On the Expressiveness of Parametric Timed Automata

Étienne André¹,², Didier Lime², Olivier H. Roux²

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

A model of the system

A property to be satisfied

\( y = \text{delay} \)

\( x := 0 \)

\( x < \text{period} \)

\( \text{is unreachable} \)
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

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

**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. Motivation
3. Definitions
4. Integers vs. Rationals
5. Comparison of the Expressiveness
6. Conclusion and Perspectives
Outline

1 Parametric Timed Automata

2 Motivation

3 Definitions

4 Integers vs. Rationals

5 Comparison of the Expressiveness

6 Conclusion and Perspectives
Timed automaton (TA)

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

- Finite state automaton (sets of locations and actions)
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

\[
\begin{align*}
x &:= 0 \\
y &:= 0 \\
y &= 5 \\
x &\geq 1
\end{align*}
\]
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

---

![Diagram of a Timed Automaton]

- \( y \leq 5 \)
- \( y \leq 8 \)
- \( x \geq 1 \)
**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

```
y ≤ 5
```

```
x ≥ 1
```

press?

```
y = 8
```

```
cup!
```

press?

```
y = 5
```

```
y ≤ 8:
```

press?
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

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

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

Possible concrete runs for the coffee machine

- Coffee with no sugar

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

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

\[ x \quad y \]
\[ 0 \quad 0 \]

\[ x \quad y \]
\[ 15.4 \quad 15.4 \]
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

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

Possible concrete runs for the coffee machine

- Coffee with no sugar
Examples of concrete runs

y = 8
coffee!

y ≤ 5

y = 5
cup!

Possible concrete runs for the coffee machine

- Coffee with no sugar

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

Possible concrete runs for the coffee machine

- Coffee with no sugar

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15.4</td>
</tr>
<tr>
<td>0</td>
<td>15.4</td>
</tr>
<tr>
<td>0</td>
<td>15.4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
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**

- **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 | 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 | 1.5 | 0 |
  | y  | 0 | 0 | 1.5 | 1.5 |
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
    
    \[
    \begin{array}{cccccccc}
    \text{x} & 0 & 15.4 & 0 & 5 & 5 & 8 & 8 \\
    \text{y} & 0 & 15.4 & 0 & 5 & 5 & 8 & 8 \\
    \end{array}
    \]
  - Coffee with 2 doses of sugar
    
    \[
    \begin{array}{cccccccc}
    \text{x} & 0 & 0 & 1.5 & 0 & 2.7 & 0 \\
    \text{y} & 0 & 0 & 1.5 & 1.5 & 4.2 & 4.2 \\
    \end{array}
    \]
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**

  - Initial state: $y = 8$
  - Transition: $x \geq 1$, press?
  - Transition: $x := 0$
  - Transition: $y \leq 5$
  - Transition: $y := 5$
  - Transition: $y = 5$
  - Transition: $x := 0$
  - Transition: $x \geq 0$

- **Coffee with 2 doses of sugar**

  - Initial state: $y = 8$
  - Transition: $x \geq 0$
  - Transition: $x := 0$
  - Transition: $y \leq 8$
  - Transition: $y := 8$
  - Transition: $x \geq 0$
  - Transition: $x := 0$
  - Transition: $x \geq 1$
  - Transition: $y := 8$
  - Transition: $x \geq 0$
  - Transition: $x := 0$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>2.7</td>
<td>0</td>
</tr>
<tr>
<td>2.7</td>
<td>0</td>
</tr>
<tr>
<td>0.8</td>
<td>0</td>
</tr>
<tr>
<td>0.8</td>
<td>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**

  ![Diagram for coffee with no sugar](image)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>press?</th>
<th>x</th>
<th>y</th>
<th>cup!</th>
<th>coffee!</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>15.4</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>15.4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

