M2 P2S

2024-2025

Formal verification
Part 3: Parametric timed automata

Étienne André

Université Sorbonne Paris Nord
Etienne.Andre@univ-paris13.fr

Version slides with holes: October 10, 2024

Towards a parametrization…

- Challenge 1: systems incompletely specified
  - Some delays may not be known yet, or may change

- Challenge 2: Robustness [Mar11]
  - What happens if 8 is implemented with 7.99?
  - Can I really get a coffee with 5 doses of sugar?

- Challenge 3: Optimization of timing constants
  - Up to which value of the delay between two actions $press?$ can I still order a coffee with 3 doses of sugar?

- Challenge 4: Avoiding numerous verifications
  - If one of the timing delays of the model changes, should I model check again the whole system?

. [Mar11] Nicolas Markey. « Robustness in Real-time Systems ». In: *SIES*. IEEE Computer Society Press, June 2011, pp. 28–34

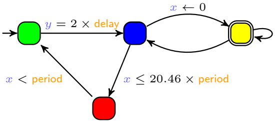
Towards a parametrization…

- Challenge 1: systems incompletely specified
  - Some delays may not be known yet, or may change
- Challenge 2: Robustness [Mar11]
  - What happens if 8 is implemented with 7.99?
  - Can I really get a coffee with 5 doses of sugar?
- Challenge 3: Optimization of timing constants
  - Up to which value of the delay between two actions $press?$ can I still order a coffee with 3 doses of sugar?
- Challenge 4: Avoiding numerous verifications
  - If one of the timing delays of the model changes, should I model check again the whole system?
- A solution: Parametric analysis
  - Consider that timing constants are unknown (parameters)
  - Find good values for the parameters s.t. the system behaves well

[Mar11] Nicolas Markey. « Robustness in Real-time Systems ». In: *SIES*. IEEE Computer Society Press, June 2011, pp. 28–34

Outline

timed model checking



A model of the system

$\overset{?}{\models}$

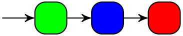
 is unreachable

A property to be verified

- Question: does the model of the system satisfy the property?

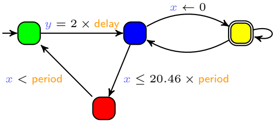
Yes

No



Counterexample

Parametric timed model checking



A model of the system

$\stackrel{?}{\models}$

 is unreachable

A property to be verified

- Question: for which values of the design parameters does the model of the system satisfy the property?

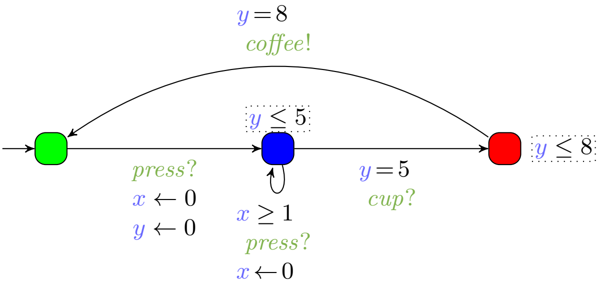
Yes if…

$$2 \times \text{delay} > 20.46 \times \text{period}$$

Outline

Parametric Timed Automaton (PTA)

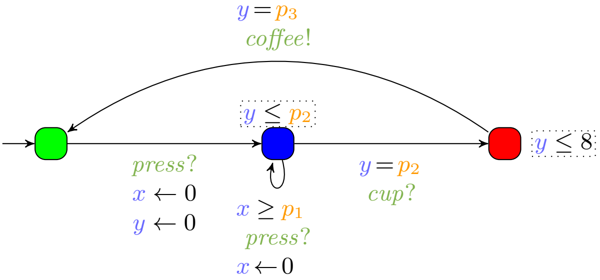
- Timed automaton (sets of locations, actions and clocks)



- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601

Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [AHV93]
  - Unknown constants compared to a clock in guards and invariants



• [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601

Formal definition of parametric timed automata

Definition (Parametric timed automaton)

A parametric timed automaton (TA) $\mathcal{A}$ is an 8-tuple of the form $\mathcal{A} = (L, \Sigma, \ell_0, F, X, P, I, E)$, where

- L is a finite set of locations,
- $\ell_0 \in L$ is the initial location,
- $F \subseteq L$ is a set of final (or accepting) locations,
- Σ is a finite set of actions,
- X is a finite set of clocks,
- P is a finite set of parameters,
- I is the invariant, assigning to every $\ell \in L$ a constraint $I(\ell)$ on the clocks and parameters, and
- E is a transition relation consisting of elements of the form $e = (\ell, g, a, R, \ell')$, where $\ell, \ell' \in L$, $a \in \Sigma$, $R \subseteq X$ is a set of clock variables to be reset by the transition, and g (the transition guard) is a constraint over the clocks and parameters.

Example 1

Draw the PTA $\mathcal{A} = (L, \Sigma, \ell_2, X, P, I, E)$ such that

- $L = \{\ell_1, \ell_2, \ell_3, \ell_4\}$,
- $\Sigma = \{a_1, a_2, a_3\}$,
- $X = \{x_1, x_2\}$,
- $P = \{p_1, p_2, p_3\}$,
- $I(\ell_1) = x_1 \leq 3$, and $I(\ell_4) = x_2 \geq 2p_1 - p_2$,
- $E = \{(\ell_1, x_2 \leq 1, a_1, \{\}, \ell_3),$
  $(\ell_2, x_2 = p_1, a_3, \{x_2\}, \ell_2),$
  $(\ell_2, \top, a_2, \{x_1, x_2\}, \ell_4),$
  $(\ell_3, x_1 \geq p_2, a_1, \{x_1\}, \ell_2),$
  $(\ell_3, \top, a_2, \{x_2\}, \ell_3),$
  $(\ell_4, x_2 > p_3, a_3, \{\}, \ell_3)\}$

Example 1: solution

Example 2: coffee machine

Give the formal PTA corresponding to the parametric timed coffee vending machine.

Example 2: coffee machine

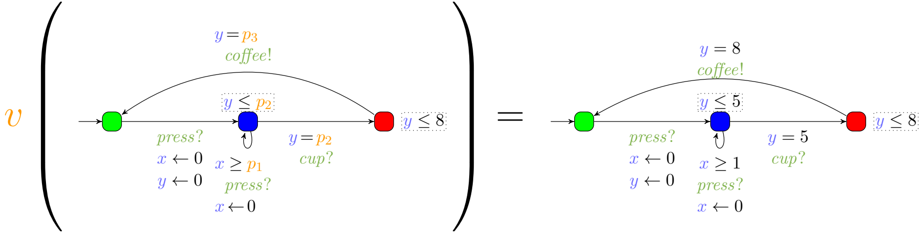
Give the formal PTA corresponding to the parametric timed coffee vending machine.

Notation: Valuation of a PTA

- Given a PTA $\mathcal{A}$ and a parameter valuation v, we denote by $v(\mathcal{A})$ the (non-parametric) timed automaton where each parameter p is valuated by $v(p)$

Notation: Valuation of a PTA

- Given a PTA $\mathcal{A}$ and a parameter valuation v, we denote by $v(\mathcal{A})$ the (non-parametric) timed automaton where each parameter p is valuated by $v(p)$



$$\text{with } v : \begin{cases} p_1 & \to & 1 \\ p_2 & \to & 5 \\ p_3 & \to & 8 \end{cases}$$

Valuation of a PTA: Example

Consider the parametric timed coffee vending machine $\mathcal{A}$.
Let v be a parameter valuation such that $v(p_1) = 3$, $v(p_2) = 4$, and $v(p_3) = 2$.
Draw $v(\mathcal{A})$.

Valuation of a PTA: Example

Consider the parametric timed coffee vending machine $\mathcal{A}$.
Let v be a parameter valuation such that $v(p_1) = 3$, $v(p_2) = 4$, and $v(p_3) = 2$.
Draw $v(\mathcal{A})$.

Outline

Example: A nuclear power plant

Design a TA modeling a nuclear power plant:

- At first, the plant is in normal mode.
- Suddenly, it may start to heat (action $startHeating$).
- At that point, a timer is set; after p_2 time units, the timer will trigger an alarm (action $alarm$).
- Then, p_3 time units later, a watering system (action $watering$) starts.
- This watering system lasts for at most p_4 time units, after which the plant is cool again (action $cool$) and goes back to the normal mode.
- However, p_1 time units after the plant starts to heat, the plant may explode at any time (action $boom$)—unless of course it is cool again.

Example: A nuclear power plant (solution)

Outline

Concrete semantics of parametric timed automata

The concrete semantics of a PTA $\mathcal{A}$ is "just" the concrete semantics of all timed automata $v(\mathcal{A})$, for all v

A parametric timed automaton can therefore be seen as an abstraction, or as a (potentially infinite) set of timed automata

Outline

Symbolic semantics of parametric timed automata

- Symbolic state of a PTA: pair $\mathbf{s} = (\ell, C)$, where
  - ℓ is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hun+02]

• [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220

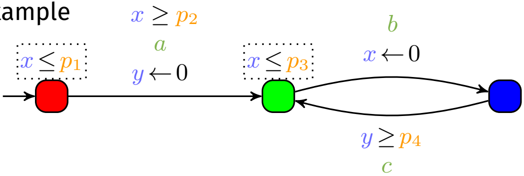
Symbolic semantics of parametric timed automata

- Symbolic state of a PTA: pair $\mathbf{s} = (\ell, C)$, where
  - ℓ is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hun+02]
- Symbolic run: alternating sequence of symbolic states and actions

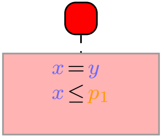
. [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220

Symbolic semantics of parametric timed automata

- Symbolic state of a PTA: pair $\mathbf{s} = (\ell, C)$, where
  - ℓ is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hun+02]
- Symbolic run: alternating sequence of symbolic states and actions
- Example



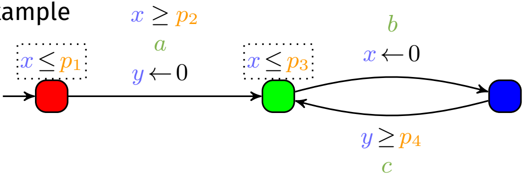
  - Possible symbolic run for this PTA



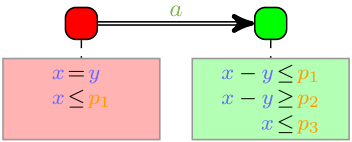
- [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220

Symbolic semantics of parametric timed automata

- Symbolic state of a PTA: pair $\mathbf{s} = (\ell, C)$, where
  - ℓ is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hun+02]
- Symbolic run: alternating sequence of symbolic states and actions
- Example



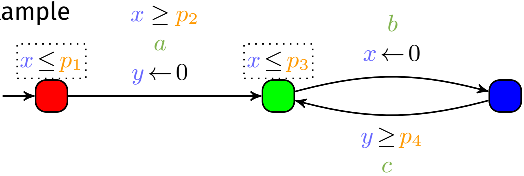
  - Possible symbolic run for this PTA



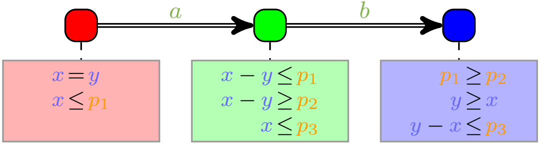
. [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220

Symbolic semantics of parametric timed automata

- Symbolic state of a PTA: pair $\mathbf{s} = (\ell, C)$, where
  - ℓ is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hun+02]
- Symbolic run: alternating sequence of symbolic states and actions
- Example



  - Possible symbolic run for this PTA



• [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220

Symbolic semantics of PTA (1/2)

Definition (State)

Let $\mathcal{A} = (\Sigma, L, \ell_0, X, P, I, E)$ be a PTA. A *state* s of $\mathcal{A}$ is a pair (ℓ, C) where $\ell \in L$ is a location, and $C \in \mathcal{L}(X \cup P)$ its associated constraint.

Symbolic semantics of PTA (1/2)

Definition (State)

Let $\mathcal{A} = (\Sigma, L, \ell_0, X, P, I, E)$ be a PTA. A *state* s of $\mathcal{A}$ is a pair (ℓ, C) where $\ell \in L$ is a location, and $C \in \mathcal{L}(X \cup P)$ its associated constraint.

The *initial state* of $\mathcal{A}$ is $s_0 = (\ell_0, C_0)$, where

$$C_0 = I(\ell_0) \wedge \bigwedge_{i=1}^{H-1} x_i = x_{i+1} \wedge x_i \geq 0$$

Symbolic semantics of PTA (1/2)

Definition (State)

Let $\mathcal{A} = (\Sigma, L, \ell_0, X, P, I, E)$ be a PTA. A *state* s of $\mathcal{A}$ is a pair (ℓ, C) where $\ell \in L$ is a location, and $C \in \mathcal{L}(X \cup P)$ its associated constraint.

The *initial state* of $\mathcal{A}$ is $s_0 = (\ell_0, C_0)$, where

$$C_0 = I(\ell_0) \wedge \bigwedge_{i=1}^{H-1} x_i = x_{i+1} \wedge x_i \geq 0$$

In this expression, $I(\ell_0)$ is the invariant of the initial state, and the rest of the expression lets clocks evolve from the same initial value.

Symbolic semantics of PTA (2/2)

Definition (Semantics of PTAs)

Let $\mathcal{A} = (\Sigma, L, \ell_0, X, P, I, E)$ be a PTA. The *semantics of* $\mathcal{A}$ is $\mathcal{LTS}(\mathcal{A}) = (\Sigma, S, S_0, \longrightarrow)$ where

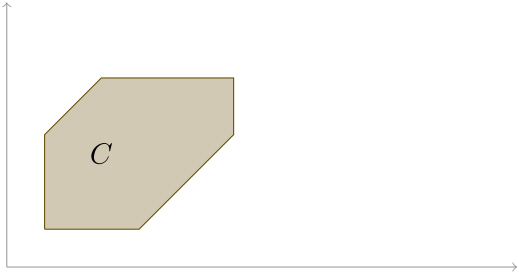
$$S = \big\{(\ell, C) \in L \times \mathcal{L}(X \cup P) \mid C \subseteq I(\ell)\big\},$$
$$S_0 = \{s_0\}$$

A transition $(\ell, C) \xrightarrow{a} (\ell', C')$ belongs to $\longrightarrow$ if $\exists e = (\ell, a, g, R, \ell')$ s.t.:

$$C' = \big([(C \cap g)]_R \cap I(\ell')\big)^{\nearrow} \cap I(\ell')$$

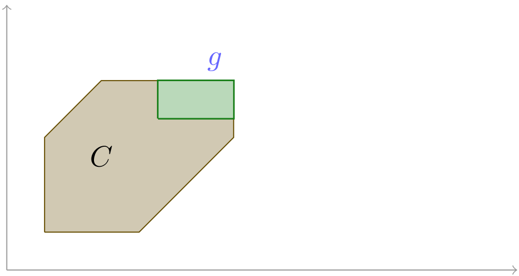
Symbolic semantics of PTA: Successor constraint

$$C' = \Big([(C \cap g)]_R \cap I(\ell') \Big)^{\nearrow} \cap I(\ell')$$



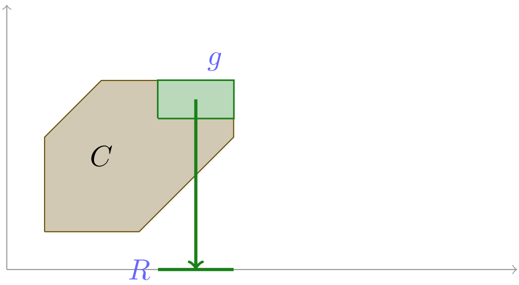
Symbolic semantics of PTA: Successor constraint

$$C' = \Big([(C \cap g)]_R \cap I(\ell') \Big)^{\nearrow} \cap I(\ell')$$



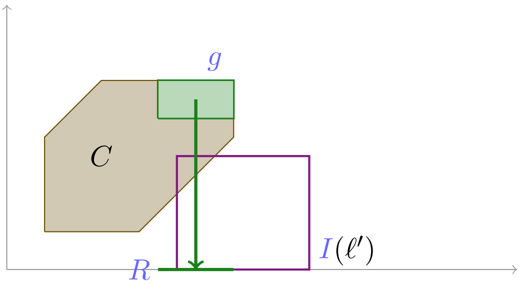
Symbolic semantics of PTA: Successor constraint

$$C' = \Big([(C \cap g)]_R \cap I(\ell') \Big)^{\nearrow} \cap I(\ell')$$



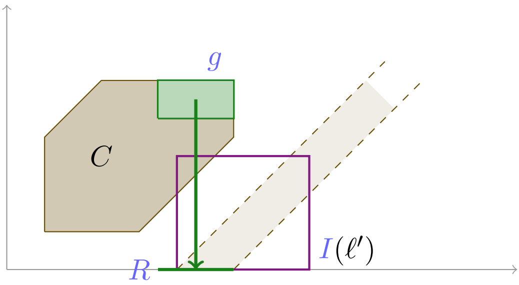
Symbolic semantics of PTA: Successor constraint

$$C' = \Big([(C \cap g)]_R \cap I(\ell') \Big)^{\nearrow} \cap I(\ell')$$



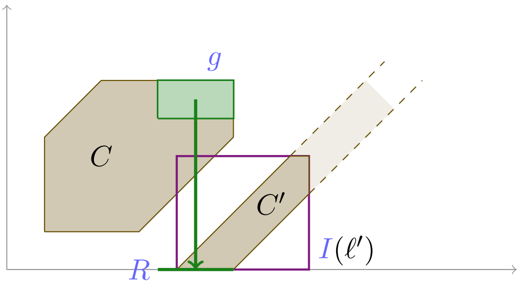
Symbolic semantics of PTA: Successor constraint

$$C' = \Big([(C \cap g)]_R \cap I(\ell') \Big)^{\nearrow} \cap I(\ell')$$

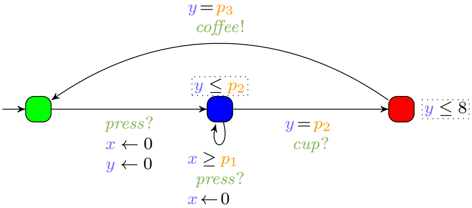


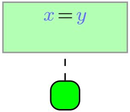
Symbolic semantics of PTA: Successor constraint

