Decision Problems for Parametric Timed Automata

Étienne André\textsuperscript{1,2}, Didier Lime\textsuperscript{2}, Olivier H. Roux\textsuperscript{2}

1 LIPN, Université Paris 13, Sorbonne Paris Cité, CNRS, France
2 École Centrale de Nantes, IRCCyN, CNRS, UMR 6597, France
Context: timed model checking

- Timed model checking

\[ y = \text{delay} \]
\[ x := 0 \]
\[ x < \text{period} \]

red box is unreachable

A model of the system  
A property to be satisfied
Context:

- Timed model checking

\[ y = \text{delay} \]
\[ x := 0 \]
\[ x < \text{period} \]

A model of the system

A property to be satisfied

Question: does the model of the system satisfy the property?
**Context:**

- Timed model checking

- **Timed model checking**

  \[
  \begin{align*}
  y & = \text{delay} \\
  x & := 0 \\
  x & < \text{period}
  \end{align*}
  \]

- A model of the system

- A property to be satisfied

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

- **Yes**

- **No**

- **Counterexample**
Context: parametric timed model checking

- Timed model checking

\[ y = \text{delay} \]
\[ x := 0 \]
\[ x < \text{period} \]

A model of the system

A property to be satisfied

Question: for what values of the parameters does the model of the system satisfy the property?

Yes if...

\[ 2\text{delay} > \text{period} \]
\[ \land \text{period} < 20.46 \]
Outline

1. Parametric timed automata
2. Decision problems
3. EF-emptiness
4. Integer-points PTAs
5. EF-universality and AF-emptiness
6. Conclusion and perspectives
Outline

1. Parametric timed automata
2. Decision problems
3. EF-emptiness
4. Integer-points PTAs
5. EF-universality and AF-emptiness
6. Conclusion and perspectives
Timed automaton (TA)

- Finite state automaton (sets of locations)
**Timed automaton (TA)**

- Finite state automaton (sets of locations and actions)

---

![Diagram of a timed automaton](image)

- **Location**: constrain to be verified to stay at a location
- **Transition guard**: constrain to be verified to enable a transition
- **Clock reset**: some of the clocks can be set to 0 at each transition

```plaintext
x := 0
y := 0
```

```
x ≥ 1
```

**Features**

```plaintext
y = 5
```
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks [Alur and Dill, 1994, Henzinger et al., 1994]
- Real-valued variables evolving linearly at the same rate
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks [Alur and Dill, 1994, Henzinger et al., 1994]
  - Real-valued variables evolving linearly at the same rate

- Features
  - Location invariant: constraint to be verified to stay at a location

\[
\begin{align*}
\text{press?} & \quad y \leq 5 \\
\text{cup!} & \quad y \leq 8
\end{align*}
\]
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set \( X \) of clocks \([\text{Alur and Dill, 1994, Henzinger et al., 1994}]\)
  - Real-valued variables evolving linearly at the same rate

- Features
  - Location invariant: constraint to be verified to stay at a location
  - Transition guard: constraint to be verified to enable a transition

\[
\begin{align*}
x &\geq 1 \\
y &\leq 5 \\
y &\leq 8
\end{align*}
\]
**Timed automaton (TA)**

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks [Alur and Dill, 1994, Henzinger et al., 1994]
  - Real-valued variables evolving linearly at the same rate

**Features**

- **Location invariant**: constraint to be verified to stay at a location
- **Transition guard**: constraint to be verified to enable a transition
- **Clock reset**: some of the clocks can be set to 0 at each transition

$$\begin{align*}
  y &= 8 \\
  \text{coffee!}
\end{align*}$$

- $$\begin{align*}
  &x \geq 1 \\
  &y = 5 \\
  &\text{cup!}
\end{align*}$$

- $$\begin{align*}
  &x := 0 \\
  &y := 0
\end{align*}$$
Concrete semantics of timed automata

- **Concrete state** of a TA: pair \( (l, w) \), where
  - \( l \) is a location,
  - \( w \) is a valuation of each clock

