Quantitative timed games

Patricia Bouyer

LSV – CNRS & ENS Cachan – France

Based on joint works with Thomas Brihaye, Ed Brinksma, Véronique Bruyère, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, Jean-François Raskin, Jiří Srba, and Jacob Illum Rasmussen
A (difficult) choice

When sending the title, I didn’t know if I would speak about:

1. timed automata with costs
2. timed automata with probabilities
A (difficult) choice

When sending the title, I didn’t know if I would speak about:

- timed automata with costs,
A (difficult) choice

When sending the title, I didn’t know if I would speak about:

- timed automata with costs, or
- timed automata with probabilities
A (difficult) choice

When sending the title, I didn’t know if I would speak about:

- timed automata with costs, or
- timed automata with probabilities
A (difficult) choice

When sending the title, I didn’t know if I would speak about:

- timed automata with costs, or
- timed automata with probabilities

leadsto talk of Vojtěch Forejt in the next session
Outline

1. Introduction

2. Weighted/priced timed automata

3. (Optimal) timed games

4. “Safe” timed games

5. Conclusion
A starting example
Natural questions

Can I reach Pontivy from Oxford?
Natural questions

- Can I reach Pontivy from Oxford?
- What is the **minimal time** to reach Pontivy from Oxford?
Natural questions

- Can I reach Pontivy from Oxford?
- What is the **minimal time** to reach Pontivy from Oxford?
- What is the **minimal fuel consumption** to reach Pontivy from Oxford?
Natural questions

- Can I reach Pontivy from Oxford?
- What is the **minimal time** to reach Pontivy from Oxford?
- What is the **minimal fuel consumption** to reach Pontivy from Oxford?
- What if there is an **unexpected event**?
Natural questions

- Can I reach Pontivy from Oxford?
- What is the **minimal time** to reach Pontivy from Oxford?
- What is the **minimal fuel consumption** to reach Pontivy from Oxford?
- What if there is an **unexpected event**?
- Can I use my computer all the way?
A first model of the system
Can I reach Pontivy from Oxford?

This is a reachability question in a finite graph: Yes, I can!
A second model of the system
How long will that take?

It is a reachability (and optimization) question in a **timed automaton**: at least $350 \text{mn} = 5\text{h}50\text{mn}$!
An example of a timed automaton

[Diagram of a timed automaton with states and transitions]

- **Safe**: Transition to **Alarm** with condition $x := 0$
  
- **Alarm**: Transition to **Repairing** with condition $y := 0$
  - Transition to **Safe** with condition $15 \leq x \leq 16$
  - Transition to **Failsafe** with condition $15 \leq x \leq 16$

- **Repairing**: Transition to **Safe** with condition $2 \leq y \land x \leq 56$
  
- **Failsafe**: Transition to **Safe** with condition $22 \leq y \leq 25$

- **Done**: Transition to **Safe** with condition $22 \leq y \leq 25$

- **Repair**: Transition to **Repairing** with condition $x \leq 15$
  
- **Delayed**: Transition to **Failsafe** with condition $y := 0$

- **Problem**: Transition to **Alarm** with condition $x := 0$

---

**Introduction**
An example of a timed automaton

\[
\begin{align*}
\text{safe} & \xrightarrow{\text{problem, } x := 0} \text{alarm} \\
\text{alarm} & \xrightarrow{\text{repair, } x \leq 15} \text{repair} \\
\text{repair} & \xrightarrow{\text{repair}} \text{failsafe} \\
\text{failsafe} & \xrightarrow{\text{done, } 22 \leq y \leq 25} \text{safe}
\end{align*}
\]
An example of a timed automaton

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td>23</td>
<td>safe</td>
</tr>
<tr>
<td>alarm</td>
<td>repair, (x \leq 15) (y = 0)</td>
<td>repairing</td>
</tr>
<tr>
<td></td>
<td>repair, (2 \leq y \wedge x \leq 56) (y = 0)</td>
<td>failsafe</td>
</tr>
<tr>
<td></td>
<td>delayed, (y = 0)</td>
<td>repairing</td>
</tr>
<tr>
<td>repair</td>
<td>done, (22 \leq y \leq 25)</td>
<td>safe</td>
</tr>
<tr>
<td>problem</td>
<td>problem, (x := 0)</td>
<td>alarm</td>
</tr>
<tr>
<td></td>
<td>alarm, (15 \leq x \leq 16) (y = 0)</td>
<td>failsafe</td>
</tr>
<tr>
<td></td>
<td>failsafe</td>
<td>repairing</td>
</tr>
</tbody>
</table>

The table shows the transitions between states with their corresponding actions and conditions.
An example of a timed automaton

\[ x := 0, x \leq 15 \]
\[ y := 0, 15 \leq x \leq 16 \]
\[ y := 0, 2 \leq y \land x \leq 56 \]
\[ y := 0, 22 \leq y \leq 25 \]

<table>
<thead>
<tr>
<th></th>
<th>done, 22 \leq y \leq 25</th>
<th>repair, \ x \leq 15</th>
<th>repair, \ 2 \leq y \land x \leq 56</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>23</td>
<td>23</td>
</tr>
</tbody>
</table>
An example of a timed automaton
An example of a timed automaton

The automaton consists of states: safe, alarm, repairing, and failsafe. The transitions between these states are as follows:

- From safe to alarm: \( x := 0 \), \( 0 \leq x \leq 15 \), \( y := 0 \), \( y = 0 \), \( 2 \leq y \land x \leq 56 \), \( 15 \leq x \leq 16 \), \( delayed \), \( y := 0 \).
- From alarm to safe: \( done \), \( 22 \leq y \leq 25 \).
- From safe to repairs: \( repair \), \( x := 0 \), \( 0 \leq x \leq 15 \), \( y := 0 \), \( 2 \leq y \land x \leq 56 \), \( 15 \leq x \leq 16 \), \( delayed \), \( y := 0 \).
- From repairs to failsafe: \( repair \), \( 2 \leq y \land x \leq 56 \), \( 15 \leq x \leq 16 \), \( delayed \), \( y := 0 \).

The table below shows the transitions and their parameters:

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>Parameter 1</th>
<th>Parameter 2</th>
<th>Parameter 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td>safe</td>
<td>23</td>
<td>safe</td>
<td>0</td>
</tr>
<tr>
<td>safe</td>
<td>alarm</td>
<td>23</td>
<td>alarm</td>
<td>15.6</td>
</tr>
<tr>
<td>alarm</td>
<td>alarm</td>
<td>23</td>
<td>alarm</td>
<td>38.6</td>
</tr>
<tr>
<td>alarm</td>
<td>failsafe</td>
<td>15.6</td>
<td>failsafe</td>
<td>0</td>
</tr>
</tbody>
</table>
An example of a timed automaton

\[ \begin{align*}
\text{safe} & \xrightarrow{23} \text{safe} \\
X & 0 \\
Y & 0 \\
\text{problem} & \xrightarrow{} \text{alarm} \\
x & 0 \\
y & 23 \\
\text{repair} & \xrightarrow{15.6} \text{alarm} \\
x & 15.6 \\
y & 38.6 \\
\text{delayed} & \xrightarrow{} \text{failsafe} \\
x & 15.6 \\
y & 0 \\
\text{done} & \xrightarrow{} \text{safe} \\
x & 23 \\
y & 0 \\
\text{failsafe} & \xrightarrow{2.3} \text{failsafe} \\
x & 15.6 \\
y & 0 \\
\cdots & \xrightarrow{} \text{failsafe} \\
x & 15.6 \\
y & 2.3
\end{align*} \]
An example of a timed automaton

safe → problem, \( x:=0 \) → alarm

- repair, \( x \leq 15 \) → repairing
- delayed, \( y:=0 \) → failsafe
- done, \( 22 \leq y \leq 25 \)

<table>
<thead>
<tr>
<th>Value</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>failsafe</td>
<td>2.3</td>
</tr>
<tr>
<td>…</td>
<td>15.6</td>
</tr>
<tr>
<td>…</td>
<td>0</td>
</tr>
</tbody>
</table>
An example of a timed automaton

**Diagram:**

- **States:**
  - Safe
  - Alarm
  - Repairing
  - Failsafe

- **Transitions:**
  - **Safe to Problem:** $x := 0$
  - **Problem to Alarm:** $y := 0$
  - **Alarm to Repairing:** $15 \leq x \leq 16$
  - **Repairing to Safe:** $22 \leq y \leq 25$
  - **Safe to Failsafe:** $23$
  - **Failsafe to Repairing:** $2.3$

**Table:**

<table>
<thead>
<tr>
<th>State</th>
<th>X</th>
<th>Y</th>
<th>Action</th>
<th>X</th>
<th>Y</th>
<th>Action</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe</td>
<td>0</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>15.6</td>
<td>23</td>
<td>15.6</td>
<td>0</td>
</tr>
<tr>
<td>Problem</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>15.6</td>
<td>38.6</td>
<td>23</td>
<td>38.6</td>
<td>0</td>
</tr>
<tr>
<td>Failsafe</td>
<td>15.6</td>
<td>0</td>
<td>2.3</td>
<td>17.9</td>
<td>17.9</td>
<td>0</td>
<td>17.9</td>
<td>40</td>
</tr>
<tr>
<td>Repairing</td>
<td>0</td>
<td>2.3</td>
<td>0</td>
<td>0</td>
<td>22.1</td>
<td>0</td>
<td>22.1</td>
<td>40</td>
</tr>
</tbody>
</table>
An example of a timed automaton

