ICECCS 2018
12 December 2018
Melbourne, Australia

Offline timed pattern matching under uncertainty

Étienne André\textsuperscript{1,2,3}, Ichiro Hasuo\textsuperscript{2,4} and Masaki Waga\textsuperscript{2,4}

\textsuperscript{1} LIPN, Université Paris 13, CNRS, France
\textsuperscript{2} National Institute of Informatics, Japan
\textsuperscript{3} JFLI, UMI CNRS, Tokyo, Japan
\textsuperscript{4} SOKENDAI (The Graduate University for Advanced Studies)

Supported by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603) and the ANR national research program PACS (ANR-14-CE28-0002).
Motivation: automotive industry

- Modern cars embed several processors and produce logs

Log: sequences of events and timestamps

start 2.3

gear1 5.8

gear2 9.2

gear3 18.5

gear2 42.1

How to ensure on-the-fly that some properties are satisfied on a log?

“Never happens that gear/one.osf and gear/three.osf are separated by less than/five.osf/s”

⇒ Monitoring

Étienne André

Parametric timed pattern matching
Motivation: automotive industry

- Modern cars embed several processors and produce logs

Log: sequences of events and timestamps

<table>
<thead>
<tr>
<th>Event</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>2.3</td>
</tr>
<tr>
<td>gear1</td>
<td>5.8</td>
</tr>
<tr>
<td>gear2</td>
<td>9.2</td>
</tr>
<tr>
<td>gear3</td>
<td>18.5</td>
</tr>
<tr>
<td>gear2</td>
<td>42.1</td>
</tr>
</tbody>
</table>

⇒ Monitoring

Étienne André
Motivation: automotive industry

- Modern cars embed several processors and produce logs

- Log: sequences of events and timestamps
  - start 2.3
  - gear1 5.8
  - gear2 9.2
  - gear3 18.5
  - gear2 42.1

- How to ensure on-the-fly that some properties are satisfied on a log?
  - “It never happens that gear1 and gear3 are separated by less than 5 s”
Motivation: automotive industry

- Modern cars embed several processors and produce logs

Log: sequences of events and timestamps

<table>
<thead>
<tr>
<th>Event</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>2.3</td>
</tr>
<tr>
<td>gear1</td>
<td>5.8</td>
</tr>
<tr>
<td>gear2</td>
<td>9.2</td>
</tr>
<tr>
<td>gear3</td>
<td>18.5</td>
</tr>
<tr>
<td>gear2</td>
<td>42.1</td>
</tr>
</tbody>
</table>

- How to ensure on-the-fly that some properties are satisfied on a log?
  - “It never happens that $\text{gear}_1$ and $\text{gear}_3$ are separated by less than 5 s”
  ⇒ Monitoring
Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

Personal mobile devices collect large amounts of data.

These data can also come in the form of a timed log:

- start walking: 2.3
- walk faster: 6.3
- receive SMS: 15.8
- read SMS: 19.2
- sound of someone bumping into a lamp: 22.5

Key challenge: manage these data

Verify properties: “has the owner bumped into a street lamp?”

key applications (health, . . . )

Deduce information:

- “what are the minimum/maximum intervals without visiting this shop?”
- “is the user visiting this place more or less periodically?” (without knowing the actual period)
Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

These data can also come in the form of a timed log:
start walking 2.3
walk faster
receive SMS
read SMS
sound of someone bumping into a lamp

Key challenge: manage these data
Verify properties: "has the owner bumped into a street lamp?"
Deduce information:
"what are the minimum/maximum intervals without visiting this shop?"
"is the user visiting this place more or less periodically?" (without knowing the actual period)
Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data
  
  These data can also come in the form of a timed log:
  
  - start walking: 2.3
  - walk faster: 6.3
Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data
  
  These data can also come in the form of a timed log:
  
<table>
<thead>
<tr>
<th>Event</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>start walking</td>
<td>2.3</td>
</tr>
<tr>
<td>walk faster</td>
<td>6.3</td>
</tr>
<tr>
<td>receive SMS</td>
<td>15.8</td>
</tr>
</tbody>
</table>

Key challenge: manage these data

Verify properties: "has the owner bumped into a street lamp?"

Deduce information:
- "what are the minimum/maximum intervals without visiting this shop?"
- "is the user visiting this place more or less periodically?" (without knowing the actual period)
Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

These data can also come in the form of a timed log:

- start walking: 2.3
- walk faster: 6.3
- receive SMS: 15.8
- read SMS: 19.2
Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

  These data can also come in the form of a timed log:
  - start walking: 2.3
  - walk faster: 6.3
  - receive SMS: 15.8
  - read SMS: 19.2
  - sound of someone bumping into a lamp: 22.5

Key challenge: manage these data

Verify properties: “has the owner bumped into a street lamp?”

Key applications (health, ...)

Deduce information:
- “what are the minimum/maximum intervals without visiting this shop?”
- “is the user visiting this place more or less periodically?” (without knowing the actual period)
Larger motivation: data collection and management

- Personal mobile devices collect large amounts of data

These data can also come in the form of a timed log:
- start walking 2.3
- walk faster 6.3
- receive SMS 15.8
- read SMS 19.2
- sound of someone bumping into a lamp 22.5

- Key challenge: manage these data
  - Verify properties: “has the owner bumped into a street lamp”?
  - key applications (health, …)
  - Deduce information:
    - “what are the minimum/maximum intervals without visiting this shop”?
    - “is the user visiting this place more or less periodically?” (without knowing the actual period)
Outline

