On the verification and control of timed systems

Patricia Bouyer-Decitre

LSV, CNRS & ENS Cachan, France

Two main parts:
1. An introduction to timed systems (this morning)
2. Modelling resources in timed systems (this afternoon)
An introduction to timed systems

Patricia Bouyer-Decitre

LSV, CNRS & ENS Cachan, France
Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. How far can we extend the model and preserve decidability?
   - Hybrid systems
   - Smaller extensions of timed automata
   - An alternative way of proving decidability

5. Timed automata in practice

6. Conclusion
Time!

**Context:** verification of critical systems

**Time**
- naturally appears in real systems (for ex. protocols, embedded systems)
- appears in properties (for ex. bounded response time)
  
  “Will the airbag oben within 5ms after the car crashes?”

→ Need of models and specification languages integrating timing aspects
Adding timing informations

- **Untimed case**: sequence of observable events
  - $a$: send message
  - $b$: receive message

  $a\ b\ a\ b\ a\ b\ a\ b\ a\ b\ \cdots = (a\ b)^\omega$
Adding timing informations

- **Untimed case:** sequence of observable events
  
  $a$: send message  
  $b$: receive message
  
  $$a \ b \ a \ b \ a \ b \ a \ b \ \cdots = (a \ b)^\omega$$

- **Timed case:** sequence of **dated** observable events
  
  $$(a, d_1) \ (b, d_2) \ (a, d_3) \ (b, d_4) \ (a, d_5) \ (b, d_6) \ \cdots$$

  $d_1$: date at which the first $a$ occurs
  $d_2$: date at which the first $b$ occurs, ...
Adding timing informations

- **Untimed case:** sequence of observable events
  
  \[ a: \text{send message} \quad b: \text{receive message} \]

  \[ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ \cdots = (a \ b)^\omega \]

- **Timed case:** sequence of **dated** observable events

  \[ (a, d_1) \ (b, d_2) \ (a, d_3) \ (b, d_4) \ (a, d_5) \ (b, d_6) \ \cdots \]

  \[ d_1: \text{date at which the first } a \text{ occurs} \]

  \[ d_2: \text{date at which the first } b \text{ occurs, } \ldots \]

- **Discrete-time semantics:** dates are e.g. taken in \( \mathbb{N} \)

  **Ex:** \((a, 1)(b, 3)(c, 4)(a, 6)\)
Adding timing informations

- **Untimed case:** sequence of observable events
  
  $a$: send message  
  $b$: receive message

  $$a\ b\ a\ b\ a\ b\ a\ b\ a\ b\ \cdots = (a\ b)\omega$$

- **Timed case:** sequence of dated observable events

  $$(a, d_1)\ (b, d_2)\ (a, d_3)\ (b, d_4)\ (a, d_5)\ (b, d_6)\ \cdots$$

  $d_1$: date at which the first $a$ occurs
  $d_2$: date at which the first $b$ occurs, . . .

  - **Discrete-time semantics:** dates are e.g. taken in $\mathbb{N}$
    
    Ex: $(a, 1)(b, 3)(c, 4)(a, 6)$

  - **Dense-time semantics:** dates are e.g. taken in $\mathbb{Q}_+$, or in $\mathbb{R}_+$
    
    Ex: $(a, 1.28). (b, 3.1). (c, 3.98)(a, 6.13)$
A case for dense-time

**Time domain:** discrete (e.g. $\mathbb{N}$) or dense (e.g. $\mathbb{Q}_+$ or $\mathbb{R}_+$)

- A compositionality problem with discrete time
- Dense-time is a more general model than discrete time
- But, can we not always discretize?
A digital circuit [Alu91]

Discussion in the context of reachability problems for asynchronous digital circuits [BS91]

A digital circuit [Alu91]

Discussion in the context of reachability problems for asynchronous digital circuits [BS91]

Start with $x=0$ and $y=[101]$ (stable configuration)
A digital circuit [Alu91]

Discussion in the context of reachability problems for asynchronous digital circuits [BS91]

Start with $x=0$ and $y=[101]$ (stable configuration)

The input $x$ changes to 1. The corresponding stable state is $y=[011]$

A digital circuit [Alu91]

Discussion in the context of reachability problems for asynchronous digital circuits [BS91]

Start with \( x=0 \) and \( y=[101] \) (stable configuration)

The input \( x \) changes to 1. The corresponding stable state is \( y=[011] \)

However, many possible behaviours, e.g.

\[
[101] \xrightarrow{y_2 \quad 1.2} [111] \xrightarrow{y_3 \quad 2.5} [110] \xrightarrow{y_1 \quad 2.8} [010] \xrightarrow{y_3 \quad 4.5} [011]
\]


A digital circuit [Alu91]

Discussion in the context of reachability problems for asynchronous digital circuits [BS91]

Start with \( x=0 \) and \( y=[101] \) (stable configuration)

The input \( x \) changes to 1. The corresponding stable state is \( y=[011] \)

However, many possible behaviours, e.g.

\[
\begin{align*}
[101] & \xrightarrow{y_2 \ 1.2} [111] & \xrightarrow{y_3 \ 2.5} [110] & \xrightarrow{y_1 \ 2.8} [010] & \xrightarrow{y_3 \ 4.5} [011]
\end{align*}
\]

**Reachable configurations:** \([ [101], [111], [110], [010], [011], [001] ]\)


Is discretizing sufficient? An example [Alu91]

- This digital circuit is not 1-discretizable.
Is discretizing sufficient? An example [Alu91]

- This digital circuit is not 1-discretizable.
- Why that? (initially $x = 0$ and $y = [11100000]$, $x$ is set to 1)
Is discretizing sufficient? An example [Alu91]

- This digital circuit is not 1-discretizable.
- Why that? (initially $x = 0$ and $y = [11100000]$, $x$ is set to 1)

$$[11100000] \xrightarrow{\frac{y_1}{1}} [01100000] \xrightarrow{\frac{y_2}{1.5}} [00100000] \xrightarrow{\frac{y_3,y_5}{2}} [00001000] \xrightarrow{\frac{y_5,y_7}{3}} [00000010] \xrightarrow{\frac{y_7,y_8}{4}} [00000001]$$
Is discretizing sufficient? An example [Alu91]

- This digital circuit is not 1-discretizable.
- Why that? (initially \( x = 0 \) and \( y = [11100000] \), \( x \) is set to 1)

\[
\begin{align*}
[11100000] \xrightarrow{y_1} [01100000] \quad &\quad [00100000] \xrightarrow{y_2} [00001000] \xrightarrow{y_3, y_5} [00000010] \xrightarrow{y_5, y_7} [00000001] \\
[11100000] \xrightarrow{y_1, y_2, y_3} &\quad [00000000]
\end{align*}
\]
Is discretizing sufficient? An example [Alu91]

- This digital circuit **is not** 1-discretizable.
- Why that? (initially $x = 0$ and $y = [11100000]$, $x$ is set to 1)

\[
\begin{align*}
[11100000] & \xrightarrow{y_1 = 1} [01100000] \xrightarrow{y_2 = 1.5} [00100000] \xrightarrow{y_3, y_5 = 2} [00001000] \xrightarrow{y_5, y_7 = 3} [00000010] \xrightarrow{y_7, y_8 = 4} [00000001] \\
[11100000] & \xrightarrow{y_1, y_2, y_3 = 1} [00000000] \\
[11100000] & \xrightarrow{y_1 = 1} [01110000] \xrightarrow{y_2, y_3, y_4, y_5 = 2} [00000000]
\end{align*}
\]
Is discretizing sufficient? An example [Alu91]

- This digital circuit **is not** 1-discretizable.
- Why that? (initially $x = 0$ and $y = [11100000]$, $x$ is set to 1)

\[
\begin{align*}
[11100000] & \xrightarrow{\frac{y_1}{1}} [01100000] & \xrightarrow{\frac{y_2}{1.5}} [00100000] & \xrightarrow{\frac{y_3, y_5}{2}} [00001000] & \xrightarrow{\frac{y_5, y_7}{3}} [00000010] & \xrightarrow{\frac{y_7, y_8}{4}} [00000001] \\
[11100000] & \xrightarrow{\frac{y_1, y_2, y_3}{1}} [00000000] \\
[11100000] & \xrightarrow{\frac{y_1}{1}} [01111000] & \xrightarrow{\frac{y_2, y_3, y_4, y_5}{2}} [00000000] \\
[11100000] & \xrightarrow{\frac{y_1, y_2}{1}} [00100000] & \xrightarrow{\frac{y_3, y_5, y_6}{2}} [00001100] & \xrightarrow{\frac{y_5, y_6}{3}} [00000000]
\end{align*}
\]
Is discretizing sufficient? An example [Alu91]

- This digital circuit **is not** 1-discretizable.
- Why that?  
  (initially $x = 0$ and $y = [11100000]$, $x$ is set to 1)

\[
\begin{align*}
[11100000] & \rightarrow [01100000] \\
[11100000] & \rightarrow [01110000] \\
[11100000] & \rightarrow [00100000] \\
[11100000] & \rightarrow [00110000] \\
\end{align*}
\]
Is discretizing sufficient?

**Theorem** [BS91]

For every \( k \geq 1 \), there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity \( \frac{1}{k} \)).

Is discretizing sufficient?

**Theorem [BS91]**

For every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

**Claim**

Finding a correct granularity is as difficult as computing the set of reachable states in dense-time.

Is discretizing sufficient?

**Theorem [BS91]**

For every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

**Claim**

Finding a correct granularity is as difficult as computing the set of reachable states in dense-time.

**Further counter-example**

There exist systems for which no granularity exists. (see later)

Is discretizing sufficient?

**Theorem** [BS91]

For every \( k \geq 1 \), there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity \( \frac{1}{k} \)).

**Claim**

Finding a correct granularity is as difficult as computing the set of reachable states in dense-time.

**Further counter-example**

There exist systems for which no granularity exists. (see later)

Hence, we better consider a dense-time domain!

Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. How far can we extend the model and preserve decidability?
   - Hybrid systems
   - Smaller extensions of timed automata
   - An alternative way of proving decidability

5. Timed automata in practice

6. Conclusion
A plethora of models...

... for real-time systems:
  - timed circuits,
  - time(d) Petri nets,
  - timed process algebra,
  - timed automata,
  - ...

... and for real-time properties:
  - timed observers,
  - real-time logics: MTL, TPTL, TCTL, QTL, MITL...
A plethora of models...

... for real-time systems:
  - timed circuits,
  - time(d) Petri nets,
  - timed process algebra,
  - timed automata,
  ... 

... and for real-time properties:
  - timed observers,
  - real-time logics: MTL, TPTL, TCTL, QTL, MITL...
Timed automata [AD90]

- A finite control structure + variables (clocks)
- A transition is of the form:
  \[ g, \ a, \ C := 0 \]

- An enabling condition (or guard) is:
  \[ g ::= x \sim c \mid x - y \sim c \mid g \wedge g \]
  where \( \sim \in \{<, \leq, =, \geq, >\} \)

[AD90] Alur, Dill. Automata for modeling real-time systems (ICALP’90).
Timed automata [AD90]

- A finite control structure + variables (clocks)
- A transition is of the form:
  \[ g, a, C := 0 \]
  
  Enabling condition
  Reset to zero

- An enabling condition (or guard) is:
  \[ g \; ::= \; x \sim c \mid x - y \sim c \mid g \land g \]
  
  where \( \sim \in \{<, \leq, =, \geq, >\} \)

[AD90] Alur, Dill. Automata for modeling real-time systems (ICALP’90).
An example of a timed automaton
An example of a timed automaton

The timed automaton model

```
safe

x 0
y 0
```

```
problem, x:=0

\[
\text{repair, } x \leq 15 \\
\text{delayed, } y:=0
\]

done, 22 \leq y \leq 25

repairing

\[
\text{repair, } 2 \leq y \land x \leq 56 \\
y:=0
\]

failsafe
```
An example of a timed automaton

The timed automaton model

<table>
<thead>
<tr>
<th>safe</th>
<th>23</th>
<th>safe</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>23</td>
</tr>
</tbody>
</table>

done, $22 \leq y \leq 25$
repair, $x \leq 15$
repair, $y = 0$
delayed, $y = 0$
An example of a timed automaton

- **Safe**
  - Transition: \( x := 0 \) to **Problem**
  - Transition: \( y := 0 \) to **Failsafe**

- **Problem**
  - Transition: \( 23 \) to **Safe**
  - Transition: \( x := 0 \) to **Repair**
  - Transition: \( 15 \leq x \leq 16 \) to **Failsafe**

- **Repair**
  - Transition: \( x \leq 15 \) to **Problem**
  - Transition: \( y := 0 \) to **Failsafe**

- **Failsafe**
  - Transition: \( 2 \leq y \land x \leq 56 \) to **Repair**

- **Done**
  - Transition: \( 22 \leq y \leq 25 \) to **Problem**
An example of a timed automaton

The timed automaton model

<table>
<thead>
<tr>
<th></th>
<th>safe</th>
<th>23</th>
<th>safe</th>
<th>problem</th>
<th>alarm</th>
<th>15.6</th>
<th>alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>23</td>
<td></td>
<td>0</td>
<td>15.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>23</td>
<td></td>
<td>23</td>
<td>38.6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An example of a timed automaton

The timed automaton model
An example of a timed automaton

The timed automaton model

<table>
<thead>
<tr>
<th>States</th>
<th>Transitions</th>
<th>Variables</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td></td>
<td>X, 0</td>
<td></td>
</tr>
<tr>
<td>alarm</td>
<td>done, $22 \leq y \leq 25$</td>
<td>Y, 0</td>
<td></td>
</tr>
<tr>
<td>failsafe</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Transition table:

- safe $\xrightarrow{23}$ safe, $x := 0$
- problem $\xrightarrow{15.6}$ alarm, $15 \leq x \leq 16$
- repair, $x \leq 15$ $\xrightarrow{y := 0}$ repairing
- repair, $2 \leq y \land x \leq 56$ $\xrightarrow{y := 0}$ failsafe
- repaired, $y := 0$ $\xrightarrow{delayed}$ failsafe

Time table:

<table>
<thead>
<tr>
<th>State</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>alarm</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>repair</td>
<td>15.6</td>
<td>38.6</td>
</tr>
<tr>
<td>failsafe</td>
<td>0</td>
<td>2.3</td>
</tr>
</tbody>
</table>

Starting state: safe

Ending state: failsafe
An example of a timed automaton

The timed automaton model
An example of a timed automaton

The timed automaton model
An example of a timed automaton

The timed automaton model

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td>23</td>
<td>safe</td>
</tr>
<tr>
<td></td>
<td>problem</td>
<td>alarm</td>
</tr>
<tr>
<td>problem</td>
<td>15.6</td>
<td>alarm</td>
</tr>
<tr>
<td></td>
<td>delayed</td>
<td>failsafe</td>
</tr>
<tr>
<td>failsafe</td>
<td>2.3</td>
<td>failsafe</td>
</tr>
<tr>
<td></td>
<td>repair</td>
<td>repairing</td>
</tr>
<tr>
<td></td>
<td>22.1</td>
<td>repairing</td>
</tr>
<tr>
<td></td>
<td>done</td>
<td>safe</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variables</th>
<th>Initial Value</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>38.6</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>15.6</td>
<td>17.9</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>2.3</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>17.9</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>15.6</td>
<td>40</td>
</tr>
<tr>
<td>y</td>
<td>22.1</td>
<td>22.1</td>
</tr>
</tbody>
</table>
An example of a timed automaton

This run read the timed word
(problem, 23)(delayed, 38.6)(repair, 40.9), (done, 63).
Timed automata semantics

- $A = (\Sigma, L, X, \rightarrow)$ is a TA

- **Configurations**: $(\ell, \nu) \in L \times T^X$ where $T$ is the time domain
  
  $\nu$ is called the (clock) valuation

- **Timed transition system**:
  
  - **action transition**: $(\ell, \nu) \xrightarrow{a} (\ell', \nu')$ if $\exists \ell \xrightarrow{g,a,r} \ell' \in A$ s.t.
    
    \[
    \begin{cases}
    \nu \models g \\
    \nu' = \nu[r \leftarrow 0]
    \end{cases}
    \]

  - **delay transition**: $(\ell, \nu) \xrightarrow{\delta(d)} (\ell, \nu + d)$ if $d \in T$
Discrete vs dense-time semantics

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

\[ x = 1, \ a, \ x := 0 \quad y < 1, \ b, \ y := 0 \]
Discrete vs dense-time semantics

\[ L_{\text{dense}} = \{ ((ab)^\omega, \tau) \mid \forall i, \; \tau_{2i-1} = i \text{ and } \tau_{2i} - \tau_{2i-1} > \tau_{2i+2} - \tau_{2i+1} \} \]
Discrete vs dense-time semantics

- **Dense-time:**
  \[ L_{dense} = \{((ab)^\omega, \tau) \mid \forall i, \tau_{2i-1} = i \text{ and } \tau_{2i} - \tau_{2i-1} > \tau_{2i+2} - \tau_{2i+1} \} \]

- **Discrete-time:** \[ L_{discrete} = \emptyset \]
Discrete vs dense-time semantics

\[ L_{\text{dense}} = \{ ((ab)^\omega, \tau) \mid \forall i, \; \tau_{2i-1} = i \text{ and } \tau_{2i} - \tau_{2i-1} > \tau_{2i+2} - \tau_{2i+1} \} \]

- **Dense-time:**

- **Discrete-time:** \( L_{\text{discrete}} = \emptyset \)

However, it does result from the following parallel composition:
Classical verification problems

- reachability of a control state
- \( S \sim S' \): bisimulation, etc...
- \( L(S) \subseteq L(S') \): language inclusion
- \( S \models \varphi \) for some formula \( \varphi \): model-checking
- \( S \parallel A_T + \) reachability: testing automata
- ...
Classical temporal logics

Path formulas:

G $\varphi$  
“Always”

F $\varphi$  
“Eventually”

$\varphi$ U $\varphi'$  
“Until”

X $\varphi$  
“Next”

State formulas:

A $\psi$  

E $\psi$  

$\leadsto$ LTL: Linear Temporal Logic [Pnu77],
CTL: Computation Tree Logic [EC82]

[Pnu77] Pnueli. The temporal logic of programs (FoCS’77).
[EC82] Emerson, Clarke. Using branching time temporal logic to synthesize synchronization skeletons (Science of Computer Programming 1982).
Adding time to temporal logics

Classical temporal logics allow us to express that
“any problem is followed by an alarm”

[ACD90] Alur, Courcoubetis, Dill. Model-checking for real-time systems (*LICS’90*).
[ACD93] Alur, Courcoubetis, Dill. Model-checking in dense real-time (*Information and Computation*).
[HNSY94] Henzinger, Nicollin, Sifakis, Yovine. Symbolic model-checking for real-time systems (*ACM Transactions on Computational Logic*).
Adding time to temporal logics

Classical temporal logics allow us to express that

“any problem is followed by an alarm”

With CTL:

$$\text{AG}(\text{problem } \Rightarrow \text{AF alarm})$$
Adding time to temporal logics

Classical temporal logics allow us to express that
“any problem is followed by an alarm”

With CTL:

\[ \text{AG}(\text{problem} \implies \text{AF alarm}) \]

How can we express:
“any problem is followed by an alarm within 20 time units”?