- **safe**
  - \( x := 0 \)
  - \( y := 0 \)
  - \( 15 \leq x \leq 16 \)
  - \( 2 \leq y \land x \leq 56 \)

- **alarm**
  - \( x := 0 \)
  - \( y := 0 \)
  - \( 15 \leq x \leq 16 \)
  - \( 2 \leq y \lor x \leq 56 \)

- **repairing**
  - \( x := 0 \)
  - \( 15.6 \leq y \leq 25 \)

- **failsafe**
  - \( x := 0 \)
  - \( y := 0 \)
  - \( 22 \leq y \leq 25 \)

### Transition Table

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>New State</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td>Problem</td>
<td>alarm</td>
<td>safe</td>
</tr>
<tr>
<td>alarm</td>
<td>Repair</td>
<td>repairing</td>
<td>alarm</td>
</tr>
<tr>
<td>alarm</td>
<td>Delayed</td>
<td>failsafe</td>
<td></td>
</tr>
<tr>
<td>failsafe</td>
<td>Repair</td>
<td>repairing</td>
<td></td>
</tr>
<tr>
<td>failsafe</td>
<td>Done</td>
<td>safe</td>
<td></td>
</tr>
</tbody>
</table>

- **Example Trace**
  - **safe** \( x := 0 \) \( y := 0 \)
  - **alarm** \( x := 0 \) \( y := 0 \)
  - **repairing** \( x := 0 \) \( y := 0 \)
  - **failsafe** \( x := 0 \) \( y := 0 \)

### Parameter Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>17.9</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>40</td>
</tr>
<tr>
<td>...</td>
<td>22.1</td>
</tr>
<tr>
<td>...</td>
<td>22.1</td>
</tr>
</tbody>
</table>

- **Safe State**
  - \( 15.6 \leq y \leq 25 \)
  - \( 22 \leq y \leq 25 \)

- **Alarm State**
  - \( 15.6 \leq y \leq 25 \)

- **Repairing State**
  - \( 15.6 \leq y \leq 25 \)

- **Failsafe State**
  - \( 15.6 \leq y \leq 25 \)
Timed automata

**Theorem [AD90]**

The reachability problem is decidable (and PSPACE-complete) for timed automata.
Timed automata

**Theorem [AD90]**

The reachability problem is decidable (and **PSPACE-complete**) for timed automata.

[AD90] Alur, Dill. Automata for modeling real-time systems *(ICALP’90).*
The region abstraction
The region abstraction

- “compatibility” between regions and constraints
“compatibility” between regions and constraints
“compatibility” between regions and time elapsing
The region abstraction

- “compatibility” between regions and constraints
- “compatibility” between regions and time elapsing
The region abstraction

- “compatibility” between regions and constraints
- “compatibility” between regions and time elapsing

\[ \leadsto \text{an equivalence of finite index} \]

\[ \leadsto \text{a time-abstract bisimulation} \]
The region abstraction

- - - - - - - - time elapsing

- - - - - - - - reset to 0

- - - - - - - -
Time-optimal reachability

**Theorem [CY92]**

The time-optimal reachability problem is decidable (and PSPACE-complete) for timed automata.

Outline

1. Introduction

2. Weighted/priced timed automata

3. (Optimal) timed games

4. “Safe” timed games

5. Conclusion
A third model of the system

Weighted/priced timed automata
How much fuel will I use?

It is a **quantitative** (optimization) problem in a **priced/weighted timed automaton**: at least 68 anti-planet units!
HSCC’01: weighted/priced timed automata

\[ \ell_0 \xrightarrow{+5} \ell_1 \xrightarrow{(y=0)} \ell_2 \xrightarrow{+10} \ell_3 \xrightarrow{+1} \text{smiley} \]

\[ x \leq 2, c, y := 0 \]


HSCC’01: weighted/priced timed automata

\[ \ell_0 \rightarrow \ell_0, x \leq 2, c, y := 0 \]
\[ \ell_1, (y = 0), u \rightarrow \ell_2, u \rightarrow \ell_3, x = 2, c \rightarrow \ell_2 \rightarrow \ell_3, c \rightarrow \text{smiley} \]

Cost:
\[ 6.5 + 0 + 0 + 0 + 7 = 14.2 \]

HSCC’01: weighted/priced timed automata

\[
\begin{align*}
\ell_0 & \quad x \leq 2, c, y := 0 \\
\ell_1 & \quad (y = 0) \\
\ell_2 & \quad x = 2, c +1 \\
\ell_3 & \quad x = 2, c +7 \\
\end{align*}
\]

\[
\begin{array}{cccc}
\ell_0 & \xrightarrow{1.3} & \ell_0 & c \\
\ell_1 & \xrightarrow{1.3} & \ell_1 & u \\
\ell_3 & \xrightarrow{0.7} & \ell_3 & c \\
\end{array}
\]

\[
\begin{align*}
x & \quad 0 & \quad 1.3 & \quad 1.3 & \quad 1.3 & \quad 2 & \quad 0.7 \\
y & \quad 0 & \quad 1.3 & \quad 0 & \quad 0 & \quad 0.7 \\
\end{align*}
\]

cost :

\[
\begin{align*}
\ell_0 & \quad +5 \\
\ell_1 & \quad (y = 0) \\
\ell_2 & \quad +10 \\
\ell_3 & \quad +1 \\
\end{align*}
\]

HSCC’01: weighted/priced timed automata

\[ \ell_0 \xrightarrow{+5} \ell_0 \rightarrow \ell_1 \rightarrow \ell_2 \rightarrow \ell_3 \rightarrow \text{smiley} \]

\[ \ell_0 \xrightarrow{1.3} \ell_0 \xrightarrow{c} \ell_1 \xrightarrow{u} \ell_3 \xrightarrow{0.7} \ell_3 \xrightarrow{c} \text{smiley} \]

\[ \begin{array}{cccccccc}
  x & 0 & 1.3 & c & 1.3 & u & 1.3 & 2 & c \\
  y & 0 & 1.3 & 0 & 0 & 0.7 & 2 & 0.7 & \text{smiley} \\
\end{array} \]

\[ \text{cost : } 6.5 \]

HSCC’01: weighted/priced timed automata

\[ \ell_0 \xrightarrow{5} \ell_1 \quad \text{(y=0)} \]

\[ \ell_1 \xrightarrow{u} \ell_2 \xrightarrow{10} \]

\[ \ell_2 \xrightarrow{x=2,c} \ell_3 \xrightarrow{1} \]

\[ \ell_3 \xrightarrow{x=2,c} \]

\[ \ell_0 \xrightarrow{1.3} \ell_0 \xrightarrow{c} \ell_1 \xrightarrow{u} \ell_3 \xrightarrow{0.7} \ell_3 \xrightarrow{c} \]

\[ x \begin{array}{c} 0 \end{array} \quad \begin{array}{c} 1.3 \end{array} \quad \begin{array}{c} 1.3 \end{array} \quad \begin{array}{c} 1.3 \end{array} \quad \begin{array}{c} 2 \end{array} \quad \begin{array}{c} \text{cost:} \end{array} \quad \begin{array}{c} 6.5 \end{array} \quad \begin{array}{c} + \end{array} \quad \begin{array}{c} 0 \end{array} \]

HSCC’01: weighted/priced timed automata

\[
\ell_0 \\ (+5)
\\xrightarrow{x \leq 2, c, y:=0}
\\ell_1 \\ (y=0)
\\xrightarrow{u}
\\ell_2 \\ (+10)
\\xrightarrow{x=2, c}
\\ell_3 \\ (+1)
\\xrightarrow{x=2, c}
\\\text{smiley}
\]

\[
\ell_0 \xrightarrow{1.3} \ell_0 \xrightarrow{c} \ell_1 \xrightarrow{u} \ell_3 \xrightarrow{0.7} \ell_3 \xrightarrow{c}
\]

\[
\begin{array}{c|c|c|c|c|c}
 & \ell_0 & \ell_0 & \ell_1 & \ell_3 & \ell_3 \\
 x & 0 & 1.3 & 1.3 & 1.3 & 2 \\
y & 0 & 1.3 & 0 & 0 & 0.7 \\
\hline
\text{cost} & 6.5 & + & 0 & + & 0 \\
\end{array}
\]

HSCC’01: weighted/priced timed automata

\[
\ell_0 \xrightarrow{5} \ell_1 \\
\xrightarrow{x \leq 2, c, y := 0} \\
\ell_1 \xrightarrow{u} \ell_2 \\
\xrightarrow{u} \ell_3 \\
\xrightarrow{x = 2, c} \smiley \\
\ell_3 \xrightarrow{c} \ell_0 \\
\ell_0 \xrightarrow{1.3} \ell_0 \\
\xrightarrow{c} \ell_1 \\
\xrightarrow{u} \ell_3 \\
\xrightarrow{0.7} \ell_3 \\
\xrightarrow{c} \smiley
\]

\[
\begin{array}{c|ccc|ccc|ccc}
 & \ell_0 & \ell_0 & \ell_1 & \ell_3 & \ell_3 & \ell_3 & \ell_0 \\
\hline
x & 0 & 1.3 & 1.3 & 1.3 & 0 & 0 & 0.7 \\
y & 0 & 1.3 & 0 & 0 & 0.7
\end{array}
\]

cost: \[6.5 + 0 + 0 + 0 + 0.7\]