1. Pattern matching
2. Methodology
3. Experiments
4. Perspectives

Étienne André
### Untimed pattern matching

<table>
<thead>
<tr>
<th>Problem</th>
<th>log (target)</th>
<th>specification (pattern)</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>string matching</td>
<td>word</td>
<td>word $pat \in \Sigma^*$</td>
<td>${(i, j) \in (\mathbb{Z}_{&gt;0})^2 \mid w(i, j) = pat}$</td>
</tr>
<tr>
<td>pattern matching (PM)</td>
<td>word</td>
<td>NFA $\mathcal{A}$</td>
<td>${(i, j) \in (\mathbb{Z}_{&gt;0})^2 \mid w(i, j) \in \mathcal{L}(\mathcal{A})}$</td>
</tr>
<tr>
<td>timed PM</td>
<td>timed word</td>
<td>TA $\mathcal{A}$</td>
<td>${(t, t') \in (\mathbb{R}_{&gt;0})^2 \mid w</td>
</tr>
<tr>
<td>parametric timed PM</td>
<td>timed word</td>
<td>PTA $\mathcal{A}$</td>
<td>${(t, t', v) \mid w</td>
</tr>
</tbody>
</table>
Untimed pattern matching: example

- Naive algorithm for pattern matching

```latex
\text{crepes} \in \mathcal{L}(\{c|id\}^* e)
```
Untimed pattern matching: example

- Naive algorithm for pattern matching

\[ \text{crepes} \in L(\{c|d\}^*e) \]

\[ c \]
Untimed pattern matching: example

- Naive algorithm for pattern matching

```latex
\texttt{crepes} \in \mathcal{L}\left(\{c|\text{id}\}^*r*e\right)
```

Étienne André
Untimed pattern matching: example

- Naive algorithm for pattern matching

\[
\text{cr} \text{e} \text{p} \text{e} \text{s} \in \mathcal{L}(\{c|d\}^*e)
\]

\[
c \text{r} \text{e}
\]
Untimed pattern matching: example

- Naive algorithm for pattern matching

\[ \text{crepes} \in L\left(\{c|\ldots|d\}^*\right) \]

\[ c r e \checkmark \]
Untimed pattern matching: example

- Naive algorithm for pattern matching

```
c r e p e s ∈?L({c|i|d}?r*e)
c r e √
r
```
Untimed pattern matching: example

- Naive algorithm for pattern matching

\[
\text{crepes} \in \mathcal{L}(\{c|d\}^*r^*e)
\]

\[
\text{cre} \quad \checkmark \\
\text{ree}
\]
Untimed pattern matching: example

- Naive algorithm for pattern matching

| c | r | e | p | e | s | ∈?L(\{c|i|d\}?r*e) |
|---|---|---|---|---|---|-----------------------|
| c | r | e | ✓ |   |   |                       |
| r | e | ✓ |   |   |   |                       |

Étienne André
Untimed pattern matching: example

- Naive algorithm for pattern matching

| c  | r  | e  | p  | e  | s  | ∈ L(\{c|d\}^* r^e) |
|----|----|----|----|----|----|-------------------|
| c  | r  | e  |    |    |    | ✓                 |
| r  | e  |    |    |    |    | ✓                 |
| e  |    |    |    |    |    | ✓                 |
Untimed pattern matching: example

- Naive algorithm for pattern matching

| crepes | ∈?L(\{c|d\}?r*e) |
|--------|------------------|
| c      | √                |
| r      | √                |
| e      | √                |
| e      | √                |

Étienne André
Parametric timed pattern matching
12 December 2018
# Untimed pattern matching: example

- Naive algorithm for pattern matching

| c | r | e | p | e | s | ∈?L({c|i|d}?r*e) |
|---|---|---|---|---|---|-----------------|
| c | r | e | ✓ |   |   |                 |
| r | e | ✓ |   |   |   |                 |
| e | ✓ |   |   |   |   |                 |
| p |   |   |   |   |   |                 |
Untimed pattern matching: example

- Naive algorithm for pattern matching

| c | r | e | p | e | s | ∈?L({c|i|d}?r*e) |
|---|---|---|---|---|---|------------------|
| c | r | e | √ |
| r | e | √ |
| e | √ |
| p | ✗ |
Untimed pattern matching: example

- Naive algorithm for pattern matching

```
\
c r e p e s ∈?L(\{c|i|d\}? r*e)
```

```
c r e ✔
  r e ✔
   e ✔
    p ✗
     e
```
Untimed pattern matching: example

- Naive algorithm for pattern matching

| c r e p e s | $\in?L(\{c|d\}?r^*e)$ |
|-------------|-------------------|
| c r e       | ✓                 |
| r e         | ✓                 |
| e           | ✓                 |
| p           | X                 |
| e           | ✓                 |
Untimed pattern matching: example

- Naive algorithm for pattern matching

```
|   c r e p e s   | ∈?L({c|i|d}?r*e) |
|---------------|------------------|
| c r e         | ✓                |
|  r e          | ✓                |
|  e            | ✓                |
|      p e      | ×                |
|        e s    | ✓                |
```
Untimed pattern matching: example

- Naive algorithm for pattern matching

<table>
<thead>
<tr>
<th>c</th>
<th>r</th>
<th>e</th>
<th>p</th>
<th>e</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>r</td>
<td>e</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>e</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Untimed pattern matching: example

- Naive algorithm for pattern matching

```
crepes ∈?L(\{c|id\}?r*e)
```

```
cre
re
  e

p
  e

s
```

```
c r e p e s
```

```
1
2
3
4
```
### Timed pattern matching

