Real-time Model Checking
— Priced timed automata —

Nicolas MARKEY

Lav. Spécification & Vérification
CNRS & ENS Cachan – France

March 3, 2010
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?

Compute $D \times (C \times (A+B)) + (A+B) + (C \times D)$ using two processors:

$P_1$ (fast):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>2 picosec.</td>
</tr>
<tr>
<td>$\times$</td>
<td>3 picosec.</td>
</tr>
</tbody>
</table>

$P_2$ (slow):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>5 picosec.</td>
</tr>
<tr>
<td>$\times$</td>
<td>7 picosec.</td>
</tr>
</tbody>
</table>
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?

Compute $D \times (C \times (A+B)) + (A+B) + (C \times D)$ using two processors:

$P_1$ (fast):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>2 picosec.</td>
</tr>
<tr>
<td>$\times$</td>
<td>3 picosec.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
<td>10 W</td>
</tr>
<tr>
<td>in use</td>
<td>90 W</td>
</tr>
</tbody>
</table>

$P_2$ (slow):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>5 picosec.</td>
</tr>
<tr>
<td>$\times$</td>
<td>7 picosec.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
<td>20 W</td>
</tr>
<tr>
<td>in use</td>
<td>30 W</td>
</tr>
</tbody>
</table>
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?

Compute \( D \times (C \times (A+B)) + (A+B) + (C \times D) \) using two processors:

\[
\begin{align*}
P_1 \text{ (fast)}: & \\
\begin{array}{|c|c|}
\hline
& \text{time} \\
+ & 2 \text{ picosec.} \\
\times & 3 \text{ picosec.} \\
\hline
\end{array} \\
\begin{array}{|c|c|}
\hline
& \text{energy} \\
\text{idle} & 10 \text{ W} \\
\text{in use} & 90 \text{ W} \\
\hline
\end{array}
\end{align*}
\]

\[
\begin{align*}
P_2 \text{ (slow)}: & \\
\begin{array}{|c|c|}
\hline
& \text{time} \\
+ & 5 \text{ picosec.} \\
\times & 7 \text{ picosec.} \\
\hline
\end{array} \\
\begin{array}{|c|c|}
\hline
& \text{energy} \\
\text{idle} & 20 \text{ W} \\
\text{in use} & 30 \text{ W} \\
\hline
\end{array}
\end{align*}
\]
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?

Compute $D \times (C \times (A+B)) + (A+B) + (C \times D)$ using two processors:

$P_1$ (fast):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>2 picosec.</td>
</tr>
<tr>
<td>$\times$</td>
<td>3 picosec.</td>
</tr>
</tbody>
</table>

Energy:

<table>
<thead>
<tr>
<th></th>
<th>idle</th>
<th>in use</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>10 W</td>
<td>90 W</td>
</tr>
</tbody>
</table>

$P_2$ (slow):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>5 picosec.</td>
</tr>
<tr>
<td>$\times$</td>
<td>7 picosec.</td>
</tr>
</tbody>
</table>

Energy:

<table>
<thead>
<tr>
<th></th>
<th>idle</th>
<th>in use</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>20 W</td>
<td>30 W</td>
</tr>
</tbody>
</table>

![Diagram with nodes T1, T2, T3, T4, T5, T6 and A, B, C, D connections showing time and energy consumption.](image-url)
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?

Compute $D \times (C \times (A+B)) + (A+B) + (C \times D)$ using two processors:

<table>
<thead>
<tr>
<th></th>
<th>$P_1$ (fast):</th>
<th>$P_2$ (slow):</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>+ 2 picosec.</td>
<td>+ 5 picosec.</td>
</tr>
<tr>
<td></td>
<td>× 3 picosec.</td>
<td>× 7 picosec.</td>
</tr>
<tr>
<td>energy</td>
<td>idle 10 W</td>
<td>idle 20 W</td>
</tr>
<tr>
<td></td>
<td>in use 90 W</td>
<td>in use 30 W</td>
</tr>
</tbody>
</table>

$P_1$ (fast):

| T1  | 12 picoseconds | 1.32 nanojoules |
| T3  | 13 picoseconds | 1.37 nanojoules  |
| T4  | 19 picoseconds | 1.32 nanojoules  |

$P_2$ (slow):

| T2  | 19 picoseconds | 1.32 nanojoules |
| T5  | 19 picoseconds | 1.32 nanojoules |
| T6  | 19 picoseconds | 1.32 nanojoules |

Diagram:

- $A$, $B$, $C$, $D$
- $T_1$, $T_2$, $T_3$, $T_4$, $T_5$, $T_6$
Time is not always sufficient

- **hybrid automata:** timed automata augmented with variables whose derivative is not constant.