HSCC’01: weighted/priced timed automata

\[ \ell_0 \xrightarrow{+5} \ell_1 \xrightarrow{\ell_2} \ell_3 \xrightarrow{+1} \text{smiley} \]

\[ \ell_0 \xrightarrow{1.3} \ell_1 \xrightarrow{u} \ell_3 \xrightarrow{0.7} \text{smiley} \]

\[
\begin{array}{c|ccc|ccc|ccc|}
   & \ell_0 & \ell_0 & \ell_1 & \ell_3 & \ell_3 & \ell_3 & \text{smiley} \\
   x & 0 & 1.3 & 1.3 & 1.3 & 2 & c \\
   y & 0 & 1.3 & 0 & 0 & 0.7 & \end{array}
\]

\[
\text{cost: } 6.5 + 0 + 0 + 0.7 + 7
\]
HSCC’01: weighted/priced timed automata

\[
\begin{align*}
\ell_0 & \xrightarrow{1.3} \ell_0 \\
& \xrightarrow{c} \ell_1 \\
& \xrightarrow{u} \ell_3 \\
& \xrightarrow{0.7} \ell_3 \\
& \xrightarrow{c} \text{smiley}
\end{align*}
\]

\[
\begin{align*}
x & = 0, c, y := 0 \\
(\ell_0) & \xrightarrow{5} \ell_1 \\
& \xrightarrow{u} \ell_2 \\
& \xrightarrow{x = 2, c} \text{smiley}
\end{align*}
\]

\[
\begin{align*}
x & \leq 2, c, y := 0 \\
(\ell_1) & \xrightarrow{u} \ell_3 \\
& \xrightarrow{x = 2, c} \text{smiley}
\end{align*}
\]

\[
\begin{align*}
\ell_2 & \xrightarrow{10} \text{smiley} \\
& \xrightarrow{x = 2, c} \text{smiley}
\end{align*}
\]

\[
\begin{align*}
\ell_3 & \xrightarrow{1} \text{smiley} \\
& \xrightarrow{x = 2, c} \text{smiley}
\end{align*}
\]

\[
\begin{align*}
x & = 2, c \\
y & = 0 \\
\text{cost} & = 6.5 + 0 + 0 + 0.7 + 7 = 14.2
\end{align*}
\]


HSCC’01: weighted/priced timed automata

Question: what is the optimal cost for reaching 😊?

HSCC’01: weighted/priced timed automata

Question: what is the optimal cost for reaching 😊?

$$5t + 10(2 - t) + 1$$


Question: what is the optimal cost for reaching 😊?

\[ 5t + 10(2 - t) + 1, \quad 5t + (2 - t) + 7 \]
HSCC’01: weighted/priced timed automata

Question: what is the optimal cost for reaching 😊?

\[
\min \left( 5t + 10(2 - t) + 1, \ 5t + (2 - t) + 7 \right)
\]

HSCC’01: weighted/priced timed automata

Question: what is the optimal cost for reaching 😊?

\[
\inf_{0 \leq t \leq 2} \min (5t + 10(2 - t) + 1, 5t + (2 - t) + 7) = 9
\]

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata (*HSCC’01*).

HSCC’01: weighted/priced timed automata

![Diagram of a weighted/priced timed automaton]

**Question:** what is the optimal cost for reaching 😊?

\[
\inf_{0 \leq t \leq 2} \min \left( 5t + 10(2 - t) + 1, \ 5t + (2 - t) + 7 \right) = 9
\]

→ **strategy:** leave immediately \( \ell_0 \), go to \( \ell_3 \), and wait there 2 t.u.


Optimal reachability

The idea “go through corners” extends in the general case.

**Theorem [ALP01,BFH+01,BBBR07]**

Optimal reachability is decidable (and *PSPACE-complete*) in timed automata.
The region abstraction is not fine enough

[Diagram showing the progression of states with arrows indicating the elapse of time and reset to 0]
The corner-point abstraction

We can somehow discretize the behaviours...
The corner-point abstraction

We can somehow discretize the behaviours...
From timed to discrete behaviours

Optimal reachability as a linear programming problem
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[ t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow \ldots \]
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[ t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow \ldots \]

\[ t_i \leq t_{i+1} \]
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[ t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow \ldots \]

\[ t_i \leq t_{i+1} \]
\[ t_2 \leq 2 \]
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[ \begin{align*}
& t_1 \quad y := 0 \\
& t_2 \quad x \leq 2 \\
& t_3 \quad y \geq 5 \\
& t_4 \quad t_i \leq t_{i+1} \\
& t_5 \quad t_2 \leq 2 \\
& \ldots \quad t_4 - t_1 \geq 5
\end{align*} \]
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[
\begin{align*}
& t_1 & t_2 & t_3 & t_4 & t_5 & \cdots \\
& y := 0 & x \leq 2 & y \geq 5 & & & \\
& & t_i \leq t_{i+1} & t_2 \leq 2 & t_4 - t_1 \geq 5 & 
\end{align*}
\]

Lemma

Let \( Z \) be a bounded zone and \( f \) be a function

\[
f : (t_1, \ldots, t_n) \mapsto \sum_{i=1}^{n} c_i t_i + c
\]

well-defined on \( \overline{Z} \). Then \( \inf_Z f \) is obtained on the border of \( \overline{Z} \) with integer coordinates.
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[
\begin{align*}
  t_1 & \quad y := 0 \\
  t_2 & \quad x \leq 2 \\
  t_3 & \quad t_4 \quad y \geq 5 \\
  t_5 & \quad \cdots
\end{align*}
\]

\[
\begin{aligned}
  t_i & \leq t_{i+1} \\
  t_2 & \leq 2 \\
  t_4 - t_1 & \geq 5
\end{aligned}
\]

Lemma

Let \( Z \) be a bounded zone and \( f \) be a function

\[
f : (t_1, \ldots, t_n) \mapsto \sum_{i=1}^{n} c_i t_i + c
\]

well-defined on \( \overline{Z} \). Then \( \inf_Z f \) is obtained on the border of \( \overline{Z} \) with integer coordinates.

\( \Rightarrow \) for every finite path \( \pi \) in \( A \), there exists a path \( \Pi \) in \( A_{cp} \) such that

\[
\text{cost}(\Pi) \leq \text{cost}(\pi)
\]

[\( \Pi \) is a “corner-point projection” of \( \pi \)]
From discrete to timed behaviours

**Approximation of abstract paths:**

For any path $\Pi$ of $A_{cp}$,

\[ \| \Pi - \pi_{\epsilon} \|_\infty < \epsilon \]

For every $\eta > 0$, there exists $\epsilon > 0$ s.t.

\[ \| \Pi - \pi_{\epsilon} \|_\infty < \epsilon \Rightarrow |\text{cost}(\Pi) - \text{cost}(\pi_{\epsilon})| < \eta \]

\[ 23/52 \]
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $A_{cp}$, for any $\varepsilon > 0$, 

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon$$

For every $\eta > 0$, there exists $\varepsilon > 0$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon \Rightarrow |\text{cost}(\Pi) - \text{cost}(\pi_\varepsilon)| < \eta$$
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $\mathcal{A}_{cp}$, for any $\varepsilon > 0$, there exists a path $\pi_\varepsilon$ of $\mathcal{A}$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon$$
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $A_{cp}$, for any $\varepsilon > 0$, there exists a path $\pi_\varepsilon$ of $A$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon$$

For every $\eta > 0$, there exists $\varepsilon > 0$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon \Rightarrow |\text{cost}(\Pi) - \text{cost}(\pi_\varepsilon)| < \eta$$
Going further 1: mean-cost optimization

\[ \dot{C} = P \quad \dot{R} = G \]
\[ \dot{C} = p \quad \dot{R} = g \]
\[ x = D \]
\[ x = 0 \]
\[ z \geq S \]
\[ z := 0 \]

\[ \text{Low} \quad \text{High} \]

\[ \text{att?} \]
\[ \text{x:=0} \]
\[ \text{Op} \]

Low

\[ \dot{C} = p \]
\[ \dot{R} = g \]
\[ x = D \]
\[ x = 0 \]
\[ \text{att?} \]

High

\[ \dot{C} = P \]
\[ \dot{R} = G \]
\[ x \leq D \]
\[ x := 0 \]
\[ \text{att?} \]

Going further 1: mean-cost optimization

\[ \begin{align*}
\dot{C} &= p \\
\dot{R} &= g
\end{align*} \]

\[ x := 0 \quad \text{att?} \]

High \( \xrightarrow{x=D} \) Low

\[ \begin{align*}
(x \leq D) \\
\dot{C} &= P \\
\dot{R} &= G \\
x := 0
\end{align*} \]

\[ \begin{align*}
\dot{C} &= p \\
\dot{R} &= g \\
x = D \\
\text{att?} \\
x := 0
\end{align*} \]