Adding time to temporal logics

Classical temporal logics allow us to express that
“any problem is followed by an alarm”

With CTL:

\[ \text{AG} (\text{problem} \Rightarrow \text{AF} \text{ alarm}) \]

How can we express:

“any problem is followed by an alarm within 20 time units”?

- Temporal logics with **subscripts**. ex: CTL +

\[ \text{E} \varphi \mathcal{U}_{\sim k} \psi \]

\[ \text{A} \varphi \mathcal{U}_{\sim k} \psi \]

---

**References**

[ACD90] Alur, Courcoubetis, Dill. Model-checking for real-time systems (*LICS’90*).

[ACD93] Alur, Courcoubetis, Dill. Model-checking in dense real-time (*Information and Computation*).

[HNSY94] Henzinger, Nicollin, Sifakis, Yovine. Symbolic model-checking for real-time systems (*ACM Transactions on Computational Logic*).
Adding time to temporal logics

Classical temporal logics allow us to express that
“any problem is followed by an alarm”

With CTL:
\[ \text{AG} (\text{problem} \Rightarrow \text{AF} \ \text{alarm}) \]

How can we express:
“any problem is followed by an alarm within 20 time units”?

- Temporal logics with \textit{subscripts}.

\[ \text{AG} (\text{problem} \Rightarrow \text{AF}_{\leq 20} \ \text{alarm}) \]

\[ \text{ACD90} \] Alur, Courcoubetis, Dill. Model-checking for real-time systems (\textit{LICS'90}).
\[ \text{ACD93} \] Alur, Courcoubetis, Dill. Model-checking in dense real-time (\textit{Information and Computation}).
\[ \text{HNSY94} \] Henzinger, Nicollin, Sifakis, Yovine. Symbolic model-checking for real-time systems (\textit{ACM Transactions on Computational Logic}).
Adding time to temporal logics

Classical temporal logics allow us to express that
“any problem is followed by an alarm”

With CTL:

\[ \text{AG}(\text{problem \Rightarrow AF alarm}) \]

How can we express:
“any problem is followed by an alarm within 20 time units”?

- Temporal logics with **subscripts**.

\[ \text{AG}(\text{problem \Rightarrow AF} \leq_{20} \text{alarm}) \]

- Temporal logics with **clocks**.

\[ \text{AG}(\text{problem \Rightarrow (x in AF (x \leq 20 \land alarm))}) \]

---

[ACD90] Alur, Courcoubetis, Dill. Model-checking for real-time systems (*LICS’90*).
[ACD93] Alur, Courcoubetis, Dill. Model-checking in dense real-time (*Information and Computation*).
[HNSY94] Henzinger, Nicollin, Sifakis, Yovine. Symbolic model-checking for real-time systems (*ACM Transactions on Computational Logic*).
Adding time to temporal logics

Classical temporal logics allow us to express that

“any problem is followed by an alarm”

With CTL:

\[ \text{AG (problem } \Rightarrow \text{ AF alarm)} \]

How can we express:

“any problem is followed by an alarm within 20 time units”?

- Temporal logics with **subscripts**.

\[ \text{AG (problem } \Rightarrow \text{ AF}_{\leq 20} \text{ alarm)} \]

- Temporal logics with **clocks**.

\[ \text{AG (problem } \Rightarrow (x \text{ in AF (x } \leq 20 \land \text{ alarm))))} \]

\[ \leadsto \text{TCTL: Timed CTL} \quad [\text{ACD90, ACD93, HNSY94}] \]

The train crossing example

Train$_i$ with $i = 1, 2, ...$

Diagram:
- **Far**
  - Transition: $10 < x_i < 20$, Exit!
  - Transition: $20 < x_i < 30$, $a, x_i := 0$

**Before**, $x_i < 30$
- Transition: $x_i := 0$
- Transition: $20 < x_i < 30$, $a, x_i := 0$
- Transition: $10 < x_i < 20$, Exit!
The train crossing example

The gate:

- **Open** \(\xrightarrow{\text{GoDown?}, H_g := 0} \text{Lowering}, H_g < 10\)
- **Raising**, \(H_g < 10\) \(\xleftarrow{\text{a}}\)
- **Close** \(\xleftarrow{\text{GoUp?}, H_g := 0}\)
- **Lowering**, \(H_g < 10\) \(\xrightarrow{\text{a}}\)
The train crossing example

The controller:

- $c_1, H_c \leq 20$
- $H_c = 20$, GoUp!

- $c_0$, Exit?, $H_c := 0$
- App?, $H_c := 0$
- $H_c \leq 10$, GoDown!

- $c_2, H_c \leq 10$
- Exit?
- App?

(3)
The train crossing example (4)

We use the synchronization function $f$:

<table>
<thead>
<tr>
<th>Train$_1$</th>
<th>Train$_2$</th>
<th>Gate</th>
<th>Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>App!</td>
<td>.</td>
<td>.</td>
<td>App?</td>
</tr>
<tr>
<td>Exit!</td>
<td>.</td>
<td>.</td>
<td>Exit?</td>
</tr>
<tr>
<td>.</td>
<td>Exit!</td>
<td>.</td>
<td>Exit?</td>
</tr>
<tr>
<td>a</td>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>a</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>a</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>GoUp?</td>
<td>GoUp!</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>GoDown?</td>
<td>GoDown!</td>
</tr>
</tbody>
</table>

to define the parallel composition $(\text{Train}_1 \ || \ \text{Train}_2 \ || \ \text{Gate} \ || \ \text{Controller})$

**NB:** the parallel composition does not add expressive power!
The train crossing example

Some properties one could check:

- Is the gate closed when a train crosses the road?
The train crossing example

Some properties one could check:
- Is the gate closed when a train crosses the road?

\( \text{AG} (\text{train.On} \Rightarrow \text{gate.Close}) \)
The train crossing example (5)

Some properties one could check:

- Is the gate closed when a train crosses the road?
  \[ \text{AG} (\text{train.On} \Rightarrow \text{gate.Close}) \]

- Is the gate always closed for less than 5 minutes?
The train crossing example

Some properties one could check:

- Is the gate closed when a train crosses the road?

\[ \text{AG} \left( \text{train.On} \Rightarrow \text{gate.Close} \right) \]

- Is the gate always closed for less than 5 minutes?

\[ \neg \text{EF} \left( \text{gate.Close} \land \text{E} \left( \text{gate.Close} \ U_{>5 \text{ min}} \neg \text{gate.Close} \right) \right) \]
Another example: A Fischer protocol

A mutual exclusion protocol with a shared variable \(id\) [AL94].

Another example: A Fischer protocol

A mutual exclusion protocol with a shared variable $id$ [AL94].

Process $i$:

$a : \text{await } (id = 0)$;

$b : \text{set } id \text{ to } i$;

$c : \text{await } (id = i)$;

$d : \text{enter critical section}.$

\[ \leadsto \text{a max. delay } k_1 \text{ between } a \text{ and } b \]
\[ \leadsto \text{a min. delay } k_2 \text{ between } b \text{ and } c \]

[AL94] Abadí, Lamport. An old-fashioned recipe for real time (*ACM Transactions on Programming Languages and Systems*).
Another example: A Fischer protocol

A mutual exclusion protocol with a shared variable \textit{id} \cite{AL94}.

Process \textit{i}:
\begin{align*}
a : & \text{ await } (id = 0); & \leadsto & \text{ a max. delay } k_1 \text{ between } a \text{ and } b \\
b : & \text{ set } id \text{ to } i; & \leadsto & \text{ a min. delay } k_2 \text{ between } b \text{ and } c \\
c : & \text{ await } (id = i); \\
d : & \text{ enter critical section.}
\end{align*}

\leadsto \text{ See the demo with the tool Uppaal} \\
\quad (\text{can be downloaded freely on http://www.uppaal.com/})

\cite{AL94} Abadí, Lamport. An old-fashioned recipe for real time \textit{(ACM Transactions on Programming Languages and Systems)}. 
Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. How far can we extend the model and preserve decidability?
   Hybrid systems
   Smaller extensions of timed automata
   An alternative way of proving decidability

5. Timed automata in practice

6. Conclusion
Verification

**Emptiness problem:** is the language accepted by a timed automaton empty?

- basic reachability/safety properties
- basic liveness properties

(final states)
(\(\omega\)-regular conditions)

[AD90] Alur, Dill. Automata for modeling real-time systems (ICALP’90).
Verification

**Emptiness problem:** is the language accepted by a timed automaton empty?

- **Problem:** the set of configurations is infinite
  \(\leadsto\) classical methods for finite-state systems cannot be applied
Verification

**Emptiness problem:** is the language accepted by a timed automaton empty?

- **Problem:** the set of configurations is infinite
  \[ \leadsto \] classical methods for finite-state systems cannot be applied

- **Positive key point:** variables (clocks) increase at the same speed

Verification

Emptiness problem: is the language accepted by a timed automaton empty?

- Problem: the set of configurations is infinite
  \(\leadsto\) classical methods for finite-state systems cannot be applied

- Positive key point: variables (clocks) increase at the same speed

Theorem [AD90,AD94]
The emptiness problem for timed automata is decidable and PSPACE-complete.

[AD90] Alur, Dill. Automata for modeling real-time systems (ICALP’90).
Verification

Emptiness problem: is the language accepted by a timed automaton empty?

- Problem: the set of configurations is infinite
  \[ \leadsto \] classical methods for finite-state systems cannot be applied

- Positive key point: variables (clocks) increase at the same speed

Theorem \[\text{[AD90, AD94]}\]
The emptiness problem for timed automata is decidable and PSPACE-complete.

Method: construct a finite abstraction
The region abstraction
The region abstraction

- “compatibility” between regions and constraints
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
The region abstraction

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

\[ \leadsto \] an equivalence of finite index
a time-abstract bisimulation
Time-abstract bisimulation

∀ → a →

Timed automata, decidability issues
Time-abstract bisimulation
Time-abstract bisimulation

\[ \forall \quad \exists \quad a \quad \delta(d) \quad \forall d > 0 \]
Time-abstract bisimulation

\[
\forall x \quad a \quad \exists y
\]

\[
\forall d > 0 \quad \delta(d) \quad \exists d' > 0 \quad \delta(d')
\]
Time-abstract bisimulation

\[
\begin{align*}
\forall \quad & (\ell_0, v_0) \xrightarrow{a_1, t_1} (\ell_1, v_1) \\
\exists \quad & (\ell_0, v_0) \xrightarrow{a} (\ell, v) \\
\forall d > 0 \quad & (\ell_1, v_1) \xrightarrow{\delta(d)} (\ell_2, v_2) \\
\exists d' > 0 \quad & (\ell_2, v_2) \xrightarrow{\delta(d')} (\ell_3, v_3) \\
\end{align*}
\]
Time-abstract bisimulation

\[
\begin{align*}
\forall & \quad (\ell_0, v_0) \xrightarrow{a_1, t_1} (\ell_1, v_1) \\
& \quad \downarrow \\
\exists & \quad (\ell_0, R_0) \xrightarrow{a_1} (\ell_1, R_1) \\
& \quad \downarrow \\
\forall d > 0 & \quad \delta(d) \\
& \quad \downarrow \\
\exists d' > 0 & \quad \delta(d')
\end{align*}
\]

with $v_i \in R_i$ for all $i$. 
Time-abstract bisimulation

\[ \forall \quad (\ell_0, v_0) \xrightarrow{a_{1, t_1}} (\ell_1, v_1) \xrightarrow{a_{2, t_2}} (\ell_2, v_2) \xrightarrow{a_{3, t_3}} \ldots \]

\[ \exists \quad (\ell_0, R_0) \xrightarrow{a_{1}} (\ell_1, R_1) \xrightarrow{a_{2}} (\ell_2, R_2) \xrightarrow{a_{3}} \ldots \]

\[ \exists d' > 0 \quad (\ell_0, v_0) \xrightarrow{\delta(d')} (\ell_0, v_0) \]

\[ \forall d > 0 \quad (\ell_0, v_0) \xrightarrow{\delta(d)} (\ell_0, v_0) \]

with \( v_i \in R_i \) for all \( i \).
The region abstraction
The region abstraction

\[
\begin{align*}
2 < x < 3 \\
1 < y < 2 \\
\{x\} < \{y\}
\end{align*}
\]
The region abstraction
The region abstraction

reset of clock $y$
The region abstraction

reset of clock x
The region graph

A finite graph representing time elapsing and reset of clocks:

\[\text{time elapsing}\]

\[\text{reset to 0}\]
Region automaton $\equiv$ finite bisimulation quotient

$\text{timed automaton} \bowtie \text{region graph}$
Region automaton $\equiv$ finite bisimulation quotient

**timed automaton $\otimes$ region graph**

\[
\ell \xrightarrow{g,a,C:=0} \ell' \text{ is transformed into:}
\]

\[(\ell, R) \xrightarrow{a} (\ell', R') \text{ if there exists } R'' \in \text{Succ}_t^*(R) \text{ s.t.}
\]

- $R'' \subseteq g$
- $[C \leftarrow 0]R'' \subseteq R'$
Region automaton $\equiv$ finite bisimulation quotient

\[
\text{timed automaton } \otimes \text{ region graph}
\]

\[
\ell \xrightarrow{g,a,C:=0} \ell' \text{ is transformed into:}
\]

\[
(\ell, R) \xrightarrow{a} (\ell', R') \text{ if there exists } R'' \in \text{Succ}_t^*(R) \text{ s.t.}
\]

- $R'' \subseteq g$
- $[C \leftarrow 0]R'' \subseteq R'$

\[
\mathcal{L}(\text{reg. aut.}) = \text{UNTIME}(\mathcal{L}(\text{timed aut.}))
\]

where $\text{UNTIME}((a_1, t_1)(a_2, t_2) \ldots) = a_1a_2 \ldots$
\[ \mathcal{L}(\text{reg. aut.}) = \text{UNTIME}(\mathcal{L}(\text{timed aut.})) \]
An example [AD94]
PSPACE membership

The size of the region graph is in $O(|X| \cdot 2^{|X|})$

- **One configuration:** a discrete location + a region
PSPACE membership

The size of the region graph is in $O(|X|!.2^{|X|})$

- **One configuration:** a discrete location + a region
  - a discrete location: log-space
PSPACE membership

The size of the region graph is in $O(|X|!.2^{|X|})$

- **One configuration:** a discrete location $+$ a region
  - a discrete location: log-space
  - a region:
    - an interval for each clock
    - an interval for each pair of clocks
PSPACE membership

The size of the region graph is in $O(|X|! \cdot 2^{|X|})$

- **One configuration**: a discrete location + a region
  - a discrete location: log-space
  - a region:
    - an interval for each clock
    - an interval for each pair of clocks
  \leadsto requires polynomial space
PSPACE membership

The size of the region graph is in $O(|X|! \cdot 2^{|X|})$

- **One configuration**: a discrete location + a region
  - a discrete location: log-space
  - a region:
    - an interval for each clock
    - an interval for each pair of clocks

  $\leadsto$ requires polynomial space

- By guessing a path of length at most exponential: needs only to store two consecutive configurations
PSPACE membership

The size of the region graph is in $O(|X|! \cdot 2^{|X|})$

- **One configuration**: a discrete location + a region
  - a discrete location: log-space
  - a region:
    - an interval for each clock
    - an interval for each pair of clocks

  $\leadsto$ requires polynomial space

- By guessing a path of length at most exponential: needs only to store two consecutive configurations

  $\leadsto$ in $\text{NPSPACE}$, thus in $\text{PSPACE}$
PSPACE-hardness

\[
\begin{align*}
\mathcal{M} \text{ LBTM} \quad w_0 \in \{a, b\}^* & \implies A_{\mathcal{M}, w_0} \text{ s.t. } \mathcal{M} \text{ accepts } w_0 \text{ iff the final state of } A_{\mathcal{M}, w_0} \text{ is reachable} \\
\end{align*}
\]

\[w_0 \quad C_j \quad \{x_j, y_j\} \]

- \(C_j\) contains an “a” if \(x_j = y_j\)
- \(C_j\) contains a “b” if \(x_j < y_j\)

(these conditions are invariant by time elapsing)

LBTM: linearly bounded Turing machine (a witness for PSPACE-complete problems)
PSPACE-hardness (cont.)

