Robustness in Timed Automata: A Game-Based Approach

Patricia Bouyer-Decitre

LSV, CNRS & ENS Cachan, France

Based on joint works with Nicolas Markey, Pierre-Alain Reynier and Ocan Sankur
Acknowledgment to Nicolas and Ocan for slides
Time-dependent systems

- We are interested in timed systems
Model-checking and control

system:

property:

[http://www.embedded.com]
Model-checking and control

system:

[http://www.embedded.com]

property:

\[ \text{AG}(\neg \text{B.overfull} \land \neg \text{B.dried_up}) \]
Model-checking and control

system:

property:

algorithm

\[ \text{AG}(\neg B.\text{overfull} \land \neg B.\text{dried\_up}) \]
Introduction

Model-checking and control

system:

property:

model-checking algorithm

AG(¬B.overfull ∧ ¬B.dried_up)

yes/no
Model-checking and control

system:

property:

control/synthesis algorithm

AG(¬B.overfull ∧ ¬B.dried_up)
Reasoning about real-time systems

The model of timed automata [AD94]
A timed automaton is made of
- a finite automaton-based structure

Example

[Diagram showing a timed automaton with states: safe, alarm, repairing, failsafe, and transitions: done, repair, delayed, and problem.]
Reasoning about real-time systems

The model of timed automata [AD94]

A timed automaton is made of
- a finite automaton-based structure
- a set of clocks

Example

```
x := 0
repair,
x \leq 15

y := 0
repair,
2 \leq y \land x \leq 56
y := 0
done,
22 \leq y \leq 25
x, y
```
Reasoning about real-time systems

The model of timed automata [AD94]

A timed automaton is made of
- a finite automaton-based structure
- a set of clocks
- timing constraints on transitions

Example

[Diagram showing states and transitions with timing constraints.]

Example – cont’d
Example – cont’d

safe

\[
\begin{align*}
x &= 0 \\
y &= 0
\end{align*}
\]
Example – cont’d

```
Example - cont'd

safe\[\rightarrow\] problem, x:=0 \rightarrow alarm
repair, x\leq15 \rightarrow repairing
repair, y:=0 \rightarrow alarm
15\leq x\leq 16 \rightarrow delayed, y:=0 \rightarrow failsafe
2\leq y \land x\leq 56 \rightarrow repairing
done, 22\leq y\leq 25 \rightarrow safe

x | 0 | 23
---|---|---
y | 0 | 23
```

5/25
Example – cont’d

\[ x = 0, \quad 0 \leq y \leq 15 \]
\[ y = 0, \quad 15 \leq x \leq 16 \]
\[ y = 0, \quad 2 \leq y \land x \leq 56 \]

\[ \text{safe} \rightarrow \text{problem}, \quad x := 0 \]
\[ \text{alarm} \rightarrow \text{repair}, \quad y := 0 \]
\[ \text{repair} \rightarrow \text{repairing}, \quad \text{delayed}, \quad y := 0 \]

\[ \text{done}, \quad 22 \leq y \leq 25 \]

\[ \text{failsafe} \rightarrow \text{repair} \rightarrow \text{repairing} \rightarrow \text{done} \rightarrow \text{safe} \]

\[ \begin{array}{|c|c|c|c|}
\hline
\text{safe} & \rightarrow & \text{safe} & \rightarrow & \text{alarm} \\
\text{x} & 0 & 23 & & 0 \\
\text{y} & 0 & 23 & & 23 \\
\hline
\end{array} \]
Example – cont’d

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>Next State</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td></td>
<td>safe</td>
<td>x:=0</td>
</tr>
<tr>
<td>problem</td>
<td></td>
<td>alarm</td>
<td>y:=0, 2≤y≤15</td>
</tr>
<tr>
<td>alarm</td>
<td></td>
<td>repairing</td>
<td>x:=0, 15≤x≤16</td>
</tr>
<tr>
<td>failsafe</td>
<td></td>
<td>repaired</td>
<td>y:=0, 2≤y≤56</td>
</tr>
</tbody>
</table>

