nu_{n+1})$ where $\ell_{n+1} \in F$.

We can now define two problems for parametric timed automata, namely the reachability problem (reaching desirable locations) and safety problem (avoiding undesirable locations). Note that the problems are not completely dual, as the safety problem contains a hidden alternation of quantifiers.

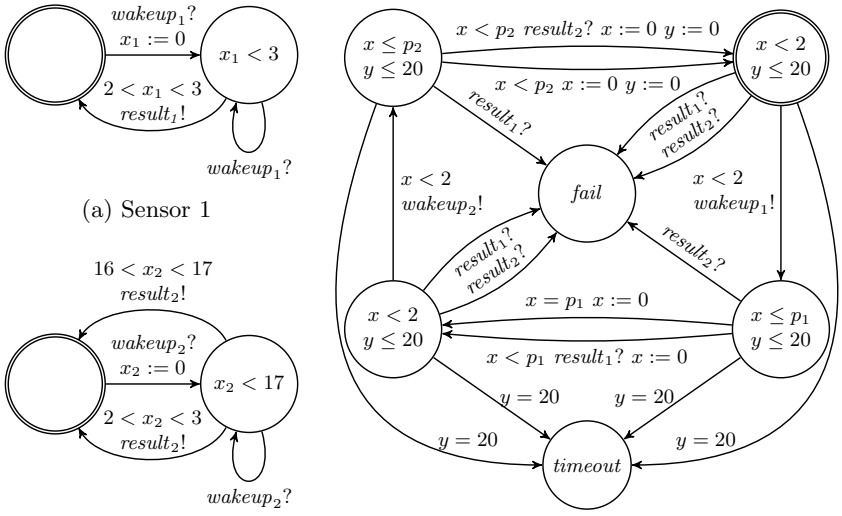
Problem 1 (Reachability Problem for PTA). Given a PTA A, is there a parameter valuation γ such that $L_\gamma(A) \neq \emptyset$?

Problem 2 (Safety Problem for PTA). Given a PTA A, is there a parameter valuation γ such that $L_\gamma(A) = \emptyset$?

We shall now present a small case study of a wireless fire alarm system [13] modelled as a parametric timed automaton. In the alarm setup, a number of wireless sensors communicate with the alarm controller over a limited number of communication channels (in our simplified example we assume just a single channel). The wireless alarm system uses a variant of Time Division Multiple Access (TDMA) protocol in order to guarantee a safe communication of multiple sensors over a shared communication channel. In TDMA the data stream is divided into frames and each frame consists of a number of time slots allocated for exclusive use by the present wireless sensors. Each sensor is assigned a single slot in each frame where it can transmit on the shared channel.

We model each sensor as a timed automaton with two locations as shown in Figure 1a and 1b. The sensor in Figure 1a waits in its initial location until it receives a $wakeup_1$ message from the controller. After this, it takes strictly between 2 to 3 seconds to gather the current status of the sensor and transmit it as $result_1$ message back to the controller. Any subsequent wakeup signals during the transmission phase are ignored and after the transmission phase is finished, the sensor is ready to receive another wakeup signal. The sensor in Figure 1b has a more complex behaviour as transmitting the answer $result_2$ can take either strictly between 2 to 3 seconds, or 16 to 17 seconds.

The controller presented in Figure 1c is responsible for synchronising the two sensors and for assigning them their time slots so that no transmissions interfere. The parametric clock x of the controller determines the size of the time slots. First, it takes at most 2 seconds for the controller to wake up the first sensor after which it waits until the elapsed time reaches the value of the parameter p_1. If it receives the result of the reading of the first sensor in this time slot, it moves immediately into the next location where it performs the wakeup of the second sensor. If the first sensor does not deliver any result and the clock x reaches



(a) Sensor 1

(b) Sensor 2

(c) Controller with parameters p_1 and p_2

Fig. 1: Wireless Fire Alarm System

the value p_1, it also moves to the next location. Now a symmetric control is performed for the second sensor. If any of the two sensors transmit during the time the controller transmits the wakeup signals, we enter the location *fail*. The fail location is also reached if $result_2$ is received in the time slot of the first sensor and vice versa. The second clock y is used to simply measure the duration of the whole frame; whenever the duration of the frame reaches 20 seconds, the controller enters the *timeout* location.