If \( q \xrightarrow{\alpha, \alpha', \delta} q' \) is a transition of \( \mathcal{M} \), then for each position \( i \) of the tape, we have a transition

\[
(q, i) \xrightarrow{g, r := 0} (q', i')
\]

where:

- \( g \) is \( x_i = y_i \) (resp. \( x_i < y_i \)) if \( \alpha = a \) (resp. \( \alpha = b \))
- \( r = \{x_i, y_i\} \) (resp. \( r = \{x_i\} \)) if \( \alpha' = a \) (resp. \( \alpha' = b \))
- \( i' = i + 1 \) (resp. \( i' = i - 1 \)) if \( \delta \) is right and \( i < n \) (resp. left)

**Enforcing time elapsing:** on each transition, add the condition \( t = 1 \) and clock \( t \) is reset.

**Initialization:** \( \text{init} \xrightarrow{t = 1, r_0 := 0} (q_0, 1) \) where \( r_0 = \{x_i \mid w_0[i] = b\} \cup \{t\} \)

**Termination:** \( (q_f, i) \longrightarrow \text{end} \)
The case of single-clock timed automata

Exercise [LMS04]

Think of the special case of single-clock timed automata. Can we do better than PSPACE?

[LMS04] Laroussinie, Markey, Schnoebelen. Model checking timed automata with one or two clocks (CONCUR’04).
Consequence of region automata construction

Region automata:
correct finite (and exponential) abstraction for checking reachability/Büchi-like properties.
Consequence of region automata construction

Region automata:
correct finite (and exponential) abstraction for checking reachability/Büchi-like properties.

However...
everything cannot be reduced to finite automata...
A model not far from undecidability

Some bad news...

- Language universality is undecidable [AD90]
- Language inclusion is undecidable [AD90]
- Complementability is undecidable [Tri03,Fin06]
- ...

[Tri03] Tripakis. Folk theorems on the determinization and minimization of timed automata (FORMATS’03).
[Fin06] Finkel. Undecidable problems about timed automata (FORMATS’06).
A model not far from undecidability

Some bad news...

- Language universality is undecidable
- Language inclusion is undecidable
- Complementability is undecidable
- ...

An example of non-determinizable/non-complementable timed aut.:

[AD90] Tripakis. Folk theorems on the determinization and minimization of timed automata (FORMATS’03).
[FIn06] Finkel. Undecidable problems about timed automata (FORMATS’06).
[AM04] Alur, Madhusudan. Decision problems for timed automata: A survey (SIM-04:RT)).
A model not far from undecidability

Some bad news...

- Language universality is undecidable
- Language inclusion is undecidable
- Complementability is undecidable
- ...

An example of non-determinizable/non-complementable timed aut.:

[AD90]}

\[ a, b \quad x \neq 1, \ a, b \]

\[ a, \ x := 0 \]

\[ a, \ x := 0 \]

[Tri03] Tripakis. Folk theorems on the determinization and minimization of timed automata (FORMATS’03).
[Fin06] Finkel. Undecidable problems about timed automata (FORMATS’06).
[AM04] Alur, Madhusudan. Decision problems for timed automata: A survey (SFM-04:RT)).
A model not far from undecidability

Some bad news...

- Language universality is undecidable [AD90]
- Language inclusion is undecidable [AD90]
- Complementability is undecidable [Tri03, Fin06]
- ...

An example of non-determinizable/non-complementable timed aut.: [AM04]

\[ L \cap \{ (a^*b^*, \tau) | \text{all } a's \text{ happen before 1 and no two } a's \text{ simultaneously} \} \]

UNTIME \( (L \cap \{ (a^*b^*, \tau) | \text{all } a's \text{ happen before 1 and no two } a's \text{ simultaneously} \} ) \) is not regular (exercise!)

[Tri03] Tripakis. Folk theorems on the determinization and minimization of timed automata (FORMATS’03).
[Fin06] Finkel. Undecidable problems about timed automata (FORMATS’06).
The two-counter machine

Definition

A two-counter machine is a finite set of instructions over two counters (c and d):

- Incrementation:
  (p): \( c := c + 1; \) goto (q)

- Decrementation:
  (p): if \( c > 0 \) then \( c := c - 1; \) goto (q) else goto (r)

Theorem [Minsky 67]

The halting problem for two counter machines is undecidable.
Undecidability of universality

Theorem [AD90]
Universality of timed automata is undecidable.

- one configuration is encoded in one time unit
- number of $c$’s: value of counter $c$
- number of $d$’s: value of counter $d$
- one time unit between two corresponding $c$’s (resp. $d$’s)

\[ \leadsto \text{We encode “non-behaviours” of a two-counter machine} \]
Example

Module to check that if instruction $i$ does not decrease counter $c$, then all actions $c$ appearing less than 1 t.u. after $b_i$ has to be followed by an other $c$ 1 t.u. later.

\[ x = 1, \neg c \]

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

\[ b_i, x := 0 \]

\[ x \neq 1 \]
Example

Module to check that if instruction $i$ does not decrease counter $c$, then all actions $c$ appearing less than 1 t.u. after $b_i$ has to be followed by an other $c$ 1 t.u. later.

The union of all small modules is not universal iff The two-counter machine has a recurring computation
Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:
  - various extensions of timed automata
    - [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
    - [Bouyer, Dufourd, Fleury, Petit 2004] · · ·
Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:
  - various extensions of timed automata
    [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
    [Bouyer, Dufourd, Fleury, Petit 2004] · · ·
  - model-checking of branching-time properties (TCTL, timed \( \mu \)-calculus)
    [Alur, Courcoubetis, Dill 1993] [Laroussinie, Larsen, Weise 1995]
Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:
  - various extensions of timed automata
    [Bérard,Diekert,Gastin,Petit 1998] [Choffrut,Goldwurm 2000]
    [Bouyer,Dufourd,Fleury,Petit 2004] · · ·
  - model-checking of branching-time properties (TCTL, timed μ-calculus)
    [Alur,Courcoubetis,Dill 1993] [Laroussinie,Larsen,Weise 1995]
  - weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
    [Bouyer,Larsen,Markey,Rasmussen 2006] [Bouyer,Larsen,Markey 2007]
Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:
  - various extensions of timed automata
    [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
    [Bouyer, Dufourd, Fleury, Petit 2004] · · ·
  - model-checking of branching-time properties (TCTL, timed μ-calculus)
    [Alur, Courcoubetis, Dill 1993] [Laroussinie, Larsen, Weise 1995]
  - weighted/-priced timed automata (e.g. WCTL model-checking, optimal games)
    [Bouyer, Larsen, Markey, Rasmussen 2006] [Bouyer, Larsen, Markey 2007]
  - o-minimal hybrid systems
    [Lafferriere, Pappas, Sastry 2000] [Brihaye 2005]
Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:
  - various extensions of timed automata
    [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
    [Bouyer, Dufourd, Fleury, Petit 2004] …
  - model-checking of branching-time properties (TCTL, timed μ-calculus)
    [Alur, Courcoubetis, Dill 1993] [Laroussinie, Larsen, Weise 1995]
  - weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
    [Bouyer, Larsen, Markey, Rasmussen 2006] [Bouyer, Larsen, Markey 2007]
  - o-minimal hybrid systems
    [Lafferriere, Pappas, Sastry 2000] [Brihaye 2005]
  - …
Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:
  - various extensions of timed automata
    [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
    [Bouyer, Dufourd, Fleury, Petit 2004] · · ·
  - model-checking of branching-time properties (TCTL, timed $\mu$-calculus)
    [Alur, Courcoubetis, Dill 1993] [Laroussinie, Larsen, Weise 1995]
  - weighted/-priced timed automata (e.g. WCTL model-checking, optimal games)
    [Bouyer, Larsen, Markey, Rasmussen 2006] [Bouyer, Larsen, Markey 2007]
  - o-minimal hybrid systems
    [Lafferriere, Pappas, Sastry 2000] [Brihaye 2005]
  - · · ·

- Note however that it might be hard to prove there is a finite bisimulation quotient!
Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. How far can we extend the model and preserve decidability?
   - Hybrid systems
   - Smaller extensions of timed automata
   - An alternative way of proving decidability

5. Timed automata in practice

6. Conclusion
How far can we extend the model and preserve decidability?

Outline

1. Introduction
2. The timed automaton model
3. Timed automata, decidability issues
4. How far can we extend the model and preserve decidability?
   Hybrid systems
   Smaller extensions of timed automata
   An alternative way of proving decidability
5. Timed automata in practice
6. Conclusion
A general model: hybrid systems

What is a hybrid system?

- a discrete control (the mode of the system)
- a continuous evolution within a mode (given by variables)

A general model: hybrid systems

What is a hybrid system?

- a discrete control (the mode of the system)
- a continuous evolution within a mode (given by variables)

Example (The thermostat)

A simple thermostat, where $T$ (the temperature) depends on the time:

\[
\begin{align*}
&\text{Off} \\
&\dot{T} = -0.5T \\
&(T \geq 18)
\end{align*}
\]

\[
\begin{align*}
&T \leq 19 \\
&T \geq 21
\end{align*}
\]

\[
\begin{align*}
&\text{On} \\
&\dot{T} = 2.25 - 0.5T \\
&(T \leq 22)
\end{align*}
\]

The thermostat example

\[\begin{align*}
\text{Off} & \quad \dot{T} = -0.5T \\
& \quad (T \geq 18) \\
\text{On} & \quad \dot{T} = 2.25 - 0.5T \\
& \quad (T \leq 22)
\end{align*}\]

How far can we extend the model and preserve decidability?
The thermostat example

\[ \begin{align*}
\text{Off} \\
\dot{T} &= -0.5T \\
(T &\geq 18) \\
\text{On} \\
\dot{T} &= 2.25 - 0.5T \\
(T &\leq 22)
\end{align*} \]
Ok...

How far can we extend the model and preserve decidability?
Ok...

Easy...
Ok...

Easy...
Ok...

Easy...

Easy...
Ok... but?

How far can we extend the model and preserve decidability?
Ok... but?

How far can we extend the model and preserve decidability?
What about decidability?

∼ almost everything is undecidable

Negative results [HKPV95]

- The class of hybrid systems with clocks and only one variable having possibly two slopes $k_1 \neq k_2$ is undecidable.
- The class of stopwatch automata is undecidable.

Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. How far can we extend the model and preserve decidability?
   Hybrid systems
   Smaller extensions of timed automata
   An alternative way of proving decidability

5. Timed automata in practice

6. Conclusion
Role of diagonal constraints

\[ x - y \sim c \quad \text{and} \quad x \sim c \]
Role of diagonal constraints

\[ x - y \sim c \quad \text{and} \quad x \sim c \]

- **Decidability:** yes, using the region abstraction
Role of diagonal constraints

\[ x - y \sim c \quad \text{and} \quad x \sim c \]

- **Decidability:** yes, using the region abstraction

- **Expressiveness:** no additional expressive power
Role of diagonal constraints (cont.)

$c$ is positive

copy where $x - y \leq c$

$\leadsto$ proof in [BDGP98]

[BDGP98] Bérard, Diekert, Gastin, Petit. Characterization of the expressive power of silent transitions in timed automata (*Fundamenta Informaticae*).
Role of diagonal constraints (cont.)

Exercise [BC05]

Consider, for every positive integer $n$, the timed language:

$$\mathcal{L}_n = \{(a, t_1) \ldots (a, t_{2^n}) \mid 0 < t_1 < \cdots < t_{2^n} < 1\}$$

- Construct a timed automaton with diagonal constraints which recognizes $\mathcal{L}_n$. What is the size of this automaton?
- Idem without diagonal constraints. Can you do better?
- Conclude.

Adding silent actions

\[ g, \varepsilon, C := 0 \]

[BDGP98]
Adding silent actions

Decidability: yes

(actions have no influence on region automaton construction)
Adding silent actions

- **Decidability:** yes
  (actions have no influence on region automaton construction)

- **Expressiveness:** strictly more expressive!

$$x = 1, \ a, \ x := 0$$

$$x = 1, \ \varepsilon, \ x := 0$$
Adding additive constraints

$$x + y \sim c \quad \text{and} \quad x \sim c$$  \[BD00\]

[BD00] Bérard, Dufourd. Timed automata and additive clock constraints (Information Processing Letters).
Adding additive constraints

\[ x + y \sim c \quad \text{and} \quad x \sim c \quad [BD00] \]

- **Decidability**: for two clocks, *decidable* using the abstraction

![Graph showing the decidability for two clocks](image)

[BD00] Bérard, Dufourd. Timed automata and additive clock constraints (*Information Processing Letters*).
Adding additive constraints

\[ x + y \sim c \quad \text{and} \quad x \sim c \]  

**Decidability:** - for two clocks, **decidable** using the abstraction

- for four clocks (or more), **undecidable**!

**Expressiveness:** more expressive! (even using two clocks)

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

\[ \{(a^n, t_1 \ldots t_n) \mid n \geq 1 \text{ and } t_i = 1 - \frac{1}{2^i}\} \]

[BD00] Bérard, Dufourd. Timed automata and additive clock constraints (*Information Processing Letters*).
Undecidability proof

\[ \leadsto \text{simulation of} \quad \begin{align*}
&\quad \bullet \text{decrementation of a counter} \\
&\quad \bullet \text{incrementation of a counter}
\end{align*} \]

We will use 4 clocks:

- \(u\), “tic” clock (each time unit)
- \(x_0, x_1, x_2\): reference clocks for the two counters

\[ \begin{align*}
&\text{“}x_i\text{ reference for }c\text{”} \quad \equiv \\
&\quad \text{“the last time } x_i \text{ has been reset is} \\
&\quad \text{the last time action } c \text{ has been performed”}
\end{align*} \]
Undecidability proof (cont.)

- **Incrementation of counter $c$:**
  \[ x_0 \leq 2, \ u + x_2 = 1, \ c, \ x_2 := 0 \]
  
  \[ x_2 := 0 \]
  
  \[ u = 1, * \], \ $u := 0$ \[ \rightarrow \]
  
  \[ x_0 > 2, \ c, \ x_2 := 0 \]
  
  \[ u + x_2 = 1 \]
  
  ref for $c$ is $x_0$

  ref for $c$ is $x_2$

- **Decrementation of counter $c$:**
  \[ x_0 < 2, \ u + x_2 = 1, \ c, \ x_2 := 0 \]
  
  \[ x_2 := 0 \]
  
  \[ u = 1, * \], \ $u := 0$ \[ \rightarrow \]
  
  \[ x_0 = 2, \ c, \ x_2 := 0 \]
  
  \[ u + x_2 = 1 \]
  
  \[ u = 1, \ x_0 = 2, * \], \ $u := 0$, \ $x_2 := 0$ \[ \rightarrow \]
Adding constraints of the form $x + y \sim c$

- **Two clocks**: decidable using the abstraction

- **Four clocks (or more)**: undecidable!
Adding constraints of the form $x + y \sim c$

- **Two clocks:** decidable using the abstraction

- **Three clocks:** open question

- **Four clocks (or more):** undecidable!
Adding new operations on clocks

Several types of updates: \( x := y + c, x :< c, x :> c \), etc…
Adding new operations on clocks

**Several types of updates:** $x := y + c$, $x :< c$, $x :> c$, etc...

- The general model is **undecidable**.
  (simulation of a two-counter machine)
Adding new operations on clocks

Several types of updates: $x := y + c$, $x <: c$, $x :> c$, etc...