<table>
<thead>
<tr>
<th>Problem</th>
<th>log (target)</th>
<th>specification (pattern)</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>string matching</td>
<td>word</td>
<td>word $pat \in \Sigma^*$</td>
<td>${(i, j) \in (\mathbb{Z}_{&gt;0})^2 \mid w(i, j) = pat}$</td>
</tr>
<tr>
<td>pattern matching (PM)</td>
<td>word</td>
<td>NFA $A$</td>
<td>${(i, j) \in (\mathbb{Z}_{&gt;0})^2 \mid w(i, j) \in L(A)}$</td>
</tr>
<tr>
<td>timed PM</td>
<td>timed word</td>
<td>TA $A$</td>
<td>${(t, t') \in (\mathbb{R}<em>{&gt;0})^2 \mid w\mid</em>{(t,t')} \in L(A)}$</td>
</tr>
<tr>
<td>parametric timed PM</td>
<td>timed word</td>
<td>PTA $A$</td>
<td>${(t, t', v) \mid w\mid_{(t,t')} \in L(v(A))}$</td>
</tr>
</tbody>
</table>

Étienne André
Timed pattern matching: timed word

Timed word = sequence of actions and timestamps

[Alur and Dill, 1994]
Timed pattern matching: timed word

Timed word

= sequence of actions and timestamps

Timed word segment

= projection of a segment of the timed word onto a given interval

[Alur and Dill, 1994]

[Waga et al., 2016]
Timed pattern matching: timed automaton

How to express a (timed) property on a log?

Example

“At least 1 time unit after the start of the segment, a is observed. Then, within strictly less than 1 time unit, another a is observed. Then, within strictly less than 1 time unit, another a is observed.”
Timed pattern matching: timed automaton

How to express a (timed) property on a log?

**Example**

“At least 1 time unit after the start of the segment, a is observed. Then, within strictly less than 1 time unit, another a is observed. Then, within strictly less than 1 time unit, another a is observed.”

A solution: **timed automata**

```
\begin{align*}
x > 1 & \quad \text{a} \\
\text{x := 0} & \quad \text{a} \\
\text{x < 1} & \quad \text{true}
\end{align*}
```

- expressive
- well-studied
- supported by well-established model-checkers

[Alur and Dill, 1994]
Timed automaton (TA)

- Finite state automaton (sets of locations)

![Diagram of a Timed Automaton with states: idle, adding sugar, delivering coffee.]

Features:
- Location invariant: property to be verified to stay at a location
- Transition guard: property to be verified to enable a transition
- Clock reset: some of the clocks can be set to 0 along transitions

Real-valued variables evolving linearly at the same rate can be compared to integer constants in invariants and guards.
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)

---

Finite state automaton (sets of locations and actions)

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

---

**Example:**

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

y = 5

x ≥ 1

press? : x := 0

coffee! : y = 8

cup! : x := 0
```

---

**States:**

- Green: idle
- Blue: adding sugar
- Red: delivering coffee
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks
- Real-valued variables evolving linearly at the same rate

[Alur and Dill, 1994]

Features

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

```
press? := 0
y := 0
y := 5
x := 0
y = 8
x ≥ 1
cup!
coffee!

idle
adding sugar
delivering coffee
```
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants

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

$$\begin{align*}
  y &\leq 5 \\
  y &\leq 8
\end{align*}$$
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

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

\[
\begin{align*}
    y &\leq 8 \\
    \text{coffee!} \\
    x &\geq 1 \\
    \text{press?} \\
    y &\leq 5 \\
    \text{cup!} \\
    x &\geq 1 \\
    \text{press?} \\
    y &\leq 8 \\
    \text{delivering coffee}
\end{align*}
\]
Timed automaton (TA)

- Finite state automaton (sets of **locations** and **actions**) augmented with a set $X$ of **clocks**
  - Real-valued variables evolving linearly **at the same rate**
  - Can be compared to integer constants in invariants and guards

**Features**

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

```
y ≤ 5
y = 5
y ≥ 8

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

coffee!

cup!