- **Coffee with 2 doses of sugar**

  ![Diagram for coffee with 2 doses of sugar](image)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>0</td>
<td>1.5</td>
<td>0</td>
<td>0.8</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>1.5</td>
<td>4.2</td>
<td>4.2</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
Parametric timed automaton (PTA)

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

```plaintext
\( \begin{align*}
    x &:= 0 \\
    y &:= 0 \\
    \text{press?} &\Rightarrow y \leq 5 \\
    \text{cup!} &\Rightarrow x \geq 1 \\
    \text{coffee!} &\Rightarrow y = 8
\end{align*} \)
Parametric timed automaton (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
coffee!
y \leq p_2
```

```
x := 0
press?
y := 0
```

```
x \geq p_1
cup!
y = p_2
press?
x := 0
```

```
y \leq 8:
```
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 $\nu$. 

\[
\begin{align*}
  &p_1 \rightarrow 1 \\
  &p_2 \rightarrow 5 \\
  &p_3 \rightarrow 8
\end{align*}
\]
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 $\nu$.

\[
\nu = \begin{cases} 
    p_1 \rightarrow 1 \\
    p_2 \rightarrow 5 \\
    p_3 \rightarrow 8 
\end{cases}
\]
Integers or rationals?

In PTAs, both the clocks and the parameters can be either integer-valued or rational-valued. This gives three possibilities:

<table>
<thead>
<tr>
<th></th>
<th>Clocks</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete time</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Dense time</td>
<td>$\mathbb{R}^+$</td>
<td>N</td>
</tr>
<tr>
<td>Dense time</td>
<td>$\mathbb{R}^+$</td>
<td>$\mathbb{Q}^+$</td>
</tr>
</tbody>
</table>
Integers or rationals?

In PTAs, both the clocks and the parameters can be either integer-valued or rational-valued. This gives three possibilities:

<table>
<thead>
<tr>
<th></th>
<th>Clocks</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete time</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Dense time</td>
<td>$\mathbb{R}^+$</td>
<td>N</td>
</tr>
<tr>
<td>Dense time</td>
<td>$\mathbb{R}^+$</td>
<td>$\mathbb{Q}^+$</td>
</tr>
</tbody>
</table>

...and this has an impact on decidability

[Alur et al., 1993, Miller, 2000, Beneš et al., 2015]
Integers or rationals?

In PTAs, both the clocks and the parameters can be either integer-valued or rational-valued. This gives three possibilities:

<table>
<thead>
<tr>
<th></th>
<th>Clocks</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete time</td>
<td>$\mathbb{N}$</td>
<td>$\mathbb{N}$</td>
</tr>
<tr>
<td>Dense time</td>
<td>$\mathbb{R}^+$</td>
<td>$\mathbb{N}$</td>
</tr>
<tr>
<td>Dense time</td>
<td>$\mathbb{R}^+$</td>
<td>$\mathbb{Q}^+$</td>
</tr>
</tbody>
</table>

...and this has an impact on decidability

[Alur et al., 1993, Miller, 2000, Beneš et al., 2015]

Here: we consider (mainly) dense time with integer-valued parameters
Outline

1 Parametric Timed Automata

2 Motivation

3 Definitions

4 Integers vs. Rationals

5 Comparison of the Expressiveness

6 Conclusion and Perspectives
The question of the syntax

Almost each work in the literature defines its own syntax:

- guards
- invariants
- accepting locations or not

Are these definitions equivalently expressive?
The question of the syntax

Almost each work in the literature defines its own syntax:

- guards
- invariants
- accepting locations or not

Are these definitions equivalently expressive?

Integer vs. rational valued parameters

- For a rational valued PTA, can we find an integer-valued PTA with the same expressiveness?
Subclasses of PTAs

Several subclasses were defined for PTAs, e.g.,:

- **Lower-bound upper-bound PTAs**
  
  [Hune et al., 2002, Bozzelli and La Torre, 2009]

- **Bounded** PTAs (i.e., with a bounded parameter domain)

Are these subclasses less expressive than PTAs?
**Subclasses of PTAs**

Several subclasses were defined for PTAs, e.g.,:

- **Lower-bound upper-bound PTAs**

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

- **Bounded PTAs** (i.e., with a bounded parameter domain)

Are these subclasses less expressive than PTAs?