$$C' = \left([(C \cap g)]_R \cap I(\ell') \right)^{\nearrow} \cap I(\ell')$$

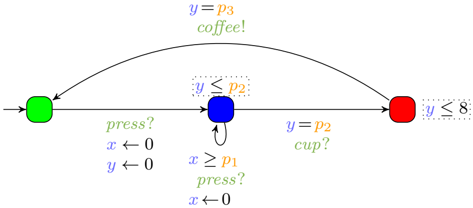


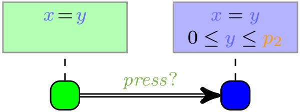
Symbolic exploration: Coffee machine



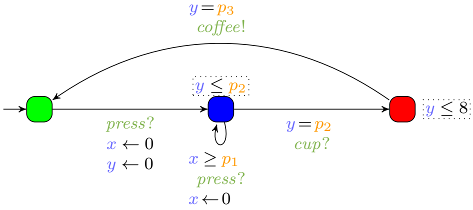


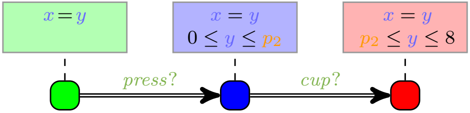
Symbolic exploration: Coffee machine



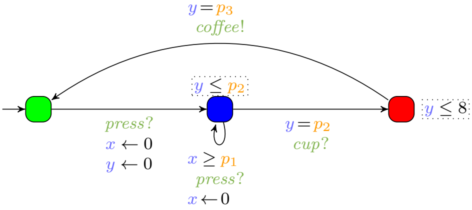


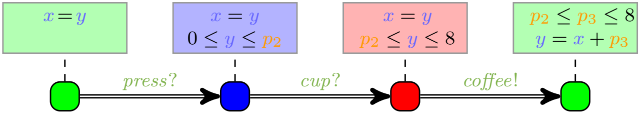
Symbolic exploration: Coffee machine



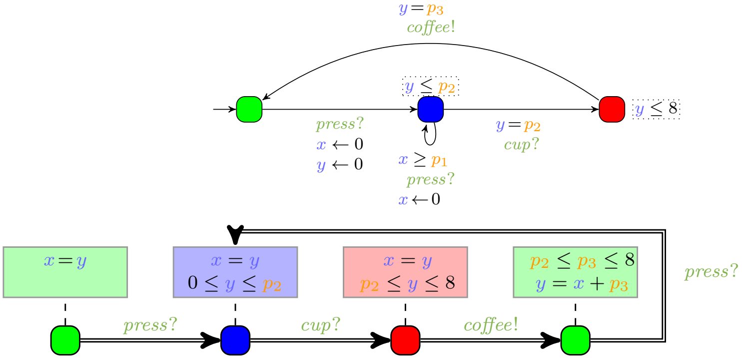


Symbolic exploration: Coffee machine

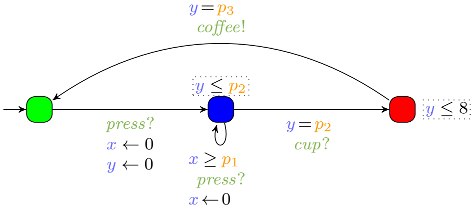


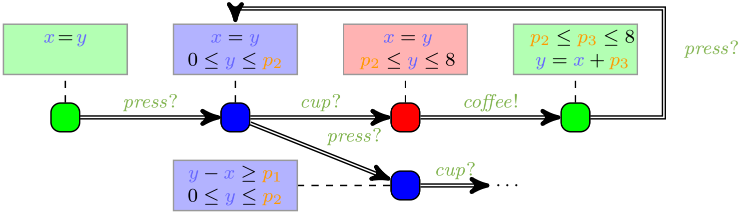


Symbolic exploration: Coffee machine

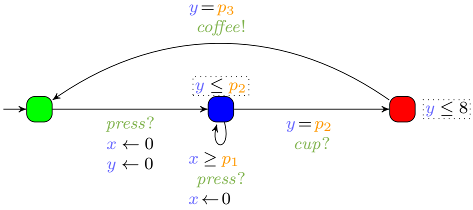


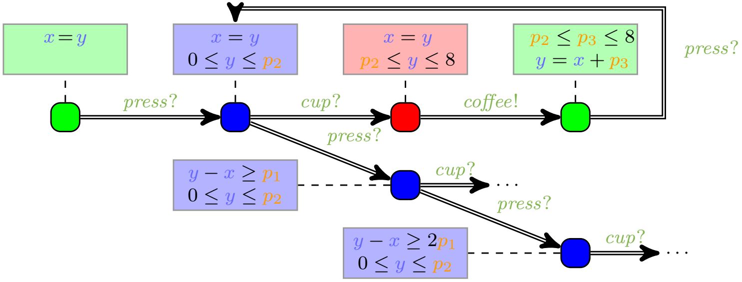
Symbolic exploration: Coffee machine



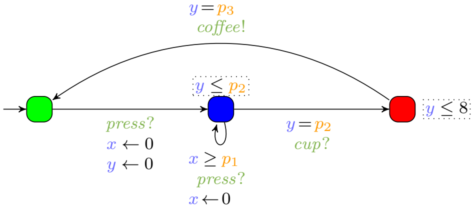


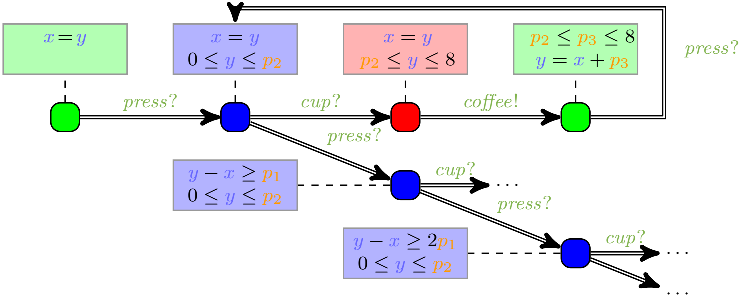
Symbolic exploration: Coffee machine





Symbolic exploration: Coffee machine





Outline

Representation of zones

No “nice” representation exists for parametric zones

- Absence of structures similar to DBMs in timed automata
- Usual representation: polyhedra [BHZ08]

This said, two attempts of “parametric DBMs” were proposed:

- Annichini et al. [AAB00]
- Hune et al. [Hun+02]
- Drawbacks
  - No available implementation (?)
  - still requires... polyhedra (over parameters)

- [BHZ08] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. « The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems ». In: *Science of Computer Programming* 72.1–2 (2008), pp. 3–21
- [AAB00] Aurore Annichini, Eugene Asarin, and Ahmed Bouajjani. « Symbolic Techniques for Parametric Reasoning about Counter and Clock Systems ». In: *CAV*. vol. 1855. Lecture Notes in Computer Science. Springer-Verlag, 2000, pp. 419–434. ISBN: 3-540-67770-4
- [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220

Lack of finite abstraction

- The symbolic state space is infinite in general
- No finite abstraction exists like for timed automata (region automaton, zone automaton)

Lack of finite abstraction

- The symbolic state space is infinite in general
- No finite abstraction exists like for timed automata (region automaton, zone automaton)

Spoiler

All non-trivial problems are undecidable for (sufficiently general) parametric timed automata.

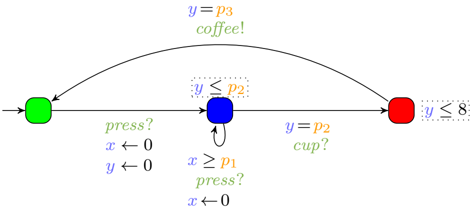
Outline

Two classes of problems

1. Emptiness (decision problem)
   - "Decide whether the set of parameter valuations satisfying some property is empty"
   - Example: reachability emptiness: "Decide the emptiness of the set of parameter valuations such that some location is reachable"
   - Dual problem: universality (all valuations)

2. Synthesis (computation problem)
   - "Synthesize the set of parameter valuations satisfying some property"
   - Example: deadlock-existence-synthesis: "Synthesize the set of parameter valuations for which there exists a deadlock"

Examples of decision problems (1/2)



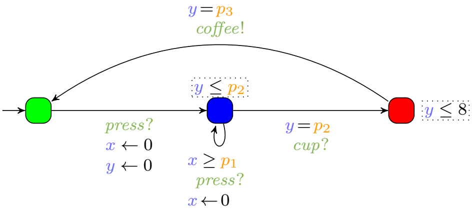
- EF-emptiness "Is the set of parameter valuations for which a given location ℓ is reachable empty?"

  Example: "Is the set of parameter valuations such that I can get a coffee with 2 sugars empty?"

- EF-universality "Do all parameter valuations allow to reach a given location ℓ?"

  Example: "Are all parameter valuations such that I may eventually get a coffee?"

Examples of decision problems (1/2)



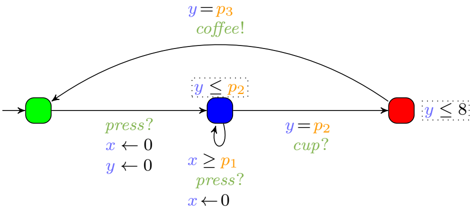
- EF-emptiness "Is the set of parameter valuations for which a given location ℓ is reachable empty?"

  Example: "Is the set of parameter valuations such that I can get a coffee with 2 sugars empty?"

- EF-universality "Do all parameter valuations allow to reach a given location ℓ?"

  Example: "Are all parameter valuations such that I may eventually get a coffee?"

Examples of decision problems (1/2)



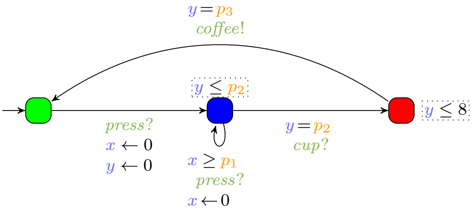
- EF-emptiness "Is the set of parameter valuations for which a given location ℓ is reachable empty?"

  Example: "Is the set of parameter valuations such that I can get a coffee with 2 sugars empty?"

- EF-universality "Do all parameter valuations allow to reach a given location ℓ?"

  Example: "Are all parameter valuations such that I may eventually get a coffee?"

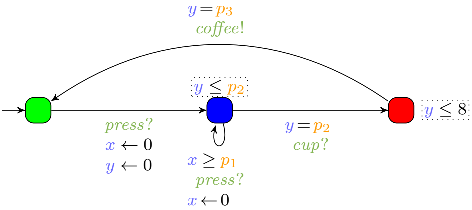
Examples of decision problems (2/2)



- AF-emptiness "Is the set of parameter valuations for which a given location ℓ is always eventually reachable empty?"

  Example: "Is the set of parameter valuations such that I can always eventually get a coffee empty?"

Examples of decision problems (2/2)



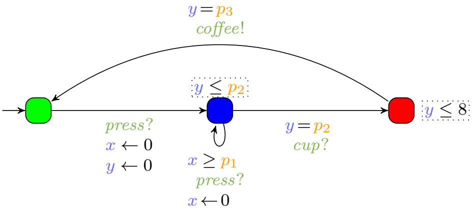
- AF-emptiness "Is the set of parameter valuations for which a given location ℓ is always eventually reachable empty?"

  Example: "Is the set of parameter valuations such that I can always eventually get a coffee empty?"

- AF-universality "Do all parameter valuations necessarily eventually reach a given location ℓ?"

  Example: "Are all parameter valuations such that I can always eventually get a coffee?"

Examples of decision problems (2/2)



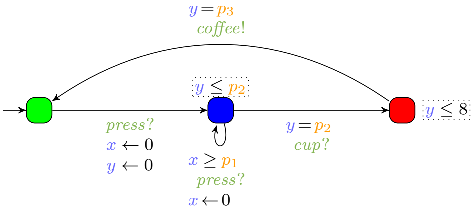
- AF-emptiness "Is the set of parameter valuations for which a given location ℓ is always eventually reachable empty?"

  Example: "Is the set of parameter valuations such that I can always eventually get a coffee empty?"

- AF-universality "Do all parameter valuations necessarily eventually reach a given location ℓ?"

  Example: "Are all parameter valuations such that I can always eventually get a coffee?"

Examples of decision problems (2/2)



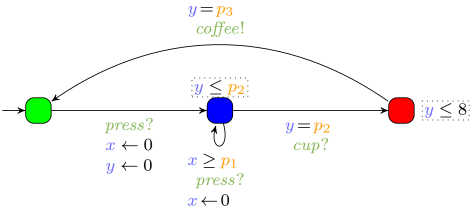
- AF-emptiness "Is the set of parameter valuations for which a given location ℓ is always eventually reachable empty?"

  Example: "Is the set of parameter valuations such that I can always eventually get a coffee empty?"

- AF-universality "Do all parameter valuations necessarily eventually reach a given location ℓ?"

  Example: "Are all parameter valuations such that I can always eventually get a coffee?"

Examples of decision problems (2/2)



- AF-emptiness "Is the set of parameter valuations for which a given location ℓ is always eventually reachable empty?"

  Example: "Is the set of parameter valuations such that I can always eventually get a coffee empty?"

- AF-universality "Do all parameter valuations necessarily eventually reach a given location ℓ?"

  Example: "Are all parameter valuations such that I can always eventually get a coffee?"

Outline

Undecidability of reachability

Reachability-emptiness:

- emptiness of the parameter valuations set for which a given location is reachable
- dual to: “given a PTA $\mathcal{A}$, does there exist a parameter valuation v for which a given location is reachable in $v(\mathcal{A})$?”

- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601

Undecidability of reachability

Reachability-emptiness:

- emptiness of the parameter valuations set for which a given location is reachable
- dual to: "given a PTA $\mathcal{A}$, does there exist a parameter valuation v for which a given location is reachable in $v(\mathcal{A})$?"

Theorem (undecidability [AHV93])

Reachability-emptiness is undecidable for PTAs with at least 3 clocks.

Proof.

By reduction from the halting problem of a 2-counter machine, which is undecidable. □

- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601

Proof of undecidability: 2-counter machine

A deterministic 2-counter machine has two non-negative counters C_1 and C_2, a finite number of states and transitions, which can be of the form:

- "when in state q_i, increment C_k and go to q_j";
- "when in state q_i, if $C_k = 0$ then go to q_l, otherwise decrement C_k and go to q_j".

The machine starts in state q_0 with the counters set to 0.

. [Min67] Marvin L. Minsky. *Computation: Finite and infinite machines.* Prentice-Hall, Inc., 1967. ISBN: 0-13-165563-9

Proof of undecidability: 2-counter machine

A deterministic 2-counter machine has two non-negative counters C_1 and C_2, a finite number of states and transitions, which can be of the form:

- "when in state q_i, increment C_k and go to q_j";
- "when in state q_i, if $C_k = 0$ then go to q_l, otherwise decrement C_k and go to q_j".

The machine starts in state q_0 with the counters set to 0.

The halting problem consists in deciding whether some distinguished state called $\mathrm{q}_{\mathrm{halt}}$ can be reached or not.
The boundedness problem asks whether the counters stay bounded or not along the execution of the machine.

Theorem (undecidability [Min67])

Both the halting problem and the boundedness problem are undecidable.

. [Min67] Marvin L. Minsky. *Computation: Finite and infinite machines.* Prentice-Hall, Inc., 1967. ISBN: 0-13-165563-9

Proof of undecidability: proof idea

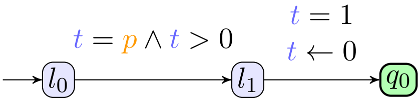
- Each instruction is encoded using a PTA fragment ("gadget")

- The counters are encoded using clocks t, x_1 and x_2 and one rational-valued parameter p (typically with values in $[0, 1]$)
- We have the following relations with the values c_1 and c_2 of counters C_1 and C_2: when $t = 0$, we have $x_1 = 1 - pc_1$ and $x_2 = 1 - pc_2$

Many proofs exist in the literature [And19]
Here: proof using one rational-valued parameter [ALM20]

- [And19] Étienne André. « What's decidable about parametric timed automata? » In: *International Journal on Software Tools for Technology Transfer* 21.2 (Apr. 2019), pp. 203–219
- [ALM20] Étienne André, Didier Lime, and Nicolas Markey. « Language Preservation Problems in Parametric Timed Automata ». In: *Logical Methods in Computer Science* 16.1 (Jan. 2020)

Proof of undecidability: initial gadget

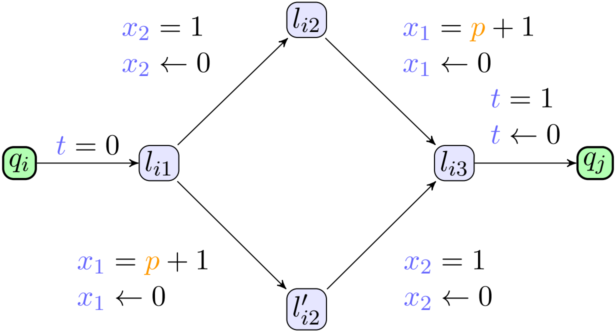


After this gadget:

- $t = 0, x_1 = x_2 = 1$
- $x_1 = 1 - pc_1 \implies c_1 = \frac{1-x_1}{p} = 0$
- $x_2 = 1 - pc_2 \implies c_2 = \frac{1-x_2}{p} = 0$
- Both counters are initially 0

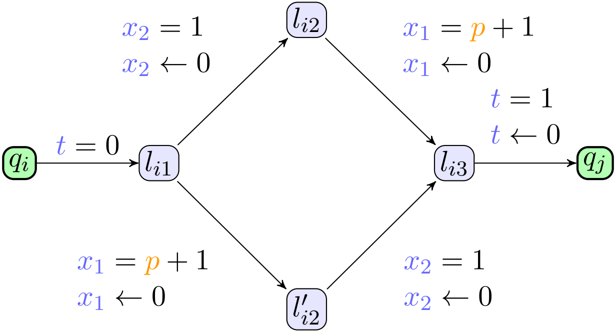
Proof of undecidability: increment gadget for C_1

"when in state q_i, increment C_1 and go to q_j"



Proof of undecidability: increment gadget for C_1

"when in state q_i, increment C_1 and go to q_j"

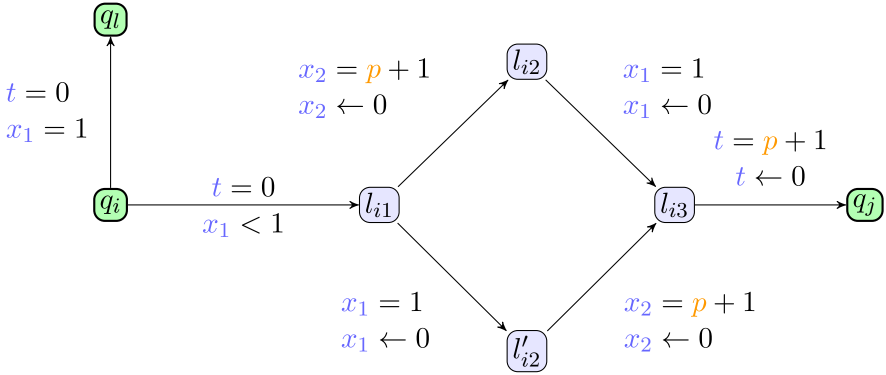


After the gadget is traversed:

- x_2 is unchanged
- x_1 is p time units smaller than before
  $\implies c'_1 = \frac{1-(x_1-p)}{p} = \frac{1-x_1+p}{p} = \frac{1-x_1}{p} + 1 = c_1 + 1$

Proof of undecidability: decrement gadget for C_1

"when in state q_i, if $C_1 = 0$ then go to q_l, otherwise decrement C_1 and go to q_j"



Undecidability of AF

Unavoidability (AF): "all runs eventually reach a target location"

Theorem (undecidability [JLR15])

Unavoidability-emptiness is undecidable for PTAs with at least 3 clocks.

Proof.