- **Concrete run**: alternating sequence of concrete states and actions or time elapse
Examples of concrete runs

- Possible concrete runs for the coffee machine
Examples of concrete runs

- Possible concrete runs for the coffee machine
  - Coffee with no sugar
Examples of concrete runs

- Possible concrete runs for the coffee machine
  - Coffee with no sugar

\[
\begin{array}{c|c}
\text{x} & 0 & 15.4 \\
\hline
\text{y} & 0 & 15.4 \\
\end{array}
\]
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>15.4</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>15.4</td>
</tr>
</tbody>
</table>
Examples of concrete runs

- Possible concrete runs for the coffee machine
  - Coffee with no sugar

| x  | 0   | 15.4 | 0   | 5   |
| y  | 0   | 15.4 | 0   | 5   |
Examples of concrete runs

- Possible concrete runs for the coffee machine

  - Coffee with no sugar

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>15.4</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>15.4</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

- y ≤ 5
- y = 5
- x ≥ 1
- x := 0
- press?
- x := 0
- y := 0
- press?
- y = 8
- coffee!
- y ≤ 8
- cup!
Examples of concrete runs

Possible concrete runs for the coffee machine
- Coffee with no sugar

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>15.4</td>
<td>0</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>15.4</td>
<td>0</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>
Examples of concrete runs

- Possible concrete runs for the coffee machine
  - Coffee with no sugar

<table>
<thead>
<tr>
<th>( \text{x} )</th>
<th>0</th>
<th>15.4</th>
<th>0</th>
<th>5</th>
<th>5</th>
<th>3</th>
<th>8</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{y} )</td>
<td>0</td>
<td>15.4</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
Examples of concrete runs

- Possible concrete runs for the coffee machine
  - Coffee with no sugar
    - Initial state:
      - $x = 0$
      - $y = 0$
    - Transition:
      - Press the button:
        - $x := 0$
        - $y := 0$
    - Intermediate states:
      - $x = 15.4$
      - $y = 15.4$
    - Final state:
      - $x = 5$
      - $y = 5$
      - $x = 8$
      - $y = 8$

  - Coffee with 2 doses of sugar
    - Initial state:
      - $x = 0$
      - $y = 0$
    - Transition:
      - Press the button:
        - $x := 0$
    - Intermediate state:
      - $x = 15.4$
      - $y = 15.4$
    - Final state:
      - $x = 3$
      - $y = 8$
      - $x = 8$
      - $y = 8$

Decision Problems for PTAs

É. André et al. (Nantes & Paris 13)
Examples of concrete runs

- Possible concrete runs for the coffee machine
  - Coffee with no sugar
    ```
    x  | 0  | 15.4 | 0  | 5  | 5  | 8  | 8  
    y  | 0  | 15.4 | 0  | 5  | 5  | 8  | 8  
    ```
  - Coffee with 2 doses of sugar
    ```
    x  | 0  | 0   |
    y  | 0  | 0   |
    ```
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Examples of concrete runs

Possible concrete runs for the coffee machine

<table>
<thead>
<tr>
<th>Coffee with no sugar</th>
</tr>
</thead>
<tbody>
<tr>
<td>x: 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coffee with 2 doses of sugar</th>
</tr>
</thead>
<tbody>
<tr>
<td>x: 0</td>
</tr>
</tbody>
</table>
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Examples of concrete runs

Possible concrete runs for the coffee machine
- Coffee with no sugar
- Coffee with 2 doses of sugar
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar
  - \( x = 0 \)
  - \( y = 0 \)
  - \( \text{press?} \quad \text{press?} \quad \text{cup!} \quad \text{coffee!} \)

- Coffee with 2 doses of sugar
  - \( x = 0 \)
  - \( y = 0 \)
  - \( \text{press?} \quad \text{press?} \quad \text{press?} \quad \text{cup!} \quad \text{coffee!} \)
Parametric timed automaton (PTA)

- Timed automaton (sets of locations, actions and clocks)
*Parametric timed automata* (PTA)