X: 0 => 23 => 0 => 15.6
Y: 0 => 23 => 23 => 38.6
Example – cont’d

- **safe**: $x := 0, y := 0$
- **problem**: $x := 0, y := 0$
- **alarm**: $2 \leq y \wedge x \leq 56, y := 0$
- **failsafe**: $x := 0$
- **repair**: $x \leq 15, y := 0$
- **repairing**: $2 \leq y \wedge x \leq 56, y := 0$
- **done**: $22 \leq y \leq 25$

\[
\begin{array}{c|c|c|c|c|c|c}
\text{safe} & \xrightarrow{23} & \text{safe} & \xrightarrow{\text{problem}} & \text{alarm} & \xrightarrow{15.6} & \text{alarm} & \xrightarrow{\text{delayed}} & \text{failsafe} \\
X & 0 & 23 & 0 & 15.6 & \text{done} & 15.6 & \cdots \\
Y & 0 & 23 & 23 & 38.6 & \text{failsafe} & 0 & \cdots \\
\end{array}
\]
Example – cont’d

\[ x := 0 \]

\[ y := 0 \]

\[ 15 \leq x \leq 16 \]

\[ y := 0 \]

\[ 2 \leq y \land x \leq 56 \]

\[ 22 \leq y \leq 25 \]

\[ 22 \leq y \leq 25 \]

\[ x := 0 \]

\[ y := 0 \]

\[ 15 \leq x \leq 16 \]

\[ y := 0 \]

\[ 2 \leq y \land x \leq 56 \]

\[ 22 \leq y \leq 25 \]

\[ x := 0 \]

\[ y := 0 \]

\[ 15 \leq x \leq 16 \]

\[ y := 0 \]

\[ 2 \leq y \land x \leq 56 \]

\[ 22 \leq y \leq 25 \]

\[ x := 0 \]

\[ y := 0 \]

\[ 15 \leq x \leq 16 \]

\[ y := 0 \]

\[ 2 \leq y \land x \leq 56 \]

\[ 22 \leq y \leq 25 \]

\[ x := 0 \]

\[ y := 0 \]

\[ 15 \leq x \leq 16 \]

\[ y := 0 \]

\[ 2 \leq y \land x \leq 56 \]

\[ 22 \leq y \leq 25 \]

\[ x := 0 \]

\[ y := 0 \]

\[ 15 \leq x \leq 16 \]

\[ y := 0 \]

\[ 2 \leq y \land x \leq 56 \]

\[ 22 \leq y \leq 25 \]

\[ x := 0 \]

\[ y := 0 \]

\[ 15 \leq x \leq 16 \]

\[ y := 0 \]

\[ 2 \leq y \land x \leq 56 \]

\[ 22 \leq y \leq 25 \]

\[ x := 0 \]

\[ y := 0 \]

\[ 15 \leq x \leq 16 \]

\[ y := 0 \]

\[ 2 \leq y \land x \leq 56 \]

\[ 22 \leq y \leq 25 \]

\[ x := 0 \]

\[ y := 0 \]

\[ 15 \leq x \leq 16 \]

\[ y := 0 \]

\[ 2 \leq y \land x \leq 56 \]

\[ 22 \leq y \leq 25 \]
Example – cont’d
Example – cont’d

### State Transition Table

<table>
<thead>
<tr>
<th>State</th>
<th>x = 0</th>
<th>x ≤ 15</th>
<th>15 ≤ x ≤ 16</th>
<th>2 ≤ y ∧ x ≤ 56</th>
<th>y = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td>23</td>
<td>0</td>
<td>15.6</td>
<td>15.6</td>
<td></td>
</tr>
<tr>
<td>alarm</td>
<td>0</td>
<td>23</td>
<td>38.6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>delayed</td>
<td>22.1</td>
<td>0</td>
<td>40</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### State Diagram

- **Safe** (x = 0, y = 0)
- **Alarm** (x = 0, y = 0)
- **Delayed** (y = 0)
- **Failsafe** (y = 0)
- **Repairing** (y = 0)