Low \( \xrightarrow{x=D} \) High

\[ \begin{align*}
z &\geq S \\
z := 0 \quad \text{att!}
\end{align*} \]

Op

\[ \leadsto \text{compute optimal infinite schedules that minimize} \]

\[ \text{mean-cost}(\pi) = \limsup_{n \to +\infty} \frac{\text{cost}(\pi_n)}{\text{reward}(\pi_n)} \]

Going further 1: mean-cost optimization

\[ \dot{C} = P \quad \dot{R} = G \]

\[ \dot{C} = p \quad \dot{R} = g \]

\[ x := 0 \quad \text{att?} \]

\[ x = D \]

\[ (x \leq D) \]

\[ \text{Low} \]

\[ \text{High} \]

\[ \text{Op} \]

\[ z \geq S \quad z := 0 \quad \text{att!} \]

\[ \leadsto \text{compute optimal infinite schedules that minimize} \]

\[ \text{mean-cost}(\pi) = \lim \sup_{n \to +\infty} \frac{\text{cost}(\pi_n)}{\text{reward}(\pi_n)} \]

\[ \text{Schedule with ratio } \approx 1.455 \]

\[ \text{Schedule with ratio } \approx 1.478 \]

Going further 1: mean-cost optimization

\[
\begin{align*}
\dot{C} &= P \\
\dot{R} &= G \\
(x \leq D) &\quad \text{High} \\
\dot{x} &= 0 &\quad \text{att?} \\
\text{att?}, x := 0
\end{align*}
\]

\[
\begin{align*}
x &= D \\
\dot{C} &= p \\
\dot{R} &= g \\
\text{Low} \\
\dot{x} &= 0 \\
\text{att?}
\end{align*}
\]

\[
\begin{align*}
\text{Op} \\
z &\geq S &\quad \text{att!} \\
z &:= 0
\end{align*}
\]

\[x := 0 \quad \text{leadsto} \quad \text{compute optimal infinite schedules that minimize}\]

\[
\text{mean-cost}(\pi) = \lim \sup_{n \to +\infty} \frac{\text{cost}(\pi_n)}{\text{reward}(\pi_n)}
\]

**Theorem [BBL08]**

The mean-cost optimization problem is decidable (and PSPACE-complete) for priced timed automata.

\[\leadsto\] the corner-point abstraction can be used

Mean-cost optimization: from timed to discrete behaviours

- **Finite behaviours:** based on the following property

**Lemma**

Let $Z$ be a bounded zone and $f$ be a function

$$f : (t_1, ..., t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

well-defined on $\overline{Z}$. Then $\inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.
Mean-cost optimization: from timed to discrete behaviours

- **Finite behaviours:** based on the following property

**Lemma**

Let $Z$ be a bounded zone and $f$ be a function

$$f : (t_1, ..., t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

well-defined on $\overline{Z}$. Then $\inf_{\overline{Z}} f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

$\leadsto$ for every finite path $\pi$ in $A$, there exists a path $\Pi$ in $A_{cp}$ such that

$$\text{mean-cost}(\Pi) \leq \text{mean-cost}(\pi)$$
Mean-cost optimization: from timed to discrete behaviours

- **Finite behaviours**: based on the following property

**Lemma**

Let $Z$ be a bounded zone and $f$ be a function

$$f : (t_1, ..., t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

well-defined on $\overline{Z}$. Then $\inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

\(
\implies \text{for every finite path } \pi \text{ in } \mathcal{A}, \text{ there exists a path } \Pi \text{ in } \mathcal{A}_{cp} \text{ such that }
\)

$$\text{mean-cost}(\Pi) \leq \text{mean-cost}(\pi)$$

- **Infinite behaviours**: decompose each sufficiently long projection into cycles

The linear part will be negligible!
Mean-cost optimization: from timed to discrete behaviours

- **Finite behaviours**: based on the following property

**Lemma**

Let $Z$ be a bounded zone and $f$ be a function

$$f : (t_1, ..., t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

well-defined on $\overline{Z}$. Then $\inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

~ for every finite path $\pi$ in $\mathcal{A}$, there exists a path $\Pi$ in $\mathcal{A}_{cp}$ such that

$$\text{mean-cost}(\Pi) \leq \text{mean-cost}(\pi)$$

- **Infinite behaviours**: decompose each sufficiently long projection into cycles

The linear part will be negligible!

~ the optimal cycle of $\mathcal{A}_{cp}$ is better than any infinite path of $\mathcal{A}$!
Mean-cost optimization: from discrete to timed behaviours

For any path $\Pi$ of $A_{cp}$, for any $\varepsilon > 0$, there exists a path $\pi_\varepsilon$ of $A$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon$$

For every $\eta > 0$, there exists $\varepsilon > 0$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon \Rightarrow |\text{mean-cost}(\Pi) - \text{mean-cost}(\pi_\varepsilon)| < \eta$$
Going further 2: concavely-priced cost functions

A general abstract framework for quantitative timed systems

Theorem [JT08]

Optimal cost in concavely-priced timed automata is computable, if we restrict to quasi-concave price functions. For the following cost functions, the (decision) problem is even PSPACE-complete:

- optimal-time and optimal-cost reachability;
- optimal discrete discounted cost;
- optimal average-time and average-cost;
- optimal mean-cost.

a slight extension of the corner-point abstraction can be used
Going further 3: discounted-time cost optimization

Globally, \( z \leq 8 \)

\[
\begin{align*}
(x \leq 3) & \quad \text{High} \\
\quad \text{deg} & \quad +2 \\
\quad \text{att} & \quad z \geq 2, x, z := 0 \\
\quad \text{deg} & \quad (x \leq 3) \\
\quad \text{att} & \quad z \geq 2, z := 0 \\
\quad x := 3 & \quad \text{Low} \\
\end{align*}
\]

\[\text{deg} + 5 \implies \text{att} + 2 \]

\[x := 3 \]

\[z \geq 2, x, z := 0 \]

\[z \geq 2, z := 0 \]

\[x := 3 \]

\[z \geq 2, x, z := 0 \]

\[z \geq 2, z := 0 \]

Going further 3: discounted-time cost optimization

Globally, \((z \leq 8)\)

\[
\begin{align*}
\text{High} & \quad \text{deg} & \quad \text{Med} & \quad \text{deg} & \quad \text{Low} \\
\Rightarrow (x \leq 3) & \quad +2 & \quad (x \leq 3) & \quad +5 & \quad +9 \\
\text{Low} & \quad \text{att} & \quad \text{Med} & \quad \text{att} & \quad \text{High} \\
\text{High} & \quad z \geq 2, x, z := 0 & \quad \text{Med} & \quad z \geq 2, z := 0 & \quad \text{High} \\
\text{Med} & \quad z \geq 2, x, z := 0 & \quad \text{Low} & \quad z \geq 2, z := 0 & \quad \text{Med} \\
\end{align*}
\]

\(\leadsto\) compute optimal infinite schedules that minimize discounted cost over time

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata (INFINITY'08).
Going further 3: discounted-time cost optimization

Globally, \((z \leq 8)\)

\[
\begin{align*}
(x \leq 3) & \quad \text{High} & +2 \\
& \quad x = 3, x := 0 \\
& \quad \text{deg} \\
& \quad z \geq 2, x, z := 0 \\
\quad \leftarrow -2 \\
& \quad \text{att} \\
(x \leq 3) & \quad \text{Med} & +5 \\
& \quad x = 3 \\
& \quad \text{deg} \\
& \quad z \geq 2, z := 0 \\
\quad \leftarrow +1 \\
& \quad \text{att} \\
& \quad +9 \\
& \quad \text{Low}
\end{align*}
\]

\[\leadsto \text{compute optimal infinite schedules that minimize}\]

\[
\text{discounted-cost}_\lambda(\pi) = \sum_{n \geq 0} \lambda^{T_n} \int_{t=0}^{\tau_{n+1}} \lambda^t \text{cost}(\ell_n) \, dt + \lambda^{T_{n+1}} \text{cost}(\ell_n \xrightarrow{a_{n+1}} \ell_{n+1})
\]

if \(\pi = (\ell_0, v_0) \xrightarrow{\tau_1, a_1} (\ell_1, v_1) \xrightarrow{\tau_2, a_2} \cdots\) and \(T_n = \sum_{i \leq n} \tau_i\)

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata (INFINITY'08).
Going further 3: discounted-time cost optimization

Globally, \((z \leq 8)\)

\[\begin{align*}
(x \leq 3) & \quad \text{deg} & \quad (x \leq 3) & \quad \text{deg} \\
\text{High} & \quad +2 & \text{Med} & \quad +5 \\
\text{Low} & \quad +9
\end{align*}\]

\[\begin{align*}
x = 3, x := 0 \\
z \geq 2, x, z := 0
\]

\[\begin{align*}
x = 3 \\
z \geq 2, z := 0
\]

\[\Rightarrow\] compute optimal infinite schedules that minimize discounted cost over time

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata (INFINITY'08).
Going further 3: discounted-time cost optimization