idle
adding sugar
delivering coffee
```
Timed pattern matching: principle

Timed pattern matching

- **Inputs**

**A log**
(timed word)

<table>
<thead>
<tr>
<th>a</th>
<th>a</th>
<th>b</th>
<th>b</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>0.9</td>
<td>1.3</td>
<td>1.7</td>
</tr>
</tbody>
</table>

**A property**
usually a specification of faults
(timed automaton)

[Alur and Dill, 1994]

\[
x > 1 \quad x < 1 \\
a \quad a \\
x := 0 \quad x := 0 \\
\]

\[\begin{array}{c}
\text{true} \\
\rightarrow (0) \quad (1) \quad (2) \quad (3) \quad (4) \\
\end{array}\]

- **Output**

- The set of time intervals where faults are detected

\[\Rightarrow \text{Set of matching intervals } \{(t, t') \mid w|_{(t,t')} \in \mathcal{L}(A)\}\]
Timed pattern matching: example

Our property:

\[
\begin{align*}
x & > 1 \\
x & < 1 \\
\end{align*}
\]

\[
\begin{align*}
a & : x := 0 \\
a & : x := 0 \\
\end{align*}
\]

\[
\begin{align*}
x & < 1 \\
\text{true} \\
\end{align*}
\]

Our log:

\[
\begin{align*}
a & 0.5 \\
a & 0.9 \\
b & 1.3 \\
b & 1.7 \\
a & 2.8 \\
a & 3.7 \\
a & 4.9 \\
a & 5.3 \\
a & 6.0 \\
\end{align*}
\]
Timed pattern matching: example

Our property:

\[
\begin{align*}
  x > 1 & \quad \text{a} \quad x := 0 \\
  x < 1 & \quad \text{a} \quad x := 0 \\
                 & \quad \text{a} \\
  x < 1 & \quad \text{true} \\
\end{align*}
\]

Our log:

\[
\begin{align*}
  w & \quad a \quad a \quad b \quad b \\
  t & \quad a \quad a \quad t \\
  t' & \quad a \quad a \quad a \quad a \\
\end{align*}
\]
Timed pattern matching: example

Our property:

\[
\begin{align*}
x > 1 & \quad a \\
x := 0 & \\
x < 1 &
\end{align*}
\]

\[
\begin{align*}
x < 1 & \quad a \\
x := 0 & \\
true &
\end{align*}
\]

Our log:

\[
\begin{array}{cccccccc}
  w & a & a & b & b & a & a & t & a & a & a & t' \\
  0 & 0.5 & 0.9 & 1.3 & 1.7 & 2.8 & 3.7 & 4.9 & 5.3 & 6.0
\end{array}
\]

Set of matching intervals:

\[
\{(t, t') \mid w|_{(t,t')} \in L(A)\} = \{(t, t') \mid t \in (3.7, 3.9), t' \in [6.0, \infty)\}
\]
Previous works

- **Timed pattern matching with signals**
  
  - Logs are encoded by **signals** (i.e., values that vary over time)
  - *state-based* view, while our timed words are *event-based*
  - Specification is encoded by timed regular expressions (TREs)

- **Timed pattern matching with timed words and timed automata**

  - [Waga et al., 2016, Waga et al., 2017]: brute-force and Boyer-Moore algorithm
  - [Waga et al., 2017]: online algorithm that employs skip values from the Franek–Jennings–Smyth string matching algorithm [Franek et al., 2007]
Goal: Extend timed pattern matching for uncertainty

Challenges

- The property may not be known with full certainty:
  - Detect a periodic event but without knowing the period
    - “is the user visiting this place more or less periodically?” (without knowing the actual period)

- Optimization problems
  - Find minimal/maximal timings for which some property holds
    - “what are the minimum/maximum intervals without visiting this shop”?
Goal: Extend timed pattern matching for uncertainty

Challenges

- The property may not be known with full certainty:
  - Detect a periodic event but without knowing the period
  - “is the user visiting this place more or less periodically?” (without knowing the actual period)

- Optimization problems
  - Find minimal/maximal timings for which some property holds
  - “what are the minimum/maximum intervals without visiting this shop”?

Objective

Find intervals of time and values of parameters for which a property holds

<table>
<thead>
<tr>
<th>Problem</th>
<th>log (target)</th>
<th>specification (pattern)</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>string matching</td>
<td>word</td>
<td>word $pat \in \Sigma^*$</td>
<td>${(i, j) \in (\mathbb{Z}_{&gt;0})^2 \mid w(i, j) = pat}$</td>
</tr>
<tr>
<td>pattern matching (PM)</td>
<td>word</td>
<td>NFA $\mathcal{A}$</td>
<td>${(i, j) \in (\mathbb{Z}_{&gt;0})^2 \mid w(i, j) \in \mathcal{L}(\mathcal{A})}$</td>
</tr>
<tr>
<td>timed PM</td>
<td>timed word</td>
<td>TA $\mathcal{A}$</td>
<td>${(t, t') \in (\mathbb{R}_{&gt;0})^2 \mid w</td>
</tr>
<tr>
<td>parametric timed PM</td>
<td>timed word</td>
<td>PTA $\mathcal{A}$</td>
<td>${(t, t', v) \mid w</td>
</tr>
</tbody>
</table>

Étienne André

Parametric timed pattern matching

12 December 2018
Methodology

Main idea

Use parametric timed model checking

- parametric timed automata
- parameter synthesis
- IMITATOR

[Alur et al., 1993]
[André et al., 2012]
Methodology

Main idea

Use parametric timed model checking

- parametric timed automata [Alur et al., 1993]
- parameter synthesis
- IMITATOR [André et al., 2012]

Methodology step by step

1. Encode the property using a PTA
2. Add two parameters $t$ and $t'$
3. Apply a (mild) transformation to the property PTA
4. Transform the timed word into a PTA
5. Perform the composition of both PTA
6. Apply reachability synthesis to the product
Methodology

Main idea

Use parametric timed model checking

- parametric timed automata
- parameter synthesis
- IMITATOR

Methodology step by step

1. Encode the property using a PTA
2. Add two parameters $t$ and $t'$
3. Apply a (mild) transformation to the property PTA
4. Transform the timed word into a PTA
5. Perform the composition of both PTA
6. Apply reachability synthesis to the product

Teaser

Our method is scalable!
Outline

1 Pattern matching

2 Methodology
   • Parametric timed automata

3 Experiments

4 Perspectives
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?

Yes

No

Counterexample
Parametric timed model checking

A model of the system

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

  Yes if...

  

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

is unreachable

A property to be satisfied
Parametric Timed Automaton (PTA)

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

\[ y \leq 5 \]
\[ x \geq 1 \]
\[ x := 0 \]

\[ y = 8 \]
coffee!

\[ y = 5 \]
cup!

\[ y \leq 8 \]
Parametric Timed Automaton (PTA)

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

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

\[
y \leq p_2 \\
\text{press?}
\]

\[
x = 0 \\
y = 0
\]

\[
x \geq p_1 \\
\text{cup!}
\]

\[
x := 0 \\
y := p_2
\]
Expressing a **parametric timed** property on a log

**Example**

“At least $p_1$ time units after the start of the segment, $a$ is observed. Then, within strictly less than $p_2$ time units, another $a$ is observed. Then, within strictly less than $p_2$ time units, another $a$ is observed.”
Property: parametric timed automaton

Expressing a **parametric timed** property on a log

**Example**

“At least $p_1$ time units after the start of the segment, $a$ is observed. Then, within strictly less than $p_2$ time units, another $a$ is observed. Then, within strictly less than $p_2$ time units, another $a$ is observed.”

Our solution: parametric timed automata

[Alur et al., 1993]
Modifying the property pattern

Add some start and end gadgets for completeness of the method

See manuscript for formal transformation and proofs
Modifying the property pattern

Add some start and end gadgets for completeness of the method

1. Add an initial transition in o-time
   - Captures segments starting from o

See manuscript for formal transformation and proofs
Modifying the property pattern

Add some start and end gadgets for completeness of the method

1. Add an initial transition in o-time
   - Captures segments starting from o

2. Add a new location with a self-loop
   - Captures segments not starting from the beginning of the word

See manuscript for formal transformation and proofs
Modifying the property pattern

Add some start and end gadgets for completeness of the method

1. Add an initial transition in o-time
   - Captures segments starting from o

2. Add a new location with a self-loop
   - Captures segments not starting from the beginning of the word

3. Add a new final transition in > 0 time
   - To match the usual definition that the segment must end in > 0 time after the last action

See manuscript for formal transformation and proofs
Transforming a log into a (parametric) timed automaton

Essentially easy:

1. Add one clock never reset (absolute time)
2. Convert pairs (action, time) into transitions
Transforming a log into a (parametric) timed automaton

Essentially easy:

1. Add one clock never reset (absolute time)
2. Convert pairs (action, time) into transitions