By reduction from the halting problem of a 2-counter machine, which is undecidable. □

• [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

A long list of undecidability results for PTA (1/2)

- EF-emptiness problem
  "Is the set of parameter valuations for which a given location ℓ is reachable empty?"
  undecidable [AHV93]

- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601
- [ALR22] Étienne André, Didier Lime, and Olivier H. Roux. « Reachability and liveness in parametric timed automata ». In: *Logical Methods in Computer Science* 18.1 (Feb. 2022), 31:1–31:41
- [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

A long list of undecidability results for PTA (1/2)

- EF-emptiness problem
  "Is the set of parameter valuations for which a given location ℓ is reachable empty?"
  undecidable [AHV93]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location ℓ?"
  undecidable [ALR22]

- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601
- [ALR22] Étienne André, Didier Lime, and Olivier H. Roux. « Reachability and liveness in parametric timed automata ». In: *Logical Methods in Computer Science* 18.1 (Feb. 2022), 31:1–31:41
- [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

A long list of undecidability results for PTA (1/2)

- EF-emptiness problem
  "Is the set of parameter valuations for which a given location ℓ is reachable empty?"
  undecidable [AHV93]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location ℓ?"
  undecidable [ALR22]

- AF-emptiness problem
  "Is the set of parameter valuations for which all runs eventually reach a given location ℓ empty?"
  undecidable [JLR15]

- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601
- [ALR22] Étienne André, Didier Lime, and Olivier H. Roux. « Reachability and liveness in parametric timed automata ». In: *Logical Methods in Computer Science* 18.1 (Feb. 2022), 31:1–31:41
- [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

A long list of undecidability results for PTA (2/2)

- AF-universality problem
  "Do all parameter valuations allow to reach a given location ℓ for all runs?"
  undecidable [ALR22]

- [ALR22] Étienne André, Didier Lime, and Olivier H. Roux. « Reachability and liveness in parametric timed automata ». In: *Logical Methods in Computer Science* 18.1 (Feb. 2022), 31:1–31:41
- [ALM20] Étienne André, Didier Lime, and Nicolas Markey. « Language Preservation Problems in Parametric Timed Automata ». In: *Logical Methods in Computer Science* 16.1 (Jan. 2020)
- [And19] Étienne André. « What's decidable about parametric timed automata? » In: *International Journal on Software Tools for Technology Transfer* 21.2 (Apr. 2019), pp. 203–219

A long list of undecidability results for PTA (2/2)

- AF-universality problem
  "Do all parameter valuations allow to reach a given location ℓ for all runs?"
  undecidable [ALR22]

- Preservation of the untimed language
  "Given a parameter valuation, does there exist another valuations with the same untimed language?"
  undecidable [ALM20]

- [ALR22] Étienne André, Didier Lime, and Olivier H. Roux. « Reachability and liveness in parametric timed automata ». In: *Logical Methods in Computer Science* 18.1 (Feb. 2022), 31:1–31:41
- [ALM20] Étienne André, Didier Lime, and Nicolas Markey. « Language Preservation Problems in Parametric Timed Automata ». In: *Logical Methods in Computer Science* 16.1 (Jan. 2020)
- [And19] Étienne André. « What's decidable about parametric timed automata? » In: *International Journal on Software Tools for Technology Transfer* 21.2 (Apr. 2019), pp. 203–219

A long list of undecidability results for PTA (2/2)

- AF-universality problem
  "Do all parameter valuations allow to reach a given location ℓ for all runs?"
  undecidable [ALR22]

- Preservation of the untimed language
  "Given a parameter valuation, does there exist another valuations with the same untimed language?"
  undecidable [ALM20]

In fact most interesting problems for PTAs are undecidable

[And19]

. [ALR22] Étienne André, Didier Lime, and Olivier H. Roux. « Reachability and liveness in parametric timed automata ». In: *Logical Methods in Computer Science* 18.1 (Feb. 2022), 31:1–31:41

. [ALM20] Étienne André, Didier Lime, and Nicolas Markey. « Language Preservation Problems in Parametric Timed Automata ». In: *Logical Methods in Computer Science* 16.1 (Jan. 2020)

. [And19] Étienne André. « What's decidable about parametric timed automata? » In: *International Journal on Software Tools for Technology Transfer* 21.2 (Apr. 2019), pp. 203–219

Outline

Low-dimension undecidability

Reachability-emptiness remains undecidable…

- ☹ for as few as 3 clocks [AHV93]
- ☹ even when only strict inequalities are used [Doy07]
- ☹ for a single parametric clock (and 3 non-parametric clocks) over dense time [Mil00]

Parametric clock: compared to a parameter at least once

- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601
- [Doy07] Laurent Doyen. « Robust Parametric Reachability for Timed Automata ». In: *Information Processing Letters* 102.5 (2007), pp. 208–213
- [Mil00] Joseph S. Miller. « Decidability and Complexity Results for Timed Automata and Semi-linear Hybrid Automata ». In: *HSCC*. vol. 1790. Lecture Notes in Computer Science. Springer, 2000, pp. 296–309. ISBN: 3-540-67259-1

Low-dimension decidability

Reachability-emptiness is decidable for...

- ☺ Only 1 clock and arbitrarily many rational-valued parameters [AHV93] [ALM20]
- ☺ A single parametric clock with arbitrarily many non-parametric clocks and integer-valued parameters [Ben+15]
- ☺ Only 2 clocks over discrete time and one parameter [BO17] [GH21]

- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601
- [ALM20] Étienne André, Didier Lime, and Nicolas Markey. « Language Preservation Problems in Parametric Timed Automata ». In: *Logical Methods in Computer Science* 16.1 (Jan. 2020)
- [Ben+15] Nikola Beneš, Peter Bezděk, Kim Guldstrand Larsen, and Jiří Srba. « Language Emptiness of Continuous-Time Parametric Timed Automata ». In: *ICALP, Part II*. vol. 9135. Lecture Notes in Computer Science. Springer, July 2015, pp. 69–81
- [BO17] Daniel Bundala and Joël Ouaknine. « On parametric timed automata and one-counter machines ». In: *Information and Computation* 253 (2017), pp. 272–303
- [GH21] Stefan Göller and Mathieu Hilaire. « Reachability in Two-Parametric Timed Automata with One Parameter Is EXPSPACE-Complete ». In: *STACS*. vol. 187. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 36:1–36:18

Main open cases

The decidability of reachability-emptiness is open for…

- 2 clocks and > 1 parameters over discrete time
- 2 clocks and ≥ 1 integer-valued parameter(s) over dense time
- 2 parametric clocks and 0 or 1 non-parametric clock over dense time
- 1 parametric clock and 1 or 2 non-parametric clock(s) over dense time

See survey [And19]

- [And19] Étienne André. « What's decidable about parametric timed automata? » In: *International Journal on Software Tools for Technology Transfer* 21.2 (Apr. 2019), pp. 203–219

Outline

L/U-PTAs: definition

Definition (L/U-PTA [Hun+02])

An L/U-PTA (lower-bound/upper-bound PTA) is a PTA where the set of parameters is partitioned into lower-bound parameters and upper-bound parameters, where each upper-bound (resp. lower-bound) parameter p_i must be such that, for every guard or invariant constraint $x \sim \sum_{1 \leq i \leq M} \alpha_i p_i + d$, we have:

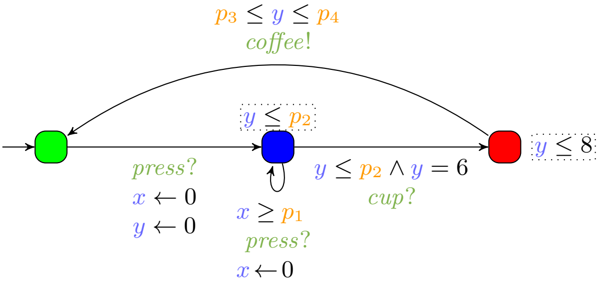
- $\sim \in \{\leq, <\}$ implies $\alpha_i \geq 0$ (resp. $\alpha_i \leq 0$), and
- $\sim \in \{\geq, >\}$ implies $\alpha_i \leq 0$ (resp. $\alpha_i \geq 0$).

Intuition:

- lower-bound parameters are compared to clocks only as lower bounds
- upper-bound parameters are compared to clocks only as upper bounds

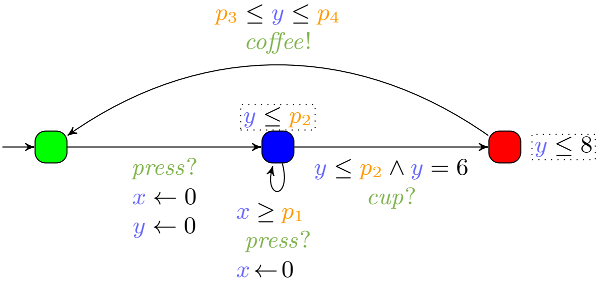
- [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220

L/U-PTAs: example



Is this an L/U-PTA?

L/U-PTAs: example

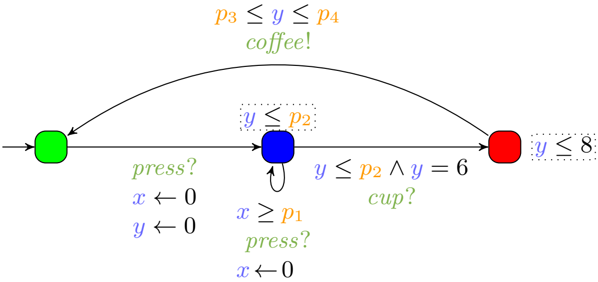


Is this an L/U-PTA?

Lower-bound parameters:

Upped-bound parameters:

L/U-PTAs: example

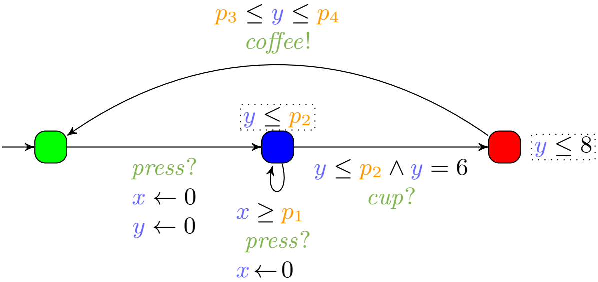


Is this an L/U-PTA?

Lower-bound parameters:

Upped-bound parameters:

L/U-PTAs: example



Is this an L/U-PTA?

Lower-bound parameters:

Upped-bound parameters:

What can we do with L/U-PTAs?

In an L/U PTA, can we syntactically...

- use an equality ($=$) in a guard or invariant?

- use an equality $x = p$ in a guard or invariant?

What can we do with L/U-PTAs?

In an L/U PTA, can we syntactically...

- use an equality $(=)$ in a guard or invariant?

- use an equality $x = p$ in a guard or invariant?

What can we do with L/U-PTAs?

In an L/U PTA, can we syntactically…

- use an equality $(=)$ in a guard or invariant?

- use an equality $x = p$ in a guard or invariant?

What fits into the class of L/U-PTAs?

- Any model with parametric delays given in the form of intervals
  - E.g.: $[p_{min}, p_{max}]$

- Many communication protocols

- All hardware circuits modeled using a bi-bounded inertial delay model

L/U-PTAs: monotonicity

Intuition

Increasing an upper-bound parameter or decreasing a lower-bound parameter can only add behaviors.

L/U-PTAs: decidability of reachability-emptiness

Theorem (decidability [Hun+02])

Reachability-emptiness is decidable for L/U-PTAs.

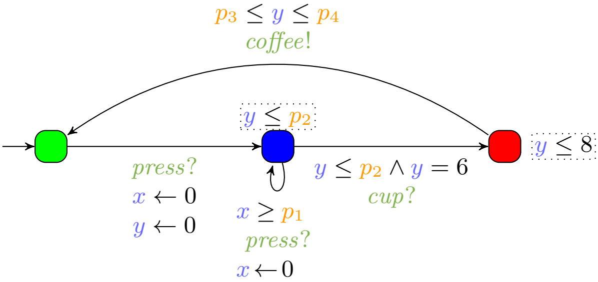
Proof intuition.

1. Intuition : valuate all upper-bound parameters with ∞ and all lower-bound parameters with 0.
2. One gets the maximum set of behaviors.
3. If the target location is unreachable, then it cannot be reached for any other valuation.
4. If the target location is reachable, there exists a concrete valuation ($\neq \infty$ for upper-bound parameters) for which it is reachable.

□

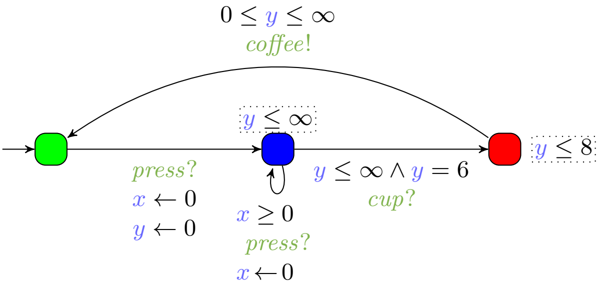
. [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220

L/U-PTAs: decidability of reachability-emptiness (example)



Does EF-emptiness hold?

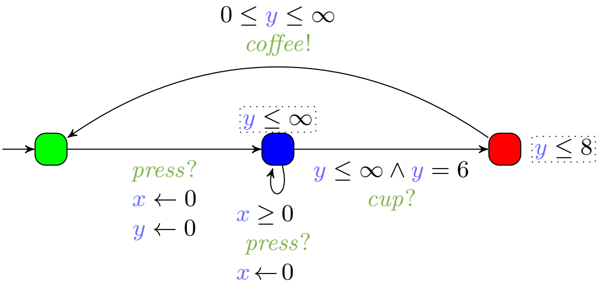
L/U-PTAs: decidability of reachability-emptiness (example)



Does EF-emptiness hold?

1. Valuate upper-bound parameters with ∞ and lower-bound parameters with 0

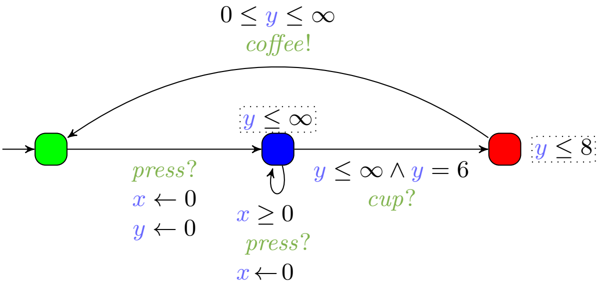
L/U-PTAs: decidability of reachability-emptiness (example)



Does EF-emptiness hold?

1. Valuate upper-bound parameters with ∞ and lower-bound parameters with 0
2. Check EF on the resulting TA

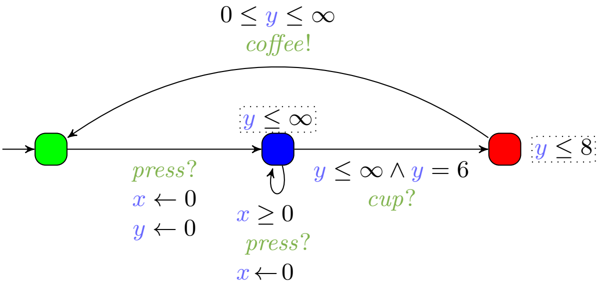
L/U-PTAs: decidability of reachability-emptiness (example)



Does EF-emptiness hold?

1. Valuate upper-bound parameters with ∞ and lower-bound parameters with 0
2. Check EF on the resulting TA

L/U-PTAs: decidability of reachability-emptiness (example)



Does EF-emptiness hold?

1. Valuate upper-bound parameters with ∞ and lower-bound parameters with 0
2. Check EF on the resulting TA

L/U-PTAs: decidability of reachability-universality

Theorem (decidability [Hun+02])

Reachability-universality is decidable for L/U-PTAs.

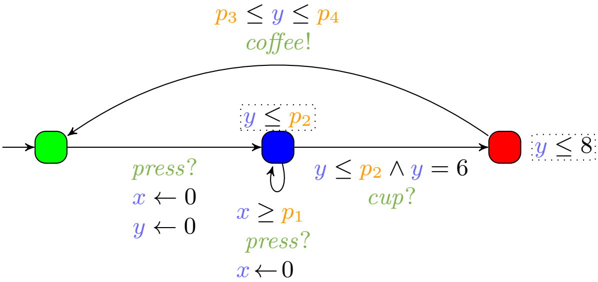
Proof intuition.

1. Dual to reachability-emptiness: valuate all upper-bound parameters with 0 and all lower-bound parameters with ∞.
2. One gets the minimum set of behaviors.
3. If the target location is reachable, then it is reachable for all valuations.

□

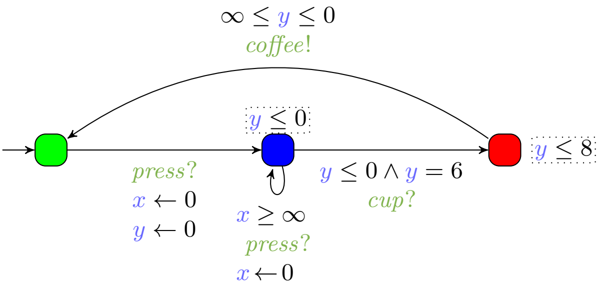
• [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220

L/U-PTAs: decidability of reachability-universality (example)



Does EF-universality hold?

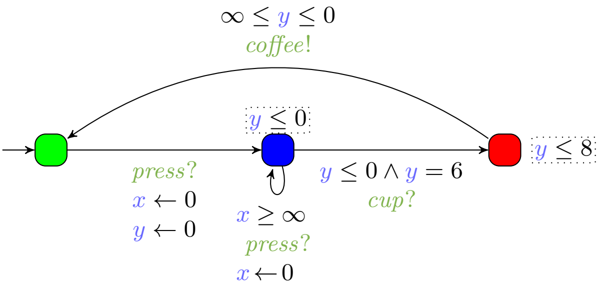
L/U-PTAs: decidability of reachability-universality (example)



Does EF-universality hold?

1. Valuate upper-bound parameters with 0 and lower-bound parameters with ∞

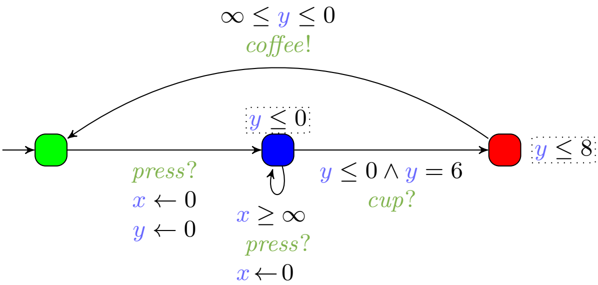
L/U-PTAs: decidability of reachability-universality (example)



Does EF-universality hold?

1. Valuate upper-bound parameters with 0 and lower-bound parameters with ∞
2. Check EF on the resulting TA

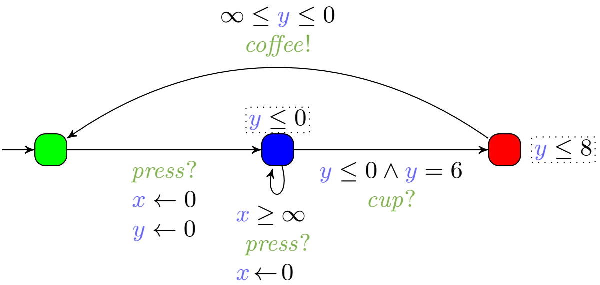
L/U-PTAs: decidability of reachability-universality (example)



Does EF-universality hold?