Globally, \((z \leq 8)\)

\[
\begin{align*}
(x \leq 3) & \quad \text{High} \\
\text{deg} & \quad +2 \\
\text{att} & \quad -2 \\
z \geq 2, x, z := 0 & \quad \text{Low} +9
\end{align*}
\]

\[
\begin{align*}
(x \leq 3) & \quad \text{Med} \\
\text{deg} & \quad +5 \\
\text{att} & \quad +1 \\
z \geq 2, z := 0 & \quad \text{Low} +9
\end{align*}
\]

\(\leadsto\) compute optimal infinite schedules that minimize discounted cost over time

if \(\lambda = e^{-1}\), the discounted cost of that infinite schedule is \(\approx 2.16\)

Going further 3: discounted-time cost optimization

Globally, \((z \leq 8)\)

\[\begin{align*}
(x \leq 3) & \quad \text{deg} & (x \leq 3) & \quad \text{deg} \\
\text{High} & \quad +2 & \text{Med} & \quad +5 & \text{Low} & \quad +9 \\
\text{att} & \quad z \geq 2, x, z := 0 \\
\downarrow & \quad z \geq 2, x, z := 0
\end{align*}\]

\[\leadsto\] compute optimal infinite schedules that minimize discounted cost over time

**Theorem** [FL08]

The optimal discounted cost is computable in \text{EXPTIME} in priced timed automata.

\[\leadsto\] the corner-point abstraction can be used
Outline

1. Introduction

2. Weighted/priced timed automata

3. (Optimal) timed games

4. “Safe” timed games

5. Conclusion
What if an unexpected event happens?

(Optimal) timed games

Flight cancelled!

On strike!!!
What if an unexpected event happens?

(Optimal) timed games
What if an unexpected event happens?

![Diagram of a network of cities with flight cancellations and strikes modeled as timed games.]

 også modelled as timed games
A simple example of timed game

\( x \leq 2, c, y := 0 \)

\( (y = 0) \)

\( x = 2, c \)

\( x = 2, c \)
A simple example of timed game

\[
x \leq 2, c, y := 0
\]

\[
\ell_0 \xrightarrow{x \leq 2, c, y := 0} \ell_1
\]

\[
(\ell_1, (y = 0)) \xrightarrow{u} \ell_2 \xrightarrow{x = 2, c} +1
\]

\[
\ell_1 \xrightarrow{u} \ell_3 \xrightarrow{x = 2, c} +1
\]

\[
\ell_3 \xrightarrow{smiley}
\]
Another example

Another example of an optimal timed game is depicted in the diagram. The game consists of four states labeled $\ell_0$, $\ell_1$, $\ell_2$, and $\ell_3$. The transitions are labeled with conditions on $x$, such as $x \leq 2$, $x \geq 1$, $x < 1$, and $x \geq 2$. The diagram shows how the game transitions between states based on these conditions.
Decidability of timed games

**Theorem [AMPS98,HK99]**

Safety and reachability control in timed automata are decidable and \text{EXPTIME}-complete.

---


Decidability of timed games

**Theorem** [AMPS98,HK99]

Safety and reachability control in timed automata are decidable and EXPTIME-complete.

(the attractor is computable...)

---


Decidability of timed games

**Theorem [AMPS98,HK99]**

Safety and reachability control in timed automata are decidable and EXPTIME-complete.

(the attractor is computable...)

\[\leadsto\] classical regions are sufficient for solving such problems

---

[AMPS98] Asarin, Maler, Pnueli, Sifakis. Controller synthesis for timed automata *(SSC’98).*

[HK99] Henzinger, Kopke. Discrete-time control for rectangular hybrid automata *(Theoretical Computer Science).*
Decidability of timed games

Theorem [AMPS98, HK99]
Safety and reachability control in timed automata are decidable and EXPTIME-complete.

(the attractor is computable...)

\[ \leadsto \text{classical regions are sufficient for solving such problems} \]

Theorem [AM99, BHPR07, JT07]
Optimal-time reachability timed games are decidable and EXPTIME-complete.

Back to the first simple example

\[ \ell_0 + 5 \quad \xrightarrow{x \leq 2, c, y = 0} \quad \ell_1 \quad (y = 0) \quad \xrightarrow{u} \quad \ell_2 \quad +10 \quad \xrightarrow{x = 2, c} \quad \ell_3 \quad +1 \quad \xrightarrow{x = 2, c} \quad \text{smiley} \]

How to automatically compute such optimal costs?
How to synthesize optimal strategies (if one exists)?

Question: what is the optimal cost we can ensure from \( \ell_0 \)?

\[ \inf_{0 \leq t \leq 2} \max \left( 5t + 10(2 - t) + 1, 5t + (2 - t) + 7 \right) = 14 + \frac{1}{3} \]
Back to the first simple example

\[ \ell_0 + 5 \xrightarrow{x \leq 2, c, y = 0} \ell_1 + 10 \xrightarrow{(y=0)} \ell_2 + 10 \xrightarrow{x = 2, c} \ell_3 + 10 \xrightarrow{x = 2, c} \text{smiley} \]

**Question:** what is the optimal cost we can ensure from \( \ell_0 \)?

\[ \inf_{0 \leq t \leq 2} \max (5t + 10(2 - t) + 1, 5t + (2 - t) + 7) = 14 + 1/3 \]
Back to the first simple example

Question: what is the optimal cost we can ensure from $\ell_0$?

$$5t + 10(2 - t) + 1$$
Back to the first simple example

**Question:** what is the optimal cost we can ensure from $l_0$?

$$5t + 10(2 - t) + 1, \quad 5t + (2 - t) + 7$$
Back to the first simple example

Question: what is the optimal cost we can ensure from $\ell_0$?

$$\max ( 5t + 10(2 - t) + 1 , 5t + (2 - t) + 7 )$$
Back to the first simple example

Question: what is the optimal cost we can ensure from $\ell_0$?

$$\inf_{0 \leq t \leq 2} \max \left( 5t + 10(2 - t) + 1, \; 5t + (2 - t) + 7 \right) = 14 + \frac{1}{3}$$
Back to the first simple example

Question: what is the optimal cost we can ensure from $\ell_0$?

$$\inf_{0 \leq t \leq 2} \max \left( 5t + 10(2 - t) + 1 , 5t + (2 - t) + 7 \right) = 14 + \frac{1}{3}$$

→ strategy: wait in $\ell_0$, and when $t = \frac{4}{3}$, go to $\ell_1$
Back to the first simple example

Question: what is the optimal cost we can ensure from $\ell_0$?

$$\inf_{0 \leq t \leq 2} \max \left( 5t + 10(2 - t) + 1, 5t + (2 - t) + 7 \right) = 14 + \frac{1}{3}$$

→ strategy: wait in $\ell_0$, and when $t = \frac{4}{3}$, go to $\ell_1$

- How to automatically compute such optimal costs?
Back to the first simple example

**Question:** what is the optimal cost we can ensure from $\ell_0$?

$$\inf_{0 \leq t \leq 2} \max (5t + 10(2 - t) + 1, 5t + (2 - t) + 7) = 14 + \frac{1}{3}$$

→ **strategy:** wait in $\ell_0$, and when $t = \frac{4}{3}$, go to $\ell_1$

- How to automatically compute such optimal costs?
- How to synthesize optimal strategies (if one exists)?
Results

This topic has been fairly hot these last couple of years...

e.g. [LMM02, ABM04, BCFL04]

---

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata (TCS’02).
Results

This topic has been fairly hot these last couple of years... e.g. \[\text{[LMM02, ABM04, BCFL04]}\]

Theorem \[\text{[BBR05, BBM06]}\]

Optimal timed games are undecidable, as soon as automata have three clocks or more.

[BBR05] Brihaye, Bruyère, Raskin. On optimal timed strategies (FORMATS’05).
[BBM06] Bouyer, Brihaye, Markey. Improved undecidability results on weighted timed automata (Information Processing Letters).
Results

This topic has been fairly hot these last couple of years... e.g. [LMM02, ABM04, BCFL04]

**Theorem [BBR05, BBM06]**
Optimal timed games are **undecidable**, as soon as automata have three clocks or more.

**Theorem [BLMR06]**
Turn-based optimal timed games are **decidable** in 3EXPTIME when automata have a single clock. They are **P-hard**.

[BBR05] Brihaye, Bruyère, Raskin. On optimal timed strategies (*FORMATS’05*).
[BBM06] Bouyer, Brihaye, Markey. Improved undecidability results on weighted timed automata (*Information Processing Letters*).
[BLMR06] Bouyer, Larsen, Markey, Rasmussen. Almost-optimal strategies in one-clock priced timed automata (*FSTTCS’06*).
The positive side

**Theorem [BLMR06]**

Turn-based optimal timed games are **decidable** in 3EXPTIME when automata have a single clock. They are **P-hard**.

- Key: resetting the clock somehow resets the history...
The positive side

Theorem [BLMR06]

Turn-based optimal timed games are decidable in 3EXPTIME when automata have a single clock. They are P-hard.

- Key: resetting the clock somehow resets the history...
- Memoryless strategies can be non-optimal...

\begin{align*}
\ell_0 & \quad (x \leq 1) \quad +2 \quad x=1 \\
\ell_1 & \quad +1 \\
& \quad x<1 \\
& \quad x:=0 \\
& \quad x>0 \\
& \quad \text{smiley}
\end{align*}
The positive side

**Theorem [BLMR06]**

Turn-based optimal timed games are **decidable in 3EXPTIME** when automata have a single clock. They are **P-hard**.