- Timed automaton (sets of *locations, actions and clocks*) augmented with a set $P$ of *parameters* [Alur et al., 1993]
- **Unknown constants** used in guards and invariants

\[
y = p_3 \\
\text{coffee!}
\]

\[
y \leq p_2 \\
x \geq p_1 \\
\text{cup!}
\]

\[
y = 0 \\
x := 0 \\
\text{press?}
\]

\[
y \leq 8
\]
L/U-PTAs

Definition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter $p$ is always compared with clocks as an upper bound or always as a lower bound. [Hune et al., 2002, Bozzelli and La Torre, 2009]
L/U-PTAs

Definition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter $p$ is always compared with clocks as an upper bound or always as a lower bound.  

[Hune et al., 2002, Bozzelli and La Torre, 2009]

![Diagram of L/U-PTA](image)

**Lower-bound parameters:** $p_1, p_3$

**Upped-bound parameters:** $p_1, p_3$
L/U-PTAs

Definition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter $p$ is always compared with clocks as an upper bound or always as a lower bound. [Hune et al., 2002, Bozzelli and La Torre, 2009]

Lower-bound parameters: $p_1, p_3$

Uppped-bound parameters: $p_2, p_4$
Symbolic semantics of PTAs

- Symbolic state $s = (l, Z)$: location + convex polyhedron constraining both clocks and parameters
  
  [Hune et al., 2002, André et al., 2009, Jovanović et al., 2015]

- Convex polyhedra obtained have a special form called parametric zone
  
  [Hune et al., 2002]

\[
Z_0 = \begin{cases} 
  x = y \\
  0 \leq y \leq p_1 \\
  p_1, p_2 \geq 0
\end{cases}
\]

\[
Z_1 = \begin{cases} 
  p_2 \leq x - y \leq p_1 \\
  (p_2 \leq p_1) \\
  x, y, p_1, p_2 \geq 0
\end{cases}
\]
Symbolic semantics of PTAs

- Symbolic state \( s = (l, Z) \): location + convex polyhedron constraining both clocks and parameters
  
  \[ \text{[Hune et al., 2002, André et al., 2009, Jovanović et al., 2015]} \]

- Convex polyhedra obtained have a special form called \textit{parametric zone}
  
  \[ \text{[Hune et al., 2002]} \]

\[
Z_0 = \begin{cases} 
  x = y \\
  0 \leq y \leq p_1 \\
  p_1, p_2 \geq 0
\end{cases} 
\]

\[
Z_1 = \begin{cases} 
  p_2 \leq x - y \leq p_1 \\
  (p_2 \leq p_1) \\
  x, y, p_1, p_2 \geq 0
\end{cases}
\]

Note: there is a potentially infinite number of symbolic states
Valuation of a PTA

Given a PTA $A$ and a parameter valuation $\nu$, we denote by $\nu(A)$ the (non-parametric) timed automaton where all parameters are valuated by $\nu$.
Valuation of a PTA

Given a PTA $\mathcal{A}$ and a parameter valuation $\nu$, we denote by $\nu(\mathcal{A})$ the (non-parametric) timed automaton where all parameters are valuated by $\nu$

\[
\nu = \begin{cases} 
    y_1 \leq p_2 \\
    y_2 \leq p_3 \\
    y_3 \leq 8
\end{cases}
\]

\[
\nu = \begin{cases} 
    y_1 \leq 5 \\
    y_2 \leq 8 \\
    y_3 \leq 8
\end{cases}
\]

with $\nu : \begin{cases} 
    p_1 \rightarrow 1 \\
    p_2 \rightarrow 5 \\
    p_3 \rightarrow 8
\end{cases}$
Outline

1. Parametric timed automata
2. Decision problems
3. EF-emptiness
4. Integer-points PTAs
5. EF-universality and AF-emptiness
6. Conclusion and perspectives
Decision problems (1/3)

Definition (reachability emptiness (EF-emptiness))

Input: a PTA $A$ and a set of locations $G$

Problem: Is the set of parameter valuations $v$ such that there exists a run of $v(A)$ reaching a location $l \in G$ empty?
Decision problems (2/3)