```
\begin{tabular}{ccccccc}
\hline
w & a & a & b & b & a & a \\
0 & 0.5 & 0.9 & 1.3 & 1.7 & 2.8 & 3.7 \\
\hline
\end{tabular}
```
Transforming a log into a (parametric) timed automaton

Essentially easy:

1. Add one clock never reset (absolute time)
2. Convert pairs (action, time) into transitions

\[
\begin{align*}
    x_{abs} &= 0.5 & x_{abs} &= 0.9 & x_{abs} &= 1.3 & x_{abs} &= 1.7 & x_{abs} &= 2.8 \\
    w_0 \rightarrow a \rightarrow w_1 \rightarrow a \rightarrow w_2 \rightarrow b \rightarrow w_3 \rightarrow b \rightarrow w_4 \rightarrow a \rightarrow w_5 \\
    x_{abs} &= 6.0 & x_{abs} &= 5.3 & x_{abs} &= 4.9 & x_{abs} &= 3.7 \\
    w_9 \leftarrow a \leftarrow w_8 \leftarrow a \leftarrow w_7 \leftarrow a \leftarrow w_6
\end{align*}
\]
Product and reachability synthesis

Result

The set of parameter valuations $t, t', p_1, p_2, \ldots$ reaching the final location of the property is exactly the answer to the parametric pattern matching problem.

Remark

This problem is decidable. In contrast to most problems using PTAs!
Result

The set of parameter valuations $t, t', p_1, p_2, \ldots$ reaching the final location of the property is exactly the answer to the parametric pattern matching problem.

Remark

This problem is decidable… in contrast to most problems using PTAs!

[André, 2018]

See formal result in paper
Product and reachability synthesis: example

Our property:

\[ x > p_1 \]
\[ x < p_2 \]
\[ a \]
\[ x := 0 \]
\[ a \]
\[ x := 0 \]
\[ a \]
\[ x < p_2 \]
\[ \text{true} \]

Our log:

\[
\begin{array}{ccccccccc}
w & a & a & b & b & a & a & a & a \\
0 & 0.5 & 0.9 & 1.3 & 1.7 & 2.8 & 3.7 & 4.9 & 5.3 & 6.0 \\
\end{array}
\]
Product and reachability synthesis: example

Our property:

\[
\begin{align*}
&x > p_1 \\
&x < p_2
\end{align*}
\]

\[
\begin{align*}
a & \quad x := 0 & a & \quad x := 0 & a & \quad x < p_2 & \text{true}
\end{align*}
\]

Our log:

\[
\begin{array}{cccccccccc}
& a & a & b & b & a & a & a & a & a \\
\hline
w & 0 & 0.5 & 0.9 & 1.3 & 1.7 & 2.8 & 3.7 & 4.9 & 5.3 & 6.0
\end{array}
\]

Set of matching intervals:

\[
\begin{align*}
1.7 < t < 2.8 & - p_1 \land 4.9 \leq t' < 5.3 \land p_2 > 1.2 \\
\lor 2.8 < t < 3.7 & - p_1 \land 5.3 \leq t' < 6 \land p_2 > 1.2 \\
\lor 3.7 < t < 4.9 & - p_1 \land t' \geq 6 \land p_2 > 0.7
\end{align*}
\]
Product and reachability synthesis: example

Our property:

\[ x > p_1 \]
\[ x < p_2 \]
\[ x := 0 \]
\[ x := 0 \]
\[ x < p_2 \]
\[ \text{true} \]

Our log:

\[ w \]

<table>
<thead>
<tr>
<th></th>
<th>( a )</th>
<th>( a )</th>
<th>( b )</th>
<th>( b )</th>
<th>( a )</th>
<th>( a )</th>
<th>( a )</th>
<th>( a )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>0.0</td>
<td>0.5</td>
<td>0.9</td>
<td>1.3</td>
<td>1.7</td>
<td>2.8</td>
<td>3.7</td>
<td>4.9</td>
</tr>
</tbody>
</table>

Set of matching intervals:

\[ 1.7 < t < 2.8 - p_1 \land 4.9 \leq t' < 5.3 \land p_2 > 1.2 \]
\[ \lor 2.8 < t < 3.7 - p_1 \land 5.3 \leq t' < 6 \land p_2 > 1.2 \]
\[ \lor 3.7 < t < 4.9 - p_1 \land t' \geq 6 \land p_2 > 0.7 \]
Product and reachability synthesis: example

Our property:

\[
\begin{align*}
    x > p_1 & \quad a \\
    x := 0 & \quad \text{true} \\
    x := 0 & \quad x < p_2 \\
    a & \quad x < p_2 \\
\end{align*}
\]

Our log:

\[
\begin{array}{cccccccc}
    w & 0 & 0.5 & 0.9 & 1.3 & 1.7 & 2.8 & 3.7 & 4.9 & 5.3 & 6.0 \\
\end{array}
\]

Set of matching intervals:

\[
\begin{align*}
    1.7 < t < 2.8 - p_1 & \land 4.9 \leq t' < 5.3 & \land p_2 > 1.2 \\
\lor 2.8 < t < 3.7 - p_1 & \land 5.3 \leq t' < 6 & \land p_2 > 1.2 \\
\lor 3.7 < t < 4.9 - p_1 & \land t' \geq 6 & \land p_2 > 0.7
\end{align*}
\]
Product and reachability synthesis: example

Our property:

\[ x > p_1 \quad \text{and} \quad x < p_2 \]

\[ x := 0 \quad \text{and} \quad x := 0 \]

Our log:

<table>
<thead>
<tr>
<th>w</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>b</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>0</td>
<td>0.5</td>
<td>0.9</td>
<td>1.3</td>
<td>1.7</td>
<td>2.8</td>
<td>3.7</td>
<td>4.9</td>
<td>5.3</td>
</tr>
</tbody>
</table>

Set of matching intervals:

\[ 1.7 < t < 2.8 - p_1 \land 4.9 \leq t' < 5.3 \land p_2 > 1.2 \]
\[ \lor \quad 2.8 < t < 3.7 - p_1 \land 5.3 \leq t' < 6 \land p_2 > 1.2 \]
\[ \lor \quad 3.7 < t < 4.9 - p_1 \land t' \geq 6 \land p_2 > 0.7 \]
Exemple: graphical representation

\[ 1.7 < t < 2.8 - p_1 \land 4.9 \leq t' < 5.3 \land p_2 > 1.2 \]
\[ \lor \ 2.8 < t < 3.7 - p_1 \land 5.3 \leq t' < 6 \land p_2 > 1.2 \]
\[ \lor \ 3.7 < t < 4.9 - p_1 \land t' \geq 6 \land p_2 > 0.7 \]

Projections in 2 dimensions:

On \( p_1 \) and \( p_2 \)