- Key: resetting the clock somehow resets the history...
- Memoryless strategies can be non-optimal...

However, we can synthesize **memoryless almost-optimal** winning strategies.
The positive side

**Theorem [BLMR06]**

Turn-based optimal timed games are **decidable** in $3\text{EXPTIME}$ when automata have a single clock. They are $P$-hard.

- Key: resetting the clock somehow resets the history...
- Memoryless strategies can be non-optimal...

\[
\ell_0 \xrightarrow{+2} \ell_0, \quad \ell_0 \xrightarrow{x \leq 1} \ell_0, \quad \ell_0 \xrightarrow{x < 1} \ell_1, \quad \ell_0 \xrightarrow{x = 0} \ell_1, \quad \ell_0 \xrightarrow{x > 0} \ell_1
\]

- However, we can synthesize **memoryless almost-optimal** winning strategies.
- Rather involved proof (by unfolding and removing one by one locations) of correctness for a simple algorithm.
The negative side: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$. 
The negative side: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

**Add$^+$($x$)**

0 \[\xrightarrow{z:=0} \] 1

$y=1, y:=0$

$x=1, x:=0$

$z=1, z:=0$

The cost is increased by $x_0$

**Add$^-$($x$)**

1 \[\xrightarrow{z:=0} \] 0

$y=1, y:=0$

$x=1, x:=0$

$z=1, z:=0$

The cost is increased by $1-x_0$
The negative side: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$. 

![Diagram showing two paths with operations and cost calculations.](image)
The negative side: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

In the game, cost $= 2x_0 + (1 - y_0) + 2$
The negative side: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

\[
\begin{align*}
\text{In } \smiley, \text{ cost } &= 2x_0 + (1 - y_0) + 2 \\
\text{In } \smiley, \text{ cost } &= 2(1 - x_0) + y_0 + 1
\end{align*}
\]
The negative side: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

- In the first green terminal state, cost $= 2x_0 + (1 - y_0) + 2$
- In the second green terminal state, cost $= 2(1 - x_0) + y_0 + 1$
- If $y_0 < 2x_0$, player 2 chooses the first branch: cost $> 3$
The negative side: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

- In $\bigcirc$, cost = $2x_0 + (1 - y_0) + 2$
- In $\bigcirc$, cost = $2(1 - x_0) + y_0 + 1$

- If $y_0 < 2x_0$, player 2 chooses the first branch: cost $> 3$
- If $y_0 > 2x_0$, player 2 chooses the second branch: cost $> 3$
The negative side: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

- In $\odot$, cost $= 2x_0 + (1 - y_0) + 2$
- In $\odot$, cost $= 2(1 - x_0) + y_0 + 1$

- If $y_0 < 2x_0$, player 2 chooses the first branch: cost $> 3$
- If $y_0 > 2x_0$, player 2 chooses the second branch: cost $> 3$
- If $y_0 = 2x_0$, in both branches, cost $= 3$
The negative side: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

- In 0, cost = $2x_0 + (1 - y_0) + 2$
- In 1, cost = $2(1 - x_0) + y_0 + 1$

- if $y_0 < 2x_0$, player 2 chooses the first branch: cost $> 3$
  - if $y_0 > 2x_0$, player 2 chooses the second branch: cost $> 3$
  - if $y_0 = 2x_0$, in both branches, cost = 3

- Player 1 has a winning strategy with cost $\leq 3$ iff $y_0 = 2x_0$
The negative side: why is that hard?

Player 1 will simulate a two-counter machine:
- each instruction is encoded as a module;
- the values $c_1$ and $c_2$ of the counters are encoded by the values of two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{3^{c_2}}$$

when entering the corresponding module.
The negative side: why is that hard?

Player 1 will simulate a two-counter machine:
- each instruction is encoded as a module;
- the values $c_1$ and $c_2$ of the counters are encoded by the values of two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{3^{c_2}}$$

when entering the corresponding module.

The two-counter machine has an halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.
The negative side: why is that hard?

Player 1 will simulate a two-counter machine:
- each instruction is encoded as a module;
- the values $c_1$ and $c_2$ of the counters are encoded by the values of two clocks:
  \[ x = \frac{1}{2c_1} \quad \text{and} \quad y = \frac{1}{3c_2} \]
  when entering the corresponding module.

The two-counter machine has an halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.

Globally, \((x \leq 1, y \leq 1, u \leq 1)\)

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

Test \(y(x=2z)\)

\[
\begin{align*}
\left( x = \frac{1}{2c} \right) & \quad \left( x = \frac{1}{2c} \right) \\
y &= \frac{1}{2d} & y &= \frac{1}{2d} \\
z = \star & z = \alpha
\end{align*}
\]
Going further: other cost functions

An easy adaptation of the previous undecidability proof yields:

**Theorem**

Optimal mean-cost games are undecidable.

Going further: other cost functions

An easy adaptation of the previous undecidability proof yields:

**Theorem**

Optimal mean-cost games are undecidable.

**Theorem [JT08]**

Turn-based optimal average-time games are decidable and EXPTIME-complete.

~ talk of Ashutosh Trivedi in the next session
Marcin Jurdziński

Outline

1. Introduction

2. Weighted/priced timed automata

3. (Optimal) timed games

4. “Safe” timed games

5. Conclusion
A fourth model of the system
Can I work on my computer all the way?
Energy is not only consumed, but can be regained. 
\[\leadsto\] the aim is to \textit{continuously} satisfy some energy constraints.
An example

Globally \((x \leq 1)\)

\[
\ell_0 \xrightarrow{-3} \ell_1 \xrightarrow{+6} \ell_2 \xrightarrow{-6}
\]

\[x := 0 \quad \text{and} \quad x = 1\]
An example

Globally ($x \leq 1$)

Lower-bound problem: can we stay above 0?
“Safe” timed games

An example

Globally \((x \leq 1)\)

Lower-bound problem: can we stay above 0?
An example

Globally \((x \leq 1)\)

- Lower-bound problem: can we stay above 0?

- Lower-weak-upper-bound problem: can we "weakly" stay within bounds?
An example

Globally ($x \leq 1$)

- Lower-bound problem: can we stay above 0?
An example

Globally \( (x \leq 1) \)

\[
\ell_0 \rightarrow \ell_1 \rightarrow \ell_2 \quad \text{with} \quad x := 0 \rightarrow x = 1
\]

- Lower-bound problem: can we stay above 0?
An example

Globally ($x \leq 1$)

Lower-bound problem: can we stay above 0?
An example

Globally \((x \leq 1)\)

- Lower-bound problem
- Lower-upper-bound problem: can we stay within bounds?
An example

Globally \((x \leq 1)\)

Lower-bound problem

Lower-upper-bound problem: can we stay within bounds?
An example

Globally ($x \leq 1$)

Lower-bound problem

Lower-upper-bound problem: can we stay within bounds?

Lower-weak-upper-bound problem: can we "weakly" stay within bounds?
An example

Globally \((x \leq 1)\)

- Lower-bound problem
- Lower-upper-bound problem: can we stay within bounds?
An example

Globally \((x \leq 1)\)

-3 \(\ell_0\) \rightarrow +6 \(\ell_1\) \rightarrow -6 \(\ell_2\)

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

- Lower-bound problem
- Lower-upper-bound problem: can we stay within bounds?
An example

Globally \((x \leq 1)\)

- Lower-bound problem
- Lower-upper-bound problem: can we stay within bounds?
An example

Globally \((x \leq 1)\)

-3 \(\ell_0\) \(\rightarrow\) +6 \(\ell_1\) \(\rightarrow\) -6 \(\ell_2\)

\(x := 0\) \(\rightarrow\) \(x = 1\)

- Lower-bound problem
- Lower-upper-bound problem: can we stay within bounds?
An example

Globally \((x \leq 1)\)

- \(\ell_0\) to \(\ell_1\) with \(-3\)
- \(\ell_1\) to \(\ell_2\) with \(+6\)
- \(\ell_2\) with \(-6\)

\[ x := 0 \quad \text{x := 1} \]

Lower-bound problem

Lower-upper-bound problem: can we stay within bounds?

lost!
An example

Globally \((x \leq 1)\)

\[
\begin{align*}
-3 & \xrightarrow{x := 0} +6 & \xrightarrow{x = 1} -6 \\
\ell_0 & \quad \ell_1 & \quad \ell_2
\end{align*}
\]

- Lower-bound problem
- Lower-upper-bound problem: can we stay within bounds?
An example

Globally ($x \leq 1$)

-3 $\xrightarrow{x:=0} +6$ $\xrightarrow{x=1} -6$

- Lower-bound problem
- Lower-upper-bound problem: can we stay within bounds?
An example

Globally ($x \leq 1$)

$$
\ell_0 \xrightarrow{x:=0} \ell_1 \xrightarrow{x=1} \ell_2
$$

-3 $\xrightarrow{}$ +6 $\xrightarrow{}$ -6

\[ x = 0 \]
\[ x = 1 \]

---

- Lower-bound problem
- **Lower-upper-bound problem:** can we stay within bounds?

lost!
An example

Globally \((x \leq 1)\)