Definition (reachability universality (EF-universality))

Input: a PTA $\mathcal{A}$ and a set of locations $G$
Problem: Are all parameter valuations $\nu$ such that there exists a run of $\nu(\mathcal{A})$ reaching a location $l \in G$?
Decision problems (3/3)

Definition (unavoidability emptiness (AF-emptiness))

Input: a PTA $A$ and a set of locations $G$
Problem: Is the set of parameter valuations $\nu$ such that all runs of $\nu(A)$ eventually reach a location $l \in G$ empty?
Outline

1. Parametric timed automata
2. Decision problems
3. EF-emptiness
4. Integer-points PTAs
5. EF-universality and AF-emptiness
6. Conclusion and perspectives
EF-emptiness in the literature

Reachability emptiness ("is the set of parameter valuations reaching a given location empty?") is undecidable for PTAs [Alur et al., 1993]

- even with a single parametric clock [Miller, 2000]
- even with a single real-valued parameter [Miller, 2000]
- even with only strict constraints [Doyen, 2007]
- even with a single integer-valued parameter [Beneš et al., 2015]
**EF-emptiness in the literature**

**EF-emptiness**

*Reachability emptiness* (“is the set of parameter valuations reaching a given location empty?”) is **undecidable** for PTAs [Alur et al., 1993]

- even with a single parametric clock [Miller, 2000]
- even with a single real-valued parameter [Miller, 2000]
- even with only strict constraints [Doyen, 2007]
- even with a single integer-valued parameter [Beneš et al., 2015]

**Proof.**

By reduction from the halting problem of a 2-counter machine, which is **undecidable** [Minsky, 1967]

See [André, 2015] for an exhaustive survey
A new construction to prove the undecidability

Reduction from the halting problem of a 2-counter machine

Not a new theoretical result

Will be used for all subsequent undecidability proofs

Our new encoding of the 2-counter machine:

- Uses a single rational-valued parameter and three (parametric) clocks
- Matches the smallest known numbers of clocks and parameters for integer-valued parameters [Beneš et al., 2015] and rational-valued parameters [Miller, 2000]
2-counter machine

Finite program reading and modifying 2 non-negative integer counters $C_1$ and $C_2$

3 instructions

- when in state $s_i$, increment $C_k$ and go to $s_j$;
- when in state $s_i$, decrement $C_k$ and go to $s_j$;
- when in state $s_i$, if $C_k = 0$ then go to $s_j$, otherwise block.

Halting is undecidable [Minsky, 1967]
Encoding

Encoding the counters $C_1$ and $C_2$

- Three clocks $x$, $y$ and $z$ and one parameter $a$
- Let $c_1, c_2$ be the values of $C_1$ and $C_2$
- In any location $s_i$, when $x = 0$ we have
  - $y = 1 - ac_1$
  - $z = 1 - ac_2$
Encoding

Encoding the counters $C_1$ and $C_2$

- Three clocks $x$, $y$ and $z$ and one parameter $a$
- Let $c_1, c_2$ be the values of $C_1$ and $C_2$
- In any location $s_i$, when $x = 0$ we have
  - $y = 1 - ac_1$
  - $z = 1 - ac_2$

$$x = 1$$
$$x := 0$$

Initializing the clocks
Encoding the instructions: Incrementing $C_1$

\[\begin{align*}
z &= 1, \\
z &= 0
\end{align*}\]

\[\begin{align*}
y &= a + 1, \\
y &= 0, \\
x &= 1, \\
x &= 0
\end{align*}\]
Encoding the instructions: Incrementing $C_1$

\[
\begin{align*}
  z &= 1, \\
  z &:= 0 \\
  x &= 0, \\
  y &= a + 1, \\
  y &:= 0 \\
  y' &= a + 1, \\
  y' &:= 0 \\
  z &= 1, \\
  z &:= 0
\end{align*}
\]
Encoding the instructions : Incrementing $C_1$

\[ z = 1, \quad y = a + 1, \]
\[ z := 0, \quad y := 0 \]
\[ x = 0, \quad x := 0 \]

\[ l_{i1} \]
\[ x \rightarrow 0 \]
\[ y \rightarrow 1 - ac_1 \]
\[ z \rightarrow 1 - ac_2 \]
Encoding the instructions: Incrementing $C_1$

\[ z = 1, \quad y = a + 1, \]
\[ z := 0, \quad y := 0 \]
\[ x = 0, \quad y := 0 \]
\[ x := 0, \]
\[ y = a + 1, \quad z = 1, \]
\[ y := 0, \quad z := 0 \]