We assume a standard handshake synchronisation of the controller and the two sensors running in parallel that results in a flat product timed automaton with two parameters p_1 and p_2. Note that x is the only parametric clock in our example. Now, our task is to find suitable values of the parameters that guide the duration of the time slots for the two sensors so that there is no behaviour of the protocol where it fails or timeouts. This question is equivalent to the safety problem on the constructed PTA where we mark *fail* and *timeout* as the accepting (undesirable) locations.

The obvious parameter valuation where $\gamma(p_1) = 5$ and $\gamma(p_2) = 19$ guarantees that the location *fail* is unreachable but it is not an acceptable solution as the duration of the frame becomes 24 and we reach *timeout*. However, there is another parameter valuation where $\gamma(p_1) = 5$ and $\gamma(p_2) = 9$ that guarantees that there is no possibility to fail or timeout. This is due to the fact that if the response time of the second sensor is too long, it skips one slot and the answer fits into an appropriate slot in the next frame.

In Section 4 we provide an algorithmic solution for finding such a parameter valuation that guarantees a given safety/reachability criterion. Note that as we are concerned with language (non)emptiness only, we employ two simplifications

in the rest of the paper: First, we assume that the considered PTA have no invariants, as moving all invariants to guards preserves the language. Second, we assume that the alphabet is a singleton set as renaming all actions into a single action preserves language (non)emptiness.

3 Undecidability for Three Parametric Clocks

We shall now provide a reduction from the halting/boundedness problems of two counter Minsky machine to the reachability/safety problems on PTA. A *Minsky machine* with two nonnegative counters c_1 and c_2 is a sequence of labelled instructions $1 : inst_1;\ 2 : inst_2;\ \ldots, n : inst_n$ where $inst_n = HALT$ and each $inst_i$, $1 \leq i < n$, is of one of the following forms (for $r \in \{1, 2\}$ and $1 \leq j, k \leq n$):

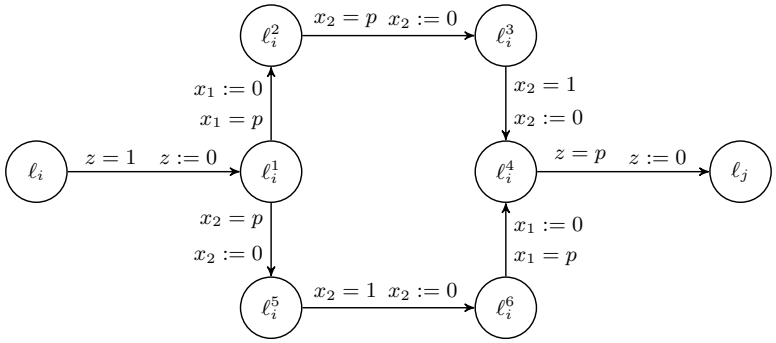
- (Increment) i: c_r`++; goto` j
- (Test and Decrement) i: `if` c_r`=0 then goto` k `else (`c_r`--; goto` j`)`

A configuration is a triple (i, v_1, v_2) where i is the current instruction and $v_1, v_2 \in \mathbb{N}_0$ are the values of the counters c_1 and c_2, respectively. A computation step between configurations is defined in the natural way. If starting from the initial configuration $(1, 0, 0)$ the machine reaches the instruction $HALT$ (note that the computation is deterministic) then we say it *halts*, otherwise it *loops*. The problem whether a given Minsky machine halts is undecidable [18]. The boundedness problem, i.e. the question whether there is a constant K such that $v_1 + v_2 \leq K$ for any configuration (i, v_1, v_2) reachable from $(1, 0, 0)$, is also undecidable [16].