- Lower-bound problem
- Lower-upper-bound problem
- **Lower-weak-upper-bound problem:** can we “weakly” stay within bounds?
An example

Globally \((x \leq 1)\)

-3 \rightarrow +6 \rightarrow -6

\(\ell_0\) \rightarrow \(\ell_1\) \rightarrow \(\ell_2\)

\[x := 0 \quad \text{or} \quad x = 1\]

- Lower-bound problem \(\leadsto L\)
- Lower-upper-bound problem \(\leadsto L + U\)
- Lower-weak-upper-bound problem \(\leadsto L + W\)
## Results in the untimed case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
<td>∈ UP ∩ coUP, P-hard</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
<td>∈ NP ∩ coNP, P-hard</td>
</tr>
<tr>
<td>L+U</td>
<td>∈ PSPACE, NP-hard</td>
<td>∈ P</td>
<td>EXPTIME-c.</td>
</tr>
</tbody>
</table>
Results in the untimed case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
<td>∈ UP ∩ coUP P-hard</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
<td>∈ NP ∩ coNP P-hard</td>
</tr>
<tr>
<td>L+U</td>
<td>∈ PSPACE NP-hard</td>
<td>∈ P</td>
<td>EXPTIME-c.</td>
</tr>
</tbody>
</table>

- Bellman-Ford algorithm
Results in the untimed case

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L</strong></td>
<td>∈ (P)</td>
<td>∈ (P) ∈ (UP \cap \text{coUP}) P-hard</td>
</tr>
<tr>
<td><strong>L+W</strong></td>
<td>∈ (P)</td>
<td>∈ (P) ∈ (NP \cap \text{coNP}) P-hard</td>
</tr>
<tr>
<td><strong>L+U</strong></td>
<td>∈ PSPACE (\text{NP-hard})</td>
<td>∈ (P) EXPTIME-c.</td>
</tr>
</tbody>
</table>

- **PSPACE**: guess an infinite path in the graph augmented with the energy level.
- **NP-hardness**: encode SUBSET-SUM:
Results in the untimed case

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$\in P$</td>
<td>$\in UP \cap coUP$ P-hard</td>
</tr>
<tr>
<td>$L+W$</td>
<td>$\in P$</td>
<td>$\in NP \cap coNP$ P-hard</td>
</tr>
<tr>
<td>$L+U$</td>
<td>$\in PSPACE$</td>
<td>$\in P$</td>
</tr>
<tr>
<td></td>
<td>NP-hard</td>
<td></td>
</tr>
</tbody>
</table>

- **EXPTIME**: play the game in the graph augmented with the energy level.
- **EXPTIME-hardness**: encode COUNTDOWN-GAME [JLS07].
Results in the untimed case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$\in P$</td>
<td>$\in P$</td>
<td>$\in \text{UP} \cap \text{coUP}$ P-hard</td>
</tr>
<tr>
<td>$L+W$</td>
<td>$\in P$</td>
<td>$\in P$</td>
<td>$\in \text{NP} \cap \text{coNP}$ P-hard</td>
</tr>
<tr>
<td>$L+U$</td>
<td>$\in \text{PSPACE}$ NP-hard</td>
<td>$\in P$</td>
<td>EXPTIME-c.</td>
</tr>
</tbody>
</table>

- Mean-payoff games
Equivalence with mean-payoff games

<table>
<thead>
<tr>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean-payoff games: in a weighted game graph, does there exist a strategy s.t. the mean-cost of any play is nonnegative?</td>
</tr>
</tbody>
</table>
Equivalence with mean-payoff games

**Definition**

Mean-payoff games: in a weighted game graph, does there exists a strategy s.t. the mean-cost of any play is nonnegative?

**Lemma**

$L$-games and $L + W$-games are determined, and memoryless strategies are sufficient to win.
Equivalence with mean-payoff games

**Definition**

Mean-payoff games: in a weighted game graph, does there exists a strategy s.t. the mean-cost of any play is nonnegative?

**Lemma**

L-games and L+W-games are determined, and memoryless strategies are sufficient to win.

- from mean-payoff games to L-games or L+W-games: play in the same game graph $G$ with initial credit $-M \geq 0$ (where $M$ is the sum of negative costs in $G$).
Equivalence with mean-payoff games

**Definition**

Mean-payoff games: in a weighted game graph, does there exists a strategy s.t. the mean-cost of any play is nonnegative?

**Lemma**

L-games and L+W-games are determined, and memoryless strategies are sufficient to win.

- **from mean-payoff games to L-games or L+W-games:** play in the same game graph $G$ with initial credit $-M \geq 0$ (where $M$ is the sum of negative costs in $G$).

- **from L-games to mean-payoff games:** transform the game as follows:

  \[ \begin{array}{c}
  \text{initial state} \\
  \end{array} \xrightarrow{p} \begin{array}{c}
  \text{new state} \\
  \end{array} \sim \begin{array}{c}
  \text{initial state} \\
  \end{array} \xrightarrow{p} \begin{array}{c}
  \text{new state} \\
  \end{array} \xrightarrow{0} \begin{array}{c}
  \text{final state} \\
  \end{array} \]

  to initial state
Results for the single-clock case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
<td>?</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
<td>?</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

"Safe" timed games
### Results for the single-clock case

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- The corner-point abstraction can be used (wait in the most profitable location) ... but only if discrete costs are not used!!

```
+2 +2
/ \ / \\
|   |   |
\ (>3) ----> (><4)
/   \\
+1 +1
x:=0 x=1
```
Results for the single-clock case

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L</strong></td>
<td>$\in \mathbb{P}$</td>
<td>$\in \mathbb{P}$</td>
</tr>
<tr>
<td><strong>L+W</strong></td>
<td>$\in \mathbb{P}$</td>
<td>$\in \mathbb{P}$</td>
</tr>
<tr>
<td><strong>L+U</strong></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- The corner-point abstraction can be used (wait in the most profitable location) ... but only if discrete costs are not used!!
Results for the single-clock case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L</strong></td>
<td>∈ P</td>
<td>∈ P</td>
<td>?</td>
</tr>
<tr>
<td><strong>L+W</strong></td>
<td>∈ P</td>
<td>∈ P</td>
<td>?</td>
</tr>
<tr>
<td><strong>L+U</strong></td>
<td>?</td>
<td>?</td>
<td><strong>undecidable</strong></td>
</tr>
</tbody>
</table>

- the corner-point abstraction can be used (wait in the most profitable location) ... but only if discrete costs are not used!!

![Diagram showing the corner-point abstraction with states l0 and l1 and transitions x:=0 and x=1]
### Results for the single-clock case

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- **the corner-point abstraction can be used** (wait in the most profitable location) ... but only if discrete costs are not used!!
Results for the single-clock case

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- the corner-point abstraction can be used (wait in the most profitable location) ... but only if discrete costs are not used!!
### Results for the single-clock case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
<td>?</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
<td>?</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

- **the corner-point abstraction can be used** (wait in the most profitable location) ... but only if discrete costs are not used!!
### Results for the single-clock case

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>$\in P$</td>
<td>$\in P$</td>
</tr>
<tr>
<td>L$+$W</td>
<td>$\in P$</td>
<td>$\in P$</td>
</tr>
<tr>
<td>L$+$U</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- the corner-point abstraction can be used (wait in the most profitable location) ... but only if discrete costs are not used!!

![Diagram](image-url)
## Results for the single-clock case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
<td>?</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
<td>?</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

- the corner-point abstraction can be used (wait in the most profitable location) ... but only if discrete costs are not used!!
Results for the single-clock case

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- the corner-point abstraction can be used (wait in the most profitable location) ... but only if discrete costs are not used!!
Results for the single-clock case

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- the corner-point abstraction can be used (wait in the most profitable location) ... but only if discrete costs are not used!!
Results for the single-clock case

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- the corner-point abstraction can be used (wait in the most profitable location) ... but only if discrete costs are not used!!
Results for the single-clock case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
<td>?</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
<td>?</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

- the corner-point abstraction can be used (wait in the most profitable location) ... but only if discrete costs are not used!!
Results for the single-clock case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$\in P$</td>
<td>$\in P$</td>
<td>?</td>
</tr>
<tr>
<td>$L+W$</td>
<td>$\in P$</td>
<td>$\in P$</td>
<td>?</td>
</tr>
<tr>
<td>$L+U$</td>
<td>?</td>
<td>?</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

- the corner-point abstraction can be used (wait in the most profitable location) ... but only if discrete costs are not used!!
### Results for the single-clock case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$\in P$</td>
<td>$\in P$</td>
<td>?</td>
</tr>
<tr>
<td>$L+W$</td>
<td>$\in P$</td>
<td>$\in P$</td>
<td>?</td>
</tr>
<tr>
<td>$L+U$</td>
<td>?</td>
<td>?</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

- **the corner-point abstraction can be used** (wait in the most profitable location) ... but only if discrete costs are not used!!
## Results for the single-clock case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$\in \mathbb{P}$</td>
<td>$\in \mathbb{P}$</td>
<td>?</td>
</tr>
<tr>
<td>$L+W$</td>
<td>$\in \mathbb{P}$</td>
<td>$\in \mathbb{P}$</td>
<td>?</td>
</tr>
<tr>
<td>$L+U$</td>
<td>?</td>
<td>?</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

- **the corner-point abstraction can be used** *(wait in the most profitable location)* ... but only if discrete costs are not used!!

```
\ell_0 \quad +2 \quad \rightarrow \quad +2
\quad x:=0

\ell_1 \quad +4 \quad \rightarrow \quad -3
\quad x=1
```

```
x = 0
```

```
x = 1
```

```
0 1 2 3 4
```

```
0 1 2 3 4
```

```
0 1 2 3 4
```

```
0 1 2 3 4
```
### Results for the single-clock case

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- **the corner-point abstraction can be used** (wait in the most profitable location) ... but only if discrete costs are not used!!
**Results for the single-clock case**

<table>
<thead>
<tr>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+W</td>
<td>∈ P</td>
<td>∈ P</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- simulation of a two-counter machine
The single-clock $\mathbf{L+U}$-games are undecidable.
Single-clock $\mathbf{L+U}$-games

**Theorem**

The single-clock $\mathbf{L+U}$-games are undecidable.