1. Valuate upper-bound parameters with 0 and lower-bound parameters with ∞
2. Check EF on the resulting TA

L/U-PTAs: decidability of reachability-universality (example)



Does EF[red location]-universality hold?

1. Valuate upper-bound parameters with 0 and lower-bound parameters with ∞
2. Check EF[red location] on the resulting TA

L/U-PTAs: decidability of liveness

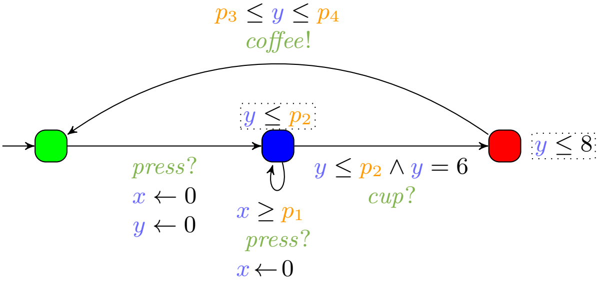
Büchi-emptiness ("liveness"): emptiness of the valuation set for which there exists a run visiting infinitely often a given set of locations

Theorem (decidability [BL09])

Büchi-emptiness is decidable for L/U-PTAs.

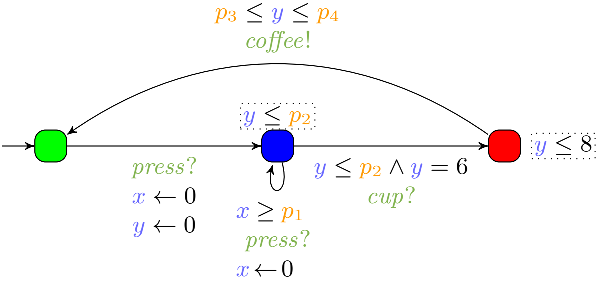
[BL09] Laura Bozzelli and Salvatore La Torre. « Decision problems for lower/upper bound parametric timed automata ». In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151

L/U-PTAs: example



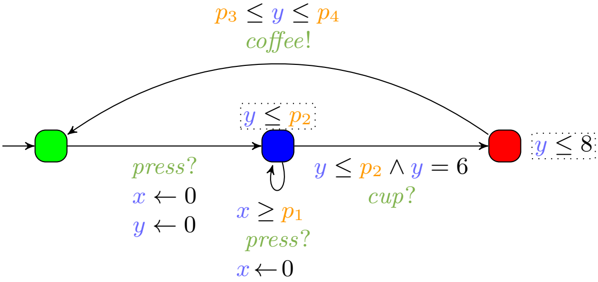
Does Büchi-emptiness holds (with  as goal location)?

L/U-PTAs: example



Does Büchi-emptiness holds (with  as goal location)?

L/U-PTAs: example



Does Büchi-emptiness holds (with  as goal location)?

Decidable problems for L/U-PTAs

- EF-emptiness problem
  "Is the set of parameter valuations allowing to reach a given location ℓ empty?"
  decidable [Hun+02]

- [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220
- [BL09] Laura Bozzelli and Salvatore La Torre. « Decision problems for lower/upper bound parametric timed automata ». In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151
- [BL09] Laura Bozzelli and Salvatore La Torre. « Decision problems for lower/upper bound parametric timed automata ». In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151

Decidable problems for L/U-PTAs

- EF-emptiness problem
  "Is the set of parameter valuations allowing to reach a given location ℓ empty?"
  decidable [Hun+02]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location ℓ?"
  decidable [BL09]

- [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220
- [BL09] Laura Bozzelli and Salvatore La Torre. « Decision problems for lower/upper bound parametric timed automata ». In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151
- [BL09] Laura Bozzelli and Salvatore La Torre. « Decision problems for lower/upper bound parametric timed automata ». In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151

Decidable problems for L/U-PTAs

- EF-emptiness problem
  "Is the set of parameter valuations allowing to reach a given location ℓ empty?"
  decidable [Hun+02]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location ℓ?"
  decidable [BL09]

- EF-finiteness problem
  "Is the set of parameter valuations allowing to reach a given location ℓ finite?"
  decidable (for integer valuations) [BL09]

. [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220

. [BL09] Laura Bozzelli and Salvatore La Torre. « Decision problems for lower/upper bound parametric timed automata ». In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151

. [BL09] Laura Bozzelli and Salvatore La Torre. « Decision problems for lower/upper bound parametric timed automata ». In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151

L/U-PTAs: intractability of synthesis

Theorem (intractability of synthesis [JLR15])

Reachability-synthesis is intractable for L/U-PTAs.

For example: the result cannot be represented using a finite union of polyhedra.

- [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

Undecidable problems for L/U-PTA (1/2)

- AF-emptiness problem
  "Is the set of parameter valuations for which a given location ℓ is always eventually reachable empty?"
  undecidable [JLR15]

- [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461
- [ALR22] Étienne André, Didier Lime, and Olivier H. Roux. « Reachability and liveness in parametric timed automata ». In: *Logical Methods in Computer Science* 18.1 (Feb. 2022), 31:1–31:41
- [ALM20] Étienne André, Didier Lime, and Nicolas Markey. « Language Preservation Problems in Parametric Timed Automata ». In: *Logical Methods in Computer Science* 16.1 (Jan. 2020)

Undecidable problems for L/U-PTA (1/2)

- AF-emptiness problem
  "Is the set of parameter valuations for which a given location ℓ is always eventually reachable empty?"
  undecidable [JLR15]

- AF-universality problem
  "Are all valuations such that a given location ℓ is always eventually reachable?"
  - decidable for closed bounded parameter domains
  - undecidable otherwise [ALR22]

- [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461
- [ALR22] Étienne André, Didier Lime, and Olivier H. Roux. « Reachability and liveness in parametric timed automata ». In: *Logical Methods in Computer Science* 18.1 (Feb. 2022), 31:1–31:41
- [ALM20] Étienne André, Didier Lime, and Nicolas Markey. « Language Preservation Problems in Parametric Timed Automata ». In: *Logical Methods in Computer Science* 16.1 (Jan. 2020)

Undecidable problems for L/U-PTA (1/2)

- AF-emptiness problem
  "Is the set of parameter valuations for which a given location ℓ is always eventually reachable empty?"
  undecidable [JLR15]

- AF-universality problem
  "Are all valuations such that a given location ℓ is always eventually reachable?"
  - decidable for closed bounded parameter domains
  - undecidable otherwise [ALR22]

- language preservation emptiness problem
  "Given a parameter valuation v, can we find another valuation with the same untimed language?"
  undecidable [ALM20]

- [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461
- [ALR22] Étienne André, Didier Lime, and Olivier H. Roux. « Reachability and liveness in parametric timed automata ». In: *Logical Methods in Computer Science* 18.1 (Feb. 2022), 31:1–31:41
- [ALM20] Étienne André, Didier Lime, and Nicolas Markey. « Language Preservation Problems in Parametric Timed Automata ». In: *Logical Methods in Computer Science* 16.1 (Jan. 2020)

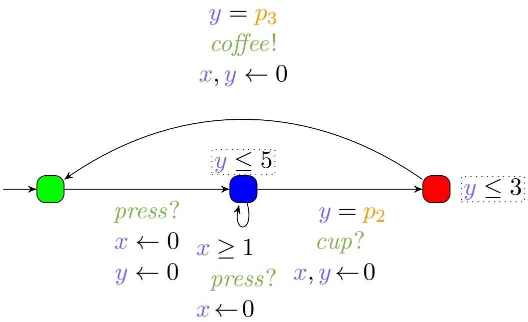
Undecidable problems for L/U-PTA (2/2)

- full TCTL-emptiness problem
  "Is the set of parameter valuations for which a TCTL formula holds empty?"
  undecidable [ALR18]
  …even in the restricted class of U-PTAs
  - U-PTAs: only upper-bound parameters
  - formula giving undecidability: $\mathsf{EG}(\mathsf{AF}_{=0}$  $)$

. [ALR18] Étienne André, Didier Lime, and Mathias Ramparison. « TCTL model checking lower/upper-bound parametric timed automata without invariants ». In: *FORMATS*. vol. 11022. Lecture Notes in Computer Science. Springer, 2018, pp. 1–17

Two promising subclasses (??) (1/2)

- reset-PTA [ALR22]
  - Principle: "every time a clock is compared to a parameter, all clocks must be reset"
    - Reminiscent of initialized rectangular automata [Hen+98]
  - EF-emptiness problem is decidable (for a bounded domain)
  - exact synthesis is possible

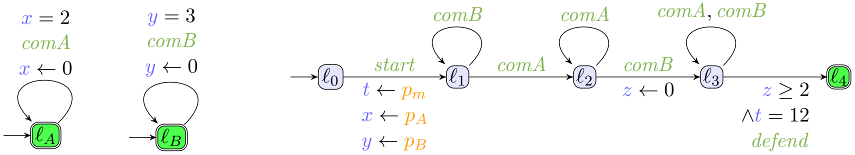


. [ALR22] Étienne André, Didier Lime, and Olivier H. Roux. « Reachability and liveness in parametric timed automata ». In: *Logical Methods in Computer Science* 18.1 (Feb. 2022), 31:1–31:41

. [Hen+98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. « What's Decidable about Hybrid Automata? » In: *Journal of Computer and System Sciences* 57.1 (1998), pp. 94–124

Two promising subclasses (??) (2/2)

- P-reset-TA [ALR21]
  - Principle: "parameters can only be used in resets" $x \leftarrow p$
  - EF-emptiness is undecidable for bounded rational parameters
  - EF-emptiness is decidable for (un)bounded integer parameters
    - in contrast to PTAs!
  - exact synthesis is possible
    - in contrast to L/U-PTAs!
  - Seems to allow for interesting applications



. [ALR21] Étienne André, Didier Lime, and Mathias Ramparison. « Parametric updates in parametric timed automata ». In: *Logical Methods in Computer Science* 17.2 (May 2021), 13:1–13:67

Outline

Outline

PTCTL (Parametric PTCTL) [BR07]

PTCTL expresses formulas on the order and the time between the future atomic propositions for some or for all paths, over a set of atomic propositions AP and involving timing parameters

Definition (Syntax of PTCTL)

$$PTCTL \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}\varphi\mathsf{U}_{\sim c}\psi \mid \mathsf{A}\varphi\mathsf{U}_{\sim c}\psi$$

where $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{Q}_+ \cup P$

Example

- $\mathsf{AG}($ $\implies \mathsf{EF}_{\leq p_1}$$)$
- $\mathsf{AF}(\mathsf{AG}_{\leq p_1}$$)$

[BR07] Véronique Bruyère and Jean-François Raskin. « Real-Time Model-Checking: Parameters everywhere ». In: *Logical Methods in Computer Science* 3.1:7 (2007), pp. 1–30

PTCTL: Examples

- "Whatever happens, the train will never crash in the next p time units"

PTCTL: Examples

- "Whatever happens, the train will never crash in the next p time units"

- "I will get a job in a month $\pm\epsilon$"

PTCTL: Examples

- “Whatever happens, the train will never crash in the next p time units”
- “I will get a job in a month $\pm\epsilon$”
- “Whenever a fire breaks, it is sure that the alarm will start ringing at least p seconds and at most 10 seconds later”

PTCTL: Examples

- “Whatever happens, the train will never crash in the next p time units”
- “I will get a job in a month $\pm\epsilon$”
- “Whenever a fire breaks, it is sure that the alarm will start ringing at least p seconds and at most 10 seconds later”
- “Whatever happens, I will love you for p years after we marry”

PTCTL: Examples

- "Whatever happens, the train will never crash in the next p time units"
- "I will get a job in a month $\pm\epsilon$"
- "Whenever a fire breaks, it is sure that the alarm will start ringing at least p seconds and at most 10 seconds later"
- "Whatever happens, I will love you for p years after we marry"

Outline

Non-parametric model against parametric formula

Theorem (decidability [BDR08])

The emptiness of the parameter valuation set for which a TA satisfies a PTCTL formula is decidable.

Theorem (synthesis)

The synthesis of the parameter valuations for which a TA satisfies a PTCTL formula can be effectively computed.

- [BDR08] Véronique Bruyère, Emmanuel Dall'Olio, and Jean-Francois Raskin. « Durations and parametric model-checking in timed automata ». In: *ACM Transactions on Computational Logic* 9.2 (2008), 12:1–12:23

Parameters everywhere

Parameters everywhere

What if we allow parameters both in the model and in the formula?

- Considering ≥ 3 clocks is useless as reachability-emptiness is undecidable

Parameters everywhere: undecidability

Theorem (undecidability [BR07])

The emptiness of the parameter valuation set for which a one-clock PTA over discrete time satisfies a PTCTL formula is undecidable.

- [BR07] Véronique Bruyère and Jean-François Raskin. « Real-Time Model-Checking: Parameters everywhere ». In: *Logical Methods in Computer Science* 3.1:7 (2007), pp. 1–30

Parameters everywhere: decidability

Theorem (decidability [BR07])

The emptiness of the parameter valuation set for which a one-clock PTA over discrete time satisfies a PTCTL formula where

- *equality is forbidden in* $\mathsf{EU}_{\sim\alpha}$ *formulas, and*
- $=$, $\geq$, $>$ *are forbidden in* $\mathsf{AU}_{\sim\alpha}$ *formulas*

is decidable.

. [BR07] Véronique Bruyère and Jean-François Raskin. « Real-Time Model-Checking: Parameters everywhere ». In: *Logical Methods in Computer Science* 3.1:7 (2007), pp. 1–30

Parameters everywhere: decidability

Theorem (decidability [BR07])

The emptiness of the parameter valuation set for which a one-clock PTA over discrete time satisfies a PTCTL formula where

- *equality is forbidden in* $\mathsf{EU}_{\sim\alpha}$ *formulas, and*
- $=, \geq, >$ *are forbidden in* $\mathsf{AU}_{\sim\alpha}$ *formulas*

is decidable.

☺ $\mathsf{EF}_{>p_1}$ (A  $\mathsf{U}_{<p_2+3}$ ) belongs to the decidable fragment

☹ $\mathsf{EF}_{=p_1}$  does not

. [BR07] Véronique Bruyère and Jean-François Raskin. « Real-Time Model-Checking: Parameters everywhere ». In: *Logical Methods in Computer Science* 3.1:7 (2007), pp. 1–30

Outline

Semi-algorithm

Definition (semi-algorithm)

A semi-algorithm is a procedure that may not terminate but, if it does, then its result is correct (sound and complete).

The projection operator

Definition

Given a parametric zone C, we denote by $C{\downarrow}_P$ its projection onto the set P.
This can be achieved using variable elimination techniques (e. g., using Fourier-Motzkin [Sch86]).

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

The projection operator

Definition

Given a parametric zone C, we denote by $C{\downarrow_P}$ its projection onto the set P.
This can be achieved using variable elimination techniques (e. g., using Fourier-Motzkin [Sch86]).

Example:

$(2 < x \leq p_1 \wedge p_1 \leq p_2){\downarrow_P}$

$=$

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

The projection operator

Definition

Given a parametric zone C, we denote by $C{\downarrow_P}$ its projection onto the set P. This can be achieved using variable elimination techniques (e. g., using Fourier-Motzkin [Sch86]).

Example:

$(2 < x \leq p_1 \wedge p_1 \leq p_2){\downarrow_P}$

$=$

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

The projection operator: examples

$(x_2 = 1 \land x_1 > x_2 \land x_1 \leq p_1 \land p_2 = 4){\downarrow_P}$
$=$

The projection operator: examples

$(x_2 = 1 \land x_1 > x_2 \land x_1 \leq p_1 \land p_2 = 4)\downarrow_P$

$=$

The projection operator: examples

$(x_2 = 1 \land x_1 > x_2 \land x_1 \leq p_1 \land p_2 = 4){\downarrow_P}$
$=$

$(x_1 < x_2 \land x_1 = x_3 \land x_1 > p_2 \land p_1 = p_2){\downarrow_P}$
$=$

The projection operator: examples

$(x_2 = 1 \wedge x_1 > x_2 \wedge x_1 \leq p_1 \wedge p_2 = 4){\downarrow_P}$
$=$

$(x_1 < x_2 \wedge x_1 = x_3 \wedge x_1 > p_2 \wedge p_1 = p_2){\downarrow_P}$
$=$

The projection operator: examples

$(x_2 = 1 \land x_1 > x_2 \land x_1 \leq p_1 \land p_2 = 4){\downarrow_P}$
$=$

$(x_1 < x_2 \land x_1 = x_3 \land x_1 > p_2 \land p_1 = p_2){\downarrow_P}$
$=$

$(x_1 < x_2 \land x_1 < x_3 \land x_1 < p_2 \land p_1 = x_3){\downarrow_P}$
$=$

The projection operator: examples

$(x_2 = 1 \land x_1 > x_2 \land x_1 \leq p_1 \land p_2 = 4){\downarrow_P}$
$=$

$(x_1 < x_2 \land x_1 = x_3 \land x_1 > p_2 \land p_1 = p_2){\downarrow_P}$
$=$

$(x_1 < x_2 \land x_1 < x_3 \land x_1 < p_2 \land p_1 = x_3){\downarrow_P}$
$=$

Semi-algorithm: EF-synthesis

Drafted in [AHV93], formalized in [JLR15]

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

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

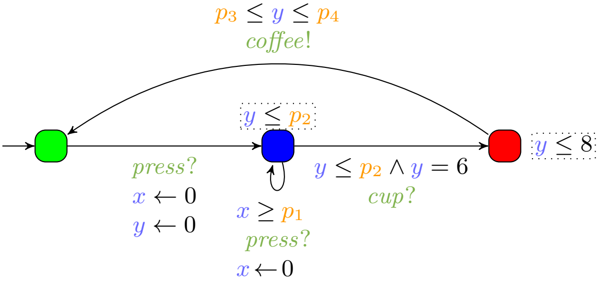
output : Parametric constraint K guaranteeing reachability

if $\ell \in G$ **then** $K \leftarrow C{\downarrow_P}$;
else
    $K \leftarrow \bot$;
    if $\mathbf{s} \notin \mathbf{Passed}$ **then**
        foreach *outgoing* e *from* ℓ *in* $\mathcal{A}$ **do**
            $K \leftarrow K \cup$ EF$\big(\mathcal{A}, \mathsf{Succ}(\mathbf{s}, e), G, \mathbf{Passed} \cup \{\mathbf{s}\}\big)$;
return K

Initial call: EF$(\mathcal{A}, \mathbf{s}_0, G, \emptyset)$

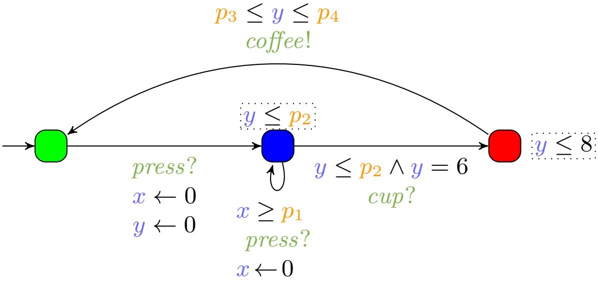
- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601
- [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

EF-synthesis: Example



What are the valuations ensuring EF?