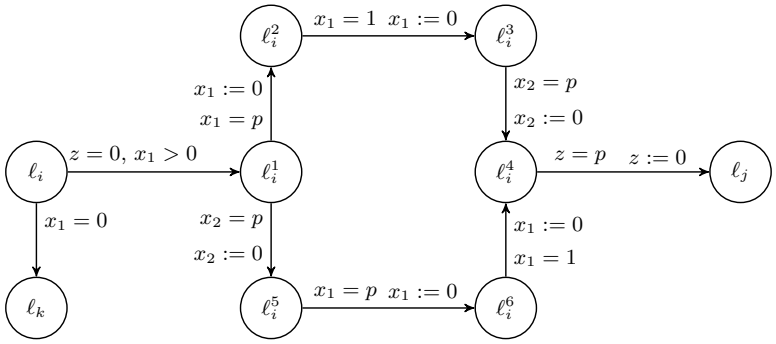
The reduction from a two counter Minsky machine to PTA with a single parameter p and three parametric clocks x_1, x_2 and z is depicted in Figure 2. The reduction rules are shown only for the instructions handling the first counter. The rules for the second counter are symmetric. We also omit the transition labels as they are not relevant for the emptiness problem. The reduction preserves the property that whenever we are in a configuration (ℓ_i, ν) where $\nu(z) = 0$ then $\nu(x_1)$ and $\nu(x_2)$ represent the exact values of the counters c_1 resp. c_2, and the next instruction to be executed is the one with label i. Note also that there are no invariants used in the constructed automaton.

Lemma 1. *Let M be a Minsky machine. Let A be the PTA built according to the rules in Figures 2a and 2b (without the transitions for safety) and where ℓ_1 is the initial location and ℓ_n is the only accepting location. The Minsky machine M halts iff there is a parameter valuation γ such that $L_\gamma(A) \neq \emptyset$.*

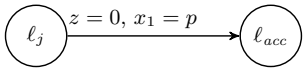
Proof (Sketch). We only sketch a part of the proof to show the basic idea. We argue that from the configuration (ℓ_i, ν) where $\nu(z) = 0$ and where $\nu(x_1)$ and $\nu(x_2)$ represent the counter values, there is a unique way to move from ℓ_i to ℓ_j (or possibly also to ℓ_k in the case of the test and decrement instruction) where again $\nu(z) = 0$ and the counter values are updated accordingly. As there are no invariants in the automaton, we can always delay long enough so that we



(a) Increment i: c_1++; goto j



(b) Test and decrement i: if c_1=0 then goto k else (c_1--; goto j)



(c) For safety, add this for every instruction i: c_1++; goto j

Fig. 2: Encoding of Minsky Machine as PTA with a single parameter p

get stuck in a given location, but this behaviour will not influence the language emptiness problem we are interested in.

Consider the automaton for the increment instruction from Figure 2a and assume we are in a configuration (ℓ_i, ν) where $\nu(z) = 0, \nu(x_1) = v_1$ and $\nu(x_2) = v_2$. First note that if $v_1 \geq p$ then there is no execution ending in ℓ_k due to the forced delay of one time unit on the transition from ℓ_i to ℓ_i^1 and the guard $x_1 = p$ tested in both the upper and lower branch in the automaton. Assume thus that $v_1 < p$. If $v_1 \geq v_2$ then we can perform the following execution with uniquely determined time delays: $(\ell_i, [x_1 \mapsto v_1, x_2 \mapsto v_2, z \mapsto 0]) \xrightarrow{1} (\ell_i^1, [x_1 \mapsto v_1 + 1, x_2 \mapsto v_2 + 1, z \mapsto 0]) \xrightarrow{p-v_1-1} (\ell_i^2, [x_1 \mapsto 0, x_2 \mapsto p - v_1 + v_2, z \mapsto p - v_1 - 1]) \xrightarrow{v_1-v_2} (\ell_i^3, [x_1 \mapsto v_1 - v_2, x_2 \mapsto 0, z \mapsto p - v_2 - 1]) \xrightarrow{1} (\ell_i^4, [x_1 \mapsto v_1 - v_2 + 1, x_2 \mapsto 0, z \mapsto p - v_2]) \xrightarrow{v_2} (\ell_j, [x_1 \mapsto v_1 + 1, x_2 \mapsto v_2, z \mapsto 0])$. In this case where $v_1 \geq v_2$, executing the lower branch of the automaton will result in getting stuck in the location ℓ_i^6 as

here necessarily $\nu(x_1) > p$. Clearly, there is a unique way of getting to ℓ_j in which the clock valuation of x_1 was incremented by one, hence faithfully simulating the increment instruction of the Minsky machine. The other cases and instructions are dealt with similarly, see [6]. □