Example conditions:
- **Safe** to **Alarm** (x = 0, y = 0)
- **Alarm** to **Delayed** (y = 0)
- **Delayed** to **Failsafe** (y = 0)
- **Failsafe** to **Repairing** (y = 0)
- **Repairing** to **Safe** (y = 0)
Example – cont’d

\[
\begin{align*}
\text{safe} & \xrightarrow{23} \text{safe} & \text{problem, } x := 0 & \xrightarrow{23} \text{alarm} & \xrightarrow{15.6} \text{alarm} & \xrightarrow{\text{delayed}} \text{failsafe} \\
X & 0 & 23 & 0 & 15.6 & 15.6 \ldots \\
Y & 0 & 23 & 23 & 38.6 & 0
\end{align*}
\]

\[
\begin{align*}
\text{failsafe} & \xrightarrow{2.3} \text{failsafe} & \text{repair} & \xrightarrow{22.1} \text{failsafe} & \xrightarrow{\text{done}} \text{safe} \\
\ldots & 15.6 & 17.9 & 17.9 & 40 & 40 \\
0 & 2.3 & 0 & 22.1 & 22.1
\end{align*}
\]
Discrete-time semantics

...because computers are digital!
Discrete-time semantics

...because computers are digital!

Example [Alur91]

- under discrete-time, the output is always 0:

Discrete-time semantics

...because computers are digital!

Example [Alur91]

- under discrete-time, the output is always 0:

Discrete-time semantics

...because computers are digital!

Example [Alur91]

- under discrete-time, the output is always 0:

Discrete-time semantics

...because computers are digital!

Example [Alur91]

- under discrete-time, the output is always 0:

Discrete-time semantics

...because computers are digital!

Example [Alur91]

- under discrete-time, the output is always 0:

Discrete-time semantics

...because computers are digital!

Example [Alur91]

- under continuous-time, the output can be 1:

Continuous-time semantics

...real-time models for real-time systems!
Continuous-time semantics

...real-time models for real-time systems!

Example

\[ \begin{align*}
x & \leq 2, \ x := 0 \\
y & \geq 2, \ y := 0
\end{align*} \]

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction
Continuous-time semantics

...real-time models for real-time systems!

Example

\[
\begin{align*}
x & \leq 2, \quad x := 0 \\
y & \geq 2, \quad y := 0
\end{align*}
\]

Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction
Continuous-time semantics

...real-time models for real-time systems!

Example

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

\[ \begin{align*}
  x &= 2, \ x := 0 \\
  x &= 0 \land \\
  y &\geq 2
\end{align*} \]

\[ \begin{align*}
  y &\geq 2, \ y := 0
\end{align*} \]

Theorem [AD94] Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction

Efficient symbolic technics based on zones, implemented in tools
Continuous-time semantics

...real-time models for real-time systems!

Example
Continuous-time semantics

...real-time models for real-time systems!

Example

\[ x = \begin{cases} 1 & \text{if } x \leq 2, \ y := 0 \\ 0 & \text{if } x = 0 \land y \geq 2, \ y := 0 \end{cases} \]
Continuous-time semantics

...real-time models for real-time systems!

Example

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

\[ x \leq 2, \quad x := 0 \]

\[ x = 0 \land y \geq 2 \]

\[ y \geq 2, \quad y := 0 \]

Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction

Efficient symbolic technics based on zones, implemented in tools
Continuous-time semantics

...real-time models for real-time systems!

Example

\[
\begin{align*}
x &= 1, \quad y := 0 \\
x &\leq 2, \quad x := 0 \\
y &\geq 2, \quad y := 0 \\
x &= 0 \land \\
y &\geq 2
\end{align*}
\]

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction
Continuous-time semantics

...real-time models for real-time systems!

Example

\[ x = 1 \]
\[ y := 0 \]
\[ x \leq 2, \; x := 0 \]
\[ y \geq 2, \; y := 0 \]

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction

Efficient symbolic technics based on zones, implemented in tools...
Continuous-time semantics

...real-time models for real-time systems!

Example

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction
Continuous-time semantics

...real-time models for real-time systems!

Example

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

\[
\begin{align*}
x &\leq 2, \quad x := 0 \\
y &\geq 2, \quad y := 0
\end{align*}
\]

\[
\begin{align*}
x &= 0 \land \\
y &\geq 2
\end{align*}
\]
Continuous-time semantics

...real-time models for real-time systems!

Example

\[
\begin{align*}
  x &= 1 \\
  y &= 0 \\
  x \leq 2, & \quad x := 0 \\
  y \geq 2, & \quad y := 0 \\
  x &= 0 \land y \geq 2 \\
  y &= 0
\end{align*}
\]
Continuous-time semantics

...real-time models for real-time systems!