〜 examples: leaking gas burner, water-level monitor, ...

\[
\begin{align*}
  x &\leq 1 \\
  \dot{x} &= 1 \\
  \dot{y} &= 1 \\
  \dot{z} &= 1 \\
  x, y, z &= 0 \\
  x &\geq 30, x := 0 \\
  true &
\end{align*}
\]

**Theorem**

Reachability is undecidable (even for timed automata with one stopwatch).

Time is not always sufficient

- **hybrid automata**: timed automata augmented with variables whose derivative is not constant.

  ∼ examples: leaking gas burner, water-level monitor, ...

\[
\begin{align*}
\dot{x} & = 1 \\
x & \leq 1 \\
\dot{y} & = 1 \\
\dot{z} & = 1 \\
x, y, z & := 0 \\
x & \geq 30, x := 0
\end{align*}
\]

- **timed automata with observers**: similar to hybrid automata, but the behavior only depends on clock variables.

Outline of the talk

1. Introduction

2. Timed automata with observers

3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies

4. Resource-management problems

5. Conclusions and perspectives
Outline of the talk

1. Introduction

2. Timed automata with observers

3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies

4. Resource-management problems

5. Conclusions and perspectives
Timed automata with (linear) observers

Example

\[ x := 0 \]

\[ x := 1 \]

Timed automata with (linear) observers

Example

\[ x := 0 \]

\[ x = 1 \]

Timed automata with (linear) observers

Example

\[
\begin{align*}
-3 & \quad \text{to} \quad +6 \\
-6 & \quad \text{to} \quad +2 \\
& \quad \text{to} \quad -3
\end{align*}
\]

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

Timed automata with (linear) observers

Example

$x := 0$  $x = 1$

Timed automata with (linear) observers

Example

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

Timed automata with (linear) observers

Example

\[ \begin{align*}
-3 & \quad -1 \quad +6 \\
-6 \quad x:=0 & \quad +2 \\
\end{align*} \]

\[ x=1 \]

\[ \begin{align*}
-3 & \quad -3 \\
\frac{1}{6} & \quad \frac{1}{2} \quad +6 \\
\end{align*} \]

Timed automata with (linear) observers

Example

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

\[
\begin{align*}
-3 & \rightarrow +6 \\
-6 & \rightarrow +2
\end{align*}
\]

Timed automata with (linear) observers

Example

$x := 0$

$x = 1$

Timed automata with (linear) observers

Example

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

Timed automata with (linear) observers

Example

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

Outline of the talk

1. Introduction

2. Timed automata with observers

3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies

4. Resource-management problems

5. Conclusions and perspectives
Outline of the talk

1. Introduction

2. Timed automata with observers

3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies

4. Resource-management problems

5. Conclusions and perspectives
Optimal reachability

Example

\[ p = 5 \]
\[ \dot{p} = 5 \]
\[ y = 0 \]
\[ \dot{x} \leq 2 \]
\[ y := 0 \]

\[ p = 7 \]
\[ \dot{p} = 7 \]
\[ x \geq 3 \]
\[ p += 1 \]

\[ p = 5 \]
\[ \dot{p} = 5 \]
\[ x \geq 3 \]
\[ p += 4 \]
Optimal reachability

Example

Minimal cost for reaching 😊:
Optimal reachability

Example

\[ \dot{p} = 5 \quad y = 0 \quad \dot{p} = 7 \quad \dot{p} = 5 \]

\[ x \leq 2 \quad y := 0 \quad x \geq 3 \quad p := 1 \quad p := 4 \]