EF-synthesis: Example



What are the valuations ensuring EF (red location)?

EF-synthesis: Correctness and completeness

Theorem ([JLR15])

Assume EF

• [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

EF-synthesis: Correctness and completeness

Theorem ([JLR15])

Assume EF *terminates with result* K.
Then

• [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

EF-synthesis: Correctness and completeness

Theorem ([JLR15])

Assume EF *terminates with result* K.

Then

Now, what happens if EF does not terminate?

. [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

EF-synthesis: Correctness and completeness

Theorem ([JLR15])

Assume EF *terminates with result* K.
Then

Now, what happens if EF does not terminate?

. [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

Semi-algorithm: AF-synthesis [JLR15]

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

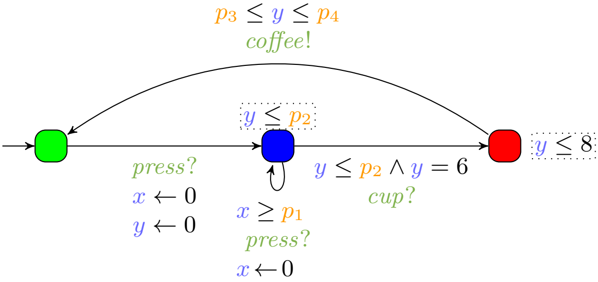
input : A bounded PTA $\mathcal{A}$, a symbolic state (ℓ, C), a set of target locations G, a set $\mathbf{Passed}$ of passed states on the current path
output : Parametric constraint K guaranteeing unavoidability

```
if ℓ ∈ G then K ← C↓_P ;
else
    if (ℓ, C) ∈ Passed then K ← ⊥ ;
    else
        K ← ⊤ ; K_Live ← ⊥;
        foreach outgoing e = (ℓ, g, a, R, ℓ') from ℓ in A do
            s' ← Succ((ℓ, C), e);
            K_Good ← AF(A, s', G, Passed ∪ {C});
            K_Block ← ⊤ \ s'↓_P;
            K ← K ∩ (K_Good ∪ K_Block);
            K_Live ← K_Live ∪ (C ∩ g)↙ ;
        K ← K \ (C \ K_Live)↓_P;
return K
```

Initial call: AF$(\mathcal{A}, \mathbf{s}_0, G, \emptyset)$

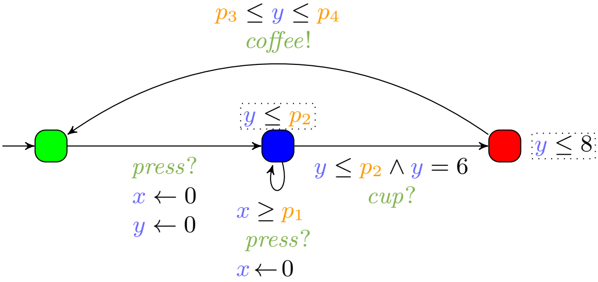
- [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

AF-synthesis: Example



What are the valuations ensuring AF?

AF-synthesis: Example

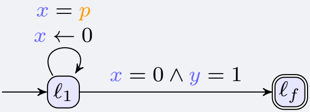


What are the valuations ensuring AF?

Outline

An open problem

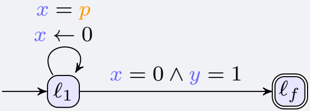
Question



What are the parameter valuations reaching ℓ_f in this PTA?

An open problem

Question

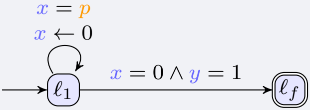


What are the parameter valuations reaching ℓ_f in this PTA?

- $p = 1$ is a solution (looping once over ℓ_1)

An open problem

Question

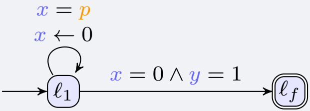


What are the parameter valuations reaching ℓ_f in this PTA?

- $p = 1$ is a solution (looping once over ℓ_1)
- $p = \frac{1}{2}$ is a solution (looping twice over ℓ_1)

An open problem

Question

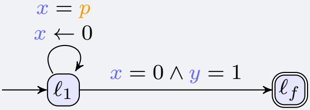


What are the parameter valuations reaching ℓ_f in this PTA?

- $p = 1$ is a solution (looping once over ℓ_1)
- $p = \frac{1}{2}$ is a solution (looping twice over ℓ_1)
- ...

An open problem

Question



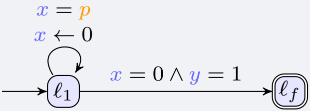
What are the parameter valuations reaching ℓ_f in this PTA?

- $p = 1$ is a solution (looping once over ℓ_1)
- $p = \frac{1}{2}$ is a solution (looping twice over ℓ_1)
- ...

Set of solutions:

An open problem

Question



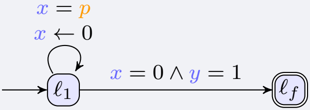
What are the parameter valuations reaching ℓ_f in this PTA?

- $p = 1$ is a solution (looping once over ℓ_1)
- $p = \frac{1}{2}$ is a solution (looping twice over ℓ_1)
- ...

Set of solutions:

An open problem

Question



What are the parameter valuations reaching ℓ_f in this PTA?

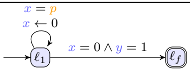
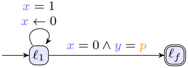
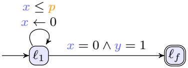
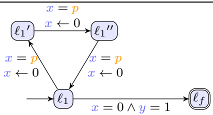
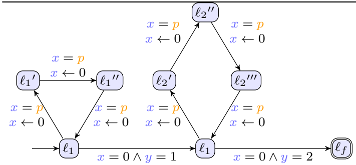
- $p = 1$ is a solution (looping once over ℓ_1)
- $p = \frac{1}{2}$ is a solution (looping twice over ℓ_1)
- ...

Set of solutions:

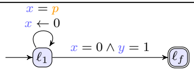
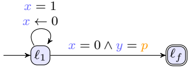
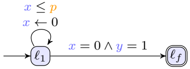
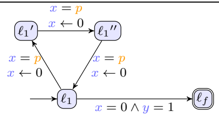
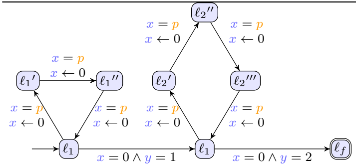
Our concrete open problem

How to compute this set automatically for this particular PTA?

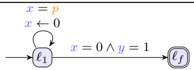
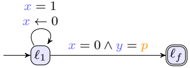
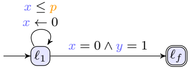
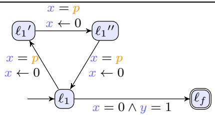
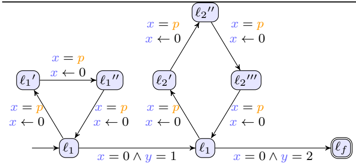
Some other concrete variants

<table>
<tr><th>Concrete model</th><th>Expected solutions</th><th>Class</th></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 rational parameter</td></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 integer parameter</td></tr>
<tr><td>
</td><td></td><td>U-PTA, 2 clocks, 1 rational parameter</td></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 rational parameter</td></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 rational parameter</td></tr>
</table>

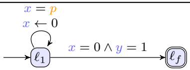
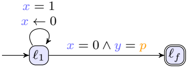
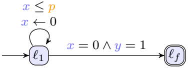
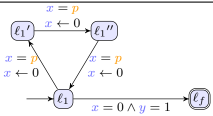
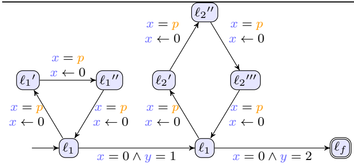
Some other concrete variants

<table>
<tr><th>Concrete model</th><th>Expected solutions</th><th>Class</th></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 rational parameter</td></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 integer parameter</td></tr>
<tr><td>
</td><td></td><td>U-PTA, 2 clocks, 1 rational parameter</td></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 rational parameter</td></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 rational parameter</td></tr>
</table>

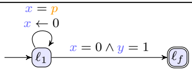
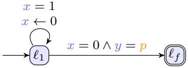
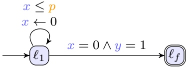
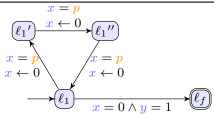
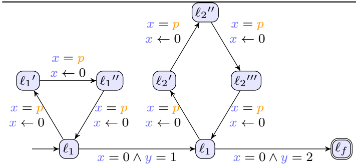
Some other concrete variants

Concrete model	Expected solutions	Class
		2 clocks, 1 rational parameter
		2 clocks, 1 integer parameter
		U-PTA, 2 clocks, 1 rational parameter
		2 clocks, 1 rational parameter
		2 clocks, 1 rational parameter

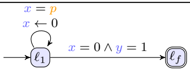
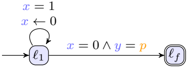
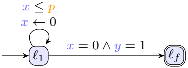
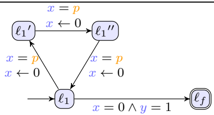
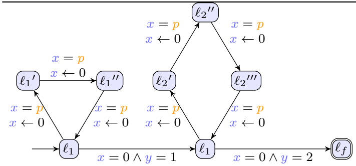
Some other concrete variants

Concrete model	Expected solutions	Class
		2 clocks, 1 rational parameter
		2 clocks, 1 integer parameter
		U-PTA, 2 clocks, 1 rational parameter
		2 clocks, 1 rational parameter
		2 clocks, 1 rational parameter

Some other concrete variants

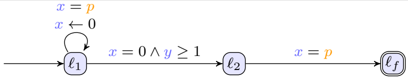
<table>
<tr><th>Concrete model</th><th>Expected solutions</th><th>Class</th></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 rational parameter</td></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 integer parameter</td></tr>
<tr><td>
</td><td></td><td>U-PTA, 2 clocks, 1 rational parameter</td></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 rational parameter</td></tr>
<tr><td>
</td><td></td><td>2 clocks, 1 rational parameter</td></tr>
</table>

Some other concrete variants

Concrete model	Expected solutions	Class
		2 clocks, 1 rational parameter
		2 clocks, 1 integer parameter
		U-PTA, 2 clocks, 1 rational parameter
		2 clocks, 1 rational parameter
		2 clocks, 1 rational parameter

Bonus: minimal-time reachability

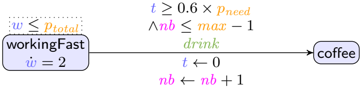
Additional problem: what is the minimum time over all parameter valuations for which the target location is reachable?

Concrete model	Expected solutions	Class
	> 1	2 clocks, 1 rational parameter

Outline

Input syntax

- Text-based (originally inspired by HyTech)
- Human-friendly




```
loc workingFast: invariant w <= pTotal flow{w' = 2}
  when t >= 0.6 * pNeed & nb <= max - 1 sync drink do {t := 0,
     nb := nb + 1} goto coffee;
```

- Conversions to other formats
  - Uppaal [LPY97] (losing parameters!)
  - JANI [Bud+17]
    - A new interchange format for automata-based formalisms

. [LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. « UPPAAL in a Nutshell ». In: *International Journal on Software Tools for Technology Transfer* 1.1-2 (1997), pp. 134–152

. [Bud+17] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. « JANI: Quantitative Model and Tool Interaction ». In: *TACAS*. vol. 10206. Lecture Notes in Computer Science. 2017, pp. 151–168

Beyond decidability

- ☺ Timed automata benefit from (some) decidability results

- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601
- [CL00] Franck Cassez and Kim Guldstrand Larsen. « The Impressive Power of Stopwatches ». In: *CONCUR*. vol. 1877. Lecture Notes in Computer Science. Springer, 2000, pp. 138–152

Beyond decidability

- ☺ Timed automata benefit from (some) decidability results
- ☹ Adding parameters yields undecidability [AHV93]
- ☹ Adding stopwatches yields undecidability [CL00]
- ☹ Adding unbounded rational variables yields undecidability

---

- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601
- [CL00] Franck Cassez and Kim Guldstrand Larsen. « The Impressive Power of Stopwatches ». In: *CONCUR*. vol. 1877. Lecture Notes in Computer Science. Springer, 2000, pp. 138–152

Beyond decidability

- ☺ Timed automata benefit from (some) decidability results
- ☹ Adding parameters yields undecidability [AHV93]
- ☹ Adding stopwatches yields undecidability [CL00]
- ☹ Adding unbounded rational variables yields undecidability

IMITATOR paradigm: "best effort"

Try to synthesize parameter valuations

- No guarantee of termination, or
- Under or over-approximations and inform the user about them
  - Evaluate whether a result is exact, over-approximated, under-approximated, or possibly invalid

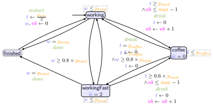
- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601
- [CL00] Franck Cassez and Kim Guldstrand Larsen. « The Impressive Power of Stopwatches ». In: *CONCUR*. vol. 1877. Lecture Notes in Computer Science. Springer, 2000, pp. 138–152

Outline

Parameter synthesis using IMITATOR 3

IMITATOR is a parametric timed model checker

An extended PTA

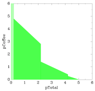


A property

`#synth AGnot(loc[researcher] = coffee)`

IMITATOR

Set of parameter valuations



Inputs

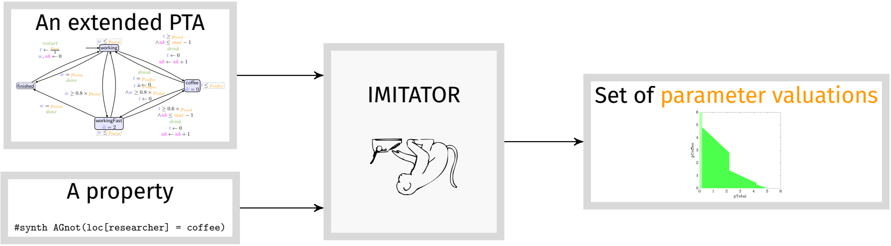
Output

The set of parameter valuations is symbolic

- Symbolic: finite set of linear constraints (polyhedra)

Parameter synthesis using IMITATOR 3

IMITATOR is a parametric timed model checker



Inputs **Output**

The set of parameter valuations is symbolic

- Symbolic: finite set of linear constraints (polyhedra)

- Two categories of properties
  - Synthesis: "(try to) synthesize all valuations for which the property holds"
  - Exhibition: "(try to) synthesize at least one valuation for which the property holds"

Safety

Synthesize all parameter valuations for which the following property holds:
"It is impossible to drink any coffee"
(i. e., the coffee location is unreachable)

```
#synth AGnot(loc[researcher] = coffee)
```

Safety

Synthesize all parameter valuations for which the following property holds:
“It is impossible to drink any coffee”
(i. e., the coffee location is unreachable)

```
#synth AGnot(loc[researcher] = coffee)
```

Result:

$$max \in [0,1) \lor \left(max \geq 1 \land p_{total} < \frac{p_{need}}{10}\right)$$

Safety: full result

```
(************************************************************
 * Result by: IMITATOR 3.0 "Cheese" (build HEAD/ea560fd)
 * Model    : 'researcher.imi'
 * Generated: Mon Feb 1, 2021 14:57:17
 * Command  : imitator3 researcher.imi researcher-AGnotcoffee.imiprop
 ************************************************************)

BEGIN CONSTRAINT
 pTotal >= 0
& pNeed >= 1
& MAXBREAK >= 0
& pCoffee >= 0
& 1 > MAXBREAK
OR
  pNeed > 10*pTotal
& pTotal >= 0
& pNeed >= 1
& MAXBREAK >= 1
& pCoffee >= 0
END CONSTRAINT

------------------------------------------------------------
Constraint soundness                    : exact
Termination                             : regular termination
------------------------------------------------------------
```

Property patterns

IMITATOR 3 offers a set of predefined property patterns

- Simple, non-compositional, commonly met
- On the system actions and parameters
- Reduce to safety or reachability synthesis
- Also called observer patterns / reachability testing [Ace+03]

[Ace+03] Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim Guldstrand Larsen. « The power of reachability testing for timed automata ». In: *Theoretical Computer Science* 300.1-3 (2003), pp. 411–475

Property patterns

IMITATOR 3 offers a set of predefined property patterns

- Simple, non-compositional, commonly met
- On the system actions and parameters
- Reduce to safety or reachability synthesis
- Also called observer patterns / reachability testing [Ace+03]

```
#synth pattern(everytime restart then eventually done within 5)
```

- [Ace+03] Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim Guldstrand Larsen. « The power of reachability testing for timed automata ». In: *Theoretical Computer Science* 300.1-3 (2003), pp. 411–475

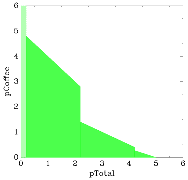
Property patterns

IMITATOR 3 offers a set of predefined property patterns

- Simple, non-compositional, commonly met
- On the system actions and parameters
- Reduce to safety or reachability synthesis
- Also called observer patterns / reachability testing [Ace+03]

```
#synth pattern(everytime restart then eventually done within 5)
```

Result projected onto 2 parameter dimensions:



• [Ace+03] Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim Guldstrand Larsen. « The power of reachability testing for timed automata ». In: *Theoretical Computer Science* 300.1-3 (2003), pp. 411–475

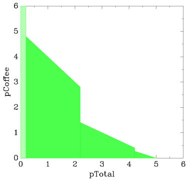
Property patterns

IMITATOR 3 offers a set of predefined property patterns

- Simple, non-compositional, commonly met
- On the system actions and parameters
- Reduce to safety or reachability synthesis
- Also called observer patterns / reachability testing [Ace+03]

```
#synth pattern(everytime restart then eventually done within 5)
```

Result projected onto 2 parameter dimensions:



IMITATOR patterns can be parameterized (e. g., `within p`)

- [Ace+03] Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim Guldstrand Larsen. « The power of reachability testing for timed automata ». In: *Theoretical Computer Science* 300.1-3 (2003), pp. 411–475

Optimal parameter reachability

Goal: synthesizing valuations for which the value of a given parameter is minimized or maximized when reaching a given state

Example: synthesize the valuations minimizing the value of p_{total} when finishing a paper after drinking (at least) 3 coffees

```
#synth EFpmin(loc[researcher] = finished & nb >= 3, pTotal)
```

Optimal parameter reachability

Goal: synthesizing valuations for which the value of a given parameter is minimized or maximized when reaching a given state

Example: synthesize the valuations minimizing the value of p_{total} when finishing a paper after drinking (at least) 3 coffees

```
#synth EFpmin(loc[researcher] = finished & nb >= 3, pTotal)
```

Result:

$$max \geq 3 \wedge p_{total} = 2.1 \wedge p_{need} = 1$$

- Note: p_{coffee} is not involved in this constraint: the time spent in drinking coffee does not impact the total duration of the *work* (p_{total}), as the progress of clock x is stopped in coffee

Liveness

Büchi acceptance condition

Example: valuations for which there exists a run s.t. the researcher completes a paper infinitely often

```
#synth CycleThrough(loc[researcher] = finished)
```