Example

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

\[\begin{align*}
x &\leq 2, \quad x := 0 \\
y &\geq 2 \\
y &:= 0
\end{align*}\]

\[\begin{align*}
x &= 0 \land \\
y &\geq 2
\end{align*}\]

Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction
Continuous-time semantics

...real-time models for real-time systems!

Example

\[
\begin{align*}
  x &= 1 \\
  y &= 0 \\
  x &\leq 2, \ x := 0 \\
  y &\geq 2, \ y := 0
\end{align*}
\]

Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

- Technical tool: region abstraction
Continuous-time semantics

...real-time models for real-time systems!

Example

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

\[
\begin{align*}
x &\leq 2, \ x := 0 \\
y &\geq 2, \ y := 0
\end{align*}
\]

Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

- Technical tool: region abstraction
- Efficient symbolic technics based on zones, implemented in tools
Are we doing the right job?

The continuous-time semantics is an *idealization* of a physical system. It is adequate for abstract design and high-level analysis.
Are we doing the right job?

The continuous-time semantics is an **idealization** of a physical system. It is adequate for abstract design and high-level analysis.

However it suffers from multiple inaccuracies:

- It might not be proper for **implementation**
  - Analysis made at the abstract level does not transfer to real world
Are we doing the right job?

The continuous-time semantics is an \textbf{idealization} of a physical system. It is adequate for abstract design and high-level analysis.

However it suffers from multiple inaccuracies:

- It might not be proper for \textit{implementation}
  $\leadsto$ analysis made at the abstract level does not transfer to real world
- It may generate \textit{timing anomalies}
Are we doing the right job?

The continuous-time semantics is an idealization of a physical system. It is adequate for abstract design and high-level analysis.

However it suffers from multiple inaccuracies:

- It might not be proper for implementation
  \(\leadsto\) analysis made at the abstract level does not transfer to real world
- It may generate timing anomalies
- It does not exclude non-realizable behaviours:
  - not only Zeno behaviours
  - many convergence phenomena are hidden
  \(\leadsto\) this requires infinite precision and might not be realizable
Example 1: Imprecision on clock values

Example 1: Imprecision on clock values

A frame will eventually be skipped

\[ 2 + \epsilon; \]

A frame will eventually be skipped

Frame capture

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

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

\[
\begin{align*}
    x \leq 2, \quad x &= 0 \\
    y \geq 2, \quad y &= 0 \\
    x &= 0 \land y \geq 2
\end{align*}
\]

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

- $x = 1$, $y := 0$
- $x \leq 2$, $x := 0$
- $y \geq 2$, $y := 0$
- $x = 0 \land y \geq 2$

References:
Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

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

\[ x \leq 2, \quad x := 0 \]

\[ y \geq 2, \quad y := 0 \]

\[ x = 0 \land y \geq 2 \]

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

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

\[
\begin{align*}
  & x \leq 2, x := 0 \\
  & y \geq 2, y := 0
\end{align*}
\]

\[
\begin{align*}
  & x = 0 \land y \geq 2
\end{align*}
\]

References:

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[ x_0, x_1, x_2 \]

\[ y_0, y_1, y_2 \]

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

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

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

\[ x = 0 \land y \geq 2 \]


Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

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

\[ x \leq 2, \ x := 0 \]
\[ y \geq 2, \ y := 0 \]
\[ x = 0 \land y \geq 2 \]

\[ \text{reachable, however small may be the jitter} \]

References:

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[
\begin{align*}
\text{\(x = 1\)} & & \text{\(y := 0\)} & \rightarrow & \text{\(x \leq 2, x := 0\)} \\
\text{\(x = 0 \land y \geq 2\)} & & \rightarrow & \text{\(y \geq 2, y := 0\)}
\end{align*}
\]

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

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

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

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

\[ x = 0 \land y \geq 2 \]

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

Introduction

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[ 1 - \delta \leq x \leq 1 + \delta \]
\[ y := 0 \]
\[ y \geq 2 - \delta, \quad y := 0 \]
\[ x \leq 2 + \delta, \quad x := 0 \]
\[ x \leq \delta \land y \geq 2 - \delta \]

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[
\begin{align*}
1 - \delta \leq x &\leq 1 + \delta, & y := 0 \\
1 - \delta \leq x &\leq 1 + \delta, & y := 0 \\
\end{align*}
\]