- The general model is **undecidable**.
  (simulation of a two-counter machine)

- Only decrementation also leads to undecidability

  **Incrementation of counter $x$**

  $z = 0 \rightarrow z = 1, z := 0 \rightarrow z = 0, y := y - 1$

  **Decrementation of counter $x$**

  $z = 0 \rightarrow x \geq 1 \rightarrow z = 0, x := x - 1$

  $x = 0$
Decidability

How far can we extend the model and preserve decidability?

The classical region automaton construction is not correct.

\[ y := 0 \quad y := 1 \quad x - y < 1 \]

image by \( y := 1 \)

\( \leadsto \) the bisimulation property is not met
Decidability (cont.)

\[ A \leadsto \text{Diophantine linear inequations system} \]
\[ \leadsto \text{is there a solution?} \]
\[ \leadsto \text{if yes, belongs to a decidable class} \]

Examples:

- constraint \( x \sim c \)

- constraint \( x - y \sim c \)

- update \( x :\sim y + c \)
  and for each clock \( z \), \( \max_{x,z} \geq \max_{y,z} + c \), \( \max_{z,x} \geq \max_{z,y} - c \)

- update \( x :< c \)
  and for each clock \( z \), \( \max_z \geq c + \max_{z,x} \)

The constants \((\max_x)\) and \((\max_{x,y})\) define a set of regions.
Decidability (cont.)

\[ y := 0 \quad y := 1 \quad x - y < 1 \]

\[
\begin{align*}
\max_y & \geq 0 \\
\max_x & \geq 0 + \max_{x,y} \\
\max_y & \geq 1 \\
\max_x & \geq 1 + \max_{x,y} \\
\max_{x,y} & \geq 1
\end{align*}
\]

implies
\[
\begin{align*}
\max_x & = 2 \\
\max_y & = 1 \\
\max_{x,y} & = 1 \\
\max_{y,x} & = -1
\end{align*}
\]

The **bisimulation property** is met.
What’s wrong when undecidable?

**Decrementation** $x := x - 1$

$$\max_x \leq \max_x - 1$$
What’s wrong when undecidable?

Decrementation $x := x - 1$

$max_x \leq max_x - 1$
What’s wrong when undecidable?

**Decrementation** $x := x - 1$

$\max_x \leq \max_x - 1$
What’s wrong when undecidable?

Decrementation $x := x - 1$

$max_x \leq max_x - 1$
What’s wrong when undecidable?

Decrementation $x := x - 1$

$max_x \leq max_x - 1$
What’s wrong when undecidable?

**Decrementation**  \( x := x - 1 \)

\[
\max_x \leq \max_x - 1
\]
What’s wrong when undecidable?

Decrementation $x := x - 1$

$max_x \leq max_x - 1$

etc...
Decidability (cont.)

<table>
<thead>
<tr>
<th></th>
<th>Diagonal-free constraints</th>
<th>General constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x := c, x := y$</td>
<td>PSPACE-complete</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>$x := x + 1$</td>
<td></td>
<td>Undecidable</td>
</tr>
<tr>
<td>$x := y + c$</td>
<td></td>
<td>Undecidable</td>
</tr>
<tr>
<td>$x := x - 1$</td>
<td></td>
<td>Undecidable</td>
</tr>
<tr>
<td>$x :&lt; c$</td>
<td></td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>$x :\geq c$</td>
<td></td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>$x :\sim y + c$</td>
<td></td>
<td>Undecidable</td>
</tr>
<tr>
<td>$y + c :&lt;: x :&lt; y + d$</td>
<td></td>
<td>Undecidable</td>
</tr>
<tr>
<td>$y + c :&lt;: x :&lt; z + d$</td>
<td></td>
<td>Undecidable</td>
</tr>
</tbody>
</table>

[BDFP00] Bouyer, Dufourd, Fleury, Petit. Updatable timed automata (*Theoretical Computer Science*).
Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. How far can we extend the model and preserve decidability?
   - Hybrid systems
   - Smaller extensions of timed automata
   - An alternative way of proving decidability

5. Timed automata in practice

6. Conclusion
The example of alternating timed automata

**Alternating timed automata** $\equiv$ ATA

[ LW05, OW05 ]

---

[LW05] Lasota, Walukiewicz. Alternating timed automata (*FoSSaCS’05*).

[ OW05 ] Ouaknine, Worrell. On the decidability of Metric Temporal Logic (*LICS’05*).
The example of alternating timed automata

\[ \text{Alternating timed automata} \equiv \text{ATA} \]

\[
\begin{align*}
\ell_0, a, \text{true} & \rightarrow \ell_0 \land (x := 0, \ell_1) \\
\ell_1, a, x \neq 1 & \rightarrow \ell_1 \\
\ell_1, a, x = 1 & \rightarrow \ell_2 \\
\ell_2, a, \text{true} & \rightarrow \ell_2
\end{align*}
\]

\[ \ell_0 \text{ initial state} \]
\[ \ell_0, \ell_1 \text{ final states} \]
\[ \ell_2 \text{ losing state} \]

“No two a’s are separated by 1 unit of time”

---

[\text{LW05}] Lasota, Walukiewicz. Alternating timed automata (FoSSaCS’05).

[\text{OW05}] Ouaknine, Worrell. On the decidability of Metric Temporal Logic (LICS’05).
The example of alternating timed automata

**Alternating timed automata** $\equiv$ ATA

**Example**

“No two $a$’s are separated by 1 unit of time”

$$
\begin{align*}
\ell_0, a, \text{true} & \quad \rightarrow \quad \ell_0 \land (x := 0, \ell_1) \\
\ell_1, a, x \neq 1 & \quad \rightarrow \quad \ell_1 \\
\ell_1, a, x = 1 & \quad \rightarrow \quad \ell_2 \\
\ell_2, a, \text{true} & \quad \rightarrow \quad \ell_2
\end{align*}
$$

- $\ell_0$ initial state
- $\ell_0, \ell_1$ final states
- $\ell_2$ losing state

[**LW05**] Lasota, Walukiewicz. Alternating timed automata (FoSSaCS’05).

[**OW05**] Ouaknine, Worrell. On the decidability of Metric Temporal Logic (LICS’05).
nice closure properties

[Sch02] Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity (Information Processing Letters).
nice closure properties

\( \leadsto \) universality is as difficult as reachability

[Sch02] Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity (Information Processing Letters).
- nice closure properties
- universality is as difficult as reachability
- more expressive than timed automata

[Sch02] Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity (Information Processing Letters).
nice closure properties

\[\rightsquigarrow\] universality is as difficult as reachability

more expressive than timed automata

**Theorem**

- Emptiness of ATA is undecidable.
- Emptiness of one-clock ATA is decidable, but non-primitive recursive.
- Emptiness for Büchi properties of one-clock ATA is undecidable.
- Emptiness of one-clock ATA with $\varepsilon$-transitions is undecidable.

[Sch02] Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity (*Information Processing Letters*).
nice closure properties

\(\leadsto\) universality is as difficult as reachability

more expressive than timed automata

**Theorem**

- Emptiness of ATA is undecidable.
- Emptiness of one-clock ATA is decidable, but non-primitive recursive.
- Emptiness for Büchi properties of one-clock ATA is undecidable.
- Emptiness of one-clock ATA with \(\varepsilon\)-transitions is undecidable.

**Lower bound:** simulation of a lossy channel system...

[Sch02] Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity (Information Processing Letters).
Example

How far can we extend the model and preserve decidability?
Execution over timed word \((a, .3) (a, .8) (a, 1.4) (a, 1.8) (a, 2)\)
Example

![Diagram of a finite state machine with states $l_0$, $l_1$, and $l_2$.]

Execution over timed word $(a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)$

$\{ (l_0, 0) \}$
Example

![Diagram](image)

**Execution over timed word** $(a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)$

$$\{ (\ell_0, 0) \}$$

$$\downarrow$$

$$\{ (\ell_0, .3), (\ell_1, 0) \}$$
Example

Execution over timed word \((a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)\)

\[
\begin{align*}
\{(l_0, 0)\} \\
\downarrow \quad \{(l_0, .3), (l_1, 0)\} \\
\downarrow \quad \{(l_0, .8), (l_1, 0), (l_1, .5)\}
\end{align*}
\]
Example

Execution over timed word \((a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)\)

\[
\begin{align*}
\{ (l_0, 0) \} \\
\{ (l_0, .3), (l_1, 0) \} \\
\{ (l_0, .8), (l_1, 0), (l_1, .5) \} \\
\{ (l_0, 1.4), (l_1, 0), (l_1, .6), (l_1, 1.1) \}
\end{align*}
\]
Example

Execution over timed word \((a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)\)

\[
\{ (l_0, 0) \} \\
\downarrow \\
\{ (l_0, .3), (l_1, 0) \} \\
\downarrow \\
\{ (l_0, .8), (l_1, 0), (l_1, .5) \} \\
\downarrow \\
\{ (l_0, 1.4), (l_1, 0), (l_1, .6), (l_1, 1.1) \} \\
\downarrow \\
\{ (l_0, 1.8), (l_1, 0), (l_1, .4), (l_2, 1), (l_1, 1.5) \}
\]
Example

Execution over timed word \((a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)\)

\[
\begin{align*}
\{ & (l_0, 0) \} \\
\downarrow \\
\{ & (l_0, .3), (l_1, 0) \} \\
\downarrow \\
\{ & (l_0, .8), (l_1, 0), (l_1, .5) \} \\
\downarrow \\
\{ & (l_0, 1.4), (l_1, 0), (l_1, .6), (l_1, 1.1) \} \\
\downarrow \\
\{ & (l_0, 1.8), (l_1, 0), (l_1, .4), (l_2, 1), (l_1, 1.5) \} \\
\downarrow \\
\{ & (l_0, 2), (l_1, 0), (l_1, .2), (l_1, .6), (l_2, 1.2), (l_1, 1.7) \}
\end{align*}
\]
An abstraction

A configuration = a finite set of pairs \((\ell, x)\)

\[(\ell, 0) \quad (\ell, 0.3) \quad (\ell, 1.2) \quad (\ell, 2.3) \quad (\ell', 0.4) \quad (\ell', 1) \quad (\ell', 0.8)\]
An abstraction

A configuration = a finite set of pairs $(\ell, x)$

\[
(\ell, 0) \quad (\ell, 0.3) \quad (\ell, 1.2) \quad (\ell, 2.3) \quad (\ell', 0.4) \quad (\ell', 1) \quad (\ell', 0.8)
\]

\[
\{(\ell, 0), (\ell', 1)\}
\]

0.0
An abstraction

A configuration = a finite set of pairs $(\ell, x)$

$\{(\ell, 0), (\ell, 0.3), (\ell, 1.2), (\ell, 2.3), (\ell', 0.4), (\ell', 1), (\ell', 0.8)\}$
An abstraction

A configuration = a finite set of pairs $(\ell, x)$

\[
\begin{align*}
(\ell, 0) & \quad (\ell, 0.3) & \quad (\ell, 1.2) & \quad (\ell, 2.3) & \quad (\ell', 0.4) & \quad (\ell', 1) & \quad (\ell', 0.8) \\
\{(\ell, 0), (\ell', 1)\} & \quad \{(\ell, 1)\} & \quad \{(\ell, 0), (\ell, 2)\} \\
0.0 & \quad 0.2 & \quad 0.3
\end{align*}
\]
An abstraction

A configuration = a finite set of pairs $(\ell, x)$
An abstraction

A configuration = a finite set of pairs \((\ell, x)\)

\[
\begin{align*}
(\ell, 0) & \quad (\ell, 0.3) & \quad (\ell, 1.2) & \quad (\ell, 2.3) & \quad (\ell', 0.4) & \quad (\ell', 1) & \quad (\ell', 0.8) \\
\{ (\ell, 0), (\ell', 1) \} & \quad \{ (\ell, 1) \} & \quad \{ (\ell, 0), (\ell, 2) \} & \quad \{ (\ell', 0) \} & \quad \{ (\ell', 0) \} \\
0.0 & \quad 0.2 & \quad 0.3 & \quad 0.4 & \quad 0.8
\end{align*}
\]
An abstraction

A configuration = a finite set of pairs \((\ell, x)\)

\[
\begin{align*}
  (\ell, 0) & \quad (\ell, 0.3) & \quad (\ell, 1.2) & \quad (\ell, 2.3) & \quad (\ell', 0.4) & \quad (\ell', 1) & \quad (\ell', 0.8) \\
  \{(\ell, 0), (\ell', 1)\} & \quad \{(\ell, 1)\} & \quad \{(\ell, 0), (\ell, 2)\} & \quad \{(\ell', 0)\} & \quad \{(\ell', 0)\}
\end{align*}
\]

0.0 \quad 0.2 \quad 0.3 \quad 0.4 \quad 0.8
An abstraction

A configuration = a finite set of pairs \((\ell, x)\)

\[
\begin{align*}
(\ell, 0) & \quad (\ell, 0.3) & \quad (\ell, 1.2) & \quad (\ell, 2.3) & \quad (\ell', 0.4) & \quad (\ell', 1) & \quad (\ell', 0.8) \\
\{(\ell, 0), (\ell', 1)\} & \quad \{(\ell, 1)\} & \quad \{(\ell, 0), (\ell, 2)\} & \quad \{(\ell', 0)\} & \quad \{(\ell', 0)\}
\end{align*}
\]

0.0 \quad 0.2 \quad 0.3 \quad 0.4 \quad 0.8

Abstracted into:

\[
\begin{align*}
\{(\ell, 0), (\ell', 1)\} & \quad \{(\ell, 1)\} & \quad \{(\ell, 0), (\ell, 2)\} & \quad \{(\ell', 0)\} & \quad \{(\ell', 0)\}
\end{align*}
\]
Abstract transition system

\{((\ell, 0), (\ell', 1))\} \cdot \{((\ell, 1))\} \cdot \{((\ell, 0), (\ell, 2))\} \cdot \{((\ell', 0))\} \cdot \{((\ell'), 0)\}
Abstract transition system

\{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\} \cdot \{(\ell', 0)\}

Time successors:
Abstract transition system

\[ \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\} \cdot \{(\ell', 0)\} \]

Time successors:

\[ \{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\} \]
Abstract transition system

\[
\{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\} \cdot \{(\ell', 0)\}
\]

Time successors:

\[
\{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\}
\]

\[
\{(\ell', 1)\} \cdot \{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\}
\]
Abstract transition system

{((l, 0), (l', 1))} · {((l, 1))} · {((l, 0), (l, 2))} · {((l', 0))} · {((l', 0))}

Time successors:

{((l', 1))} · {((l, 0), (l', 1))} · {((l, 1))} · {((l, 0), (l, 2))} · {((l', 0))}

{((l', 1))} · {((l', 1))} · {((l, 0), (l', 1))} · {((l, 1))} · {((l, 0), (l, 2))}

{((l, 1), (l, 3))} · {((l', 1))} · {((l', 1))} · {((l, 0), (l', 1))} · {((l, 1))}
Abstract transition system

Time successors:
Abstract transition system

Time successors:
Abstract transition system

\{(l, 0), (l', 1)\} \cdot \{(l, 1)\} \cdot \{(l, 0), (l, 2)\} \cdot \{(l', 0)\} \cdot \{(l', 0)\}

Time successors:

\{(l', 1)\} \cdot \{(l, 0), (l', 1)\} \cdot \{(l, 1)\} \cdot \{(l, 0), (l, 2)\} \cdot \{(l', 0)\}

\{(l', 1)\} \cdot \{(l', 1)\} \cdot \{(l, 0), (l', 1)\} \cdot \{(l, 1)\} \cdot \{(l, 0), (l, 2)\}

\{(l, 1), (l, 3)\} \cdot \{(l', 1)\} \cdot \{(l, 1)\} \cdot \{(l, 0), (l', 1)\} \cdot \{(l, 1)\}

\{(l, 2)\} \cdot \{(l, 1), (l, 3)\} \cdot \{(l', 1)\} \cdot \{(l', 1)\} \cdot \{(l, 0), (l', 1)\}

\{(l, 1), (l', 2)\} \cdot \{(l, 2)\} \cdot \{(l, 1), (l, 3)\} \cdot \{(l', 1)\} \cdot \{(l', 1)\}

Transition \( l \xrightarrow{x > 2, x = 0} l'' \) :
Abstract transition system

Time successors:

Transition $\ell \xrightarrow{x>2,x:=0} \ell''$: 

{(ℓ'', 0)} · {(ℓ, 1), (ℓ', 2)} · {(ℓ, 1)} · {(ℓ', 1)} · {(ℓ', 1)}
What can we do with that abstract transition system?

Correctness?
What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.
What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?
What can we do with that abstract transition system?

Correctness?
The previous abstraction is (almost) a **time-abstract bisimulation**.

Termination?

😊 possibly infinitely many abstract configurations
What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a **time-abstract bisimulation**.

Termination?