And are PTAs any more expressive than TAs...?
Towards an expressiveness for PTAs

Problem
What is the expressiveness of a PTA?
Towards an expressiveness for PTAs

Problem
What is the expressiveness of a PTA?

Goal
Propose a definition of expressiveness for PTAs, and compare the syntax / subclasses of PTAs.
Outline

1 Parametric Timed Automata
2 Motivation
3 Definitions
4 Integers vs. Rationals
5 Comparison of the Expressiveness
6 Conclusion and Perspectives
Definition (PTA with hidden parameters)

A parametric timed automaton with hidden parameters (hereafter hPTA) $A$ is a tuple $(\Sigma, L, l_0, F, X, P, I, E)$, where:

1. $\Sigma$ is a finite set of actions,
2. $L$ is a finite set of locations,
3. $l_0 \in L$ is the initial location,
4. $F \subseteq L$ is a set of accepting locations,
5. $X$ is a finite set of clocks,
6. $P = P_v \uplus P_v$ is a finite set of parameters partitioned into hidden parameters $P_v$ and visible parameters $P_v$,
7. $I$ is the invariant, assigning to every $l \in L$ a guard $I(l)$,
8. $E$ is a finite set of edges $e = (l, g, a, R, l')$ where $l, l' \in L$ are the source and target locations, $a \in \Sigma \cup \{\epsilon\}$ ($\epsilon$: silent action), $R \subseteq X$ is a set of clocks to be reset, and $g$ is a guard.
Lower-bound / Upper-bound PTAs

Definition (hL/U-PTA [Hune et al., 2002, Bozzelli and La Torre, 2009])

An hL/U-PTA is an hPTA where the set of parameters is partitioned into:

1. a set of lower-bound parameters $P^-$ where each parameter can only be compared to a clock as a lower bound ("$p \leq x$") and,
2. a set of upper-bound parameters $P^+$ ("$x \leq p$").

L/U-PTAs (without hidden parameters) benefit from some decidability properties, in contrast to general PTAs [Hune et al., 2002, Bozzelli and La Torre, 2009]
A first definition of expressiveness

Definition (untimed language of an hPTA)

Given an hPTA $\mathcal{A}$, the un timed language $UL(\mathcal{A})$ of $\mathcal{A}$ is

$$\bigcup_{v \in V(\mathcal{P})} \{ w \mid w \text{ is an un timed word accepted by } v(\mathcal{A}) \}$$

Untimed word accepted by a TA: sequence of actions associated with a finite (timed) run ending in an accepting location
A second definition of expressiveness

Definition (constrained untimed language of an hPTA)

Given an hPTA $A$, the constrained untimed language $\text{CUL}(A)$ of $A$ is

$$\bigcup_{\nu \in \mathcal{N}(P_{\nu})} \{(w, \nu) \mid \exists \overline{\nu} \in \mathcal{N}(P_{\overline{\nu}}) \text{ s.t. } w \text{ is an untimed word of } \nu(\overline{\nu}(A))\}$$

Remarks:

- Only the visible parameters are considered
- Constrained because an alternative way to represent $\text{CUL}(A)$ is a set of pairs $(w, K)$ (where $K$ denotes all valuations accepting $w$)
- Only PTAs with the same number of visible parameters can be compared using $\text{CUL}$
**Example**

\[ \text{UL}(\mathcal{A}) = \{a\} \cup \{ba^n \mid n \in \mathbb{N}\} \text{ (also written } \text{UL}(\mathcal{A}) = a + ba^*) \]
Example

UL(\(A\)) = \{a\} \cup \{ba^n \mid n \in \mathbb{N}\} (also written UL(\(A\)) = a + ba^*)