- [NPP18] Hoang Gia Nguyen, Laure Petrucci, and Jaco van de Pol. « Layered and Collecting NDFS with Subsumption for Parametric Timed Automata ». In: *ICECCS*. IEEE Computer Society, Dec. 2018, pp. 1–9
- [And+21] Étienne André, Jaime Arias, Laure Petrucci, and Jaco van de Pol. « Iterative Bounded Synthesis for Efficient Cycle Detection in Parametric Timed Automata ». In: *TACAS*. vol. 12651. Lecture Notes in Computer Science. Springer, 2021, pp. 311–329

Liveness

Büchi acceptance condition

Example: valuations for which there exists a run s.t. the researcher completes a paper infinitely often

```
#synth CycleThrough(loc[researcher] = finished)
```

Result: True (i. e., all valuations may yield such behavior)

- [NPP18] Hoang Gia Nguyen, Laure Petrucci, and Jaco van de Pol. « Layered and Collecting NDFS with Subsumption for Parametric Timed Automata ». In: *ICECCS*. IEEE Computer Society, Dec. 2018, pp. 1–9
- [And+21] Étienne André, Jaime Arias, Laure Petrucci, and Jaco van de Pol. « Iterative Bounded Synthesis for Efficient Cycle Detection in Parametric Timed Automata ». In: *TACAS*. vol. 12651. Lecture Notes in Computer Science. Springer, 2021, pp. 311–329

Liveness

Büchi acceptance condition

Example: valuations for which there exists a run s.t. the researcher completes a paper infinitely often

```
#synth CycleThrough(loc[researcher] = finished)
```

Result: True (i. e., all valuations may yield such behavior)

In the box:

- Variants of BFS
- NDFS extended with parametric subsumption and pruning [NPP18] [And+21]

- [NPP18] Hoang Gia Nguyen, Laure Petrucci, and Jaco van de Pol. « Layered and Collecting NDFS with Subsumption for Parametric Timed Automata ». In: *ICECCS*. IEEE Computer Society, Dec. 2018, pp. 1–9
- [And+21] Étienne André, Jaime Arias, Laure Petrucci, and Jaco van de Pol. « Iterative Bounded Synthesis for Efficient Cycle Detection in Parametric Timed Automata ». In: *TACAS*. vol. 12651. Lecture Notes in Computer Science. Springer, 2021, pp. 311–329

Trace preservation (robustness)

Quantifying the admissible variations of some parameters w.r.t. the discrete (untimed) behavior

```
#synth TracePreservation(pTotal=10, pNeed=5, pCoffee=3, max=3)
```

Trace preservation (robustness)

Quantifying the admissible variations of some parameters w.r.t. the discrete (untimed) behavior

```
#synth TracePreservation(pTotal=10, pNeed=5, pCoffee=3, max=3)
```

Result:

$$\big(3\times p_{need} > p_{total} \geq 2\times p_{need} \wedge max \in [2,3)\big) \vee \big(2.1\times p_{need} > p_{total} \geq 2\times p_{need} \wedge max \geq 3\big)$$

And also…

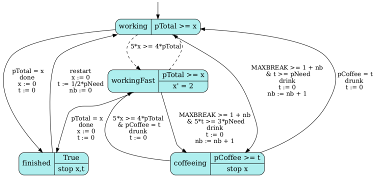
- Deadlock freeness
- Minimal-time reachability
- Parametric reachability preservation
- Behavioral cartography
- …

Results

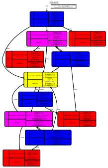
- Normalized text results

```
 * Result by: IMITATOR 3.0 "Cheese" (build HEAD/ea560fd)
 * Model     : 'researcher.imi'
 * Generated: Mon Feb 1, 2021 14:57:17
 * Command   : imitator3 researcher.imi researcher-AGnotcoffee.imiprop
 *************************************************************)

BEGIN CONSTRAINT
 pTotal >= 0
& pNeed >= 1
& MAXBREAK >= 0
& pCoffee >= 0
& 1 > MAXBREAK
OR
  pNeed > 10*pTotal
& pTotal >= 0
& pNeed >= 1
& MAXBREAK >= 1
& pCoffee >= 0
END CONSTRAINT

------------------------------------------------------------
Constraint soundness                        : exact
Termination                                 : regular termination
------------------------------------------------------------
```

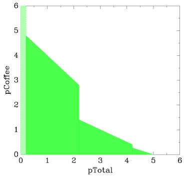
- Graphical results



PTA visualization



State space (zone graph)



Constraints representation

Outline

Under the box

Entirely written in OCaml

Strongly relies on polyhedra for symbolic computations

- Parma polyhedra library [BHZ08]

• [BHZ08] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. « The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems ». In: *Science of Computer Programming* 72.1–2 (2008), pp. 3–21

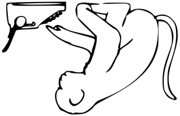
Distribution

Free and open source software: Available under the GNU-GPL license

Distribution:

- Binaries available for Linux platforms (no dependency, no install)
- Docker version
- Integrated as a virtual machine doi.org/10.5281/zenodo.4723415
- Comes with a user manual and an extensive benchmarks library [AMP21]

Try it!



www.imitator.fr

[AMP21] Étienne André, Dylan Marinho, and Jaco van de Pol. « A Benchmarks Library for Extended Timed Automata ». In: *TAP*. vol. 12740. Lecture Notes in Computer Science. Springer, 2021, pp. 39–50

Outline

Some success stories using PTAs

- Variants of train controllers [AHV93]
- The root contention protocol
- Philip's bounded retransmission protocol [Hun+02]
- An asynchronous circuit commercialized by ST-Microelectronics [Che+09]
- A 4-phase handshake protocol [KP12]
- The alternating bit protocol [JLR15]
- (non-preemptive) schedulability problems

- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601
- [Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220
- [Che+09] Rémy Chevallier, Emmanuelle Encrenaz-Tiphène, Laurent Fribourg, and Weiwen Xu. « Timed Verification of the Generic Architecture of a Memory Circuit Using Parametric Timed Automata ». In: *Formal Methods in System Design* 34.1 (Feb. 2009), pp. 59–81
- [KP12] Mirosław Kurkowski and Wojciech Penczek. « Applying Timed Automata to Model Checking of Security Protocols ». In: *Handbook of Finite State Based Models and Applications*. Chapman and Hall/CRC, 2012, pp. 223–254
- [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

Some success stories using IMITATOR

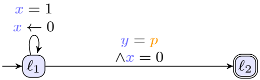
- Verification of an asynchronous memory circuit by ST-Microelectronics
- Parametric schedulability analyses for flight control systems for ASTRIUM Space Transportation / ArianeGroup [Fri+12]
- Verification of software product lines [Lut+17]
- Formal timing analysis of music scores [FJ13]
- Solution to a challenge related to a distributed video processing system by Thales [Alt+23]
- Parametric timed pattern matching and online monitoring [WAH23]

- [Fri+12] Laurent Fribourg, David Lesens, Pierre Moro, and Romain Soulat. « Robustness Analysis for Scheduling Problems using the Inverse Method ». In: *TIME*. IEEE Computer Society Press, Sept. 2012, pp. 73–80
- [Lut+17] Lars Luthmann, Andreas Stephan, Johannes Bürdek, and Malte Lochau. « Modeling and Testing Product Lines with Unbounded Parametric Real-Time Constraints ». In: *SPLC, Volume A*. ACM, 2017, pp. 104–113
- [FJ13] Léa Fanchon and Florent Jacquemard. « Formal Timing Analysis Of Mixed Music Scores ». In: *ICMC*. Michigan Publishing, Aug. 2013
- [Alt+23] Sebastian Altmeyer, Étienne André, Silvano Dal Zilio, Loïc Fejoz, Susanne Graf, J. Javier Gutiérrez, Michael González Harbour, Rafik Henia, Didier Le Botlan, Giuseppe Lipari, Julio Medina, Nicolas Navet, Sophie Quinton, Juan M. Rivas, and Youcheng Sun. « From FMTV to WATERS: Lessons Learned from the First Verification Challenge at ECRTS (invited paper) ». In: *ECRTS*. vol. 262. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 19:1–19:18. ISBN: 978-3-95977-280-8
- [WAH23] Masaki Waga, Étienne André, and Ichiro Hasuo. « Parametric Timed Pattern Matching ». In: *ACM Transactions on Software Engineering and Methodology* 32.1 (Feb. 2023), 10:1–10:35

Outline

What's not in IMITATOR?

- Solving problems not representable by a finite union of polyhedra
  - Toy benchmark for which the answer is $\{p = i, i \in \mathbb{N}\}$



- Discrete parameters (as in population protocols [AJ03])
  - "Arbitrary number of concurrent coffee drinkers"

- Integration to higher-level formalisms
  - Logics: MITL, STL
  - Real-time systems

. [AJ03] Parosh Aziz Abdulla and Bengt Jonsson. « Model checking of systems with many identical timed processes ». In: *Theoretical Computer Science* 290.1 (2003), pp. 241–264

Outline

Modeling real-time systems with timed automata

- Using timed automata [AM01]
- Using stopwatch automata [AM02]
- Using parametric timed automata [CPR08]
- Using parametric stopwatch automata [Fri+12] [Sun+13] [Lip+14]
- Using task automata [NWY99] [Fer+07] [And17]

- [AM01] Yasmina Abdeddaïm and Oded Maler. « Job-Shop Scheduling Using Timed Automata ». In: *CAV*. vol. 2102. Lecture Notes in Computer Science. Springer, 2001, pp. 478–492. ISBN: 3-540-42345-1
- [AM02] Yasmina Adbeddaïm and Oded Maler. « Preemptive Job-Shop Scheduling using Stopwatch Automata ». In: *TACAS*. vol. 2280. Lecture Notes in Computer Science. Springer-Verlag, Apr. 2002, pp. 113–126
- [CPR08] Alessandro Cimatti, Luigi Palopoli, and Yusi Ramadian. « Symbolic Computation of Schedulability Regions Using Parametric Timed Automata ». In: *RTSS*. IEEE Computer Society, 2008, pp. 80–89. ISBN: 978-0-7695-3477-0
- [Fri+12] Laurent Fribourg, David Lesens, Pierre Moro, and Romain Soulat. « Robustness Analysis for Scheduling Problems using the Inverse Method ». In: *TIME*. IEEE Computer Society Press, Sept. 2012, pp. 73–80
- [Sun+13] Youcheng Sun, Romain Soulat, Giuseppe Lipari, Étienne André, and Laurent Fribourg. « Parametric Schedulability Analysis of Fixed Priority Real-Time Distributed Systems ». In: *FSTCS*. vol. 419. Communications in Computer and Information Science. Springer, Oct. 2013, pp. 212–228
- [Lip+14] Giuseppe Lipari, Youcheng Sun, Étienne André, and Laurent Fribourg. « Toward Parametric Timed Interfaces for Real-Time Components ». In: *SynCoP*. vol. 145. Electronic Proceedings in Theoretical Computer Science. Apr. 2014, pp. 49–64
- [NWY99] Christer Norström, Anders Wall, and Wang Yi. « Timed Automata as Task Models for Event-Driven Systems ». In: *RTCSA*. IEEE Computer Society, 1999, pp. 182–189
- [Fer+07] Elena Fersman, Pavel Krcál, Paul Pettersson, and Wang Yi. « Task automata: Schedulability, decidability and undecidability ». In: *Information and Computation* 205.8 (2007), pp. 1149–1172
- [And17] Étienne André. « A unified formalism for monoprocessor schedulability analysis under uncertainty ». In: *FMICS-AVoCS*. vol. 10471. Lecture Notes in Computer Science. Springer, 2017, pp. 100–115

Modeling a periodic task T (exercise)

Periodic task T with period $periodT$

- "action $actT$ must occur every $periodT$ time units"

Modeling a periodic task T (exercise)

Periodic task T with period $periodT$

- “action $actT$ must occur every $periodT$ time units”

Modeling a periodic task T (exercise)

Periodic task T with period $periodT$

- "action $actT$ must occur every $periodT$ time units"

Periodic task T with period $periodT$ and $offsetT$

- "after $offsetT$ time units, action $actT$ must occur every $periodT$ time units"

Modeling a periodic task T (exercise)

Periodic task T with period $periodT$

- "action $actT$ must occur every $periodT$ time units"

Periodic task T with period $periodT$ and $offsetT$

- "after $offsetT$ time units, action $actT$ must occur every $periodT$ time units"

Modeling a sporadic task T (exercise)

Sporadic task T with minimum interarrival time $miaT$ and $offsetT$

- "after at least $offsetT$ time units, action actT can occur sporadically, with at least $miaT$ time units between two consecutive occurrences"

Modeling a sporadic task T (exercise)

Sporadic task T with minimum interarrival time $miaT$ and $offsetT$

- "after at least $offsetT$ time units, action actT can occur sporadically, with at least $miaT$ time units between two consecutive occurrences"

Modeling a sporadic task T (exercise)

Sporadic task T with minimum interarrival time $miaT$ and $offsetT$

- "after at least $offsetT$ time units, action actT can occur sporadically, with at least $miaT$ time units between two consecutive occurrences"

A more efficient modeling to avoid clock divergence in IMITATOR

- and hence optimize the computation

Modeling a sporadic task T (exercise)

Sporadic task T with minimum interarrival time $miaT$ and $offsetT$

- "after at least $offsetT$ time units, action actT can occur sporadically, with at least $miaT$ time units between two consecutive occurrences"

A more efficient modeling to avoid clock divergence in IMITATOR

- and hence optimize the computation

Trick: stop the computation of $xactT$ to avoid diverging

Modeling a task / pipeline

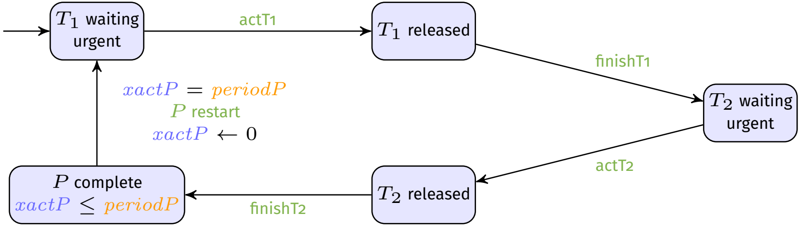
Pipeline P of two tasks T_1 and T_2

The pipeline has a period $periodP$

Modeling a task / pipeline

Pipeline P of two tasks T_1 and T_2

The pipeline has a period $periodP$

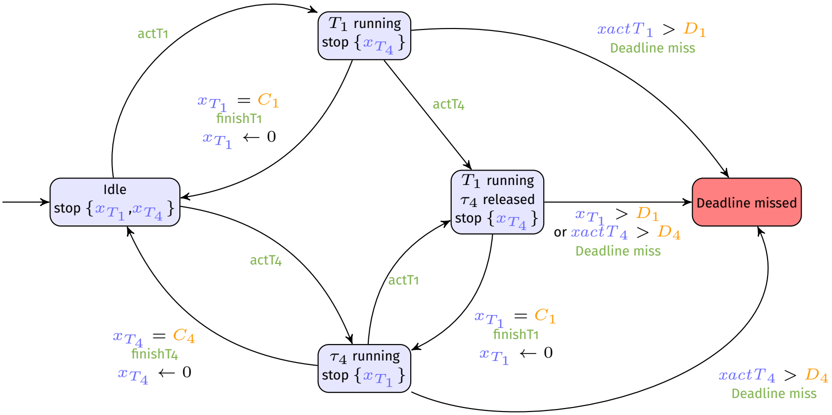


Modeling the preemptive fixed priority scheduler

A fixed-priority preemptive processor with two tasks T_1 and T_4

Timings for T_1: period $periodT1$, execution time period C_1, deadline period D_1

- and similarly for T_4



Outline

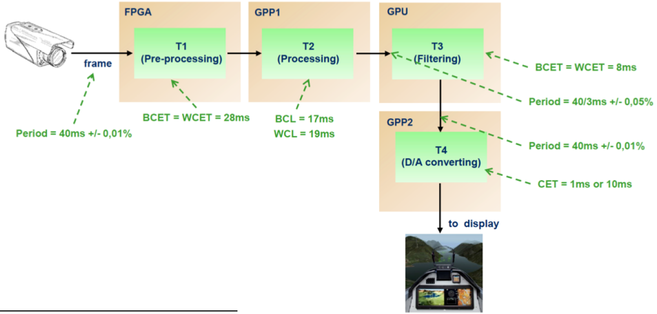
The FMTV 2015 Challenge (1/2)

Challenge by Thales proposed during the WATERS 2014 workshop

Solutions presented at WATERS 2015 [Alt+23]

System: an unmanned aerial video system with uncertain periods

- Period constant but with a small uncertainty (typically 0.01 %)
- Not a jitter!



[Alt+23] Sebastian Altmeyer, Étienne André, Silvano Dal Zilio, Loïc Fejoz, Susanne Graf, J. Javier Gutiérrez, Michael González Harbour, Rafik Henia, Didier Le Botlan, Giuseppe Lipari, Julio Medina, Nicolas Navet, Sophie Quinton, Juan M. Rivas, and Youcheng Sun. « From FMTV to WATERS: Lessons Learned from the First Verification Challenge at ECRTS (invited paper) ». In: *ECRTS*. vol. 262. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 19:1–19:18. ISBN: 978-3-95977-280-8

The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of $n = 1$ and $n = 3$

The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of $n = 1$ and $n = 3$

- ☹ Not a typical parameter synthesis problem?
  - No parameters in the specification

The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of $n = 1$ and $n = 3$

☹ Not a typical parameter synthesis problem?

- No parameters in the specification

☺ A typical parameter synthesis problem

- The end-to-end time can be set as a parameter… to be synthesized
- The uncertain period is typically a parameter (with some constraint, e. g., $P1 \in [40 - 0.004, 40 + 0.004]$)

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EF-synth (implemented in IMITATOR) w.r.t. that location

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EF-synth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EF-synth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)
5. Eliminate all parameters but the end-to-end time

Note: not eliminating parameters allows one to know for which values of the periods the best / worst case execution times are obtained.

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EF-synth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)
5. Eliminate all parameters but the end-to-end time
6. Exhibit the minimum and the maximum

Note: not eliminating parameters allows one to know for which values of the periods the best / worst case execution times are obtained.

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$
- Parameters:
  - P1_uncertain
  - P3_uncertain
  - P4_uncertain

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$
- Parameters:
  - P1_uncertain
  - P3_uncertain
  - P4_uncertain

- The end-to-end latency (another parameter): E2E

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$
- Parameters:
  - P1_uncertain
  - P3_uncertain
  - P4_uncertain

- The end-to-end latency (another parameter): E2E

- Others:
  - the register between task 2 and task 3: discrete variable $\mathrm{reg}_{2,3}$
  - the buffer between task 3 and task 4: $n = 1$ or $n = 3$

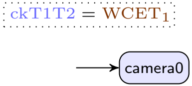
Simplification

- T1 and T2 are synchronized; T1, T3 and T4 are asynchronized
  - (exact modeling of the system behavior is too heavy)

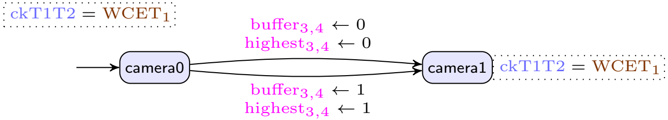
Simplification

- T1 and T2 are synchronized; T1, T3 and T4 are asynchronized
  - (exact modeling of the system behavior is too heavy)
- We choose a single arbitrary frame, called the target one
- We assume the system is initially in an arbitrary status
  - This is our only uncertain assumption (in other words, can the periods deviate from each other so as to yield any arbitrary deviation?)