<table>
<thead>
<tr>
<th>$l_{i1}$</th>
<th>$l_{i2}$</th>
<th>$l_{i3}$</th>
<th>$s_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\chi$</td>
<td>0</td>
<td>$ac_2$</td>
<td>$ac_2$</td>
</tr>
<tr>
<td>$y$</td>
<td>$1 - ac_1$</td>
<td>$1 - ac_1 + ac_2$</td>
<td>$1 - ac_1 + ac_2$</td>
</tr>
<tr>
<td>$z$</td>
<td>$1 - ac_2$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
</tbody>
</table>
Encoding the instructions: Incrementing $C_1$

$z = 1,$ $y = a + 1,$
$z := 0,$ $y := 0$
$x = 1,$ $x := 0$

$y = a + 1,$ $z = 1,$
$y := 0,$ $z := 0$

$\begin{array}{c}
\text{l}_i \leftarrow \text{s}_i \\
\text{x} = 0 \\
y = a + 1 \\
y := 0 \\
z = 1 \\
z := 0
\end{array}$
Encoding the instructions: Incrementing $C_1$

\[
\begin{align*}
z &= 1, \\
&\quad z := 0 \\
&\quad \text{if } z = 1, \\
&\quad \text{if } z = 0
\end{align*}
\]

\[
\begin{align*}
y &= a + 1, \\
&\quad y := 0 \\
&\quad \text{if } y = a + 1, \\
&\quad \text{if } y = 0
\end{align*}
\]

\[
\begin{align*}
x &= 0, \\
&\quad x := 0 \\
&\quad \text{if } x = 1, \\
&\quad \text{if } x = 0
\end{align*}
\]

\[
\begin{align*}
l_{i1} &\quad \longrightarrow \quad l_{i2} \\
&\quad \text{if } x = 0, \\
&\quad \text{if } x = 1
\end{align*}
\]

\[
\begin{align*}
l_{i1} &\quad \longrightarrow \quad l_{i3} \\
&\quad \text{if } y = a + 1, \\
&\quad \text{if } y = 0
\end{align*}
\]

\[
\begin{align*}
l_{i2} &\quad \longrightarrow \quad l_{i3} \\
&\quad \text{if } z = 1, \\
&\quad \text{if } z = 0
\end{align*}
\]

\[
\begin{align*}
l_{i2} &\quad \longrightarrow \quad s_j \\
&\quad \text{if } y = a + 1, \\
&\quad \text{if } y = 0
\end{align*}
\]

\[
\begin{align*}
l_{i2} &\quad \longrightarrow \quad s_i \\
&\quad \text{if } x = 0, \\
&\quad \text{if } x = 1
\end{align*}
\]

\[
\begin{align*}
l_{i1} &\quad \longrightarrow \quad s_i \\
&\quad \text{if } y = 1 - ac_1 \\
&\quad \text{if } z = 1 - ac_2
\end{align*}
\]

\[
\begin{align*}
l_{i2} &\quad \longrightarrow \quad l_{i3} \\
&\quad \text{if } y = 1 - ac_1 + ac_2 \\
&\quad \text{if } z = 0
\end{align*}
\]

\[
\begin{align*}
l_{i2} &\quad \longrightarrow \quad a(c_1 + 1) \\
&\quad \text{if } y = a + 1 \\
&\quad \text{if } z = a(c_1 + 1) - ac_2
\end{align*}
\]

\[
\begin{align*}
l_{i2} &\quad \longrightarrow \quad l_{i2} \\
&\quad \text{if } y = 1 - ac_1 \\
&\quad \text{if } z = 1 - ac_2
\end{align*}
\]
Encoding the instructions: Incrementing $C_1$