Minimal cost for reaching 😊:

\[ 5t + 7(3 - t) + 1 \]
Optimal reachability

Example

\[ \dot{p} = 5, \quad \dot{y} = 0 \]

\[ \dot{p} = 7, \quad \dot{p} = 5 \]

\[ x \geq 3 \]

\[ p + = 1 \]

\[ p + = 4 \]

\[ y := 0 \]

\[ x \leq 2 \]

Minimal cost for reaching 😊:

\[ 5t + 7(3 - t) + 1 \]

\[ 5t + 5(3 - t) + 4 \]
Optimal reachability

Example

Minimal cost for reaching 😊:

$$\min \left( 5t + 7(3 - t) + 1, 5t + 5(3 - t) + 4 \right)$$
Optimal reachability

Example

Minimal cost for reaching 😊:

$$\inf_{0 \leq t \leq 2} \min \left( \begin{array}{l} 5t + 7(3 - t) + 1 \\ 5t + 5(3 - t) + 4 \end{array} \right)$$
Optimal reachability

Example

\[ p = 5 \]
\[ y = 0 \]
\[ \dot{p} = 5 \]
\[ x \leq 2 \]
\[ y := 0 \]
\[ x \geq 3 \]
\[ x \geq 3 \]
\[ p += 1 \]
\[ p += 4 \]

Minimal cost for reaching 😊:

\[
\inf_{0 \leq t \leq 2} \min \left( \frac{5t + 7(3 - t) + 1}{5t + 5(3 - t) + 4} \right) = 18
\]
Optimal reachability

Example

\[ \dot{p} = 5 \]
\[ y = 0 \]
\[ \dot{p} = 7 \]
\[ x \geq 3 \]
\[ p += 1 \]
\[ \dot{p} = 5 \]
\[ x \geq 3 \]
\[ p += 4 \]

Minimal cost for reaching 😊:

\[ \inf_{0 \leq t \leq 2} \min \left( \frac{5t + 7(3 - t) + 1}{5t + 5(3 - t) + 4} \right) = 18 \]

The optimal schedule consists in

- waiting 2 time units in ☺;
- going through ☹.
Optimal reachability

**Theorem**

*Optimal reachability in priced timed automata is PSPACE-complete.*

Refs:  
Optimal reachability

**Theorem**

*Optimal reachability in priced timed automata is PSPACE-complete.*

**Proof.**

- The region abstraction is not fine enough:

```
<table>
<thead>
<tr>
<th>p=3</th>
<th>p=3</th>
<th>p=3</th>
<th>p=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>x:0</td>
<td>p+=2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Optimal reachability

Theorem

*Optimal reachability in priced timed automata is PSPACE-complete.*

Proof.

- The idea is: “take transitions close to integer dates”;

Refs:
Optimal reachability

**Theorem**

*Optimal reachability in priced timed automata is PSPACE-complete.*

**Proof.**

- The idea is: "take transitions close to integer dates";
- **Corner-point abstraction:** only consider *corners* of regions:

```
\dot{p} = 3
```

Optimal reachability

Theorem

**Optimal reachability in priced timed automata is PSPACE-complete.**

Proof.

- The idea is: “take transitions close to integer dates”;
- Corner-point abstraction: only consider corners of regions:

Optimal reachability

Theorem

Optimal reachability in priced timed automata is PSPACE-complete.

Proof.

- The idea is: “take transitions close to integer dates”;
- Corner-point abstraction: only consider corners of regions:

Outline of the talk

1. Introduction
2. Timed automata with observers
3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies
4. Resource-management problems
5. Conclusions and perspectives
Weighted temporal logic

Example

Decorate temporal modalities with constraints on cost:
Weighted temporal logic

Example

Decorate temporal modalities with constraints on cost:

\[ G(\text{failure} \Rightarrow F \leq 250) \]

\[ A \ G(\text{failure} \Rightarrow E F \text{ time} \leq 5 (\text{repair} \land A F \text{ cost} \leq 150 \text{ running})) \]
Weighted temporal logic

Example

Decorate temporal modalities with constraints on cost:

\begin{align*}
\text{failure} & \Rightarrow F_{\leq 250} \text{repaired} \\
A G & (\text{failure} \Rightarrow E F_{\leq 5} \text{time} \leq 150 \text{repair} \land A F_{\leq 150} \text{running})
\end{align*}
Weighted temporal logic

Example

Decorate temporal modalities with constraints on cost:

Example

\( G(failure \Rightarrow F_{\leq 250} \text{repaired}) \)
Weighted temporal logic

Example

Decorate temporal modalities with constraints on cost:

\[ \text{Example} \]

\[ \text{G}(\text{failure} \Rightarrow \text{F}_{\leq 250} \text{repaired}) \]

\[ \text{A G}(\text{failure} \Rightarrow \text{E F}_{\text{time} \leq 5}(\text{repair} \land \text{A F}_{\text{cost} \leq 150} \text{running})) \]
Undecidability results

Theorem

**WMTL model-checking is undecidable.**

Undecidability results

Theorem

WMTL model-checking is undecidable.

Proof.

- encoding of a two-counter machine;

Undecidability results

Theorem

WMTL model-checking is undecidable.

Proof.

- encoding of a two-counter machine;
- Holds even for one clock and one cost variable.

Undecidability results

**Theorem**

*WMTL model-checking is undecidable.*

*Proof.*
- encoding of a two-counter machine;
- Holds even for one clock and one cost variable.

**Theorem**

*WCTL model-checking is undecidable.*

Undecidability results

Theorem

\textit{WMTL model-checking is undecidable.}

\textit{Proof.}

\begin{itemize}
  \item encoding of a \textit{two-counter machine};
  \item Holds even for one clock and one cost variable.
\end{itemize}

Theorem

\textit{WCTL model-checking is undecidable.}

\textit{Proof.}

\begin{itemize}
  \item encoding of a \textit{two-counter machine};
\end{itemize}

Undecidability results

**Theorem**

*WMTL model-checking is undecidable.*

**Proof.**
- encoding of a two-counter machine;
- Holds even for one clock and one cost variable.

**Theorem**

*WCTL model-checking is undecidable.*

**Proof.**
- encoding of a two-counter machine;
- requires three clocks.

Decidable subcases

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{WCTL} model-checking is \textit{PSPACE-complete} on \textit{1-clock weighted timed automata}.</td>
</tr>
</tbody>
</table>

Decidable subcases

Theorem

WCTL model-checking is \textit{PSPACE-complete} on 1-clock weighted timed automata.

Proof.

- region-based algorithm;

Decidable subcases

Theorem

*WCTL model-checking is \textit{PSPACE-complete} on 1-clock weighted timed automata.*

\textit{Proof.}

- region-based algorithm;
- but region are not fine enough:

\[
\begin{align*}
\dot{p} &= 2 \\
\dot{x} &= 1 \\
\dot{p} &= 1 \\
x &= 1
\end{align*}
\]

Ref: \[1\] Bouyer, Larsen, M. \textit{Model-Checking One-Clock Priced Timed Automata} (2007).
Decidable subcases

**Theorem**

*WCTL model-checking is PSPACE-complete on 1-clock weighted timed automata.*

*Proof.*

- region-based algorithm;
- but region are not fine enough:

\[ p' = 2 \quad x = 1 \quad p' = 1 \]

Decidable subcases

**Theorem**

WCTL model-checking is **PSPACE-complete** on 1-clock weighted timed automata.

**Proof.**

- region-based algorithm;
- but region are not fine enough:

\[
\begin{align*}
\dot{p} &= 2 \\
\dot{p} &= 1 \\
\dot{p} &= 1
\end{align*}
\]

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

\[
E[\neg (E F_{\leq 1} \circ) U_{\geq 1} \bullet]
\]

Decidable subcases

Theorem

WCTL model-checking is \textit{PSPACE-complete} on 1-clock weighted timed automata.

Proof.

- region-based algorithm;
- but region are not fine enough:

\[
\begin{align*}
\dot{p} &= 2 \\
x &= 1 \\
\dot{p} &= 1
\end{align*}
\]

\[
\begin{align*}
\dot{p} &= 1 \\
x &= 1 \\
\dot{p} &= 1
\end{align*}
\]

\[
E \left[ \neg (E F_{\leq 1} \swarrow) U_{\geq 1} \searrow \right]