Lemma 2. *Let M be a Minsky machine. Let A be the PTA built according to the rules in Figures 2a, 2b and 2c (including the transitions for safety) and where ℓ_1 is the initial location and ℓ_{acc} is the only accepting location. The Minsky machine M is bounded iff there is a parameter valuation γ such that $L_\gamma(A) = \emptyset$.*

Proof. If the computation of the Minsky machine is unbounded then clearly, for any parameter value of p, the Minsky machine will eventually try to make one of the counters larger or equal than p (using the increment instruction). Necessarily, we will then have $\nu(x_1) = p$ or $\nu(x_2) = p$ in the location ℓ_j where we end after performing the increment instruction i, implying that we can reach the accepting location ℓ_{acc} due to the transition added in Figure 2c and hence the language is nonempty. On the other hand, if the parameter p is large enough and the computation bounded (note that the boundedness condition $\exists K.\ v_1 + v_2 \leq K$ is equivalent to $\exists K.\ \max\{v_1, v_2\} \leq K$), we will not be able to enter the accepting location ℓ_{acc} and the language is empty. □

We now conclude with the main theorem of this section, tightening the previously known undecidability result that used six parameters and three parametric clocks [3]. The theorem is valid for both the continuous-time and the discrete-time semantics due to the exact guards in all transitions of the constructed PTA that allow to take transitions only after integer delays.

Theorem 1. *The reachability and safety problems are undecidable for PTA with one integer parameter, three parametric clocks and no further nonparametric clocks in the continuous-time as well as the discrete-time semantics.*

4 Decidability for One Parametric Clock

In this section, we show that both the reachability and safety problems for PTA with a single parametric clock are decidable. Our general strategy is similar to that of [3], i.e. reducing the original PTA (which has continuous-time semantics in our case) into a so-called *parametric 0/1-timed automaton* with just a single clock. It is shown in [3] that the set of parameter valuations that ensure language nonemptiness of a given parametric 0/1-timed automaton with single clock is effectively computable. Moreover, in [10] the authors show that the reachability problem for parametric 0/1-timed automata is polynomial-time reducible to the halting problem of parametric bounded one-counter machines, which is in NP. As the parametric 0/1-timed automaton is going to be exponential in the size of the original PTA, this makes the reachability problem for PTA with a single parametric clock belong to the NEXPTIME complexity class.

A 0/1-timed automaton is a timed automaton with discrete time, in which all the delays are explicitly encoded via two kinds of delay transitions: 0-transitions

and 1-transitions. Formally, we enrich the syntax of a timed automaton with two transition relations $\xrightarrow{0}, \xrightarrow{1} \subseteq L \times L$ and modify the semantics so that $(\ell, \nu) \xrightarrow{0} (\ell', \nu)$ iff $\ell \xrightarrow{0} \ell'$ and $(\ell, \nu) \xrightarrow{1} (\ell', \nu + 1)$ iff $\ell \xrightarrow{1} \ell'$; other delays in the timed transition system are no longer possible.

Corner-Point Abstraction. As we are concerned with continuous time, our reduction to 0/1-timed automata is more convoluted than that of [3], in which the nonparametric clocks were eliminated by moving their integer values into locations. In our setting, using region abstraction to eliminate nonparametric clocks will not allow us to correctly identify the 0/1 delays. We thus choose to use *corner-point abstraction* [5] that is finer than the region-based one. In this abstraction, each region is associated with a set of its corner points. Note that the original definition only deals with timed automata that are bounded, while we want to be more general here. For this reason, we extend the original definition with extra corner points for unbounded regions.

We first define the region equivalence [2]. Let $M \in \mathbb{N}_0$ be the largest constant appearing in the constraints of a given timed automaton. Note that in the original definition the largest constant is considered for each clock independently. For the sake of readability, we consider M to be a common upper bound for each clock. Let ν, ν' be clock valuations. Let further $fr(t)$ be the fractional part of t and $\lfloor t \rfloor$ be the integral part of t. We define an equivalence relation $\equiv$ on clock valuations by $\nu \equiv \nu'$ if and only if the following three conditions are satisfied:

- for all $x \in \mathcal{C}$ either $\nu(x) \geq M$ and $\nu'(x) \geq M$ or $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$;
- for all $x, y \in \mathcal{C}$ such that $\nu(x) \leq M$ and $\nu(y) \leq M$, $fr(\nu(x)) \leq fr(\nu(y))$ if and only if $fr(\nu'(x)) \leq fr(\nu'(y))$;
- for all $x \in \mathcal{C}$ such that $\nu(x) \leq M$, $fr(\nu(x)) = 0$ if and only if $fr(\nu'(x)) = 0$.