- ☹ possibly infinitely many abstract configurations
- ☀ there is a well-quasi ordering on the set of abstract configurations! (subword relation \(\sqsubseteq\))
What can we do with that abstract transition system?

Correctness?
The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

❌ possibly infinitely many abstract configurations

✅ there is a well-quasi ordering on the set of abstract configurations!
   (subword relation \( \sqsubseteq \))
   + downward compatibility:
     \[
     (\gamma_1 \sqsubseteq \gamma'_1 \text{ and } \gamma'_1 \sim \gamma'_2) \Rightarrow (\gamma_1 \sim^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma'_2)
     \]
What can we do with that abstract transition system?

Correctness?
The previous abstraction is (almost) a \textit{time-abstract bisimulation}.

Termination?

☐ possibly infinitely many abstract configurations

☐ there is a well-quasi ordering on the set of abstract configurations!
   (subword relation $\sqsubseteq$)
   + downward compatibility:
   \[
   (\gamma_1 \sqsubseteq \gamma'_1 \text{ and } \gamma'_1 \sim \gamma'_2) \Rightarrow (\gamma_1 \sim^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma'_2)
   \]
   + downward-closed objective (all states are accepting)
What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

😊 possibly infinitely many abstract configurations
😊 there is a well-quasi ordering on the set of abstract configurations!
   (subword relation $\sqsubseteq$)
   + downward compatibility:
     \[(\gamma_1 \sqsubseteq \gamma_1' \text{ and } \gamma_1' \rightsquigarrow \gamma_2') \Rightarrow (\gamma_1 \rightsquigarrow^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma_2')\]
   + downward-closed objective (all states are accepting)

A recipe:
What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

- possibly infinitely many abstract configurations
- there is a well-quasi ordering on the set of abstract configurations!
  (subword relation $\sqsubseteq$)
  + downward compatibility:
    \[(\gamma_1 \sqsubseteq \gamma_1' \text{ and } \gamma_1' \leadsto \gamma_2') \Rightarrow (\gamma_1 \leadsto^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma_2')\]
  + downward-closed objective (all states are accepting)

A recipe:

\[(\text{Higman's lemma } + \text{ Koenig's lemma}) \Rightarrow \text{termination}\]
What can we do with that abstract transition system?

Correctness?
The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

owo Possibly infinitely many abstract configurations

☑️ There is a well-quasi ordering on the set of abstract configurations!
  (subword relation \( \sqsubseteq \))
  + Downward compatibility:
    \[ (\gamma_1 \sqsubseteq \gamma'_1 \text{ and } \gamma'_1 \sim \gamma'_2) \Rightarrow (\gamma_1 \sim^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma'_2) \]
  + Downward-closed objective (all states are accepting)

A recipe:

\[ (\text{Higman's lemma + Koenig's lemma}) \Rightarrow \text{termination} \]

Alternative

The abstract transition system can be simulated by a kind of FIFO channel machine.
A digression on timed automata

How far can we extend the model and preserve decidability?
A digression on timed automata

\[ x, y \in r_0, \{y\} < \{x\} \]

\[(y, r_0) \cdot (x, r_0)\]
A digression on timed automata

\[ x \in r_1, y \in r_0, \{x\} < \{y\} \]

\[(x, r_1) \cdot (y, r_0)\]
A digression on timed automata

\[ x, y \in r_1, \{y\} < \{x\} \]

\[(y, r_1) \cdot (x, r_1)\]
A digression on timed automata

The classical region automaton can be simulated by a channel machine (with a single bounded channel).
Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata

[Abdulla, Jonsson 1998]
Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata
- timed Petri nets

[Abdulla, Jonsson 1998]
[Abdulla, Nylén 2001]
Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata  
  [Abdulla, Jonsson 1998]
- timed Petri nets  
  [Abdulla, Nylén 2001]
- MTL model checking  
  [Ouaknine, Worrell 2005, 2007]
Partial conclusion

Similar technics apply to:
- networks of single-clock timed automata [Abdulla, Jonsson 1998]
- timed Petri nets [Abdulla, Nylén 2001]
- MTL model checking [Ouaknine, Worrell 2005, 2007]
- coFlatMTL model checking [Bouyer, Markey, Ouaknine, Worrell 2007]
  (using channel machines with a bounded number of cycles)
Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata  [Abdulla, Jonsson 1998]
- timed Petri nets  [Abdulla, Nylén 2001]
- MTL model checking  [Ouaknine, Worrell 2005, 2007]
- coFlatMTL model checking  [Bouyer, Markey, Ouaknine, Worrell 2007]
  (using channel machines with a bounded number of cycles)
- single-clock automata inclusion checking  [Ouaknine, Worrell 2004]
Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata [Abdulla, Jonsson 1998]
- timed Petri nets [Abdulla, Nylén 2001]
- MTL model checking [Ouaknine, Worrell 2005, 2007]
- coFlatMTL model checking [Bouyer, Markey, Ouaknine, Worrell 2007]
  (using channel machines with a bounded number of cycles)
- single-clock automata inclusion checking [Ouaknine, Worrell 2004]
- ...
Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. How far can we extend the model and preserve decidability?
   - Hybrid systems
   - Smaller extensions of timed automata
   - An alternative way of proving decidability

5. Timed automata in practice

6. Conclusion
What about the practice?

- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

- Need of a symbolic representation:
  Finite representation of infinite sets of configurations
What about the practice?

- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

- Need of a symbolic representation:
  
  Finite representation of infinite sets of configurations

- in the plane, a line represented by two points.
What about the practice?

- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

- Need of a symbolic representation:
  
  Finite representation of infinite sets of configurations

  - in the plane, a line represented by two points.
  - set of words $aa, aaaa, aaaaaa...$
    represented by a rational expression $aa(aa)^*$
What about the practice?

- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

- Need of a symbolic representation:
  
  **Finite representation of infinite sets of configurations**

- in the plane, a line represented by two points.

- set of words \( aa, aaaa, aaaaaa... \) represented by a rational expression \( aa(aa)^* \)

- set of integers, represented using **semi-linear** sets
What about the practice?

- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

- Need of a symbolic representation:
  - Finite representation of infinite sets of configurations

- in the plane, a line represented by two points.
- set of words \textit{aa, aaaa, aaaaaa...} represented by a rational expression \textit{aa(aa)*}
- set of integers, represented using \textit{semi-linear} sets
- sets of constraints, \textit{polyhedra}, zones, regions
What about the practice?

- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

- Need of a symbolic representation:
  
  Finite representation of infinite sets of configurations

- in the plane, a line represented by two points.
- set of words \( aa, aaaa, aaaaaa \ldots \) represented by a rational expression \( aa(aa)^* \)
- set of integers, represented using semi-linear sets
- sets of constraints, polyhedra, zones, regions
- BDDs, DBMs (see later), CDDs, etc...
What about the practice?

- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

- Need of a symbolic representation:
  - Finite representation of infinite sets of configurations
  - in the plane, a line represented by two points.
  - set of words \textit{aa}, \textit{aaaa}, \textit{aaaaaa}... represented by a rational expression \textit{aa(aa)}^*
  - set of integers, represented using \textit{semi-linear} sets
  - sets of constraints, \textit{polyhedra}, zones, regions
  - BDDs, DBMs (see later), CDDs, etc...

- Need of abstractions, heuristics, etc...
An example of computation with HyTech

command: /usr/local/bin/hytech gas_burner

================================================================================
HyTech: symbolic model checker for embedded systems
Version 1.04f (last modified 1/24/02) from v1.04a of 12/6/96
For more info:
    email: hytech@eecs.berkeley.edu
    http://www.eecs.berkeley.edu/~tah/HyTech
Warning: Input has changed from version 1.00(a). Use -i for more info
================================================================================
Backward computation
Number of iterations required for reachability: 6
System satisfies non-leaking duration property

Location: not_leaking
x >= 0 & t >= 3 & y <= 20t & y >= 0
| x + 20t >= y + 11 & y <= 20t + 19 & t >= 2 & x >= 0 & y >= 0
| y >= 0 & t >= 1 & x + 20t >= y + 22 & y <= 20t + 8 & x >= 0
| y >= 0 & x + 20t >= y + 33 & 20t >= y + 3 & x >= 0

Location: leaking
19x + y <= 20t + 19 & y >= x + 59 & x <= 1 & x >= 0
| t >= x + 2 & x <= 1 & y >= 0 & 19x + y <= 20t + 19 & x >= 0
| t >= x + 1 & x <= 1 & y >= 0 & 19x + y <= 20t + 8 & x >= 0
| 20t >= 19x + y + 3 & y >= 0 & x <= 1 & x >= 0

================================================================================
Max memory used = 0 pages = 0 bytes = 0.00 MB
Time spent = 0.02u + 0.00s = 0.02 sec total
================================================================================
Zones: A symbolic representation for timed systems

Example of a zone and its DBM representation

\[ Z = (x_1 \geq 3) \land (x_2 \leq 5) \land (x_1 - x_2 \leq 4) \]

DBM: Difference Bound Matrice [BM83,Dill89]

[BM83] Berthomieu, Menasche. An enumerative approach for analyzing time Petri nets World Comuter Congress.
Zones: A symbolic representation for timed systems

Example of a zone and its DBM representation

\[ Z = (x_1 \geq 3) \land (x_2 \leq 5) \land (x_1 - x_2 \leq 4) \]

DBM: Difference Bound Matrice [BM83,Dill89]

[BM83] Berthomieu, Menasche. An enumerative approach for analyzing time Petri nets World Comupter Congress.
Backward computation
Backward computation
Backward computation
Backward computation

Init

Final
Backward computation
Note on the backward analysis of TA

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \]

\(Z\)

\(g, a, C := 0\)
Note on the backward analysis of TA

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \]

\[ Z \]
Note on the backward analysis of TA

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \]

\[ Z \]

\[ g, \ a, \ C := 0 \]
Note on the backward analysis of TA

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \]

\[ g, \ a, \ C := 0 \]

\[ Z \]

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \]

\[ Z \]
Note on the backward analysis of TA

\[ g, \ a, \ C := 0 \]

\[ Z \]

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \]

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \]

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \]
Note on the backward analysis of TA

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \]

\( g, \ a, \ C := 0 \)

😊 the backward computation always terminates!
😊😊 ... and it is correct!!!
Note on the backward analysis (cont.)

If $\mathcal{A}$ is a timed automaton, we construct its corresponding set of regions.

Because of the bisimulation property, we get that:

“Every set of valuations which is computed along the backward computation is a finite union of regions”
Note on the backward analysis (cont.)

If $\mathcal{A}$ is a timed automaton, we construct its corresponding set of regions.

Because of the bisimulation property, we get that:

“Every set of valuations which is computed along the backward computation is a finite union of regions”

Let $R$ be a region. Assume:

1. $v \in \overrightarrow{R}$ (for ex. $v + t \in R$)
2. $v' \equiv_{\text{reg.}} v$

There exists $t'$ s.t. $v' + t' \equiv_{\text{reg.}} v + t$, which implies that $v' + t' \in R$ and thus $v' \in \overrightarrow{R}$. 
Note on the backward analysis (cont.)

If $\mathcal{A}$ is a timed automaton, we construct its corresponding set of regions.

Because of the bisimulation property, we get that:

“All set of valuations which is computed along the backward computation is a finite union of regions”

**But**, the backward computation is not so nice, when also dealing with integer variables...

$$i := j \cdot k + \ell \cdot m$$
Forward computation
Forward computation
Forward computation
Forward computation
Forward computation
Forward analysis of timed automata

\[ g, a, C := 0 \]

zones \( Z \)

\[ [C \leftarrow 0](\bar{Z} \cap g) \]
Forward analysis of timed automata

\[ g, \ a, \ C := 0 \]

zones \( Z \)

\[ [C \leftarrow 0](\bar{Z} \cap g) \]
Forward analysis of timed automata

\[ g,\ a,\ C := 0 \]

zones \[ Z \]

\[ [C \leftarrow 0](\vec{Z} \cap g) \]

\[ Z \]

\[ \vec{Z} \]
Forward analysis of timed automata

$g, a, C := 0$

zones $Z$

$[C \leftarrow 0](\overrightarrow{Z} \cap g)$

$Z$

$\overrightarrow{Z}$

$\overrightarrow{Z} \cap g$
Forward analysis of timed automata

$g, \ a, \ C := 0$

zones

$Z$

$[C \leftarrow 0](\overrightarrow{Z} \cap g)$

$\overrightarrow{Z}$

$\overrightarrow{Z} \cap g$

$[y \leftarrow 0](\overrightarrow{Z} \cap g)$
Forward analysis of timed automata

\[ g, \ a, \ C := 0 \]

zones \[ Z \]

\[ [C \leftarrow 0](\overrightarrow{Z} \cap g) \]

\[ Z \]

\[ \overrightarrow{Z} \]

\[ \overrightarrow{Z} \cap g \]

\[ [y \leftarrow 0](\overrightarrow{Z} \cap g) \]

⚠️ the forward computation may not terminate...
Non termination of the forward analysis

\[ y := 0, \]
\[ x := 0 \]
\[ x \geq 1 \land y = 1, \]
\[ y := 0 \]
Non termination of the forward analysis

\[ y := 0, \]
\[ x := 0 \]
\[ x \geq 1 \land y = 1, \]
\[ y := 0 \]
Non termination of the forward analysis

\[ y := 0, \]
\[ x := 0 \]
\[ x \geq 1 \land y = 1, \]
\[ y := 0 \]
Non termination of the forward analysis

\[ y := 0, \]
\[ x := 0 \]
\[ x \geq 1 \land y = 1, \]
\[ y := 0 \]
Non termination of the forward analysis

\[ y := 0, \]
\[ x := 0 \]
\[ x \geq 1 \land y = 1, \]
\[ y := 0 \]

\[ \leadsto \] an infinite number of steps...
“Solutions” to this problem

(f.ex. in [Larsen, Pettersson, Yi 1997] or in [Daws, Tripakis 1998])

- **inclusion checking**: if $Z \subseteq Z'$ and $Z'$ already considered, then we don’t need to consider $Z$

  $\leadsto$ correct w.r.t. reachability
“Solutions” to this problem

(f.ex. in [Larsen, Pettersson, Yi 1997] or in [Daws, Tripakis 1998])

- **inclusion checking**: if $Z \subseteq Z'$ and $Z'$ already considered, then we don't need to consider $Z$
  \[ \leadsto \text{correct w.r.t. reachability} \]

- **activity**: eliminate redundant clocks
  \[ \text{[Daws, Yovine 1996]} \]
  \[ \leadsto \text{correct w.r.t. reachability} \]

\[ q \xrightarrow{g,a,C:=0} q' \quad \text{implies} \quad \text{Act}(q) = \text{clocks}(g) \cup (\text{Act}(q') \setminus C) \]

\[ \ldots \]
“Solutions” to this problem (cont.)

- **convex-hull approximation**: if $Z$ and $Z'$ are computed then we overapproximate using “$Z \sqcup Z'$”.

  $\leadsto$ “semi-correct” w.r.t. reachability
“Solutions” to this problem (cont.)

- **convex-hull approximation**: if $Z$ and $Z'$ are computed then we overapproximate using \( Z \uplus Z' \).

  \( \leadsto \) “semi-correct” w.r.t. reachability

- **extrapolation**, an abstraction operator on zones
An abstraction: the extrapolation operator

$\text{Approx}_2(Z)$: “the smallest zone containing $Z$ that is defined only with constants no more than 2”

\[
\begin{pmatrix}
0 & -3 & 0 \\
9 & 0 & 4 \\
5 & 2 & 0
\end{pmatrix}
\]

$\leadsto$ The extrapolation operator ensures termination of the computation!
An abstraction: the extrapolation operator

Approx_2(Z): “the smallest zone containing Z that is defined only with constants no more than 2”

\[
\begin{pmatrix}
0 & -3 & 0 \\
9 & 0 & 4 \\
5 & 2 & 0 \\
\end{pmatrix}
\xrightarrow{\text{Approx}_2}
\begin{pmatrix}
0 & -2 & 0 \\
\infty & 0 & \infty \\
\infty & 2 & 0 \\
\end{pmatrix}
\]

\[\leadsto\] The extrapolation operator ensures termination of the computation!
Classical algorithm, focus on correctness

Challenge

Choose a good constant for the extrapolation so that the forward computation is correct. Classical algorithm, the choice goes to the maximal constant.

Classical algorithm, focus on correctness

Challenge

Choose a good constant for the extrapolation so that the forward computation is correct. Classical algorithm, the choice goes to the maximal constant.

- Implemented in tools like Uppaal, Kronos, RT-Spin...
- Successfully used on many real-life examples

Classical algorithm, focus on correctness

Challenge
Choose a **good** constant for the extrapolation so that the forward computation is correct. Classical algorithm, the choice goes to the maximal constant.

- Implemented in tools like Uppaal, Kronos, RT-Spin...
- Successfully used on many real-life examples

Theorem
The classical algorithm is correct for diagonal-free timed automata.