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[ 1 - \delta \leq x \leq 1 + \delta \]
\[ y := 0 \]

\[ x \leq 2 + \delta, \quad x := 0 \]

\[ x \leq \delta \land y \geq 2 - \delta \]
\[ y \geq 2 - \delta, \quad y := 0 \]


Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[ 1 - \delta \leq x \leq 1 + \delta, \quad y := 0 \]

\[ x \leq 2 + \delta, \quad x := 0 \]

\[ y \geq 2 - \delta, \quad y := 0 \]

\[ x \leq \delta \land y \geq 2 - \delta \]

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[ 1 - \delta \leq x \leq 1 + \delta, \quad y := 0 \]
\[ x \leq 2 + \delta, \quad x := 0 \]
\[ x \leq \delta \land y \geq 2 - \delta \]
\[ y \geq 2 - \delta, \quad y := 0 \]

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

![Graph showing the impact of small timing jitters]

- $1 - \delta \leq x \leq 1 + \delta$
- $y := 0$
- $x \leq 2 + \delta$, $x := 0$
- $y \geq 2 - \delta$, $y := 0$
- $x \leq \delta \land y \geq 2 - \delta$

References:
Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[\begin{align*}
1-\delta \leq x &\leq 1+\delta \\
y &:= 0
\end{align*}\]

\[\begin{align*}
x &\leq 2+\delta, \quad x := 0 \\
y &\geq 2-\delta, \quad y := 0
\end{align*}\]

\[\begin{align*}
x &\leq \delta \land y \geq 2-\delta
\end{align*}\]

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[ 1 - \delta \leq x \leq 1 + \delta \quad y := 0 \]

\[ x \leq 2 + \delta, \quad x := 0 \]

\[ x \leq \delta \land y \geq 2 - \delta \]

\[ y \geq 2 - \delta, \quad y := 0 \]


Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[ \begin{align*}
1 - \delta \leq x &\leq 1 + \delta \\
y &:= 0
\end{align*} \]

\[ \begin{align*}
x &\leq 2 + \delta, \ x := 0
\end{align*} \]

\[ \begin{align*}
x &\leq \delta \land y \geq 2 - \delta
\end{align*} \]

\[ \begin{align*}
y \geq 2 - \delta, \ y := 0
\end{align*} \]

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[ \begin{align*}
1 - \delta \leq x & \leq 1 + \delta, \\
y & := 0
\end{align*} \]

\[ \begin{align*}
x & \leq 2 + \delta, \\
x & := 0
\end{align*} \]

\[ \begin{align*}
x & \leq \delta \land y \geq 2 - \delta
\end{align*} \]

\[ \begin{align*}
y & \geq 2 - \delta, \\
y & := 0
\end{align*} \]

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

\[ 1 - \delta \leq x \leq 1 + \delta \]

\[ y := 0 \]

\[ x \leq 2 + \delta, \quad x := 0 \]

\[ y \geq 2 - \delta, \quad y := 0 \]

\[ x \leq \delta \land y \geq 2 - \delta \]

---


Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

〜 ◯ is reachable, however small may be the jitter

Example 3: Scheduling and timing anomaly

- Scheduling analysis with timed automata [AAM06]
- **Goal:** analyze a *work-conserving* scheduling policy on given scenarios (no machine is idle if a task is waiting for execution)

**Example of a scenario**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>A</td>
<td></td>
<td>D</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>C</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

with the dependency constraints: $A \rightarrow B$ and $C \rightarrow D, E$.

1. $A, D, E$ must be scheduled on machine $M_1$
2. $B, C$ must be scheduled on machine $M_2$
3. $C$ starts no sooner than 2 time units

Example 3: Scheduling and timing anomaly

Example of a scenario

\[ \begin{array}{c|cc|cc|cc|cc|cc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline M_1 & A & & & D & E & & & & \\ M_2 & C & & & B & & & & & \\ \end{array} \]

\[ \sim \] Schedulable in 6 time units
Example 3: Scheduling and timing anomaly

Example of a scenario

![Scenario Diagram]

\[ M_1 \]
\[ M_2 \]

\( \sim \) Schedulable in 6 time units

- Unexpectedly, the duration of A drops to 1.999
Example 3: Scheduling and timing anomaly

Example of a scenario

![Scenario Diagram]