We define a *region* as an equivalence class of clock valuations induced by $\equiv$. A region r' is a time successor of a region r if for all $\nu \in r$ there exists $d \in \mathbb{R}_{>0}$ such that $\nu + d \in r'$ and for all d', $0 \leq d' \leq d$, we have $\nu + d' \in r \cup r'$. As the time successor is unique if it exists, we use $succ(r)$ to denote the time successor of r. Moreover, if no time successor of r exists, we let $succ(r) = r$.

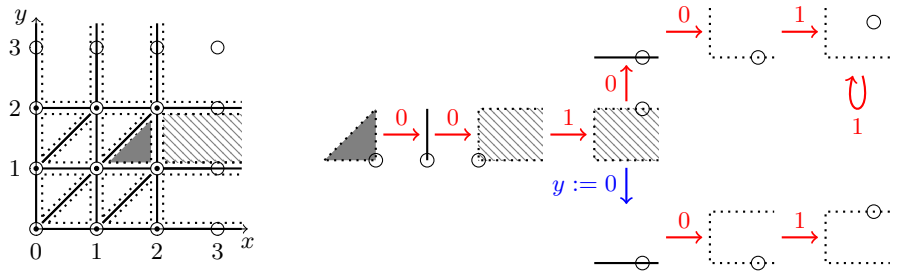
An (M^{+1})*-corner point* $\alpha : \mathcal{C} \to \mathbb{N}_0 \cap [0, M+1]$ is a function which assigns an integer value from the interval $[0, M+1]$ to each clock. We define the successor of the M^{+1}-corner point α, denoted by $succ(\alpha)$, as follows:

$$\text{for each } x \in \mathcal{C}, \quad succ(\alpha)(x) = \begin{cases} \alpha(x) + 1 & \alpha(x) \leq M \\ M + 1 & \text{otherwise .} \end{cases}$$

For $R \subseteq \mathcal{C}$, we define the reset of the corner point α, denoted by $\alpha[R]$, as follows:

$$\text{for each } x \in \mathcal{C}, \quad \alpha[R](x) = \begin{cases} \alpha(x) & x \notin R \\ 0 & x \in R \ . \end{cases}$$

We say α is a corner point of a region r whenever α is in the topological closure of r. The construction of the corner-point abstraction is illustrated in Figure 3. Notice the corner points in unbounded regions.



(a) Corner points where $M = 2$ and $\mathcal{C} = \{x, y\}$

(b) Fragment of an evolution of a region with a corner point (locations are omitted for simplicity)

Fig. 3: Corner point abstraction

Construction of the Parametric 0/1-Timed Automaton. Now we show how to construct for a given PTA with one parametric clock an equivalent 0/1-PTA with just one clock. Let $A = (\Sigma, L, \ell_0, F, I, \to)$ be the original PTA over the set of clocks $\mathcal{C}$ and parameters $\mathcal{P}$. Let x_p denote the only parametric clock.

We first modify the automaton by adding a fresh clock z as follows: every transition $\ell \xrightarrow{g,a,R} \ell'$ is changed into $\ell \xrightarrow{g \wedge z<1,a,R'} \ell'$ where $R' = R$ if $x_\mathrm{p} \notin R$, and $R' = R \cup \{z\}$ otherwise. To every location ℓ we then add a new self-loop transition $\ell \xrightarrow{z=1,a,\{z\}} \ell$. Intuitively, the new clock z will always contain the fractional part of x_p. We call this new automaton A'. Clearly, this modification preserves the language (non)emptiness of the original automaton A.