On \( t \) and \( t' \)

On \( t \) and \( p_1 \)
Outline

1. Pattern matching
2. Methodology
3. Experiments
4. Perspectives
Outline

1. Pattern matching

2. Methodology

3. Experiments
   - IMITATOR in a nutshell
   - Benchmarks

4. Perspectives
IMITATOR

- A tool for modeling and verifying timed concurrent systems with unknown constants modeled with parametric timed automata
  - Communication through (strong) broadcast synchronization
  - Rational-valued shared discrete variables
  - Stopwatches, to model schedulability problems with preemption

- Synthesis algorithms
  - (non-Zeno) parametric model checking (using a subset of TCTL)
  - Language and trace preservation, and robustness analysis
  - Parametric deadlock-freeness checking

Étienne André
Parametric timed pattern matching

IMITATOR

Under continuous development since 2008

A library of benchmarks

- Communication protocols
- Schedulability problems
- Asynchronous circuits
- …and more

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

Under continuous development since 2008

A library of benchmarks
  ■ Communication protocols
  ■ Schedulability problems
  ■ Asynchronous circuits
  ■ …and more

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

Try it!

www.imitator.fr
Outline

1 Pattern matching

2 Methodology

3 Experiments
   - IMITATOR in a nutshell
   - Benchmarks

4 Perspectives
Experimental environment

Toolkit

- Simple Python script to transform timed words into IMITATOR PTAs
- Slightly modified version of IMITATOR
  - To handle PTAs with dozens of thousands of locations
  - To manage $n$-parameter constraints with dozens of thousands of disjuncts

Two algorithms:

- $\text{PTPM}$: parametric timed pattern matching
- $\text{PTPM}_{\text{opt}}$: parametric timed pattern matching with parameter optimization
  - e.g., “where in the log is the smallest value of the parameter $p$ s.t. the property is satisfied/violated?”

Sources, binaries, models, logs can be found at www.imitator.fr/static/ICECCS18
Case study 1: GEAR (description)

Monitoring the gear change of an automatic transmission system

- Obtained by simulation of the Simulink model of an automatic transmission system [Hoxha et al., 2014]
- S-TaLiRo [Annpureddy et al., 2011] used to generate an input to this model (generates a gear change signal that is fed to the model)
- Gear chosen from \( \{g_1, g_2, g_3, g_4\} \)
- Generated gear change recorded in a timed word

Property

“If the gear is changed to 1, it should not be changed to 2 within \( p \) seconds.”

This condition is related to the requirement \( \phi_{AT}^5 \) proposed in [Hoxha et al., 2014] (the nominal value for \( p \) in [Hoxha et al., 2014] is 2).
Case study 1: GEAR (experiments)

Property: “If the gear is changed to 1, it should not be changed to 2 within $p$ seconds.”

Experiments data:

<table>
<thead>
<tr>
<th>Model</th>
<th>Length</th>
<th>Time frame</th>
<th>States</th>
<th>Matches</th>
<th>Parsing (s)</th>
<th>Comp. (s)</th>
<th>States</th>
<th>Comp. (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTPM</td>
<td>1,467</td>
<td>1,000</td>
<td>4,453</td>
<td>379</td>
<td>0.02</td>
<td>1.60</td>
<td>3,322</td>
<td>0.94</td>
</tr>
<tr>
<td>PTPM_opt</td>
<td>2,837</td>
<td>2,000</td>
<td>8,633</td>
<td>739</td>
<td>0.33</td>
<td>2.14</td>
<td>6,422</td>
<td>1.70</td>
</tr>
<tr>
<td></td>
<td>4,595</td>
<td>3,000</td>
<td>14,181</td>
<td>1,247</td>
<td>0.77</td>
<td>3.63</td>
<td>10,448</td>
<td>2.85</td>
</tr>
<tr>
<td></td>
<td>5,839</td>
<td>4,000</td>
<td>17,865</td>
<td>1,546</td>
<td>1.23</td>
<td>4.68</td>
<td>13,233</td>
<td>3.74</td>
</tr>
<tr>
<td></td>
<td>7,301</td>
<td>5,000</td>
<td>22,501</td>
<td>1,974</td>
<td>1.94</td>
<td>5.88</td>
<td>16,585</td>
<td>4.79</td>
</tr>
<tr>
<td></td>
<td>8,995</td>
<td>6,000</td>
<td>27,609</td>
<td>2,404</td>
<td>2.96</td>
<td>7.28</td>
<td>20,413</td>
<td>5.76</td>
</tr>
<tr>
<td></td>
<td>10,316</td>
<td>7,000</td>
<td>31,753</td>
<td>2,780</td>
<td>4.00</td>
<td>8.38</td>
<td>23,419</td>
<td>6.86</td>
</tr>
<tr>
<td></td>
<td>11,831</td>
<td>8,000</td>
<td>36,301</td>
<td>3,159</td>
<td>5.39</td>
<td>9.75</td>
<td>26,832</td>
<td>7.87</td>
</tr>
<tr>
<td></td>
<td>13,183</td>
<td>9,000</td>
<td>40,025</td>
<td>3,414</td>
<td>6.86</td>
<td>10.89</td>
<td>29,791</td>
<td>8.61</td>
</tr>
<tr>
<td></td>
<td>14,657</td>
<td>10,000</td>
<td>44,581</td>
<td>3,816</td>
<td>8.70</td>
<td>12.15</td>
<td>33,141</td>
<td>9.89</td>
</tr>
</tbody>
</table>

PTPM$_{opt}$: alternative procedure to find the minimum/maximum value of a parameter along the log