[Bou03] Bouyer. Untameable timed automata! (*STACS’03*).
Classical algorithm, focus on correctness

Challenge

Choose a good constant for the extrapolation so that the forward computation is correct. Classical algorithm, the choice goes to the maximal constant.

- Implemented in tools like Uppaal, Kronos, RT-Spin...
- Successfully used on many real-life examples

Theorem

The classical algorithm is correct for diagonal-free timed automata.

This theorem does not extend to timed automata using diagonal clock constraints... [Bou03,Bou04]

A problematic automaton

\[ x_2 - x_1 > 2 \]
\[ x_4 - x_3 < 2 \]

\[ x_1 = 3 \]
\[ x_1 := 0 \]

\[ x_2 = 2 \]
\[ x_2 := 0 \]

\[ x_1 = 2, \ x_1 := 0 \]
\[ x_2 = 2, \ x_2 := 0 \]

The loop

Error
A problematic automaton

\begin{align*}
\nu(x_1) &= 0 \\
\nu(x_2) &= d \\
\nu(x_3) &= 2\alpha + 5 \\
\nu(x_4) &= 2\alpha + 5 + d
\end{align*}

\begin{itemize}
  \item \( x_2 - x_1 > 2 \)
  \item \( x_4 - x_3 < 2 \)
  \item \( x_2 = 2, \ x_2 := 0 \)
  \item \( x_1 = 2, \ x_1 := 0 \)
\end{itemize}
A problematic automaton

\begin{align*}
    &\begin{cases}
        v(x_1) = 0 \\
        v(x_2) = d \\
        v(x_3) = 2\alpha + 5 \\
        v(x_4) = 2\alpha + 5 + d
    \end{cases} \\
    &x_1 = 0 \quad x_1 := 0 \\
    &x_2 = 2 \quad x_2 := 0 \\
    &x_3 \leq 3
\end{align*}

The loop

Error
The problematic zone

\[ x_1 - x_2 = x_3 - x_4. \]
The problematic zone

If $\alpha$ is sufficiently large, after extrapolation:

It does not imply $x_1 - x_2 = x_3 - x_4$.

It implies $x_1 - x_2 = x_3 - x_4$. 

\[ [1; 3] \quad [2\alpha + 5] \]

\[ [2\alpha + 2; 2\alpha + 4] \quad [2\alpha + 6; 2\alpha + 8] \]

\[ x_1 \quad x_2 \quad x_3 \quad x_4 \]
The problematic zone

If $\alpha$ is sufficiently large, after extrapolation:

Hence, any choice of constant is erroneous!
General abstractions

Criteria for a good abstraction operator Abs:
General abstractions

Criteria for a good abstraction operator $\text{Abs}$:

- easy computation

$\text{Abs}(Z)$ is a zone if $Z$ is a zone

[Effectiveness]
General abstractions

Criteria for a good abstraction operator $\text{Abs}$:

- easy computation
  $\text{Abs}(Z)$ is a zone if $Z$ is a zone
- finiteness of the abstraction
  $\{\text{Abs}(Z) \mid Z \text{ zone}\}$ is finite
Criteria for a good abstraction operator $\text{Abs}$:

- easy computation
  \[ \text{Abs}(Z) \text{ is a zone if } Z \text{ is a zone} \]

- finiteness of the abstraction
  \[ \{ \text{Abs}(Z) \mid Z \text{ zone} \} \text{ is finite} \]

- completeness of the abstraction
  \[ Z \subseteq \text{Abs}(Z) \]
General abstractions

**Criteria for a good abstraction operator** **Abs:**

- easy computation
  - \( \text{Abs}(Z) \) is a zone if \( Z \) is a zone
- finiteness of the abstraction
  - \( \{ \text{Abs}(Z) \mid Z \text{ zone} \} \) is finite
- completeness of the abstraction
  - \( Z \subseteq \text{Abs}(Z) \)
- soundness of the abstraction
  - the computation of \( (\text{Abs} \circ \text{Post})^* \) is correct w.r.t. reachability
General abstractions

Criteria for a good abstraction operator $\text{Abs}$:

- easy computation
  $\text{Abs}(Z)$ is a zone if $Z$ is a zone

- finiteness of the abstraction
  $\{\text{Abs}(Z) \mid Z \text{ zone}\}$ is finite

- completeness of the abstraction
  $Z \subseteq \text{Abs}(Z)$

- soundness of the abstraction
  the computation of $(\text{Abs} \circ \text{Post})^*$ is correct w.r.t. reachability

For the previous automaton,

no abstraction operator can satisfy all these criteria!
Why that?

Assume there is a “nice” operator $\text{Abs}$.

The set $\{M \text{ DBM representing a zone } \text{Abs}(Z)\}$ is finite.

$\leadsto k$ the max. constant defining one of the previous DBMs

We get that, for every zone $Z$,

$$Z \subseteq \text{Extra}_k(Z) \subseteq \text{Abs}(Z)$$
Problem!

**Open questions:**  
- which conditions can be made weaker?  
- find a clever termination criterium?  
- use an other data structure than zones/DBMs?
Improving the classical algorithm

- the extrapolation operator can be made coarser:
  - local extrapolation constants [BBFL03];
  - distinguish between lower- and upper-bounded contraints [BBLP03,BBLP06]

[BBFL03] Behrmann, Bouyer, Fleury, Larsen. Static Guard Analysis in Timed Automata Verification (TACAS’03).
[BBLP04] Behrmann, Bouyer, Larsen, Pelánek. Lower and Upper Bounds in Zone Based Abstractions of Timed Automata (TACAS’04).
[HBL+03] Hendriks, Behrmann, Larsen, Niebert, Vaandrager. Adding symmetry reduction to Uppaal (FORMATS’03).
[DHLP06] David, Håkansson, Larsen, Pettersson. Model checking timed automata with priorities using DBM subtraction (FORMATS'06).
Improving the classical algorithm

- the extrapolation operator can be made coarser:
  - local extrapolation constants [BBFL03];
  - distinguish between lower- and upper-bounded contraints [BBLP03,BBLP06];

- heuristics can be added
  - order for exploration
  - symmetry reduction [HBL+03]

---

[BBFL03] Behrmann, Bouyer, Fleury, Larsen. Static Guard Analysis in Timed Automata Verification (TACAS’03).
[BBLP04] Behrmann, Bouyer, Larsen, Pelánek. Lower and Upper Bounds in Zone Based Abstractions of Timed Automata (TACAS’04).
[HBL+03] Hendriks, Behrmann, Larsen, Niebert, Vaandrager. Adding symmetry reduction to Uppaal (FORMATS’03).
[DHLP06] David, Håkansson, Larsen, Pettersson. Model checking timed automata with priorities using DBM subtraction (FORMATS’06).
Improving the classical algorithm

- the extrapolation operator can be made coarser:
  - local extrapolation constants [BBFL03];
  - distinguish between lower- and upper-bounded contraints [BBLP03,BBLP06]

- heuristics can be added
  - order for exploration
  - symmetry reduction [HBL+03]

- the representation of zones can be improved [DHL06]

---

[BBFL03] Behrmann, Bouyer, Fleury, Larsen. Static Guard Analysis in Timed Automata Verification (TACAS’03).
[BBLP04] Behrmann, Bouyer, Larsen, Pelánek. Lower and Upper Bounds in Zone Based Abstractions of Timed Automata (TACAS’04).
[HBL+03] Hendriks, Behrmann, Larsen, Niebert, Vaandrager. Adding symmetry reduction to Uppaal (FORMATS’03).
[DHL06] David, Häkansson, Larsen, Pettersson. Model checking timed automata with priorities using DBM subtraction (FORMATS’06).
Improving the classical algorithm

- the extrapolation operator can be made coarser:
  - local extrapolation constants [BBFL03];
  - distinguish between lower- and upper-bounded contraints [BBLP03,BBLP06]
- heuristics can be added
  - order for exploration
  - symmetry reduction [HBL+03]
- the representation of zones can be improved [DHL06]

→ the tool Uppaal is under development since 1995...
Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. How far can we extend the model and preserve decidability?
   - Hybrid systems
   - Smaller extensions of timed automata
   - An alternative way of proving decidability

5. Timed automata in practice

6. Conclusion
Conclusion

- Justification of the dense-time paradigm
- Several technics for proving decidability of real-time systems
  - finite time-abstract bisimulation
  - well-quasi-order on the time-abstract transition system
- Timed automata are implemented in several model checking tools
  - Other timed models have been developed and have concurrent tools: for instance Romeo and Tina for time Petri nets
Conclusion

- Justification of the dense-time paradigm
- Several technics for proving decidability of real-time systems
  - finite time-abstract bisimulation
  - well-quasi-order on the time-abstract transition system
- Timed automata are implemented in several model checking tools
  - Other timed models have been developed and have concurrent tools: for instance Romeo and Tina for time Petri nets

Some current streams of research in timed systems:

- quantitative model-checking,
- real-time logics,
- robustness, implementability issues,
- timed games,
- modelling of resources,
- ...
Conclusion

- Justification of the dense-time paradigm
- Several technics for proving decidability of real-time systems
  - finite time-abstract bisimulation
  - well-quasi-order on the time-abstract transition system
- Timed automata are implemented in several model checking tools
  - Other timed models have been developed and have concurrent tools: for instance Romeo and Tina for time Petri nets

Some current streams of research in timed systems:
- quantitative model-checking,
- real-time logics,
- robustness, implementability issues,
- timed games,
- modelling of resources,
- ...
Modelling and analyzing resources in timed systems

Patricia Bouyer-Decitre

LSV, CNRS & ENS Cachan, France
Outline

1. Introduction

2. Modelling and optimizing resources in timed systems

3. Managing resources

4. Conclusion
A starting example
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

\[
\begin{align*}
&\text{done, } 22 \leq y \leq 25 \\
&\text{repair, } x \leq 15 \\
&\text{repair, } 2 \leq y \wedge x \leq 56 \\
&\text{delayed, } y := 0 \\
&\text{done, } 22 \leq y \leq 25
\end{align*}
\]
An example of a timed automaton

\begin{align*}
    &\text{safe} \\
    &x = 0 \\
    &y = 0
\end{align*}
An example of a timed automaton

\[
\begin{align*}
\text{safe} & \xrightarrow{23} \text{safe} \\
X &= 0, \quad 23 \\
y &= 0, \quad 23
\end{align*}
\]
An example of a timed automaton

The automaton has states labeled as "safe", "alarm", "repairing", and "failsafe". The transitions between states are as follows:

- From "safe" to "alarm" with the condition \( x = 0 \) and \( y = 0 \) and the guard \( 15 \leq y \leq 16 \).
- From "alarm" to "repairing" with the condition \( 2 \leq y \wedge x \leq 56 \) and the guard \( y = 0 \).
- From "repairing" to "done" with the condition \( 22 \leq y \leq 25 \).
- From "done" to "safe" with the condition \( 15 \leq x \leq 16 \).
- From "failsafe" to "repairing" with the condition \( 2 \leq y \wedge x \leq 56 \) and the guard \( y = 0 \).
- From "safe" to "failsafe" with the condition \( 15 \leq x \leq 16 \) and the guard \( y = 0 \).

The table below shows the transitions with their corresponding inputs and outputs:

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td>23</td>
<td>X 0</td>
<td>23</td>
</tr>
<tr>
<td>safe</td>
<td>problem</td>
<td>y 0</td>
<td>0</td>
</tr>
</tbody>
</table>

The equation is:
\[
x := 0, \quad y := 0, \quad 15 \leq y \leq 16, \quad 2 \leq y \wedge x \leq 56, \quad x := 0, \quad 15 \leq x \leq 16.
\]
An example of a timed automaton

Introduction

An example of a timed automaton
An example of a timed automaton

**State Transitions:**
- **Safe:**
  - $x := 0$
  - $y := 0$
- **Problem:**
  - $x := 0$
- **Alarm:**
  - $y = 0$
  - $15 \leq x \leq 16$
  - $2 \leq y \leq x \leq 56$
- **Repair:**
  - $y := 0$
- **Failsafe:**
  - $y := 0$
  - $22 \leq y \leq 25$

**Timed Transitions:**
- **Safe to Safe:**
  - $23$
- **Problem to Alarm:**
  - $23$
  - $0$
- **Alarm to Alarm:**
  - $15.6$
- **Alarm to Failsafe:**
  - $15.6$

**Tables:**

<table>
<thead>
<tr>
<th>State</th>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Problem</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>Alarm</td>
<td>0</td>
<td>15.6</td>
</tr>
<tr>
<td>Repair</td>
<td>0</td>
<td>15.6</td>
</tr>
<tr>
<td>Failsafe</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
An example of a timed automaton

\[
\begin{align*}
\text{safe} & \xrightarrow{23} \text{safe} \quad \text{problem, } x := 0 \\
\text{alarm} & \quad \text{repair, } x \leq 15 \\
\text{repairing} & \quad \text{repair, } y := 0 \\
\text{failsafe} & \quad \text{delayed, } y := 0 \\
\end{align*}
\]

\[
\begin{array}{c|c|c|c|c}
\text{safe} & \text{problem} & \text{alarm} & \text{repair} & \text{failsafe} \\
\hline
x & 0 & 23 & 0 & 15.6 \\
y & 0 & 23 & 23 & 38.6 \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{failsafe} & \text{delayed} \\
\hline
\cdots & 15.6 \\
0 & 0 \\
\end{array}
\]
An example of a timed automaton
An example of a timed automaton

\[
\begin{align*}
\text{safe} & \xrightarrow{\text{problem}, \ x=0} \text{alarm} \\
\text{alarm} & \xrightarrow{\text{repair, } x \leq 15} \text{repairing} \\
\text{repairing} & \xrightarrow{\text{delayed, } y=0} \text{failsafe} \\
\text{failsafe} & \xrightarrow{\text{done, } 22 \leq y \leq 25} \text{safe}
\end{align*}
\]

\[
\begin{array}{cccccccc}
\text{safe} & \xrightarrow{23} & \text{safe} & \xrightarrow{\text{problem}} & \text{alarm} & \xrightarrow{15.6} & \text{alarm} & \xrightarrow{\text{delayed}} & \text{failsafe} \\
X & 0 & 23 & 0 & 15.6 & \text{...} & 15.6 & \text{...} \\
Y & 0 & 23 & 23 & 38.6 & 0 & \text{...} & \text{...} \\
\text{failsafe} & \xrightarrow{2.3} & \text{failsafe} & \xrightarrow{\text{repair}} & \text{repairing} & \xrightarrow{22.1} & \text{repairing} \\
\text{...} & 15.6 & 17.9 & 17.9 & 40 & \text{...} & 22.1 & \text{...} \\
\text{...} & 0 & 2.3 & 0 & 22.1 & \text{...} & \text{...} & \text{...}
\end{array}
\]
An example of a timed automaton
Timed automata

Theorem [AD90,CY92]

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

[AD90] Alur, Dill. Automata for modeling real-time systems (ICALP’90).
Theorem [AD90, CY92]

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

The region abstraction
The region abstraction

- “compatibility” between regions and constraints
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
The region abstraction

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

\[ \sim \text{ an equivalence of finite index} \]
\[ \text{a time-abstract bisimulation} \]
The region abstraction
Outline

1. Introduction

2. Modelling and optimizing resources in timed systems

3. Managing resources

4. Conclusion
Modelling resources in timed systems

- System resources might be relevant and even crucial information
Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - price to pay,
  - bandwidth,
  - ...

Theorem [HKPV95]
The reachability problem is undecidable in hybrid automata.

An alternative: weighted/priced timed automata [ALP01, BFH+01]

Hybrid variables do not constrain the system
Hybrid variables are observer variables
Modelling resources in timed systems

- System **resources** might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - price to pay,
  - bandwidth,
  - ...

  → timed automata are not powerful enough!
Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - price to pay,
  - bandwidth,
  - ...

  \(\Rightarrow\) timed automata are not powerful enough!

- A possible solution: use hybrid automata
Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - price to pay,
  - bandwidth,
  - ...

  \(\implies\) timed automata are not powerful enough!

- A possible solution: use hybrid automata

The thermostat example

\[
\begin{align*}
\text{Off} & \quad \dot{T} = -0.5T \\
& \quad (T \geq 18) \\
\text{On} & \quad \dot{T} = 2.25 - 0.5T \\
& \quad (T \leq 22)
\end{align*}
\]
Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - price to pay,
  - bandwidth,
  - ...

→ timed automata are not powerful enough!

- A possible solution: use hybrid automata

The thermostat example

\[
\begin{align*}
\text{Off} & : \dot{T} = -0.5 T \\
& \quad (T \geq 18) \\
\text{On} & : \dot{T} = 2.25 - 0.5 T \\
& \quad (T \leq 22)
\end{align*}
\]
Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - price to pay,
  - bandwidth,
  - ...

  \[ \leadsto \text{timed automata are not powerful enough!} \]

- A possible solution: use hybrid automata

**Theorem** [HKPV95]

The reachability problem is **undecidable** in hybrid automata.