In the second step, we use the corner-point abstraction of A' with respect to all clocks except for x_p to create the 0/1-timed automaton with a single clock. Let $\hat{\mathcal{C}} = (\mathcal{C} \cup \{z\}) \setminus \{x_\mathrm{p}\}$ and let M be the largest constant appearing in the guards concerning the clocks in $\hat{\mathcal{C}}$. In the following, we consider regions and corner-points with respect to clocks in $\hat{\mathcal{C}}$ and the bound M. Let *Reg* denote the set of all such regions and let *Cp* denote the set of all corresponding corner-points, i.e. $Cp = (\mathbb{N}_0 \cap [0, M+1])^{\hat{\mathcal{C}}}$.

We use the following auxiliary notation. Let $r \in Reg$ and $\alpha \in Cp$.

$$\iota(r, \alpha) = \begin{cases} LESS & \alpha(z) = 1 \text{ and } r \not\models z = 1 \\ MORE & \alpha(z) = 0 \text{ and } r \not\models z = 0 \\ EXACT & \text{otherwise} \end{cases}$$

The 0/1-timed automaton over the singleton set of clocks $\{\hat{x}_\mathrm{p}\}$ is $\hat{A} = (\Sigma, L \times Reg \times Cp, (\ell_0, r_0, \alpha_0), F \times Reg \times Cp, I, \to)$ where r_0 is the initial region and $\alpha_0(x) = 0$ for all $x \in \hat{\mathcal{C}}$ is the initial corner-point. The transition relation is defined as follows:

- zero delay: $(\ell, r, \alpha) \xrightarrow{0} (\ell, r', \alpha)$ if $r' = succ(r)$ and α is a corner-point of both r and r';

– unit delay: $(\ell, r, \alpha) \xrightarrow{1} (\ell, r, \alpha')$ if $\alpha' = succ(\alpha)$ and both α and α' are corner-points of r;
– action: whenever $\ell \xrightarrow{g,a,R} \ell'$ in A' then let $g_1, \ldots, g_k$ be all the simple clock constraints appearing in g comparing clocks from $\hat{\mathcal{C}}$ and let $h_1, \ldots, h_n$ be the remaining simple clock constraints, i.e. those that consider x_{p}. For every (ℓ, r, α) that satisfies (1) $r \models g_1 \wedge \cdots \wedge g_k$ and (2) if $\iota(r, \alpha) \neq EXACT$ then no h_i contains equality ($=$), we set $(\ell, r, \alpha) \xrightarrow{\hat{h}_1 \wedge \cdots \wedge \hat{h}_n, \hat{R}} (\ell', r[R \setminus \{x_{\mathrm{p}}\}], \alpha[R \setminus \{x_{\mathrm{p}}\}])$, where $\hat{R} = \{\hat{x}_{\mathrm{p}}\}$ if $x_{\mathrm{p}} \in R$ and $\hat{R} = \emptyset$ otherwise. The constraints $\hat{h}_i$ are created as follows: all x_{p} are changed into $\hat{x}_{\mathrm{p}}$; if $\iota(r, \alpha) = LESS$, all $<$ are changed into $\leq$ and all $\geq$ are changed into $>$; if $\iota(r, \alpha) = MORE$, all $\leq$ are changed into $<$ and all $>$ are changed into $\geq$.

Theorem 2. *The reachability and safety problems for parametric timed automata over integer parameters with one parametric clock in the continuous-time semantics are decidable. Moreover, the reachability problem is in NEXPTIME.*

Proof (Idea). Due to space constraints, the complete proof can be found in [6]. As mentioned above, the modification from A to A' preserves the language (non)emptiness. The idea of the proof is to show that for every given parameter valuation, every run of A' has a corresponding run in $\hat{A}$ and vice versa. This shows that the reachability and safety problems for parametric timed automata with one parametric clock reduce to the reachability and safety problems for parametric 0/1-timed automata. These problems were shown decidable in [3]. The complexity argument is discussed in the beginning of this section. □