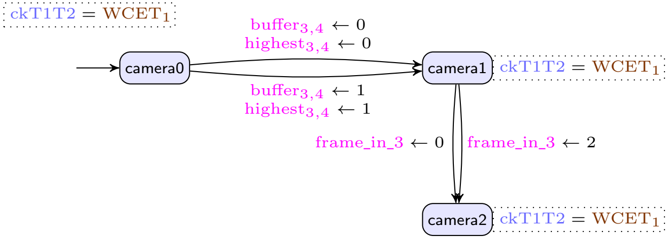
The initialization automaton



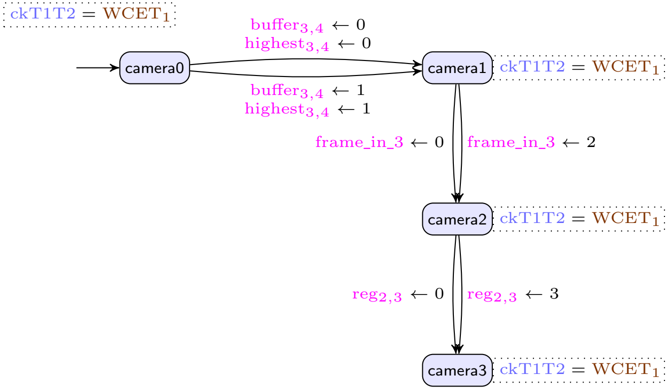
The initialization automaton



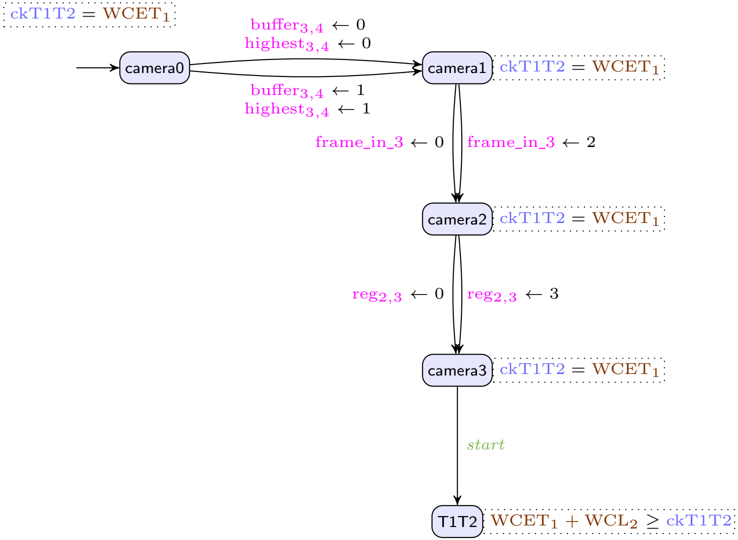
The initialization automaton



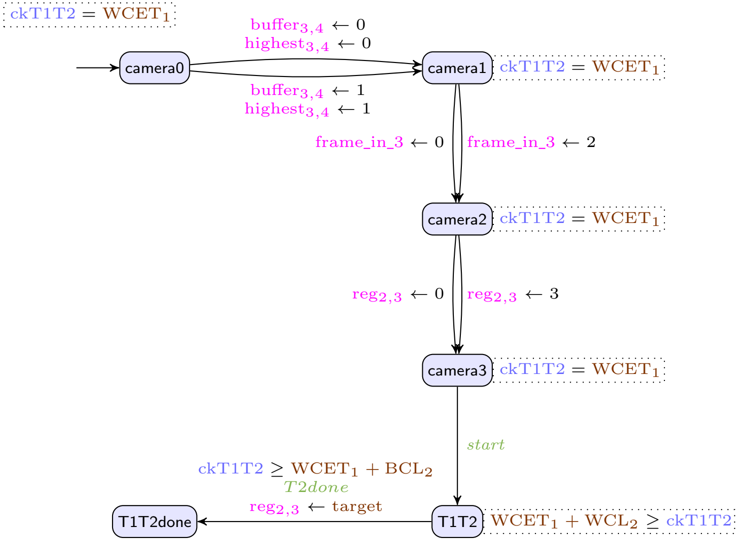
The initialization automaton



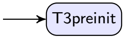
The initialization automaton



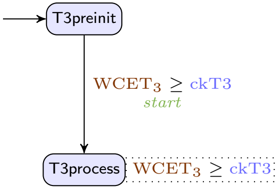
The initialization automaton



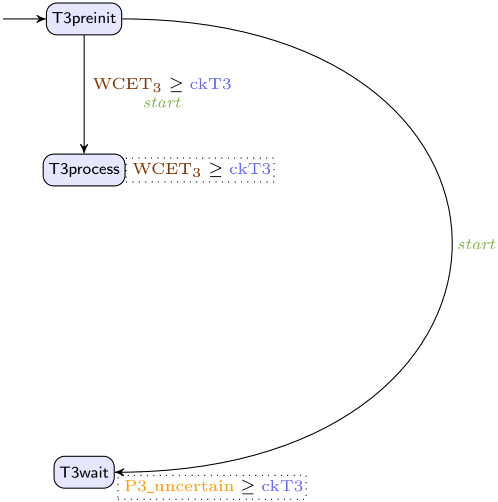
Task T3



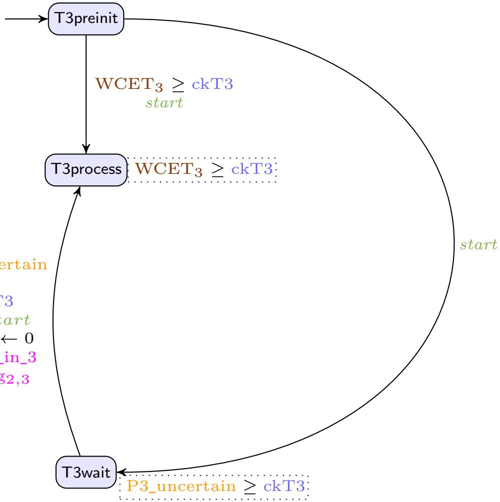
Task T3



Task T3

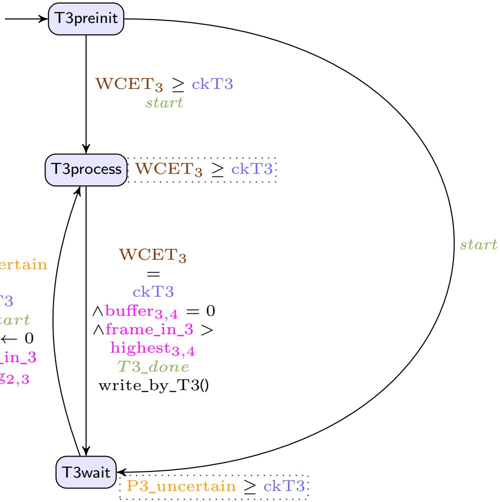

T3preinit
WCET3 ≥ ckT3
start
T3process
WCET3 ≥ ckT3
start
T3wait
P3_uncertain ≥ ckT3

Task T3

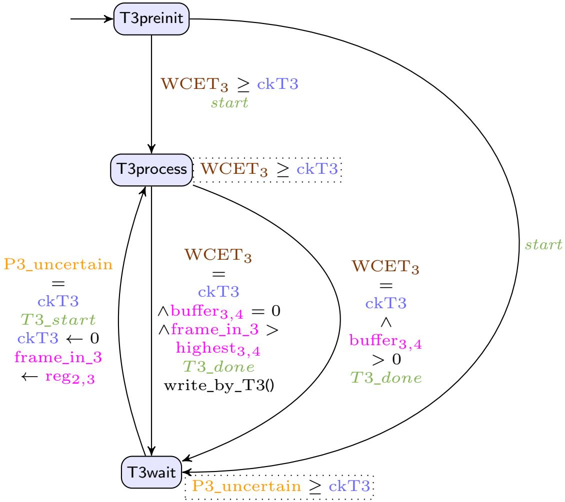


T3preinit
$\mathrm{WCET}_3 \geq \mathrm{ckT3}$
start
T3process
$\mathrm{WCET}_3 \geq \mathrm{ckT3}$
start
P3_uncertain
=
ckT3
T3_start
ckT3 ← 0
frame_in_3
← $\mathrm{reg}_{2,3}$
T3wait
P3_uncertain ≥ ckT3

Task T3

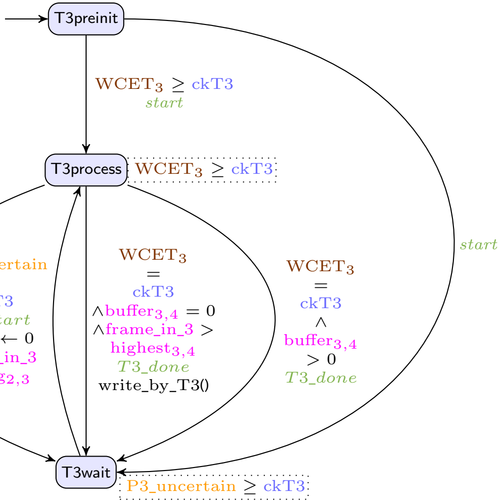

T3preinit
WCET3 ≥ ckT3
start
T3process
WCET3 ≥ ckT3
start
P3_uncertain = ckT3
T3_start
ckT3 ← 0
frame_in_3 ← reg2,3
WCET3 = ckT3 ∧buffer3,4 = 0 ∧frame_in_3 > highest3,4
T3_done
write_by_T3()
T3wait
P3_uncertain ≥ ckT3

Task T3

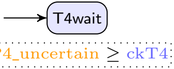


T3preinit
WCET$_3$ ≥ ckT3
start
T3process
WCET$_3$ ≥ ckT3
start
P3_uncertain = ckT3
T3_start
ckT3 ← 0
frame_in_3 ← reg$_{2,3}$
WCET$_3$ = ckT3 ∧buffer$_{3,4}$ = 0 ∧frame_in_3 > highest$_{3,4}$
T3_done
write_by_T3()
WCET$_3$ = ckT3 ∧ buffer$_{3,4}$ > 0
T3_done
T3wait
P3_uncertain ≥ ckT3

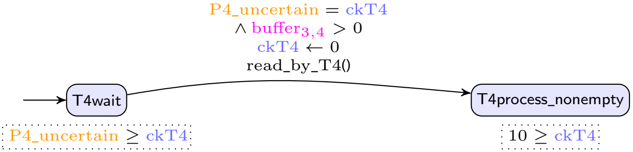
Task T3


T3preinit
WCET3 ≥ ckT3
start
T3process
WCET3 ≥ ckT3
start
WCET3 = ckT3 ∧ buffer3,4 = 0 ∧ highest3,4 ≥ frame_in_3
T3_done
P3_uncertain = ckT3
T3_start
ckT3 ← 0
frame_in_3 ← reg2,3
WCET3 = ckT3 ∧buffer3,4 = 0 ∧frame_in_3 > highest3,4
T3_done
write_by_T3()
WCET3 = ckT3 ∧ buffer3,4 > 0
T3_done
T3wait
P3_uncertain ≥ ckT3

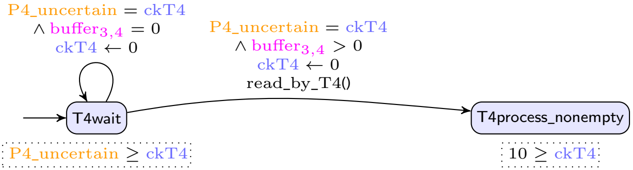
Task T4



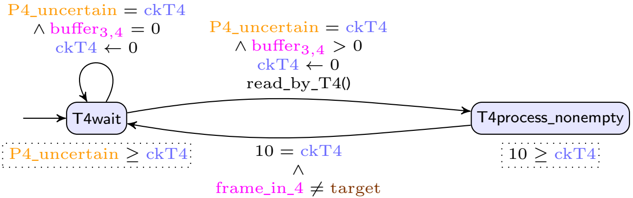
Task T4


P4_uncertain = ckT4
∧ buffer3,4 > 0
ckT4 ← 0
read_by_T4()
T4wait
T4process_nonempty
P4_uncertain ≥ ckT4
10 ≥ ckT4

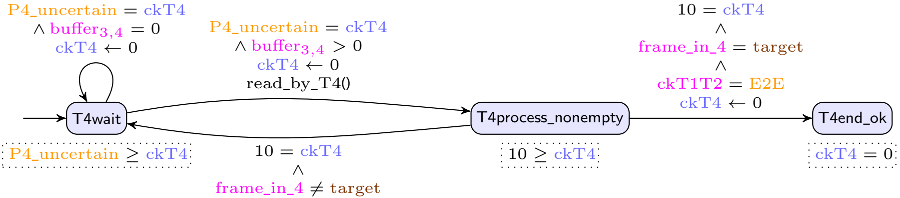
Task T4


P4_uncertain = ckT4
∧ buffer3,4 = 0
ckT4 ← 0
P4_uncertain = ckT4
∧ buffer3,4 > 0
ckT4 ← 0
read_by_T4()
T4wait
T4process_nonempty
P4_uncertain ≥ ckT4
10 ≥ ckT4

Task T4


P4_uncertain = ckT4
∧ buffer3,4 = 0
ckT4 ← 0
P4_uncertain = ckT4
∧ buffer3,4 > 0
ckT4 ← 0
read_by_T4()
T4wait
T4process_nonempty
P4_uncertain ≥ ckT4
10 = ckT4
∧
frame_in_4 ≠ target
10 ≥ ckT4

Task T4


P4_uncertain = ckT4
∧ buffer3,4 = 0
ckT4 ← 0
P4_uncertain = ckT4
∧ buffer3,4 > 0
ckT4 ← 0
read_by_T4()
10 = ckT4
∧
frame_in_4 = target
∧
ckT1T2 = E2E
ckT4 ← 0
T4wait
T4process_nonempty
T4end_ok
P4_uncertain ≥ ckT4
10 = ckT4
∧
frame_in_4 ≠ target
10 ≥ ckT4
ckT4 = 0

Results

E2E latency results for $n = 1$ and $n = 3$

	$n = 1$	$n = 3$
min E2E	63 ms	63 ms
max E2E	145.008 ms	225.016 ms

Results obtained using IMITATOR in a few seconds

[SAL15]

- [SAL15] Youcheng Sun, Étienne André, and Giuseppe Lipari. « Verification of Two Real-Time Systems Using Parametric Timed Automata ». In: *WATERS*. July 2015

Outline

Other parametric timed formalisms

- (Parametric) hybrid systems [Hen+98]
- Parametric time Petri nets [TLR09]
- Parametric timed CSP [And+14]

- [Hen+98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. « What's Decidable about Hybrid Automata? » In: *Journal of Computer and System Sciences* 57.1 (1998), pp. 94–124
- [TLR09] Louis-Marie Traonouez, Didier Lime, and Olivier H. Roux. « Parametric Model-Checking of Stopwatch Petri Nets ». In: *Journal of Universal Computer Science* 15.17 (2009), pp. 3273–3304
- [And+14] Étienne André, Yang Liu, Jun Sun, and Jin Song Dong. « Parameter Synthesis for Hierarchical Concurrent Real-Time Systems ». In: *Real-Time Systems* 50.5-6 (Sept. 2014), pp. 620–679

Software

Tool	Start	Formalism	Ref
HyTech	1997?	(Parametric) hybrid automata	[HHW97]
Roméo	2000	Parametric time Petri nets	[Lim+09]
PHAVer	2008?	(Parametric) hybrid automata	[Fre08]
IMITATOR	2008	Parametric timed automata	[And21]
SpaceEx	2009	Hybrid systems	[Fre+11]

- [HHW97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. « HyTech: A Model Checker for Hybrid Systems ». In: *International Journal on Software Tools for Technology Transfer* 1.1-2 (1997), pp. 110–122
- [Lim+09] Didier Lime, Olivier H. Roux, Charlotte Seidner, and Louis-Marie Traonouez. « Romeo: A Parametric Model-Checker for Petri Nets with Stopwatches ». In: *TACAS*. vol. 5505. Lecture Notes in Computer Science. Springer, Mar. 2009, pp. 54–57
- [Fre08] Goran Frehse. « PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech ». In: *International Journal on Software Tools for Technology Transfer* 10.3 (May 2008), pp. 263–279. ISSN: 1433-2779
- [And21] Étienne André. « IMITATOR 3: Synthesis of timing parameters beyond decidability ». In: *CAV*. vol. 12759. Lecture Notes in Computer Science. Springer, 2021, pp. 1–14
- [Fre+11] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. « SpaceEx: Scalable Verification of Hybrid Systems ». In: *CAV*. vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 379–395

Outline

Outline

Summary

- Finite-state automata
  - ☺ Mostly decidable results
  - ☺ Efficient model checking algorithms
  - ☹ Miss the quantitative aspects
  - ☺ Many powerful tools

- Timed automata
  - ☺ Finite abstract semantics
  - ☺ Some decidable results
  - ☹ Some undecidable results
  - ☺ Several powerful tools

- Parametric timed automata
  - ☺ Very expressive
  - ☹ No finite abstract semantics
  - ☹ Mostly undecidability results
    - With some decidable subclasses
  - ☺ Some powerful tools

Parametric timed automata: features and theory

Advantages of parametric timed automata

- allow to verify partially specified systems
- allow to verify classes of systems
- allow to synthesize parameter valuations so that some property holds

Parametric timed automata in theory

- a very expressive formalism
- most problems are undecidable
  - yet several decidable subclasses exist

Parametric timed automata: practice

Parametric timed automata in practice

- several semi-algorithms
- a number of success stories
- IMITATOR model checker
- library of benchmarks

Promising applications

- Real-time systems and parametric schedulability analyses
- Monitoring under uncertainty
- Analyses of opacity

Outline

Beyond (parametric) timed automata

Beyond time...

- Cost, temperature, energy
  - Hybrid automata [Alu+93] [Alu+95]
    - Very expressive, but often undecidable
    - Some interesting software (including SpaceEx [Fre+11])

Probabilities

- Useful when a property cannot be proved with full certainty
  - Communication protocols, failures...
- Another way to model systems known with limited precision

- [Alu+93] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. « Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems ». In: *Hybrid Systems 1992*. Vol. 736. Lecture Notes in Computer Science. Springer, 1993, pp. 209–229
- [Alu+95] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. « The Algorithmic Analysis of Hybrid Systems ». In: *Theoretical Computer Science* 138.1 (1995), pp. 3–34
- [Fre+11] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. « SpaceEx: Scalable Verification of Hybrid Systems ». In: *CAV*. vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 379–395

Perspectives

- Exhibit decidable subclasses of parametric timed automata
  - Good candidates: L-PTAs and U-PTAs
- Design efficient semi-algorithms with good termination conditions
  - e. g., reuse the integer hull of [JLR15]
- Broaden the use of formal methods in the industry

• [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

Perspectives

- Exhibit decidable subclasses of parametric timed automata
  - Good candidates: L-PTAs and U-PTAs
- Design efficient semi-algorithms with good termination conditions
  - e. g., reuse the integer hull of [JLR15]
- Broaden the use of formal methods in the industry
  - ☺ …and save lives!!