CUL(\(A\)) = \{(a, p_1 = i) \mid 0 \leq i \leq 1\} \cup \{(ba^n, p_1 = i) \mid i \in \mathbb{N}, n \in \mathbb{N}\}
(also written CUL(\(A\)) = \{(a, p_1 \leq 1), (ba^*, p_1 \geq 0)\}, with p_1 \in \mathbb{N}\)

Note: both the parameter p_2 and the fact that p_2 must be at least 1 to go to \(\square\) are hidden.
Outline

1. Parametric Timed Automata
2. Motivation
3. Definitions
4. Integers vs. Rationals
5. Comparison of the Expressiveness
6. Conclusion and Perspectives
Integers vs. Rationals

**Proposition (Rationals and integers are equivalent for UL)**

*PTAs with rational parameters and PTAs with unbounded integer parameters are equivalent with respect to the untimed language.*
Proposition (Rationals and integers are equivalent for UL)

*PTAs with rational parameters and PTAs with unbounded integer parameters are equivalent with respect to the untimed language.*

Proof idea

- Any integer-valued PTA can be simulated by a rational-valued PTA using appropriate gadgets
- A rational-valued PTA can be simulated by an integer-valued by scaling up all expressions: since we do not know by how much we need to scale, we add... a fresh parameter.

For example, $x \leq 3p_1 + 2p_2 + 7$ becomes $x \leq 3p_1 + 2p_2 + 7p$. 
Integers vs. Rationals

Proposition (Rationals and integers are equivalent for $\text{UL}$)

$\text{PTAs with rational parameters and PTAs with unbounded integer parameters are equivalent with respect to the untimed language.}$

Proof idea

- Any integer-valued PTA can be simulated by a rational-valued PTA using appropriate gadgets
- A rational-valued PTA can be simulated by an integer-valued by scaling up all expressions: since we do not know by how much we need to scale, we add... a fresh parameter.

For example $x \leq 3p_1 + 2p_2 + 7$ becomes $x \leq 3p_1 + 2p_2 + 7p$.

A similar result is shown for $L/U$-PTAs [see paper]
Outline

1. Parametric Timed Automata
2. Motivation
3. Definitions
4. Integers vs. Rationals
5. Comparison of the Expressiveness
6. Conclusion and Perspectives
General PTAs: type 0

Lemma

*Turing-recognizable languages are also recognizable by PTAs.*
General PTAs: type 0

Lemma

**Turing-recognizable languages** are also recognizable by PTAs.

Proof idea

- all proofs of undecidability for PTAs reduce from the halting problem of a 2-counter machine.
- 2-counter machines are Turing equivalent

e.g., [Alur et al., 1993]
Single-clock PTAs: type 3

Lemma

The untimed language recognized by a PTA with a single clock (and arbitrarily many rational-valued parameters) is regular.
**Lemma**

The untimed language recognized by a PTA with a single clock (and arbitrarily many rational-valued parameters) is *regular*.

**Proof idea**

From the fact that the parametric zone graph of a PTA with a single (necessarily parametric) clock and arbitrarily many parameters is finite.

[A. and Markey, 2015, Theorem 20]
Two clocks: at least type 1

Theorem

PTAs with 1 parametric clock, 1 non-parametric clock and 1 parameter can recognize languages that are context-sensitive.
Two clocks: at least type 1

Theorem

PTAs with 1 parametric clock, 1 non-parametric clock and 1 parameter can recognize languages that are context-sensitive.