5 Conclusion

We have shown that for three parametric clocks with a single integer parameter, both the reachability and safety problems are undecidable in the discrete as well as the continuous semantics. This improves the previously known undecidability result by Alur, Henzinger and Vardi [3] where six parameters were needed. For the case with a single parametric clock with an unrestricted number of integer parameters and with any number of additional nonparametric clocks, we contributed to the solution of an open problem stated more than 20 years ago by proving a decidability result for reachability and safety problems in the continuous semantics, extending the previously known decidability result for the discrete-time semantics [3]. To achieve this result, we used the corner-point abstraction technique that had to be modified to handle also corner-points in unbounded regions, contrary to the use of the technique in [5]. Not surprisingly, the decidability of the problem in case of two parametric clocks in the continuous-time setting remains open, as it is the case also for a number of other problems over timed automata with two real-time clocks [7]. On the other hand, as demonstrated by our wireless fire alarm case study, the parameter synthesis problem for one parametric clock and an unlimited number of parameters is sufficiently expressive in order to describe nontrivial scheduling problems. As a next step, we will

consider moving from corner-point regions into zones and provide an efficient implementation of the presented techniques.

Acknowledgements. We acknowledge a funding from the EU FP7 grant agreement nr. 318490 (SENSATION) and grant agreement nr. 601148 (CASSTING) and from the Sino-Danish Basic Research Center IDEA4CPS.

References

1. Alur, R., Courcoubetis, C., Dill, D.: Model-checking for real-time systems. In: LICS'90. pp. 414–425. IEEE (1990)
2. Alur, R., Dill, D.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
3. Alur, R., Henzinger, T., Vardi, M.: Parametric real-time reasoning. In: Proceedings of 25th Annual Symposium on Theory of Computing (STOC'93). pp. 592–601. ACM Press (1993)
4. Étienne André, Chatain, T., Fribourg, L., Encrenaz, E.: An inverse method for parametric timed automata. ENTCS 223(0), 29–46 (2008)
5. Behrmann, G., Fehnker, A., Hune, T., Larsen, K.G., Pettersson, P., Romijn, J., Vaandrager, F.W.: Minimum-cost reachability for priced timed automata. In: HSCC'01. LNCS, vol. 2034, pp. 147–161. Springer (2001)
6. Beneš, N., Bezděk, P., Larsen, K.G., Srba, J.: Language emptiness of continuous-time parametric timed automata. CoRR abs/1504.07838 (2015)
7. Bouyer, P., Brihaye, T., Markey, N.: Improved undecidability results on weighted timed automata. Inform. Proc. Letters 98(5), 188–194 (2006)
8. Bozzelli, L., La Torre, S.: Decision problems for lower/upper bound parametric timed automata. Formal Methods in Syst. Design 35(2), 121–151 (2009)
9. Bruyère, V., Raskin, J.F.: Real-time model-checking: Parameters everywhere. In: FST TCS'03. LNCS, vol. 2914, pp. 100–111. Springer (2003)
10. Bundala, D., Ouaknine, J.: Advances in parametric real-time reasoning. In: MFCS'14. LNCS, vol. 8634, pp. 123–134. Springer (2014)
11. Bérard, B., Haddad, S., Jovanović, A., Lime, D.: Parametric interrupt timed automata. In: RP'13, LNCS, vol. 8169, pp. 59–69. Springer (2013)
12. Doyen, L.: Robust parametric reachability for timed automata. Information Processing Letters 102(5), 208 – 213 (2007)
13. Feo-Arenis, S., Westphal, B., Dietsch, D., Muñiz, M., Andisha, S.: The wireless fire alarm system: Ensuring conformance to industrial standards through formal verification. In: FM'14. LNCS, vol. 8442, pp. 658–672. Springer (2014)
14. Hune, T., Romijn, J., Stoelinga, M., Vaandrager, F.: Linear parametric model checking of timed automata. Springer (2001)
15. Jovanović, A., Lime, D., Roux, O.H.: Integer parameter synthesis for timed automata. In: TACAS 2013. LNCS, vol. 7795, pp. 401–415. Springer (2013)
16. Kuzmin, E., Chalyy, D.: Decidability of boundedness problems for Minsky counter machines. Automatic Control and Computer Sciences 44(7), 387–397 (2010)
17. Miller, J.S.: Decidability and complexity results for timed automata and semi-linear hybrid automata. In: Hybrid Systems: Computation and Control, pp. 296–310. Springer (2000)
18. Minsky, M.: Computation: Finite and Infinite Machines. Prentice (1967)
19. Wang, F.: Parametric timing analysis for real-time systems. Information and Computation 130(2), 131–150 (1996)