---

Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - price to pay,
  - bandwidth,
  - ...

  \[ \Rightarrow \text{timed automata are not powerful enough!} \]

- A possible solution: use hybrid automata

**Theorem** [HKPV95]
The reachability problem is **undecidable** in hybrid automata.

- An alternative: weighted/priced timed automata [ALP01,BFH+01]
  \[ \Rightarrow \text{hybrid variables do not constrain the system} \]
  \[ \Rightarrow \text{hybrid variables are observer variables} \]
A third model of the system
How much fuel will I use?

It is a quantitative (optimization) problem in a priced timed automaton: at least 68 anti-planet units!
Weighted/priced timed automata [ALP01,BFH+01]

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

Weighted/priced timed automata [ALP01,BFH+01]

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

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

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC'01).
Weighted/priced timed automata [ALP01,BFH+01]

\[
\ell_0 \xrightarrow{5} \ell_1 \xrightarrow{\ell_1} \ell_2 \xrightarrow{+10} \ell_3 \xrightarrow{+1} \ell_4
\]

\[
\ell_0 \xrightarrow{1.3} \ell_0 \xrightarrow{c} \ell_1 \xrightarrow{u} \ell_3 \xrightarrow{0.7} \ell_3 \xrightarrow{c} \ell_4
\]

Cost:

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c}
\ell_0 & \ell_0 & \ell_1 & \ell_3 & \ell_3 & \ell_4 \\
\hline
x & 0 & 1.3 & 1.3 & 1.3 & 2 \\
y & 0 & 1.3 & 0 & 0 & 0.7 \\
\end{array}
\]

Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC'01).

Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata (HSCC'01).
Weighted/priced timed automata [ALP01,BFH+01]

\[ \ell_0 \xrightarrow{5} \ell_1, \quad x \leq 2, c, y := 0 \]

\[ \ell_1 \xrightarrow{u} \ell_2, \quad (y = 0) \]

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

\[ \ell_3 \xrightarrow{u} \ell_4, \quad c \]

\[ \text{cost} : \ 6.5 \]

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


Weighted/priced timed automata \cite{ALP01,BFH+01}

\begin{align*}
\ell_0 & \xrightarrow{1.3} \ell_0 \xrightarrow{c} \ell_1 \xrightarrow{u} \ell_3 \xrightarrow{0.7} \ell_3 \xrightarrow{c} \ell_3, \\
x & = 2, c \\
y & = 0
\end{align*}

\begin{align*}
\text{cost : } & 6.5 + 0
\end{align*}

\cite{ALP01} Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC’01).

\cite{BFH+01} Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata (HSCC’01).
Weighted/priced timed automata [ALP01,BFH+01]

Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC'01).
Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata (HSCC'01).
Weighted/priced timed automata [ALP01,BFH+01]

\[
\ell_0 \xrightarrow{c} \ell_0, \quad \ell_0 \xrightarrow{u} \ell_0, \quad \ell_0 \xrightarrow{c} \ell_1, \quad \ell_1 \xrightarrow{u} \ell_2, \quad \ell_2 \xrightarrow{u} \ell_3, \quad \ell_3 \xrightarrow{c} \ell_3.
\]

<table>
<thead>
<tr>
<th>State</th>
<th>(x)</th>
<th>(y)</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\ell_0)</td>
<td>0</td>
<td>0</td>
<td>6.5</td>
</tr>
<tr>
<td>(\ell_1)</td>
<td>1.3</td>
<td>1.3</td>
<td>0 + 0 + 0.7</td>
</tr>
<tr>
<td>(\ell_2)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\ell_3)</td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Weighted/priced timed automata [ALP01,BFH+01]

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

\[
\begin{array}{c|c|c|c|c|c|c|c}
 & \ell_0 & \ell_0 & \ell_1 & \ell_3 & \ell_3 & \ell_3 \\
 x & 0 & 1.3 & 1.3 & 1.3 & 2 & \\
y & 0 & 1.3 & 0 & 0 & 0.7 & \\
\hline
\text{cost} & 6.5 & + & 0 & + & 0.7 & + & 7
\end{array}
\]
Weighted/priced timed automata [ALP01,BFH+01]

\[
\begin{align*}
\ell_0 \xrightarrow{+5} & x \leq 2, c, y := 0 \\
\ell_0 \xrightarrow{1.3} & \ell_0 \\
\ell_1 \xrightarrow{c} & \ell_1 \\
\ell_1 \xrightarrow{u} & \ell_2 \\
\ell_3 \xrightarrow{0.7} & \ell_3 \\
\ell_3 \xrightarrow{c} & \text{smiley} \end{align*}
\]

Cost:

\[
\begin{align*}
\text{cost} : & \quad 6.5 + 0 + 0 + 0.7 + 7 = 14.2
\end{align*}
\]


Weighted/priced timed automata \[ \text{[ALP01,BFH+01]} \]

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

**Question:** what is the optimal cost for reaching \( \text{smiley} \)?
Weighted/priced timed automata [ALP01,BFH+01]

Question: what is the optimal cost for reaching 😊?

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

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC'01).
Weighted/priced timed automata [ALP01,BFH+01]

Question: what is the optimal cost for reaching \( \smiley \)?

\[
5t + 10(2 - t) + 1 , \quad 5t + (2 - t) + 7
\]


Question: what is the optimal cost for reaching 🌞?

$$\min \left( 5t + 10(2 - t) + 1, \ 5t + (2 - t) + 7 \right)$$
Weighted/priced timed automata [ALP01,BFH+01]

Question: what is the optimal cost for reaching 😊?

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

Weighted/priced timed automata \cite{ALP01,BFH+01}

\begin{equation*}
\ell_0 \xrightarrow{x \leq 2,c,y:=0} \ell_1 \xrightarrow{(y=0)} \ell_1 \xrightarrow{u} \ell_2 \xrightarrow{x=2,c} \ell_3 \xrightarrow{x=2,c} \ell_2 \xrightarrow{+1} \ell_3 \xrightarrow{+1} \ell_2 \xrightarrow{+10} \ell_0
\end{equation*}

**Question:** what is the optimal cost for reaching \(\smiley\)?

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

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

\cite{ALP01} Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC’01).

\cite{BFH+01} Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata (HSCC’01).
The region abstraction is not fine enough

\[\text{\ldots\ldots\ldots\ldots\ time elapsing}\]
\[\text{\ldots\ldots\ldots\ldots reset to 0}\]
The corner-point abstraction
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

\[ t_1 \quad t_2 \quad t_3 \quad t_4 \quad t_5 \quad \cdots \]
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[
\begin{align*}
    t_1 & \quad t_2 & \quad t_3 & \quad t_4 & \quad t_5 & \quad \cdots \\
    x \leq 2 & & & & & \\
\end{align*}
\]

\[
\begin{cases}
    t_1 + t_2 \leq 2
\end{cases}
\]
Optimal reachability as a linear programming problem

\begin{array}{l}
  t_1 \\
  y := 0 \\
  t_2 \\
  x \leq 2 \\
  t_3 \\
  t_4 \\
  y \geq 5 \\
  t_5 \\
  \cdots \\
\end{array}

\begin{align*}
  t_1 + t_2 & \leq 2 \\
  t_2 + t_3 + t_4 & \geq 5
\end{align*}
From timed to discrete behaviours

Optimal reachability as a linear programming problem

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**

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

\[ \begin{aligned} t_1 + t_2 & \leq 2 \\ t_2 + t_3 + t_4 & \geq 5 \end{aligned} \]

**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.

\[ \leadsto \text{ for every finite path } \pi \text{ in } \mathcal{A}, \text{ there exists a path } \Pi \text{ in } \mathcal{A}_{cp} \text{ such that} \]

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

[\Pi \text{ is a “corner-point projection” of } \pi]
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $A_{cp}$,
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $\mathcal{A}_{cp}$, for any $\varepsilon > 0$,
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.

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

Approximation of abstract paths:

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

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

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

$$||\Pi - \pi_\epsilon||_\infty < \epsilon \Rightarrow |\text{cost}(\Pi) - \text{cost}(\pi_\epsilon)| < \eta$$
Optimal-cost reachability

Theorem [ALP01,BFH+01,BBBR07]

The optimal-cost reachability problem is decidable (and PSPACE-complete) in (priced) timed automata.

Going further 1: mean-cost optimization

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

\[ \text{High} \]

\[ \text{Low} \]

\[ \text{Op} \]

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

\[ x = D \Rightarrow \text{att?} \]

\[ (x \leq D) \Rightarrow \dot{C} = P \quad \dot{R} = G \]

\[ x \neq D \Rightarrow x := 0 \]

\[ z \geq S \Rightarrow z := 0 \]

Going further 1: mean-cost optimization

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

\[ x = D \quad x := 0 \]

\[ x \leq D \]

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

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

\[ \text{High} \]

\[ \text{Low} \]

\[ \text{Op} \]

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

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

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

[20/45]

Going further 1: mean-cost optimization

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

\( x \leq D \)

\( \dot{C} = p, \quad \dot{R} = g \)

\( x = D \)

\( \dot{C} = 0, \quad \dot{R} = 0 \)

\( x := 0 \)

\( \text{att} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( z \geq S \)

\( z := 0 \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)

\( \text{High} \)

\( \text{Op} \)

\( \text{att}? \)

\( x := 0 \)

\( \text{Low} \)
Going further 1: mean-cost optimization

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

\( x \leq D \)

\( x = D \)

\( x := 0 \)

\( x :\!: 0 \)

\( z :\!: 0 \)

\( z \geq S \)

\sim \text{ compute optimal infinite schedules that minimize } \]

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

**Theorem [BBL08]**

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

\sim \text{ the corner-point abstraction can be used}

From timed to discrete behaviours

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

**Lemma**

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

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

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

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

**Lemma**

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

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

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

\[ \leadsto \text{for every finite path } \pi \text{ in } \mathcal{A}, \text{ there exists a path } \Pi \text{ in } \mathcal{A}_{cp} \text{ s.t.} \]

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

$\leadsto$ for every finite path $\pi$ in $\mathcal{A}$, there exists a path $\Pi$ in $\mathcal{A}_{\text{cp}}$ s.t.

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

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

The (acyclic) linear part will be negligible!
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.

\[ \Rightarrow \text{ for every finite path } \pi \text{ in } \mathcal{A}, \text{ there exists a path } \Pi \text{ in } \mathcal{A}_{cp} \text{ s.t. } \text{mean-cost}(\Pi) \leq \text{mean-cost}(\pi) \]

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

The (acyclic) linear part will be negligible!

\[ \Rightarrow \text{ the optimal cycle of } \mathcal{A}_{cp} \text{ is better than any infinite path of } \mathcal{A}! \]
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $\mathcal{A}_{cp}$, 

$$\forall \varepsilon > 0, \exists \delta > 0 \text{ s.t. } \|\Pi - \Pi^\delta\|_\infty < \varepsilon$$

$$\Rightarrow |\text{mean-cost}(\Pi) - \text{mean-cost}(\Pi^\delta)| < \varepsilon$$
From discrete to timed behaviours

**Approximation of abstract paths:**

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

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

\[\leadsto\] 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.

\[\leadsto\] a slight extension of corner-point abstraction can be used

---

[JT08] Judziński, Trivedi. Concavely-priced timed automata (*FORMATS’08*).
Going further 3: discounted-time cost optimization

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

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

Going further 3: discounted-time cost optimization

Globally, \( z \leq 8 \)

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

\[
\begin{align*}
x = 3, x := 0 & \quad (x \leq 3) \\
(x \leq 3) & \quad x = 3 \\
\end{align*}
\]

\[
\begin{align*}
z \geq 2, x, z := 0 & \quad (x \leq 3) \\
z \geq 2, z := 0 & \quad (x \leq 3) \\
\end{align*}
\]

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

Going further 3: discounted-time cost optimization

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

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, \( (z \leq 8) \)

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

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

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

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

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

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

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

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

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

\[
\begin{align*}
\text{if } \lambda = e^{-1}, \text{ the discounted cost of}
\text{that infinite schedule is } \approx 2.16
\end{align*}
\]

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

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

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

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

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

**Theorem [FL08]**

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

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

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata (INFINITY'08).
Modelling and optimizing resources in timed systems

A fourth model of the system

What if there is an unexpected event?
A fourth model of the system
What if there is an unexpected event?
A fourth model of the system
What if there is an unexpected event?

⧵ modelled as timed games
A simple example of timed game

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

\[ x = 2, c \]

\[ u \]
A simple example of timed game

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

\[ x = 2, c \]
Another example

\( \ell_0 \)  
\( x \leq 2 \)

\( \ell_1 \)  
\( x \geq 1 \)

\( \ell_2 \)  
\( x \leq 1 \)

\( \ell_3 \)  
\( x < 1 \), \( x := 0 \)

\( x < 1 \), \( x := 0 \)

\( x \leq 1 \), \( x = 2 \)

\( x \geq 2 \)

\( x \leq 1 \)
Decidability of timed games

**Theorem [AMPS98,HK99]**

Safety and reachability control in timed automata are decidable and 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.

(For certain cases, the attractor is computable.)

∽ classical regions are sufficient for solving such problems.
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.

Decidability of timed games

Theorem [AMPS98,HK99]

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

(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 \( \text{EXPTIME-complete} \).

\[ \leadsto \text{let's play with Uppaal Tiga!} \] [BCD+07]

Back to the simple example

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

Question: what is the optimal cost we can ensure while reaching the smiley?

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

Strategy: wait in \(\ell_0\), and when \(t = \frac{4}{3}\), go to \(\ell_1\).
Back to the simple example

\[
\begin{align*}
\ell_0 & \xrightarrow{x \leq 2, c, y := 0} \ell_1 \\
+5 & \quad (y = 0) \\
\ell_2 & \xrightarrow{x = 2, c} +10 \\
\ell_3 & \xrightarrow{x = 2, c} +1 \\
\end{align*}
\]

Question: what is the optimal cost we can ensure while reaching?
Back to the simple example

Question: what is the optimal cost we can ensure while reaching 😊?
Back to the simple example

Question: what is the optimal cost we can ensure while reaching 😊?  

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

Question: what is the optimal cost we can ensure while reaching 😊 ?

\[ 5t + 10(2 - t) + 1, \quad 5t + (2 - t) + 7 \]
Question: what is the optimal cost we can ensure while reaching 🌟 ?

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

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

**Question:** what is the optimal cost we can ensure while reaching \( \text{smiley} \)?

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

\[ \ell_0 + 5 \rightarrow \ell_1 \quad x \leq 2, c, y := 0 \quad (y = 0) \]

\[ +10 \rightarrow \ell_2 \quad x = 2, c \]

\[ +1 \rightarrow \ell_3 \quad x = 2, c \]

\[ +1 \rightarrow \text{smiley} \]

**Question:** what is the optimal cost we can ensure while reaching \( \text{smiley} \)?

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

\( \leadsto \) **strategy:** wait in \( \ell_0 \), and when \( t = \frac{4}{3} \), go to \( \ell_1 \)
Optimal reachability in priced timed games

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).
Optimal reachability in priced timed games

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

\[ \text{e.g. [LMM02, ABM04, BCFL04]} \]

**Theorem** [BBR05, BBM06]

Optimal timed games are **undecidable**, as soon as automata have three clocks or more.
Optimal reachability in priced timed games

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

\[ \text{e.g. [LMM02, ABM04, BCFL04]} \]

**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 \(3\text{EXPTIME}\) when automata have a single clock. They are **PTIME-hard**.

[BRR05] 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** $3\text{EXPTIME}$ when automata have a single clock. They are **PTIME-hard**.

- Key: resetting the clock somehow resets the history...

**[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 **PTIME-hard**.

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

![Diagram](attachment:image.png)

---

The positive side

Theorem [BLMR06]

Turn-based optimal timed games are \textit{decidable} in 3EXPTIME when automata have a single clock. They are \textit{PTIME-hard}.

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

\[
\begin{align*}
\ell_0 & \quad (x \leq 1) \\
\ell_0 & \quad x < 1 \\
\ell_0 & \quad x := 0 \\
\ell_1 & \quad +1 \\
\ell_1 & \quad x = 1 \\
\ell_1 & \quad x > 0 \\
\end{align*}
\]

- However, by unfolding and removing one by one the locations, we can synthesize \textit{memoryless almost-optimal} winning strategies.

The positive side

**Theorem [BLMR06]**

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

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

```
(\ell_0) +2 \quad (x \leq 1) \quad x = 1 \quad x > 0
```

- However, by unfolding and removing one by one the locations, we can synthesize **memoryless almost-optimal** winning strategies.
- Rather involved proof 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 $y = 2x$.

The cost is increased by $x_0$

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 $y = 2x$. 

\[
\begin{align*}
\text{Add}^+(x) & \rightarrow \text{Add}^+(x) & \rightarrow \text{Add}^-(y) & \rightarrow \text{smiley} & \text{cost} = 2 \cdot 0 + (1 - y) + 2 \\
\text{Add}^-(x) & \rightarrow \text{Add}^-(x) & \rightarrow \text{Add}^+(y) & \rightarrow \text{smiley} & \text{cost} = 2(1 - x) + y + 1
\end{align*}
\]

If $y < 2x$, player 2 chooses the first branch: cost $> 3$.
If $y > 2x$, player 2 chooses the second branch: cost $> 3$.
If $y = 2x$, in both branches, cost $= 3$.