\]

Decidable subcases

Theorem

WCTL model-checking is \textit{PSPACE-complete} on 1-clock weighted timed automata.

Proof.

- region-based algorithm;
- but region are not fine enough:
- Refine regions: granularity $\frac{1}{M|\varphi|}$ is sufficient.
Outline of the talk

1. Introduction

2. Timed automata with observers

3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies

4. Resource-management problems

5. Conclusions and perspectives
Weighted timed games

Example
Timed games can also be extended with weights:

A strategy for a player indicates which (action or delay) transition to play; a strategy is winning if all its outcomes are.
Weighted timed games

Example

Timed games can also be extended with weights:

\[ \dot{p} = 2 \quad \dot{p} = 5 \quad \dot{p} = 0 \quad \dot{p} = 3 \]

\[ x \leq 1 \quad p = 4 \quad x = 1 \quad x \leq 1 \]

A strategy for a player indicates which (action or delay) transition to play; a strategy is winning if all its outcomes are.
Weighted timed games

Example

Timed games can also be extended with weights:

A strategy for a player indicates which (action or delay) transition to play;

A strategy is winning if all its outcomes are.
Optimal winning strategy

Example

\[ \dot{p} = 5 \]
\[ y = 0 \]
\[ \dot{p} = 6 \]
\[ x \geq 3 \]
\[ p += 1 \]
\[ \dot{p} = 3 \]
\[ x \geq 3 \]
\[ p += 9 \]

Minimal cost for reaching \( y = 0 \):

\[
\inf_{0 \leq t \leq 2} \max \left( 5t + 6(3 - t) + 1, 5t + 3(3 - t) + 9 \right) = \frac{56}{3}
\]

which is achieved with \( t = \frac{1}{3} \)

Corollary

Regions are not sufficient for solving priced timed games.
Optimal winning strategy

Example

Minimal cost for reaching 😊:

$$\inf_{0 \leq t \leq 2} \max (5t + 6(3 - t) + 1, 5t + 3(3 - t) + 9) = \frac{56}{3}$$

which is achieved with $t = \frac{1}{3}$.
Optimal winning strategy

Example

Minimal cost for reaching 😊:

\[5t + 6(3 - t) + 1\]
Optimal winning strategy

Example

Mineral cost for reaching 😊:

\[ 5t + 6(3 - t) + 1 \]
\[ 5t + 3(3 - t) + 9 \]
Optimal winning strategy

Example

Minimal cost for reaching 🙂:

\[
\max \left( \frac{5t + 6(3 - t) + 1}{5t + 3(3 - t) + 9} \right)
\]

\[\inf_{0 \leq t \leq 2} \max \left( \frac{5t + 6(3 - t) + 1}{5t + 3(3 - t) + 9} \right) = \frac{56}{3}\]

which is achieved with \(t = \frac{1}{3}\)
Optimal winning strategy

Example

Minimal cost for reaching 😊:

$$\inf_{0 \leq t \leq 2} \max \left( \begin{array}{c} 5t + 6(3 - t) + 1 \\ 5t + 3(3 - t) + 9 \end{array} \right)$$

which is achieved with $$t = \frac{1}{3}$$.
Optimal winning strategy

Example

Optimal winning strategy

Minimal cost for reaching 😊:

\[ \inf_{0 \leq t \leq 2} \max \left( \frac{5t + 6(3 - t) + 1}{5t + 3(3 - t) + 9} \right) = \frac{56}{3} \]
Optimal winning strategy

**Example**

**Minimal cost for reaching 😊:**

\[
\inf_{0 \leq t \leq 2} \max \left( \frac{5t + 6(3 - t) + 1}{5t + 3(3 - t) + 9} \right) = \frac{56}{3}
\]

which is achieved with \( t = \frac{1}{3} \)
Optimal winning strategy

Example

Minimal cost for reaching 😊:

$$\inf_{0 \leq t \leq 2} \max \left( \frac{5t + 6(3 - t) + 1}{5t + 3(3 - t) + 9} \right) = \frac{56}{3}$$

which is achieved with $$t = 1/3$$

Corollary

Regions are not sufficient for solving priced timed games.
Computing optimal winning strategies is undecidable

**Theorem**

*Computing optimal strategies in priced timed games is undecidable.*

Computing optimal winning strategies is undecidable

Theorem

*Computing optimal strategies in priced timed games is undecidable.*

Proof.

The proof relies on simple modules that will allow encoding a two-counter machine:

Computing optimal winning strategies is undecidable

**Theorem**

*Computing optimal strategies in priced timed games is undecidable.*

**Proof.**

The proof relies on simple modules that will allow encoding a two-counter machine:

- Adding the value of clock $x$ to the cost:

  ![Diagram](image)

  - $y = 1, y := 0$
  - $y = 1, y := 0$
  - $z = 0$
  - $x = 1, x := 0$
  - $p = 0$
  - $p = 1$
  - $z = 1$
  - $z := 0$

  $\text{Add}^+(x)$

Computing optimal winning strategies is undecidable

Theorem

*Computing optimal strategies in priced timed games is undecidable.*

Proof.

The proof relies on simple modules that will allow encoding a two-counter machine:

- Adding the value of clock $x$ to the cost:
- Adding $1 - x$ to the cost:

![Diagram](image)

Computing optimal winning strategies is undecidable

Theorem

*Computing optimal strategies in priced timed games is undecidable.*

Proof.

The proof relies on simple modules that will allow encoding a two-counter machine:

- Checking that \( y = 2x \):

![Diagram]

Computing optimal winning strategies is undecidable

**Theorem**

*Computing optimal strategies in priced timed games is undecidable.*

**Proof.**

The proof relies on simple modules that will allow encoding a two-counter machine:

- Checking that $y = 2x$:

![Diagram](attachment:image.png)

Computing optimal winning strategies is undecidable

Theorem

Computing optimal strategies in priced timed games is undecidable.

Proof.

The proof relies on simple modules that will allow encoding a two-counter machine:

- Checking that $y = 2x$:
- Dividing clock $x$ by 2:

\[
\begin{align*}
z &= 0 \\
\dot{p} &= 0 \\
x &= 1 \\
x &:= 0 \\
\dot{p} &= 0 \\
\dot{p} &= 0 \\
y &:= 0 \\
z &= 1 \\
z &:= 0 \\
\dot{p} &= 0 \\
z &= 0 \\
\text{Test}(x = 2y)
\end{align*}
\]

Divide\(_2\)(x)

Theorem

Computing optimal strategies in priced timed games is undecidable.

Proof.

The proof relies on simple modules that will allow encoding a two-counter machine:

- encode counter $c_1$ as $x_1 = 2^{-c_1}$ and counter $c_2$ as $x_2 = 3^{-c_1}$;
- by cleverly juggling with clocks, we can achieve this encoding with three clocks.

Example

- Optimal strategies do not always exist:

![Diagram](https://via.placeholder.com/150)

\[ \dot{p} = 2, \quad \dot{p} = 1, \quad x = 1, \quad x = 0 \]
Turn-based 1-clock priced timed games are decidable

Example

- Optimal strategies do not always exist:

  \[ \dot{p} = 2, \quad \dot{p} = 1, \quad x = 1, \quad x = 0 \]

- Optimal strategies may require memory:

  \[ \dot{p} = 2, \quad x = 1, \quad x < 1, \quad x := 0, \quad x > 0 \]
Turn-based 1-clock priced timed games are decidable

Theorem

*Turn-based 1-clock priced timed games* always admit \( \varepsilon \)-optimal *winning strategies*, and such strategies can be computed.

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games* always admit \( \varepsilon \)-optimal winning strategies, and such strategies can be computed.

**Proof.**

![Diagram](image)


Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.*

**Proof.**

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games* always admit $\epsilon$-optimal winning strategies, and such strategies can be computed.

**Proof.**

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.*

**Proof.**

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games* always admit \( \varepsilon \)-optimal winning strategies, and such strategies can be computed.

**Proof.**

Turn-based 1-clock priced timed games are decidable

Theorem

*Turn-based 1-clock priced timed games* always admit \( \varepsilon \)-optimal winning strategies, and such strategies can be computed.

Proof.

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games* always admit \( \epsilon \)-optimal winning strategies, and such strategies can be computed.

**Proof.**


Turn-based 1-clock priced timed games are decidable

Theorem

*Turn-based 1-clock priced timed games always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.*

Proof.


Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games* always admit \( \varepsilon \)-optimal winning strategies, and such strategies can be computed.

**Proof.**


Turn-based 1-clock priced timed games are decidable

Theorem

Turn-based 1-clock priced timed games always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.

Proof.

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.*

**Proof.**


Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games* always admit *ε*-optimal winning strategies, and such strategies can be computed.

**Proof.**

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.*

**Proof.**

References:
Turn-based 1-clock priced timed games are decidable

Theorem

*Turn-based 1-clock priced timed games* always admit \( \varepsilon \)-optimal winning strategies, and such strategies can be computed.

Proof.

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games always admit \( \varepsilon \)-optimal winning strategies, and such strategies can be computed.*

**Proof.**
Turn-based 1-clock priced timed games are decidable

Theorem

**Turn-based 1-clock priced timed games** always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.

**Proof.**

- The procedure terminates;
- There is a positive granularity for with the region abstraction is correct;
- The optimal cost functions are piecewise affine, continuous, decreasing functions. Their slopes are rates of the automaton.

Outline of the talk

1. Introduction

2. Timed automata with observers

3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies

4. Resource-management problems

5. Conclusions and perspectives
Managing resources

Example

In some cases, resources can both be consumed and regained.

The aim is then to keep the level of resources within given bounds.
Managing resources

Example

Three variants of the problem:

1. lower bound: the aim is to maintain the level of resources above a given bound.
2. interval: the aim is to keep the level of resources within an interval.