Case study 2: **ACCEL (description)**

Monitoring the acceleration of an automated transmission system

- Also obtained by simulation from the Simulink model of [Hoxha et al., 2014]
- (discretized) value of three state variables recorded in the log:
  - engine RPM (discretized to “high” and “low” with a certain threshold)
  - velocity (discretized to “high” and “low” with a certain threshold)
  - 4 gear positions

**Property**

“If a gear changes from 1 to 2, 3, and 4 in this order in \( p \) seconds and engine RPM becomes large during this gear change, then the velocity of the car must be sufficiently large in one second.”

This condition models the requirement \( \phi_{AT} \) proposed in [Hoxha et al., 2014] (the nominal value for \( p \) in [Hoxha et al., 2014] is 10).
Case study 2: **AcCel** (experiments)

Property: “If a gear changes from 1 to 2, 3, and 4 in this order in $p$ seconds and engine RPM becomes large during this gear change, then the velocity of the car must be sufficiently large in one second.”

Experiments data:

<table>
<thead>
<tr>
<th>Model</th>
<th>PTPM Length</th>
<th>Time frame</th>
<th>States</th>
<th>Matches</th>
<th>Parsing (s)</th>
<th>Comp. (s)</th>
<th>PTPMopt States</th>
<th>Comp. (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/two.osf</td>
<td>2,559</td>
<td>1,000</td>
<td>6,504</td>
<td>2</td>
<td>0.27</td>
<td>1.60</td>
<td>6,502</td>
<td>1.85</td>
</tr>
<tr>
<td>/two.osf</td>
<td>4,894</td>
<td>2,000</td>
<td>12,429</td>
<td>2</td>
<td>0.86</td>
<td>3.04</td>
<td>12,426</td>
<td>3.57</td>
</tr>
<tr>
<td>/two.osf</td>
<td>7,799</td>
<td>3,000</td>
<td>19,922</td>
<td>7</td>
<td>2.21</td>
<td>4.98</td>
<td>19,908</td>
<td>6.06</td>
</tr>
<tr>
<td>/two.osf</td>
<td>10,045</td>
<td>4,000</td>
<td>25,520</td>
<td>3</td>
<td>3.74</td>
<td>6.51</td>
<td>25,514</td>
<td>7.55</td>
</tr>
<tr>
<td>/two.osf</td>
<td>12,531</td>
<td>5,000</td>
<td>31,951</td>
<td>9</td>
<td>6.01</td>
<td>8.19</td>
<td>31,926</td>
<td>9.91</td>
</tr>
<tr>
<td>/two.osf</td>
<td>15,375</td>
<td>6,000</td>
<td>39,152</td>
<td>7</td>
<td>9.68</td>
<td>10.14</td>
<td>39,129</td>
<td>12.39</td>
</tr>
<tr>
<td>/two.osf</td>
<td>17,688</td>
<td>7,000</td>
<td>45,065</td>
<td>9</td>
<td>13.40</td>
<td>11.61</td>
<td>45,039</td>
<td>14.06</td>
</tr>
<tr>
<td>/two.osf</td>
<td>20,299</td>
<td>8,000</td>
<td>51,660</td>
<td>10</td>
<td>18.45</td>
<td>13.52</td>
<td>51,629</td>
<td>16.23</td>
</tr>
<tr>
<td>/two.osf</td>
<td>22,691</td>
<td>9,000</td>
<td>57,534</td>
<td>11</td>
<td>24.33</td>
<td>15.33</td>
<td>57,506</td>
<td>18.21</td>
</tr>
<tr>
<td>/two.osf</td>
<td>25,137</td>
<td>10,000</td>
<td>63,773</td>
<td>13</td>
<td>31.35</td>
<td>16.90</td>
<td>63,739</td>
<td>20.61</td>
</tr>
</tbody>
</table>

Étienne André
Parametric timed pattern matching

12 December 2018

35 / 39
Case study 3: BLOWUP

Property made on purpose to test our scalability

Experiments data:

<table>
<thead>
<tr>
<th>Model</th>
<th>PTPM</th>
<th>PTPM\text{\text{opt}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>Time frame</td>
<td>States</td>
</tr>
<tr>
<td>200</td>
<td>101</td>
<td>20,602</td>
</tr>
<tr>
<td>400</td>
<td>202</td>
<td>81,202</td>
</tr>
<tr>
<td>600</td>
<td>301</td>
<td>181,802</td>
</tr>
<tr>
<td>800</td>
<td>405</td>
<td>322,402</td>
</tr>
<tr>
<td>1,000</td>
<td>503</td>
<td>503,002</td>
</tr>
</tbody>
</table>
Outline

1. Pattern matching
2. Methodology
3. Experiments
4. Perspectives
Summary

- New original method to monitor logs of real-time systems

- Methodology: parametric timed model checking

- Applications: automotive industry
  - Linear in the size of the log
  - Able to handle logs of dozens of thousands of events
    ⇒ scalable
Summary

- New original method to monitor logs of real-time systems

- Methodology: parametric timed model checking

- Applications: automotive industry
  - Linear in the size of the log
  - Able to handle logs of dozens of thousands of events
    ⇒ scalable

- An offline online algorithm
  - We believe our algorithm is in fact essentially online
    - No need for the whole log to start the analysis
    - The word could be fed to IMITATOR in an incremental manner
  - But the speed may need to be improved further
Perspectives

- **Extensions**
  - Improve the efficiency with **skipping**
  - Exploit the **polarity of parameters**
  - Use and extend the MONAA library

- **Graphical representation and interpretation**
  - How to interpret dozens of thousands of matches?
Perspectives

- **Extensions**
  - Improve the efficiency with skipping