Proof idea

The following PTA recognizes $a^n b^n c^n$

$x_1 = 1 \land x_2 = p \
\begin{array}{c}
x_1 = 0 \\
\end{array}
\begin{array}{c}
x_1 \leq 1 \\
\land x_2 \leq p \\
\end{array}$
Outline

5 Comparison of the Expressiveness
- PTAs in the Chomsky hierarchy
- Comparison w.r.t. the untimed language
- Comparison w.r.t. the constrained untimed language
Proposition

Most subclasses of PTAs are not more expressive than timed automata w.r.t. the untimed language.
**TAs** = **L/U-PTAs**

### Proposition

*Most subclasses of PTAs are not more expressive than timed automata w.r.t. the untimed language.*

### Proof idea

- **L/U-PTAs**: union over all parameter valuations equivalent to the TA replacing lower-bound parameters with 0 and upper-bound parameter with a sufficiently large constant

[Bozzelli and La Torre, 2009]
**TAs = L/U-PTAs = bounded PTAs**

**Proposition**

*Most subclasses of PTAs are not more expressive than timed automata w.r.t. the untimed language.*

**Proof idea**

- **L/U-PTAs**: union over all parameter valuations equivalent to the TA replacing lower-bound parameters with 0 and upper-bound parameter with a sufficiently large constant

  [Bozzelli and La Torre, 2009]

- **integer-valued bounded PTAs**: from the closure of regular languages
TAs < PTAs

Proposition

PTAs are strictly more expressive than TAs w.r.t. the untimed language.

Proof idea

From the results on the hierarchy of Chomsky.
PTAs = fcp-PTAs

**Proposition**

*Fully parametric constraints PTAs* (fpc-PTAs), i.e., allowing *parametric linear terms* \((3p_1 - p_2 > 5p_3)\) in guards, are not more expressive than PTAs w.r.t. the untimed language.

**Proof idea**

1. By translation to an equivalent PTA w.r.t. the untimed language, that follows the most restrictive syntax (that of [Alur et al., 1993])
Outline

5 Comparison of the Expressiveness
- PTAs in the Chomsky hierarchy
- Comparison w.r.t. the untimed language
- Comparison w.r.t. the constrained untimed language
Expressiveness w.r.t. the CUL

Main results:
L/U-PTAs < PTAs
Expressiveness w.r.t. the CUL

Main results:
- $L/U$-PTAs $<$ PTAs
- $hL/U$-PTAs $=$ $L/U$-PTAs
Expressiveness w.r.t. the CUL

Main results:
- $\text{L/U-PTAs} < \text{PTAs}$
- $\text{hL/U-PTAs} = \text{L/U-PTAs}$
- $\text{bL/U-PTA}$ incomp. with $\text{L/U}$
Expressiveness w.r.t. the CUL

Main results:
- \( L/U\text{-PTAs} < PTAs \)
- \( hL/U\text{-PTAs} = L/U\text{-PTAs} \)
- bounded PTAs < PTAs
- bL/U-PTA incomp. with L/U
Expressiveness w.r.t. the **CUL**

Main results:
- \( \text{L/U-PTAs} < \text{PTAs} \)
- \( \text{hL/U-PTAs} = \text{L/U-PTAs} \)
- \( \text{bL/U-PTA} \) incomp. with \( \text{L/U} \)
- \( \text{bounded PTAs} < \text{PTAs} \)
- \( \text{hPTAs} > \text{PTAs} \)

*(more in paper)*
Outline

1. Parametric Timed Automata
2. Motivation
3. Definitions
4. Integers vs. Rationals
5. Comparison of the Expressiveness
6. Conclusion and Perspectives
Summary

- First attempt to define the expressiveness of parametric timed automata and subclasses
  - Including by hiding parameters

- Untimed language: most formalisms are not more expressive than timed automata

- Constrained untimed language: proposed a first classification of the main subclasses of PTAs
Perspectives

Consider the timed language and constrained timed language as well

- (But in fact, results do not seem to differ from their untimed counterparts)

The expressiveness of two-clock PTAs remains open

- And close to well-known open decision problems

Power of silent transitions in our definitions?

Relationship between the guard syntax and the number of clocks and parameters

- Is using a syntax “\(x \sim p + c\)” equivalent to a syntax “\(x \sim p | c\)” at the cost of one extra clock in the PTA?
Bibliography
References I


In *STOC*, pages 592–601. ACM.


References II


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

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