Player 1 has a winning strategy with cost $\leq 3$ if $y = 2x$. 

32/45
The negative side: why is that hard?

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

$\text{In } \varnothing, \text{ cost } = 2x_0 + (1 - y_0) + 2$
The negative side: why is that hard?

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

- In $\begin{array}{c}
\text{z:=0} \\
\text{z:=0}
\end{array}$, cost = $2x_0 + (1 - y_0) + 2$
- In $\begin{array}{c}
\text{z:=0} \\
\text{z:=0}
\end{array}$, cost = $2(1 - x_0) + y_0 + 1$
The negative side: why is that hard?

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

- In the first case, cost = $2x_0 + (1 - y_0) + 2$
- In the second case, 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 $y = 2x$.

- $\text{In } \smiley, \text{ cost } = 2x_0 + (1 - y_0) + 2$
- $\text{In } \frowny, \text{ cost } = 2(1 - x_0) + y_0 + 1$

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

- In $\bigodot$, 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 $\smiley$, cost $= 2x_0 + (1 - y_0) + 2$
- In $\smiley$, 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:
  \[
  x = \frac{1}{2c_1} \quad \text{and} \quad y = \frac{1}{3c_2}
  \]
  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.
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, $(x \leq 1, y \leq 1, u \leq 1)$

$$x=1, x:=0$$
$$\lor \quad y=1, y:=0$$

$$x=1, x:=0$$
$$\lor \quad y=1, y:=0$$

Test$_y(x=2z)$
Outline

1. Introduction

2. Modelling and optimizing resources in timed systems

3. Managing resources

4. Conclusion
A fifth model of the system
Can I work with my computer all the way?
Can I work with my computer all the way?
Can I work with 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 of resource management

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 of resource management

Globally ($x \leq 1$)

-3 → +6 → -6

$x := 0$ → $x = 1$

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

Globally \((x \leq 1)\)

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

Globally \((x \leq 1)\)

![Diagram showing resource management](image)

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

Globally \((x \leq 1)\)

\[
\ell_0 \rightarrow \ell_1 \rightarrow \ell_2
\]

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

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

Globally \((x \leq 1)\)

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

Globally \((x \leq 1)\)

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

Globally \( (x \leq 1) \)

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

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

Globally \((x \leq 1)\)

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

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

Globally \((x \leq 1)\)

\[
\ell_0 - 3 \xrightarrow{x:=0} \ell_1 + 6 \xrightarrow{x=1} \ell_2 - 6
\]

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

Globally \( (x \leq 1) \)

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

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

Globally ($x \leq 1$)

Lower-bound problem

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

Globally \((x \leq 1)\)

-3 \(\ell_0\) → +6 \(\ell_1\) → −6 \(\ell_2\)

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

Globally \((x \leq 1)\)

-3 \rightarrow 6 \rightarrow -6

\(x:=0\) \hspace{1cm} \(x=1\)

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

Globally \((x \leq 1)\)

\[
\begin{align*}
-3 & \quad \ell_0 \\
+6 & \quad \ell_1 \\
-6 & \quad \ell_2
\end{align*}
\]

\[
\begin{array}{c}
x := 0 \\
x := 1
\end{array}
\]

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

lost!
An example of resource management

Globally \((x \leq 1)\)

\[
\ell_0 \rightarrow \ell_1 \rightarrow \ell_2
\]

\[
-3 + 6 - 6
\]

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

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

Globally \((x \leq 1)\)

-3 \(\ell_0\) \(\xrightarrow{x:=0}\) +6 \(\ell_1\) \(\xrightarrow{x=1}\) -6 \(\ell_2\)

Lower-bound problem

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

Globally \( (x \leq 1) \)

\[
\ell_0 \rightarrow \ell_1 \rightarrow \ell_2
\]

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

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

Globally \((x \leq 1)\)

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

Globally \((x \leq 1)\)

-3 \[\ell_0\] \[x:=0\] \[x=1\] +6 \[\ell_1\] \[-6 \[\ell_2\]

- Lower-bound problem \(\leadsto L\)
- Lower-upper-bound problem \(\leadsto L+U\)
- Lower-weak-upper-bound problem \(\leadsto L+W\)
Only partial results so far \[\text{[BFLMS08]}\]

<table>
<thead>
<tr>
<th>0 clock!</th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>(\in) PTIME</td>
<td>(\in) PTIME</td>
<td>(\in) UP (\cap) co-UP PTIME-hard</td>
</tr>
<tr>
<td>L+W</td>
<td>(\in) PTIME</td>
<td>(\in) PTIME</td>
<td>(\in) NP (\cap) co-NP PTIME-hard</td>
</tr>
<tr>
<td>L+U</td>
<td>(\in) PSPACE \ NP-hard</td>
<td>(\in) PTIME</td>
<td>EXPTIME-c.</td>
</tr>
</tbody>
</table>

\[\text{[BFLMS08]}\] Bouyer, Fahrenberg, Larsen, Markey, Srba. Infinite runs in weighted timed automata with energy constraints \(\text{(FORMATS'08)}\).
Only partial results so far \[BFLMS08\]

<table>
<thead>
<tr>
<th>1 clock</th>
<th>exist. problem</th>
<th>univ. problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>$\in$ PTIME</td>
<td>$\in$ PTIME</td>
<td>?</td>
</tr>
<tr>
<td>L+W</td>
<td>$\in$ PTIME</td>
<td>$\in$ PTIME</td>
<td>?</td>
</tr>
<tr>
<td>L+U</td>
<td>?</td>
<td>?</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

\[BFLMS08\] Bouyer, Fahrenberg, Larsen, Markey, Srba. Infinite runs in weighted timed automata with energy constraints (FORMATS'08).
Only partial results so far [BFLMS08]

<table>
<thead>
<tr>
<th>n clocks</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>undecidable</td>
</tr>
</tbody>
</table>

Relation 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?
Relation with mean-payoff games

Definition
Mean-payoff games: in a weighted game graph, does there exist a strategy so that the mean-cost of any play is nonnegative?

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

Relation with mean-payoff games

Definition

Mean-payoff games: in a weighted game graph, does there exist 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 $G$).
Relation with mean-payoff games

Definition

Mean-payoff games: in a weighted game graph, does there exist a strategy such that 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:

\[ \begin{array}{c}
\text{initial state} \\
\circ \quad p \quad \square \quad 0 \quad \circ
\end{array} \quad \sim \quad \begin{array}{c}
\text{to initial state} \\
\circ \quad p \quad \circ
\end{array} \]
Single-clock $\mathbf{L}+\mathbf{U}$-games

**Theorem**

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

Theorem

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

\textbf{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.

\[ \leadsto \text{We present a generic construction for incrementing/decrementing the counters.} \]
Generic module for incrementing/decrementing

Managing resources
Generic module for incrementing/decrementing

Managing resources

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

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

module ok

module ok

energy

\[ 5 - e \]

\[ x \]

\[ 0 \]

\[ 1 \]
Managing resources

Generic module for incrementing/decrementing

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

\[ x := 0 \]
\[ x = 1 \]
\[ \text{module ok} \]

\[ x := 0 \]
\[ x = 1 \]
\[ \text{module ok} \]

\[ 5 - e \]

\[ x \]

energy

\[ 0 \]

\[ 1 \]
Generic module for incrementing/decrementing

Managing resources

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

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

\[ x := 1 \quad \text{module ok} \quad x := 1 \quad \text{module ok} \]

energy

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

\[ x = 0 \quad 1 \]

\( \alpha \)
Generic module for incrementing/decrementing

Managing resources
Managing resources

Generic module for incrementing/decrementing

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

\[
x = 0 \rightarrow +5 \rightarrow x = 1 \rightarrow \text{module ok}
\]

\[
x = 0 \rightarrow -5 \rightarrow x = 1 \rightarrow \text{module ok}
\]

\[
\begin{align*}
\text{energy} & \\
5 - e & \\
0 & \quad 1
\end{align*}
\]
Managing resources

Generic module for incrementing/decrementing

\[ x := 0 \to m \to m_1 \to m_2 \to m_3 \to n \to x = 1 \]

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

\[ \text{module ok} \]

\[ \text{module ok} \]

energy

\[ 5 - e \]

\[ 5 - \frac{\alpha e}{6} \]

\[ x \]

\[ \frac{5 - e}{6} \]

\[ 0 \]

\[ 1 \]
Managing resources

Generic module for incrementing/decrementing

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

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

\[ x = 1 \rightarrow -5 \rightarrow x = 1 \]

\[ \text{module ok} \]

\[ \alpha = 3: \text{ increment } c_1 \]

\[ \alpha = 2: \text{ increment } c_2 \]

\[ \alpha = 12: \text{ decrement } c_1 \]

\[ \alpha = 18: \text{ decrement } c_2 \]
Outline

1. Introduction

2. Modelling and optimizing resources in timed systems

3. Managing resources

4. Conclusion
Some applications

Tools
- Uppaal (timed automata)
- Uppaal Cora (priced timed automata)
- Uppaal Tiga (timed games)

Case studies
- A lacquer production scheduling problem [BBHM05]
- Task graph scheduling problems [AKM03]
- An oil pump control problem [CJL+09]

[BBHM05] Behrmann, Brinksma, Hendriks, Mader. Scheduling lacquer production by reachability analysis - A case study (IFAC'05).
[AKM03] Abdelhaïm, Kerbaa, Maler. Task graph scheduling using timed automata (IPDPS'03).
Task graph scheduling problems

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

$P_1$ (fast):

<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
</table>
| +                  | 2 picoseconds   
| $\times$           | 3 picoseconds   

<table>
<thead>
<tr>
<th>energy</th>
</tr>
</thead>
</table>
| idle               | 10 Watt          
| in use             | 90 Watts         

$P_2$ (slow):

<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
</table>
| +                  | 5 picoseconds  
| $\times$           | 7 picoseconds  

<table>
<thead>
<tr>
<th>energy</th>
</tr>
</thead>
</table>
| idle               | 20 Watts         
| in use             | 30 Watts         

Task graph scheduling problems

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

$P_1$ (fast):

<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
</tr>
<tr>
<td>×</td>
</tr>
</tbody>
</table>

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

$P_2$ (slow):

<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
</tr>
<tr>
<td>×</td>
</tr>
</tbody>
</table>

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

1. $13$ picoseconds
2. $1.37$ nanojoules
Task graph scheduling problems

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

**$P_1$ (fast):**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (picoseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>2</td>
</tr>
<tr>
<td>$\times$</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Energy (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
</tr>
<tr>
<td>In use</td>
</tr>
</tbody>
</table>

**$P_2$ (slow):**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (picoseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>5</td>
</tr>
<tr>
<td>$\times$</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Energy (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
</tr>
<tr>
<td>In use</td>
</tr>
</tbody>
</table>

Diagram showing the task graph and scheduling: Sch1 and Sch2 with respective times and energy consumption.
Task graph scheduling problems

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

**$P_1$ (fast):**

<table>
<thead>
<tr>
<th>Time</th>
<th>2 picoseconds</th>
<th>3 picoseconds</th>
</tr>
</thead>
</table>

| Energy | idle 10 Watt | in use 90 Watts |

**$P_2$ (slow):**

<table>
<thead>
<tr>
<th>Time</th>
<th>5 picoseconds</th>
<th>7 picoseconds</th>
</tr>
</thead>
</table>

| Energy | idle 20 Watts | in use 30 Watts |

![Diagram of task dependencies and scheduling]

1. **Sch1**
   - $T_1$, $T_2$, $T_3$, $T_4$, $T_5$, $T_6$
   - 13 picoseconds
   - 1.37 nanojoules

2. **Sch2**
   - $T_1$, $T_3$, $T_5$, $T_4$, $T_6$
   - 12 picoseconds
   - 1.39 nanojoules

3. **Sch3**
   - $T_1$, $T_3$, $T_4$
   - $T_2$, $T_5$, $T_6$
   - 19 picoseconds
   - 1.32 nanojoules
Modelling the task graph scheduling problem
Modelling the task graph scheduling problem

- Processors

\[ P_1: \]

- \( x \leq 2 \)
  - \( x = 2 \)
    - \( \text{done}_1 \)
    - \( \text{add}_1 \)
    - \( x = 0 \)
- \( x = 3 \)
  - \( \text{done}_1 \)
  - \( \text{mult}_1 \)
  - \( x = 0 \)
- \( x \leq 3 \)

\[ P_2: \]

- \( y \leq 5 \)
  - \( y = 5 \)
    - \( \text{done}_2 \)
    - \( \text{add}_2 \)
    - \( x = 0 \)
- \( y = 7 \)
  - \( \text{done}_2 \)
  - \( \text{mult}_2 \)
  - \( x = 0 \)
- \( y \leq 7 \)
Modelling the task graph scheduling problem

- **Processors**

  $P_1$:  
  \[
  \begin{align*}
  &\text{idle} & x \leq 2 \\
  &\text{add}_1 & x \leq 2 \\
  &\text{mult}_1 & x \leq 3 \\
  &\text{done}_1 & x = 2 \\
  &\text{idle} & x = 3 \\
  &\text{add}_1 & x = 3 \\
  &\text{mult}_1 & x = 3 \\
  \end{align*}
  \]

  $P_2$:  
  \[
  \begin{align*}
  &\text{idle} & y \leq 5 \\
  &\text{add}_2 & y \leq 5 \\
  &\text{mult}_2 & y \leq 7 \\
  &\text{done}_2 & y = 5 \\
  &\text{idle} & y = 7 \\
  &\text{add}_2 & y = 7 \\
  &\text{mult}_2 & y = 7 \\
  \end{align*}
  \]

- **Tasks**

  $T_4$:  
  \[
  \begin{align*}
  t_1 \land t_2 & \rightarrow \text{idle} \\
  \text{add}_i & \rightarrow \text{done}_i \\
  t_4 := 1 & \\
  \end{align*}
  \]

  $T_5$:  
  \[
  \begin{align*}
  t_3 & \rightarrow \text{idle} \\
  \text{add}_i & \rightarrow \text{done}_i \\
  t_5 := 1 & \\
  \end{align*}
  \]
Modelling the task graph scheduling problem

- **Processors**
  - $P_1$: (idle) $x=2$ \(\rightarrow\) done\(_1\) add\(_1\) $x=0$ \(\rightarrow\) done\(_1\) mult\(_1\) $x=3$ \(\rightarrow\) (idle)
  - $P_2$: (idle) $y=5$ \(\rightarrow\) done\(_2\) add\(_2\) $x=0$ \(\rightarrow\) done\(_2\) mult\(_2\) $y=7$ \(\rightarrow\) (idle)

- **Tasks**
  - $T_4$: $t_1 \land t_2$ \(\rightarrow\) $t_4 := 1$
  - $T_5$: $t_3$ \(\rightarrow\) $t_5 := 1$

- **Modelling energy**
  - $P_1$: (idle) $x=2$ \(\rightarrow\) done\(_1\) add\(_1\) $x=0$ \(\rightarrow\) done\(_1\) mult\(_1\) $x=3$ \(\rightarrow\) (idle)
  - $P_2$: (idle) $y=5$ \(\rightarrow\) done\(_2\) add\(_2\) $x=0$ \(\rightarrow\) done\(_2\) mult\(_2\) $y=7$ \(\rightarrow\) (idle)
Modelling the task graph scheduling problem

- **Processors**
  - **$P_1$:**
    - $x=2$
    - $x=3$
    - $x:=0$
    - $x:=0$
  - **$P_2$:**
    - $y=5$
    - $y=7$
    - $x:=0$
    - $x:=0$

- **Tasks**
  - **$T_4$:**
    - $t_1 \land t_2$
    - $t_4 := 1$
  - **$T_5$:**
    - $t_3$
    - $t_5 := 1$

- **Modelling energy**
  - **$P_1$:**
    - $x=2$
    - $x=3$
    - $x:=0$
    - $x:=0$
  - **$P_2$:**
    - $y=5$
    - $y=7$
    - $x:=0$
    - $x:=0$

- **Modelling uncertainty**
  - **$P_1$:**
    - $x \geq 1$
    - $x:=0$
    - $x:=0$
  - **$P_2$:**
    - $y \geq 3$
    - $y:=2$
    - $y:=2$
Conclusion

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

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

- **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
  - various tools have been developed:
    Uppaal, Uppaal Cora, Uppaal Tiga
Priced/weighted timed automata, a model for representing quantitative constraints on timed systems:
- useful for modelling resources in timed systems
- natural (optimization/management) questions have been posed...
  ... and not all of them have been answered!

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

Current and further work:
- further cost functions (e.g. exponential)
- 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?
- computation of equilibria
- ...