• [JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461

Bibliography

References I

[AAB00] Aurore Annichini, Eugene Asarin, and Ahmed Bouajjani. « Symbolic Techniques for Parametric Reasoning about Counter and Clock Systems ». In: *CAV* (July 15–19, 2000). Ed. by E. Allen Emerson and A. Prasad Sistla. Vol. 1855. Lecture Notes in Computer Science. Chicago, IL, USA: Springer-Verlag, 2000, pp. 419–434. ISBN: 3-540-67770-4. DOI: 10.1007/10722167_32.

[Ace+03] Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim Guldstrand Larsen. « The power of reachability testing for timed automata ». In: *Theoretical Computer Science* 300.1-3 (2003), pp. 411–475. DOI: 10.1016/S0304-3975(02)00334-1.

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC* (May 16–18, 1993). Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, USA: ACM, 1993, pp. 592–601. DOI: 10.1145/167088.167242.

[AJ03] Parosh Aziz Abdulla and Bengt Jonsson. « Model checking of systems with many identical timed processes ». In: *Theoretical Computer Science* 290.1 (2003), pp. 241–264. DOI: 10.1016/S0304-3975(01)00330-9.

[ALM20] Étienne André, Didier Lime, and Nicolas Markey. « Language Preservation Problems in Parametric Timed Automata ». In: *Logical Methods in Computer Science* 16.1 (Jan. 2020). DOI: 10.23638/LMCS-16(1:5)2020.

[ALR18] Étienne André, Didier Lime, and Mathias Ramparison. « TCTL model checking lower/upper-bound parametric timed automata without invariants ». In: *FORMATS* (Sept. 4–6, 2018). Ed. by David N. Jansen and Pavithra Prabhakar. Vol. 11022. Lecture Notes in Computer Science. Beijing, China: Springer, 2018, pp. 1–17. DOI: 10.1007/978-3-030-00151-3_3.

References II

[ALR21] Étienne André, Didier Lime, and Mathias Ramparison. « Parametric updates in parametric timed automata ». In: *Logical Methods in Computer Science* 17.2 (May 2021), 13:1–13:67. DOI: 10.23638/LMCS-17(2:13)2021.

[ALR22] Étienne André, Didier Lime, and Olivier H. Roux. « Reachability and liveness in parametric timed automata ». In: *Logical Methods in Computer Science* 18.1 (Feb. 2022), 31:1–31:41. DOI: 10.46298/lmcs-18(1:31)2022.

[Alt+23] Sebastian Altmeyer, Étienne André, Silvano Dal Zilio, Loïc Fejoz, Susanne Graf, J. Javier Gutiérrez, Michael González Harbour, Rafik Henia, Didier Le Botlan, Giuseppe Lipari, Julio Medina, Nicolas Navet, Sophie Quinton, Juan M. Rivas, and Youcheng Sun. « From FMTV to WATERS: Lessons Learned from the First Verification Challenge at ECRTS (invited paper) ». In: *ECRTS* (July 11–14, 2023). Ed. by Alessandro V. Papadopoulos. Vol. 262. Leibniz International Proceedings in Informatics (LIPIcs). Vienna, Austria: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 19:1–19:18. ISBN: 978-3-95977-280-8. DOI: 10.4230/LIPIcs.ECRTS.2023.19.

[Alu+93] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. « Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems ». In: *Hybrid Systems 1992* (Oct. 19–21, 1992). Ed. by Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel. Vol. 736. Lecture Notes in Computer Science. Lyngby, Denmark: Springer, 1993, pp. 209–229. DOI: 10.1007/3-540-57318-6_30.

References III

[Alu+95] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. « The Algorithmic Analysis of Hybrid Systems ». In: *Theoretical Computer Science* 138.1 (1995), pp. 3–34. DOI: 10.1016/0304-3975(94)00202-T.

[AM01] Yasmina Abdeddaïm and Oded Maler. « Job-Shop Scheduling Using Timed Automata ». In: *CAV* (July 18–22, 2001). Ed. by Gérard Berry, Hubert Comon, and Alain Finkel. Vol. 2102. Lecture Notes in Computer Science. Paris, France: Springer, 2001, pp. 478–492. ISBN: 3-540-42345-1. DOI: 10.1007/3-540-44585-4_46.

[AM02] Yasmina Adbeddaïm and Oded Maler. « Preemptive Job-Shop Scheduling using Stopwatch Automata ». In: *TACAS* (Apr. 8–12, 2002). Ed. by Joost-Pieter Katoen and Perdita Stevens. Vol. 2280. Lecture Notes in Computer Science. Grenoble, France: Springer-Verlag, Apr. 2002, pp. 113–126. DOI: 10.1007/3-540-46002-0_9.

[AMP21] Étienne André, Dylan Marinho, and Jaco van de Pol. « A Benchmarks Library for Extended Timed Automata ». In: *TAP* (June 21–25, 2021). Ed. by Frédéric Loulergue and Franz Wotawa. Vol. 12740. Lecture Notes in Computer Science. virtual: Springer, 2021, pp. 39–50. DOI: 10.1007/978-3-030-79379-1_3.

[And+14] Étienne André, Yang Liu, Jun Sun, and Jin Song Dong. « Parameter Synthesis for Hierarchical Concurrent Real-Time Systems ». In: *Real-Time Systems* 50.5-6 (Sept. 2014), pp. 620–679. DOI: 10.1007/s11241-014-9208-6.

References IV

[And+21] Étienne André, Jaime Arias, Laure Petrucci, and Jaco van de Pol. « Iterative Bounded Synthesis for Efficient Cycle Detection in Parametric Timed Automata ». In: *TACAS* (Mar. 27–Apr. 1, 2021). Ed. by Jan Friso Groote and Kim G. Larsen. Vol. 12651. Lecture Notes in Computer Science. Virtual: Springer, 2021, pp. 311–329. DOI: 10.1007/978-3-030-72016-2_17.

[And17] Étienne André. « A unified formalism for monoprocessor schedulability analysis under uncertainty ». In: *FMICS-AVoCS* (Sept. 18–20, 2017). Ed. by Ana Cavalcanti, Laure Petrucci, and Cristina Seceleanu. Vol. 10471. Lecture Notes in Computer Science. Torino, Italy: Springer, 2017, pp. 100–115. DOI: 10.1007/978-3-319-67113-0_7.

[And19] Étienne André. « What's decidable about parametric timed automata? » In: *International Journal on Software Tools for Technology Transfer* 21.2 (Apr. 2019), pp. 203–219. DOI: 10.1007/s10009-017-0467-0.

[And21] Étienne André. « IMITATOR 3: Synthesis of timing parameters beyond decidability ». In: *CAV* (July 18–23, 2021). Ed. by Rustan Leino and Alexandra Silva. Vol. 12759. Lecture Notes in Computer Science. virtual: Springer, 2021, pp. 1–14. DOI: 10.1007/978-3-030-81685-8_26.

[BDR08] Véronique Bruyère, Emmanuel Dall'Olio, and Jean-Francois Raskin. « Durations and parametric model-checking in timed automata ». In: *ACM Transactions on Computational Logic* 9.2 (2008), 12:1–12:23. DOI: 10.1145/1342991.1342996.

References V

[Ben+15] Nikola Beneš, Peter Bezděk, Kim Guldstrand Larsen, and Jiří Srba. « Language Emptiness of Continuous-Time Parametric Timed Automata ». In: *ICALP, Part II* (July 6–10, 2015). Ed. by Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann. Vol. 9135. Lecture Notes in Computer Science. Kyoto, Japan: Springer, July 2015, pp. 69–81. DOI: 10.1007/978-3-662-47666-6_6.

[BHZ08] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. « The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems ». In: *Science of Computer Programming* 72.1–2 (2008), pp. 3–21. DOI: 10.1016/j.scico.2007.08.001.

[BL09] Laura Bozzelli and Salvatore La Torre. « Decision problems for lower/upper bound parametric timed automata ». In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151. DOI: 10.1007/s10703-009-0074-0.

[BO17] Daniel Bundala and Joël Ouaknine. « On parametric timed automata and one-counter machines ». In: *Information and Computation* 253 (2017), pp. 272–303. DOI: 10.1016/j.ic.2016.07.011.

[BR07] Véronique Bruyère and Jean-François Raskin. « Real-Time Model-Checking: Parameters everywhere ». In: *Logical Methods in Computer Science* 3.1:7 (2007), pp. 1–30. DOI: 10.2168/LMCS-3(1:7)2007.

References VI

[Bud+17] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. « JANI: Quantitative Model and Tool Interaction ». In: *TACAS* (Apr. 22–29, 2017). Ed. by Axel Legay and Tiziana Margaria. Vol. 10206. Lecture Notes in Computer Science. Uppsala, Sweden, 2017, pp. 151–168. DOI: 10.1007/978-3-662-54580-5_9.

[Che+09] Rémy Chevallier, Emmanuelle Encrenaz-Tiphène, Laurent Fribourg, and Weiwen Xu. « Timed Verification of the Generic Architecture of a Memory Circuit Using Parametric Timed Automata ». In: *Formal Methods in System Design* 34.1 (Feb. 2009), pp. 59–81. DOI: 10.1007/s10703-008-0061-x.

[CL00] Franck Cassez and Kim Guldstrand Larsen. « The Impressive Power of Stopwatches ». In: *CONCUR* (Aug. 22–25, 2000). Ed. by Catuscia Palamidessi. Vol. 1877. Lecture Notes in Computer Science. University Park, PA, USA: Springer, 2000, pp. 138–152. DOI: 10.1007/3-540-44618-4_12.

[CPR08] Alessandro Cimatti, Luigi Palopoli, and Yusi Ramadian. « Symbolic Computation of Schedulability Regions Using Parametric Timed Automata ». In: *RTSS* (Nov. 30–Dec. 3, 2008). Barcelona, Spain: IEEE Computer Society, 2008, pp. 80–89. ISBN: 978-0-7695-3477-0. DOI: 10.1109/RTSS.2008.36.

[Doy07] Laurent Doyen. « Robust Parametric Reachability for Timed Automata ». In: *Information Processing Letters* 102.5 (2007), pp. 208–213. DOI: 10.1016/j.ipl.2006.11.018.

[Fer+07] Elena Fersman, Pavel Krcál, Paul Pettersson, and Wang Yi. « Task automata: Schedulability, decidability and undecidability ». In: *Information and Computation* 205.8 (2007), pp. 1149–1172. DOI: 10.1016/j.ic.2007.01.009.

References VII

[FJ13] Léa Fanchon and Florent Jacquemard. « Formal Timing Analysis Of Mixed Music Scores ». In: *ICMC* (Aug. 12–16, 2013). Perth, Australia: Michigan Publishing, Aug. 2013.

[Fre+11] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. « SpaceEx: Scalable Verification of Hybrid Systems ». In: *CAV* (July 14–20, 2011). Ed. by Ganesh Gopalakrishnan and Shaz Qadeer. Vol. 6806. Lecture Notes in Computer Science. Snowbird, UT, USA: Springer, 2011, pp. 379–395. DOI: 10.1007/978-3-642-22110-1_30.

[Fre08] Goran Frehse. « PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech ». In: *International Journal on Software Tools for Technology Transfer* 10.3 (May 2008), pp. 263–279. ISSN: 1433-2779. DOI: 10.1007/s10009-007-0062-x.

[Fri+12] Laurent Fribourg, David Lesens, Pierre Moro, and Romain Soulat. « Robustness Analysis for Scheduling Problems using the Inverse Method ». In: *TIME* (Sept. 12–14, 2012). Ed. by Mark Reynolds, Paolo Terenziani, and Ben Moszkowski. Leicester, UK: IEEE Computer Society Press, Sept. 2012, pp. 73–80. DOI: 10.1109/TIME.2012.10.

[GH21] Stefan Göller and Mathieu Hilaire. « Reachability in Two-Parametric Timed Automata with One Parameter Is EXPSPACE-Complete ». In: *STACS* (Mar. 16–19, 2021). Ed. by Markus Bläser and Benjamin Monmege. Vol. 187. LIPIcs. Saarbrücken, Germany: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 36:1–36:18. DOI: 10.4230/LIPIcs.STACS.2021.36.

[Hen+98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. « What's Decidable about Hybrid Automata? » In: *Journal of Computer and System Sciences* 57.1 (1998), pp. 94–124. DOI: 10.1006/jcss.1998.1581.

References VIII

[HHW97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. « HyTech: A Model Checker for Hybrid Systems ». In: *International Journal on Software Tools for Technology Transfer* 1.1-2 (1997), pp. 110–122. DOI: 10.1007/s100090050008.

[Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. « Linear parametric model checking of timed automata ». In: *Journal of Logic and Algebraic Programming* 52-53 (2002), pp. 183–220. DOI: 10.1016/S1567-8326(02)00037-1.

[JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. « Integer Parameter Synthesis for Real-Time Systems ». In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461. DOI: 10.1109/TSE.2014.2357445.

[KP12] Mirosław Kurkowski and Wojciech Penczek. « Applying Timed Automata to Model Checking of Security Protocols ». In: *Handbook of Finite State Based Models and Applications*. Ed. by Jiacun Wang. Chapman and Hall/CRC, 2012, pp. 223–254. DOI: 10.1201/b13055-12.

[Lim+09] Didier Lime, Olivier H. Roux, Charlotte Seidner, and Louis-Marie Traonouez. « Romeo: A Parametric Model-Checker for Petri Nets with Stopwatches ». In: *TACAS* (Mar. 22–29, 2009). Ed. by Stefan Kowalewski and Anna Philippou. Vol. 5505. Lecture Notes in Computer Science. York, United Kingdom: Springer, Mar. 2009, pp. 54–57. DOI: 10.1007/978-3-642-00768-2_6.

[Lip+14] Giuseppe Lipari, Youcheng Sun, Étienne André, and Laurent Fribourg. « Toward Parametric Timed Interfaces for Real-Time Components ». In: *SynCoP* (Apr. 6, 2014). Ed. by Étienne André and Goran Frehse. Vol. 145. Electronic Proceedings in Theoretical Computer Science. Grenoble, France, Apr. 2014, pp. 49–64. DOI: 10.4204/EPTCS.145.6.

References IX

[LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. « UPPAAL in a Nutshell ». In: *International Journal on Software Tools for Technology Transfer* 1.1-2 (1997), pp. 134–152. DOI: 10.1007/s100090050010.

[Lut+17] Lars Luthmann, Andreas Stephan, Johannes Bürdek, and Malte Lochau. « Modeling and Testing Product Lines with Unbounded Parametric Real-Time Constraints ». In: *SPLC, Volume A* (Sept. 25–29, 2017). Ed. by Myra B. Cohen, Mathieu Acher, Lidia Fuentes, Daniel Schall, Jan Bosch, Rafael Capilla, Ebrahim Bagheri, Yingfei Xiong, Javier Troya, Antonio Ruiz Cortés, and David Benavides. Sevilla, Spain: ACM, 2017, pp. 104–113. DOI: 10.1145/3106195.3106204.

[Mar11] Nicolas Markey. « Robustness in Real-time Systems ». In: *SIES*. Ed. by Iain Bate and Roberto Passerone. Västerås, Sweden: IEEE Computer Society Press, June 2011, pp. 28–34. DOI: 10.1109/SIES.2011.5953652.

[Mil00] Joseph S. Miller. « Decidability and Complexity Results for Timed Automata and Semi-linear Hybrid Automata ». In: *HSCC* (Mar. 23–25, 2000). Ed. by Nancy A. Lynch and Bruce H. Krogh. Vol. 1790. Lecture Notes in Computer Science. Pittsburgh, PA, USA: Springer, 2000, pp. 296–309. ISBN: 3-540-67259-1. DOI: 10.1007/3-540-46430-1_26.

[Min67] Marvin L. Minsky. *Computation: Finite and infinite machines*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1967. ISBN: 0-13-165563-9.

[NPP18] Hoang Gia Nguyen, Laure Petrucci, and Jaco van de Pol. « Layered and Collecting NDFS with Subsumption for Parametric Timed Automata ». In: *ICECCS* (Dec. 12–14, 2018). Ed. by Anthony Widjaja Lin and Jun Sun. Melbourne, Australia: IEEE Computer Society, Dec. 2018, pp. 1–9. DOI: 10.1109/ICECCS2018.2018.00009.

References X

[NWY99] Christer Norström, Anders Wall, and Wang Yi. « Timed Automata as Task Models for Event-Driven Systems ». In: *RTCSA* (Dec. 13–16, 1999). Hong Kong, China: IEEE Computer Society, 1999, pp. 182–189. DOI: 10.1109/RTCSA.1999.811218.

[SAL15] Youcheng Sun, Étienne André, and Giuseppe Lipari. « Verification of Two Real-Time Systems Using Parametric Timed Automata ». In: *WATERS* (July 7, 2015). Ed. by Sophie Quinton and Tullio Vardanega. Lund, Sweden, July 2015.

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

[Sun+13] Youcheng Sun, Romain Soulat, Giuseppe Lipari, Étienne André, and Laurent Fribourg. « Parametric Schedulability Analysis of Fixed Priority Real-Time Distributed Systems ». In: *FSTCS* (Oct. 29–30, 2013). Ed. by Cyrille Artho and Peter Ölveczky. Vol. 419. Communications in Computer and Information Science. Auckland, New Zealand: Springer, Oct. 2013, pp. 212–228. DOI: 10.1007/978-3-319-05416-2_14.

[TLR09] Louis-Marie Traonouez, Didier Lime, and Olivier H. Roux. « Parametric Model-Checking of Stopwatch Petri Nets ». In: *Journal of Universal Computer Science* 15.17 (2009), pp. 3273–3304. DOI: 10.3217/jucs-015-17-3273.

[WAH23] Masaki Waga, Étienne André, and Ichiro Hasuo. « Parametric Timed Pattern Matching ». In: *ACM Transactions on Software Engineering and Methodology* 32.1 (Feb. 2023), 10:1–10:35. DOI: 10.1145/3517194.

Credits

Source of the graphics used I

Titre : Smiley green alien big eyes (aaah)
Auteur : LadyofHats
Source : `https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg`
Licence : PUBLIC DOMAIN

Titre : Smiley green alien big eyes (cry)
Auteur : LadyofHats
Source : `https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg`
Licence : PUBLIC DOMAIN

Titre : Coffee machine drawing
Auteur : Ysangkok
Source : `https://commons.wikimedia.org/wiki/File:Coffee_machine.svg`
Licence : PUBLIC DOMAIN

Titre : taking a coffee break
Auteur : chris
Source : `https://commons.wikimedia.org/wiki/File:SMirC-coffeebreak.svg`
Licence : CC BY

License of this document

Author: **Étienne André**

(LaTeX source available to academic teachers upon request)

UNIVERSITÉ SORBONNE PARIS NORD

Institut GALILÉE
Université Sorbonne Paris Nord

Version: October 10, 2024