3. lower bound with finite capacity: the aim is to keep the level of resources above a given lower bound, but with a finite capacity.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
Managing resources

Example

Three variants of the problem:

1. lower bound: the aim is to maintain the level of resources above a given bound.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
2. **interval**: the aim is to keep the level of resources within an interval.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
2. **interval**: the aim is to keep the level of resources within an interval.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.

2. **interval**: the aim is to keep the level of resources within an interval.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.

2. **interval**: the aim is to keep the level of resources within an interval.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.

2. **interval**: the aim is to keep the level of resources within an interval.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
2. **interval**: the aim is to keep the level of resources within an interval.
3. **lower bound with finite capacity**: the aim is to keep the level of resources above a given lower bound, but with a finite capacity.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
2. **interval**: the aim is to keep the level of resources within an interval.
3. **lower bound with finite capacity**: the aim is to keep the level of resources above a given lower bound, but with a finite capacity.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
2. **interval**: the aim is to keep the level of resources within an interval.
3. **lower bound with finite capacity**: the aim is to keep the level of resources above a given lower bound, but with a finite capacity.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
2. **interval**: the aim is to keep the level of resources within an interval.
3. **lower bound with finite capacity**: the aim is to keep the level of resources above a given lower bound, but with a finite capacity.
### Results in the untimed case

**Theorem**

In the untimed case, the following results hold:

<table>
<thead>
<tr>
<th></th>
<th><strong>existential problem</strong></th>
<th><strong>universal problem</strong></th>
<th><strong>games</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lower bound</strong></td>
<td>∈ PTIME</td>
<td>∈ PTIME</td>
<td>∈ UP ∩ coUP</td>
</tr>
<tr>
<td><strong>Lower bound, finite capacity</strong></td>
<td>∈ PTIME</td>
<td>∈ PTIME</td>
<td>∈ NP</td>
</tr>
<tr>
<td><strong>Interval</strong></td>
<td>∈ PSPACE NP-hard</td>
<td>∈ PTIME</td>
<td>EXPTIME-c.</td>
</tr>
</tbody>
</table>

---

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in $\text{EXPTIME}$.

Proof.

- Corner-point abstraction:

Results in the 1-clock case

**Theorem**

In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.

**Proof.**

- **Corner-point abstraction:**

  ![Diagram](image)

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.

Proof.

- Corner-point abstraction: Only correct if no discrete costs!

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.

Proof.

- Corner-point abstraction: Only correct if no discrete costs!

\[
+2 \quad -3
\]

\[x=1, x:=0\]

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.

Proof.

- Corner-point abstraction: Only correct if no discrete costs!
- In the presence of discrete costs:

Results in the 1-clock case

**Theorem**

*In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in **EXPTIME**.*

**Proof.**

- **Corner-point abstraction:** Only correct if no discrete costs!
- In the presence of discrete costs:
  - compute optimal final resource-level along a non-resetting path;

![Diagram of a 1-clock case with resource levels and transitions](image)

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.

Proof.

- Corner-point abstraction: Only correct if no discrete costs!
- In the presence of discrete costs:
  - compute optimal final resource-level along a non-resetting path;
  - compose the resulting functions for general paths.

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of a strategy for maintaining the resource level within a given interval is undecidable.

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of a strategy for maintaining the resource level within a given interval is undecidable.

Proof.

- Encoding of a two-counter machine: both counters are stored in one cost, as \( \ell = 5 - 2^{-c_1} \cdot 3^{-c_2} \).

Results in the 1-clock case

**Theorem**

In the 1-clock case, the existence of a strategy for maintaining the resource level within a given interval is undecidable.

**Proof.**

- Encoding of a two-counter machine: both counters are stored in one cost, as \( \ell = 5 - 2^{-c_1} \cdot 3^{-c_2} \).
- The following module is used to increment and decrement:

```
\begin{align*}
&x := 0 & m & -6 & m_1 & -6 & m_2 & 30 & m_3 & 30 & m' & -n & x = 1 \\
&5 & \xRightarrow{\text{module ok}} & x = 1 & 5 & \xRightarrow{\text{module ok}} & x = 1 \\
\end{align*}
```

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of a strategy for maintaining the resource level within a given interval is undecidable.

Proof.

- Encoding of a two-counter machine: both counters are stored in one cost, as $\ell = 5 - 2^{-c_1} \cdot 3^{-c_2}$.
- The following module is used to increment and decrement:

```
x := 0 -> m : -6
      |    \  
      |     v
      |    m1: -6
      |      \ x:=0
      |       v
      |      m2: 30
      |        \ x:=0
      |         v
      |        m3: 30
      |          \ x:=0
      |           v
      |           m': -n
      |            x=1
```

Initial level $5 - e$

module ok

Final level $5 - \frac{ne}{6}$

Outline of the talk

1. Introduction

2. Timed automata with observers

3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies

4. Resource-management problems

5. Conclusions and perspectives
Conclusions and perspectives

- **Weighted timed automata** are a powerful formalism for modeling **resources**:
  - expressive enough for many applications;
  - several problems remain **decidable**;
  - some algorithms can be made **symbolic** and are implemented in Uppaal CORA.
Conclusions and perspectives

- **Weighted timed automata** are a powerful formalism for modeling resources:
  - expressive enough for many applications;
  - several problems remain decidable;
  - some algorithms can be made symbolic and are implemented in Uppaal CORA.

- Many open problems:
  - energy constraints for automata with several clocks;
  - timed automata with observers having richer dynamics.

\[
\begin{align*}
\frac{dp}{dt} &= 2 \times p \\
x &= 0 \\
-1 \\
x &= 1 \\
-3 &\rightarrow +6 \\
-6 &\rightarrow +2
\end{align*}
\]
Conclusions and perspectives

- **Weighted timed automata** are a powerful formalism for modeling resources:
  - expressive enough for many applications;
  - several problems remain decidable;
  - some algorithms can be made symbolic and are implemented in Uppaal CORA.

- **Many open problems:**
  - energy constraints for automata with several clocks;
  - timed automata with observers having richer dynamics.

\[ x := 0 \]
\[ x = 1 \]
\[ \frac{dp}{dt} = 2 \times p \]