$$z = 1, \quad z := 0$$

$$y = a + 1, \quad y := 0$$

$$x = 0, \quad x := 0$$

$$y = a + 1, \quad y := 0$$

$$z = 1, \quad z := 0$$

<table>
<thead>
<tr>
<th>State</th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_i$</td>
<td>0</td>
<td>$1 - ac_1$</td>
<td>$1 - ac_2$</td>
</tr>
<tr>
<td>$l_{i1}$</td>
<td>0</td>
<td>$1 - ac_1$</td>
<td>$1 - ac_2$</td>
</tr>
<tr>
<td>$l_{i2}$</td>
<td>$ac_2$</td>
<td>$1 - ac_1 + ac_2$</td>
<td>$a(c_1 + 1) - ac_2$</td>
</tr>
<tr>
<td>$l_{i3}$</td>
<td>$a(c_1 + 1)$</td>
<td>0</td>
<td>$a(c_1 + 1) - ac_2$</td>
</tr>
<tr>
<td>$s_j$</td>
<td>$ac_2$</td>
<td>$a(c_1 + 1)$</td>
<td>0</td>
</tr>
</tbody>
</table>
Encoding the instructions: Incrementing $C_1$

\begin{align*}
z &= 1, & y &= a + 1, \\
z &:= 0 & y &:= 0 \\
x &= 0 & x &:= 0 \\
\end{align*}

\begin{align*}
\text{l}_{i1} &\xrightarrow{\text{a} \text{c}_2} \text{l}_{i2} & \text{l}_{i2} &\xrightarrow{\text{a} (c_1 + 1) - \text{a} \text{c}_2} \text{l}_{i3} \\
x &= 0 & \text{a} \text{c}_2 &\text{a} \text{c}_2 & \text{a} (c_1 + 1) &\text{0} \\
y &= 1 - \text{a} \text{c}_1 & \text{1 - a} \text{c}_1 + \text{a} \text{c}_2 &\text{a} (c_1 + 1) - \text{a} \text{c}_2 \\
z &= 1 - \text{a} \text{c}_2 &\text{0} &\text{a} (c_1 + 1) - \text{a} \text{c}_2 \\
\text{l}_{i3} &\xrightarrow{1 - a(c_1 + 1)} \text{l}_{i3} \\
x &= 1 & \text{1} &\text{1} & \text{1 - a} (c_1 + 1) \\
y &= 1 - \text{a} (c_1 + 1) & \text{1 - a} (c_1 + 1) &\text{1 - a} \text{c}_2 \\
z &= 1 - \text{a} \text{c}_2 &\text{1 - a} \text{c}_2 \\
\end{align*}
Encoding the instructions: Incrementing $C_1$

\[
\begin{align*}
z &= 1, \\
z &= 0 \\
x &= 0 \\
x &= 1, \\
y &= a + 1, \\
y &= 0 \\
y &= a + 1, \\
y &= 0 \\
z &= 1, \\
z &= 0
\end{align*}
\]

\[
\begin{align*}
l_{i1} &\xrightarrow{\text{ac}_2} l_{i2} \\
l_{i2} &\xrightarrow{\text{ac}_2} l'_{i2} \\
l'_{i2} &\xrightarrow{\text{ac}_2} l_{i3} \\
l_{i3} &\xrightarrow{\text{ac}(c_1 + 1) - \text{ac}_2} s_j
\end{align*}
\]
Encoding the instructions: Decrementing $C_1$

Replacing guards $y = a + 1$ with $y = 1$, and guards $x = 1$ and $z = 1$ with $x = a + 1$ and $z = a + 1$, respectively.
Encoding the instructions : Decrementing \( C_1 \)

Replacing guards \( y = a + 1 \) with \( y = 1 \), and guards \( x = 1 \) and \( z = 1 \) with \( x = a + 1 \) and \( z = a + 1 \), respectively.