We encode the behaviour of a two-counter machine:
- each instruction is encoded as a module;
- the values $c_1$ and $c_2$ of the counters are encoded by the energy level

$$e = 5 - \frac{1}{2^{c_1} \cdot 3^{c_2}}$$

when entering the corresponding module.
Single-clock $\mathbf{L}+\mathbf{U}$-games

**Theorem**

The single-clock $\mathbf{L}+\mathbf{U}$-games are undecidable.

We encode the behaviour of a two-counter machine:
- each instruction is encoded as a module;
- the values $c_1$ and $c_2$ of the counters are encoded by the energy level

$$e = 5 - \frac{1}{2^{c_1} \cdot 3^{c_2}}$$

when entering the corresponding module.

There is an infinite execution in the two-counter machine iff there is a strategy in the single-clock timed game under which the energy level remains between 0 and 5.
Single-clock $\textbf{L+U}$-games

**Theorem**

The single-clock $\textbf{L+U}$-games are undecidable.

We encode the behaviour of a two-counter machine:
- each instruction is encoded as a module;
- the values $c_1$ and $c_2$ of the counters are encoded by the energy level

$$e = 5 - \frac{1}{2^{c_1} \cdot 3^{c_2}}$$

when entering the corresponding module.

There is an infinite execution in the two-counter machine iff there is a strategy in the single-clock timed game under which the energy level remains between 0 and 5.

$\leadsto$ We present a generic construction for incrementing/decrementing the counters.
Generic module for incrementing/decrementing

"Safe" timed games
Generic module for incrementing/decrementing

-6
\[m\]
+5
\[x:=0\]
\[m_1\]

-6
\[m_2\]
+30
\[x:=0\]
\[m_3\]

+30
\[x:=0\]
\[m'\]

-5
\[x=1\]

\[\text{module ok}\]

\[\text{module ok}\]

 energy

\[5-e\]

\[\rightarrow\]

\[\rightarrow\]

\[0\]

\[1\]
Generic module for incrementing/decrementing

\[
\begin{align*}
\text{module } m &: x := 0 \\
\text{module } m_1 &: x := 0, \quad -6 \quad \text{increment} \quad c_1 \\
\text{module } m_2 &: x := 0, \quad +30 \quad \text{increment} \quad c_2 \\
\text{module } m_3 &: x := 0, \quad +30 \quad \text{increment} \quad c_2 \\
\text{module } m' &: x := 0, \quad -n \quad \text{decrement} \quad c_1 \\
\end{align*}
\]

Energy:

\[
\begin{align*}
x &= 0, \quad 5 - e \\
x &= 1, \quad 5 - e
\end{align*}
\]
“Safe” timed games

Generic module for incrementing/decrementing

\[
x := 0 \quad \rightarrow \quad -6 \quad \rightarrow \quad -6 \quad \rightarrow \quad +30 \quad \rightarrow \quad +30 \quad \rightarrow \quad -n \quad \rightarrow \quad x = 1
\]

\[
x := 0 \quad \rightarrow \quad +5 \quad \rightarrow \quad \text{module ok} \quad \rightarrow \quad -5 \quad \rightarrow \quad \text{module ok}
\]

energy

\[
\begin{align*}
5 - e & \quad \rightarrow \quad \text{energy} \\
0 & \quad \rightarrow \quad x \\
\frac{5 - e}{6} & \quad \rightarrow \quad 1
\end{align*}
\]
Generic module for incrementing/decrementing

```
x:=0
```

```
x=0
```

```
x:=0
```

```
x=0
```

```
x:=0
```

```
x=0
```

```
x:=0
```

```
x=0
```

```
x:=0
```

```
x=0
```

```
x:=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```

```
x=0
```
Generic module for incrementing/decrementing

\[ x := 0 \] 
\[ m \]
\[ -6 \]
\[ m_1 \]
\[ -6 \]
\[ m_2 \]
\[ +30 \]
\[ m_3 \]
\[ +30 \]
\[ m' \]
\[ -n \]
\[ x = 1 \]

\[ x := 0 \]
\[ +5 \]
\[ x = 1 \]

module **ok**

\[ x := 0 \]
\[ -5 \]
\[ x = 1 \]

module **ok**

**energy**

\[ \begin{align*}
5 - e & \quad x = 0 \\
5 - e & \quad x = 1
\end{align*} \]
Generic module for incrementing/decrementing

\[
x := 0 \quad m_{-6} \quad m_1 \quad m_2 \quad m_3 \quad m'_{-n} \quad x = 1
\]

\[
\begin{align*}
x := 0 \quad x := 0 \\
+5 \quad +5 \quad -5 \quad -5
\end{align*}
\]

\[
\begin{align*}
\text{module } & \text{ok} \\
\text{module } & \text{ok}
\end{align*}
\]

\[
\begin{align*}
\text{energy} & \quad 5 - e \quad 5 - \frac{ne}{6}
\end{align*}
\]
Generic module for incrementing/decrementing

\[ x := 0 \quad \rightarrow \quad m \quad \rightarrow \quad m_1 \quad \rightarrow \quad m_2 \quad \rightarrow \quad m_3 \quad \rightarrow \quad m' \quad \rightarrow \quad x = 1 \]

\[ \begin{align*}
  &x := 0 \\
  &\text{module } \textbf{ok} \\
  &x := 0 \\
  &\text{module } \textbf{ok} \\
\end{align*} \]

energy

\[ x = 0 \quad \rightarrow \quad \frac{5 - e}{6} \quad \rightarrow \quad 1 \]

- \( n = 3 \): increment \( c_1 \)
- \( n = 2 \): increment \( c_2 \)
- \( n = 12 \): decrement \( c_1 \)
- \( n = 18 \): decrement \( c_2 \)
### Results for the general case

<table>
<thead>
<tr>
<th></th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$L+W$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$L+U$</td>
<td>?</td>
<td>?</td>
<td>undecided</td>
</tr>
</tbody>
</table>

"Safe" timed games
Outline

1. Introduction

2. Weighted/priced timed automata

3. (Optimal) timed games

4. “Safe” timed games

5. Conclusion
Conclusion

- **Priced/weighted timed automata**, a model for representing quantitative constraints on timed systems:
  - useful in embedded systems verification
  - natural (optimization) questions have been posed...
    
    ... and not all of them have been answered yet!

Not mentioned here: all works on model-checking issues (extensions of CTL, LTL) models based on hybrid automata weighted o-minimal hybrid games [BBC07] weighted strong reset hybrid games [BBJLR07] leadsto talk of Michał Rutkowski in the next session

Various tools have been developed: Uppaal, Uppaal Cora, Uppaal Tiga

Current and further work: computation of approximate optimal values further investigation of safe games + several cost variables? discounted-time optimal games link between discounted-time games and mean-cost games?
Conclusion

- **Priced/weighted timed automata**, a model for representing quantitative constraints on timed systems:
  - useful in embedded systems verification
  - natural (optimization) questions have been posed...
    ... and not all of them have been answered yet!

- **Not mentioned here:**
  - all works on model-checking issues (extensions of CTL, LTL)
  - models based on hybrid automata
    - weighted o-minimal hybrid games
    - weighted strong reset hybrid games
  - current and further work:
    - computation of approximate optimal values
    - further investigation of safe games + several cost variables?
    - discounted-time optimal games
    - link between discounted-time games and mean-cost games?

- various tools have been developed:
  Uppaal, Uppaal Cora, Uppaal Tiga
Conclusion

- **Priced/weighted timed automata**, a model for representing quantitative constraints on timed systems:
  - useful in embedded systems verification
  - natural (optimization) questions have been posed...
    ... and not all of them have been answered yet!

- **Not mentioned here:**
  - all works on model-checking issues (extensions of CTL, LTL)
  - models based on hybrid automata
    - weighted o-minimal hybrid games
    - weighted strong reset hybrid games

  [BBC07] [BBJLR07]

  $\leadsto$ talk of Michał Rutkowski in the next session

- various tools have been developed:
  Uppaal, Uppaal Cora, Uppaal Tiga

- **Current and further work:**
  - computation of approximate optimal values
  - further investigation of safe games + several cost variables?
  - discounted-time optimal games
  - link between discounted-time games and mean-cost games?
  - ...