  - Exploit the polarity of parameters
  - Use and extend the MONAA library

- **Graphical representation and interpretation**
  - How to interpret dozens of thousands of matches?
Bibliography
References

A theory of timed automata.

Parametric real-time reasoning.

André, É. (2018).
A benchmarks library for parametric timed model checking.

André, É. (2018).
What’s decidable about parametric timed automata?
*International Journal on Software Tools for Technology Transfer.*
To appear.

IMITATOR 2.5: A tool for analyzing robustness in scheduling problems.

S-TaLiRo: A tool for temporal logic falsification for hybrid systems.


References III

Online timed pattern matching using derivatives.

A Boyer-Moore type algorithm for timed pattern matching.
In Fränzle, M. and Markey, N., editors, FORMATS, volume 9884 of LNCS, pages 121–139. Springer.

Efficient online timed pattern matching by automata-based skipping.

MONAA: A tool for timed pattern matching with automata-based acceleration.
In MT@CPSWeek, pages 14–15. IEEE.
Additional explanation
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with two.osf doses of sugar

idle
adding sugar
delivering coffee
The most critical system: The coffee machine

Example of concrete run for the coffee machine

- Coffee with 2 doses of sugar

$x = 0$

$y = 0$
The most critical system: The coffee machine

- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

\[
x = 0 \\
y = 0
\]
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>1.5</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>press?</th>
<th>press?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>2.7</td>
<td>4.2</td>
<td>2.7</td>
<td></td>
</tr>
</tbody>
</table>
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>press?</th>
<th>x</th>
<th>y</th>
<th>press?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>1.5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>1.5</td>
<td></td>
<td>2.7</td>
<td>4.2</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>4.2</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

\[
\begin{align*}
x &= 0 & y &= 0 \\
x &= 0 & y &= 1.5 \\
x &= 0 & y &= 1.5 \\
x &= 0 & y &= 2.7 \\
x &= 0 & y &= 4.2 \\
x &= 0 & y &= 4.2 \\
x &= 0.8 & y &= 5
\end{align*}
\]
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

\[
\begin{align*}
\text{x} &= 0 & 0 & 1.5 & 0 & 2.7 & 0 & 0.8 & 0.8 \\
\text{y} &= 0 & 0 & 1.5 & 1.5 & 4.2 & 4.2 & 5 & 5
\end{align*}
\]
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

x = 0 0 1.5 0 2.7 0 0.8 0.8 0.8 3.8 3.8
y = 0 0 1.5 1.5 4.2 4.2 5 5 8 8
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

Example: \((\text{ }, (x=1.2, y=3.7))\)

- **Concrete run**: alternating sequence of concrete states and actions or time elapse
Symbolic semantics of parametric timed automata

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a convex polyhedron over \(X\) and \(P\) with a special form, called parametric zone

[Hune et al., 2002]
Symbolic semantics of parametric timed automata

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a convex polyhedron over \(X\) and \(P\) with a special form, called parametric zone [Hune et al., 2002]

- **Symbolic run**: alternating sequence of **symbolic states** and **actions**
Symbolic semantics of parametric timed automata

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a convex polyhedron over \(X\) and \(P\) with a special form, called parametric zone

[Hune et al., 2002]

- **Symbolic run**: alternating sequence of symbolic states and actions

**Example**

\[
\begin{align*}
    x &\leq p_1 \\
    x &\geq p_2 \\
    y &:= 0 \\
    x &\leq p_3 \\
    b &
\end{align*}
\]

- **Possible symbolic run for this PTA**
Symbolic semantics of parametric timed automata

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a convex polyhedron over \(X\) and \(P\) with a special form, called parametric zone

- **Symbolic run**: alternating sequence of symbolic states and actions

- **Example**

- **Possible symbolic run for this PTA**
Symbolic semantics of parametric timed automata

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a convex polyhedron over \(X\) and \(P\) with a special form, called parametric zone

- **Symbolic run**: alternating sequence of symbolic states and actions

**Example**

Possible symbolic run for this PTA
Licensing
Source of the graphics used

Title: 1960 Citroen DS19
Author: Joc281
License: CC by-sa 3.0

Title: A Cartoon Businessman Reading A Text Message
Author: Vector Toons
License: CC by-sa 4.0

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/