0-test: Straightforward (when \( x = 0 \), then \( y \) shall be 1)
Outline

1. Parametric timed automata
2. Decision problems
3. EF-emptiness
4. Integer-points PTAs
5. EF-universality and AF-emptiness
6. Conclusion and perspectives
Definition of IP-PTAs

Definition (IP-PTA)

A PTA $\mathcal{A}$ is an integer points PTA (in short IP-PTA) if, in any reachable symbolic state $(l, C)$ of $\mathcal{A}$, $C$ contains at least one integer point.

$C_1$: 
\[
p + 1 \leq x \\
\land x \leq 2p + 3
\]
contains integer points

$C_2$: 
\[
x > 1 \\
\land p < 5 \\
\land x < p - 3
\]
contains no integer points
EF-emptiness for bounded IP-PTAs

Theorem

EF-emptiness is \textit{decidable} for bounded IP-PTAs.

Proof idea

- There is a parameter valuation $\nu$ and a path to $l$ in $\nu(A)$ iff there is \textit{symbolic} path $\pi$ to some $(l, C)$;
- There is a path $\pi$ to $(l, C)$ iff $C \neq \emptyset$;
- $C \neq \emptyset$ iff $C$ contains an \textit{integer} point (IP-PTA);
- $C$ contains an integer point iff there is an \textit{integer} parameter valuation $\nu^*$ such that $\nu^*(\pi)$ is feasible in $\nu^*(A)$;
- There is a \textit{finite} number of integer parameter valuations (bounded).
Expressiveness of IP-PTAs

- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]
Expressiveness of IP-PTAs

- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]
- IP-PTAs incomparable with L/U-PTAs
Expressiveness of IP-PTAs

- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]
- IP-PTAs incomparable with L/U-PTAs
- IP-PTAs strictly larger than closed L/U-PTAs
Expressiveness of IP-PTAs

- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]
- IP-PTAs incomparable with L/U-PTAs
- IP-PTAs strictly larger than closed L/U-PTAs
- bounded IP-PTAs strictly larger than bounded closed L/U-PTAs
Expressiveness of IP-PTAs

- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]
- IP-PTAs incomparable with L/U-PTAs
- IP-PTAs strictly larger than closed L/U-PTAs
- bounded IP-PTAs strictly larger than bounded closed L/U-PTAs
- bounded IP-PTAs incomparable with bounded L/U-PTAs
Expressiveness of IP-PTAs

- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]
- IP-PTAs incomparable with L/U-PTAs
- IP-PTAs strictly larger than closed L/U-PTAs
- bounded IP-PTAs strictly larger than bounded closed L/U-PTAs
- bounded IP-PTAs incomparable with bounded L/U-PTAs

\( \leadsto \) We strictly extend the class of PTAs for which the EF-emptiness problem is decidable
Proposition

The EF-synthesis for closed bounded L/U-PTAs (and therefore for IP-PTAs) is intractable in practice.

Proof idea

- Assume a closed bounded PTA $A$.
- Split each parameter $p_i$ into $p_i^-$ and $p_i^+$.
- Assume complete synthesis is possible, giving constraint $K$.
- Then if one can test the emptiness of $K \land (\bigwedge_i p_i^- = p_i^+)$ then this contradicts the undecidability of EF-emptiness for PTAs.

(following a reasoning from [Bozzelli and La Torre, 2009, Jovanović et al., 2015])
Undecidability of the membership

Theorem

*It is undecidable whether a PTA is an IP-PTA.*

Proof idea

From a reduction to the halting problem of a 2-counter machine: the model is IP iff the machine does not halt.
Undecidability of the membership

Theorem

It is undecidable whether a PTA is an IP-PTA.

Proof idea

From a reduction to the halting problem of a 2-counter machine: the model is IP iff the machine does not halt.

Consequence:

- Limited practical interest of IP-PTAs?
- ...unless we can exhibit sufficient syntactic conditions for membership
**Reset-PTA**

**Definition (reset-PTA)**

A PTA is a reset-PTA if, whenever a clock is compared to a parameter, all clocks are reset.
**Reset-PTA**

**Definition (reset-PTA)**

A PTA is a reset-PTA if, whenever a clock is compared to a parameter, all clocks are reset.

```
y ≤ 5
press? x, y := 0

y = 8
coffee!

y ≥ 5

y = p2
x, y := 0
cup!

x ≥ p1
press? x, y := 0
```

**Theorem**

_A reset-PTA is an IP-PTA._
**Reset-PTA**

**Definition (reset-PTA)**

A PTA is a reset-PTA if, whenever a clock is compared to a parameter, all clocks are reset.

**Theorem**

A reset-PTA is an IP-PTA.

**Corollary**

The EF-emptiness problem is *decidable* for bounded reset-PTAs.
Outline

1. Parametric timed automata
2. Decision problems
3. EF-emptiness
4. Integer-points PTAs
5. EF-universality and AF-emptiness
6. Conclusion and perspectives
**EF-universality and AF-emptiness**

**Theorem**

*EF-universality and AF-emptiness are *undecidable* for bounded IP-PTAs.*

See paper for details
EF-universality and AF-emptiness

Theorem

EF-universality and AF-emptiness are undecidable for bounded IP-PTAs.

See paper for details

Corollary

EF-universality and AF-emptiness are undecidable for unbounded IP-PTAs, for bounded PTAs and for PTAs.
Outline

1 Parametric timed automata

2 Decision problems

3 EF-emptiness

4 Integer-points PTAs

5 EF-universality and AF-emptiness

6 Conclusion and perspectives
## Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>bL/U-PTAs</th>
<th>bIP-PTAs</th>
<th>L/U-PTAs</th>
<th>IP-PTAs</th>
<th>bPTAs</th>
<th>PTAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF-empt.</td>
<td>✓</td>
<td>✓</td>
<td>[HRSV02]</td>
<td>×</td>
<td>[Miller00]</td>
<td>[AHV93]</td>
</tr>
<tr>
<td>EF-univ.</td>
<td>✓</td>
<td>×</td>
<td>[BlT09]</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>AF-empt.</td>
<td>×</td>
<td>×</td>
<td>[JLR15]</td>
<td>×</td>
<td>×</td>
<td>[JLR15]</td>
</tr>
</tbody>
</table>

---

**EF-emptiness**

- Bounded L/U
- Closed L/U
- IP-PTA
- PTAs

**EF-universality**

- Bounded L/U
- Closed L/U
- IP-PTA
- PTAs

**AF-emptiness**

- Bounded L/U
- Closed L/U
- IP-PTA
- PTAs
Conclusion

- PTAs extensively studied

- A new **decidable** subclass: bounded IP-PTAs... but of limited interest
  - Other problems than EF-emptiness are **undecidable**
  - Membership **undecidable**
  - Exact synthesis **intractable**

- A syntactic subclass of IP-PTAs: reset-PTAs
  - Promising decidability results
Perspectives

- **Extend reset-PTAs**
  - Using the same restrictions as in hybrid systems
    
    [Henzinger et al., 1998]

- **AF-universality**: studied in another paper with a subtle border between decidability and undecidability for L/U-PTAs
  
  [André and Lime, 2016]

- **Study L-PTAs and U-PTAs**
  - Entirely open classes
Bibliography


References II

On the expressiveness of parametric timed automata.

Language emptiness of continuous-time parametric timed automata.
In ICALP, Part II, volume 9135 of Lecture Notes in Computer Science, pages 69–81. Springer.

Decision problems for lower/upper bound parametric timed automata.

Robust parametric reachability for timed automata.

What’s decidable about hybrid automata?
References III


*Journal of Logic and Algebraic Programming, 52-53:*183–220.


Licensing
Source of the graphics used I

Title: Smiley green alien big eyes (aaah)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain

Title: Smiley green alien big eyes (cry)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain
License of this document

This presentation can be published, reused and modified under the terms of the license Creative Commons Attribution-ShareAlike 4.0 Unported (CC BY-SA 4.0)

(LATEX source available on demand)

Author: Étienne André, Didier Lime, Olivier H. Roux

https://creativecommons.org/licenses/by-sa/4.0/