Schedulable in 6 time units

- Unexpectedly, the duration of A drops to 1.999

is not work-conserving
Example 3: Scheduling and timing anomaly

Example of a scenario

\[ \begin{array}{|c|c|c|c|c|c|c|c|} 
\hline
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 
\hline
M_1 & A & & & & D & E & & \\ 
\hline
M_2 & & & C & & B & & & \\ 
\hline
\end{array} \]

\( \leadsto \) Schedulable in 6 time units

- Unexpectedly, the duration of A drops to 1.999

\[ \begin{array}{|c|c|c|c|c|c|c|c|} 
\hline
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 
\hline
M_1 & A & & D & E & & & & \\ 
\hline
M_2 & & & C & & B & & & \\ 
\hline
\end{array} \]

is not work-conserving

\[ \begin{array}{|c|c|c|c|c|c|c|c|} 
\hline
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 
\hline
M_1 & A & B & D & E & & & & & \\ 
\hline
M_2 & & & & & & C & & & \\ 
\hline
\end{array} \]

is work-conserving and completes in 7.999 t.u.
Example 3: Scheduling and timing anomaly

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>$M_1$</td>
<td>A</td>
<td></td>
<td>D</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td></td>
<td>C</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

~ Schedulable in 6 time units

- Unexpectedly, the duration of $A$ drops to 1.999

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>$M_1$</td>
<td>A</td>
<td>D</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>C</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

is not work-conserving

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>$M_1$</td>
<td>A</td>
<td></td>
<td>D</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

is work-conserving and completes in 7.999 t.u.

~ Standard analysis does not capture this **timing anomaly**
Example 4: Zeno behaviours

\[ x < 1 \land y < 1 \]
\[ x := 0 \]
\[ y = 1 \]

Those are easy to detect and can be handled; [HS11]

They are easy to remove by construction.
Example 4: Zeno behaviours

Those are easy to detect and can be handled;

\[ x \leq 1 \land y < 1 \]
\[ x := 0 \]
\[ y = 1 \]

Example 4: Zeno behaviours

- Those are easy to detect and can be handled;
- They are easy to remove by construction.

Example 5: More complex convergence phenomena

![Diagram](image)
Example 5: More complex convergence phenomena

\[ x = 1, \quad y = 0 \]

Values of clock \( x \) when hitting \( x \leq 2, x := 0 \)

Values of clock \( y \) when hitting \( y \geq 2, y := 0 \)

Even though global time diverges, the value of the clock converges.
Example 5: More complex convergence phenomena

Value of clock \( x \) when hitting \( y \) is converging, even though global time diverges.
Example 5: More complex convergence phenomena

The figure illustrates a state transition diagram with conditions:
- $x = 1$ and $y = 0$
- $x \leq 2$, $x := 0$
- $y \geq 2$, $y := 0$

The diagram shows the convergence of the clock value even though the global time diverges.
Example 5: More complex convergence phenomena

\[ x = 1, \quad y = 0 \]
\[ x \leq 2, \quad x := 0 \]
\[ y \geq 2, \quad y := 0 \]
Example 5: More complex convergence phenomena

Introduction

Value of clock $x$ when hitting $y$ is converging, even though global time diverges.
Example 5: More complex convergence phenomena

$\begin{align*}
x &\leq 2, \ x := 0 \\
y &\geq 2, \ y := 0
\end{align*}$

Value of clock $x$ when hitting $y_0$ is converging, even though global time diverges.
Example 5: More complex convergence phenomena

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

\[ x \leq 2, \quad x:=0 \]

\[ y \geq 2, \quad y:=0 \]
Example 5: More complex convergence phenomena

\[ x = 1, \quad y = 0 \]

\[ x \leq 2, \quad x = 0 \]
\[ y \geq 2, \quad y = 0 \]
Example 5: More complex convergence phenomena

$\begin{align*}
  x &= 1 \\
  y &= 0
\end{align*}$

$x \leq 2, x := 0$

$y \geq 2, y := 0$

Value of clock is converging, even though global time diverges.
Example 5: More complex convergence phenomena

\[ x = 1, \quad y = 0 \]
\[ x \leq 2, \quad x := 0 \]
\[ y \geq 2, \quad y := 0 \]

Value of clock when hitting is converging, even though global time diverges.
Example 5: More complex convergence phenomena

$x = 1$

$y := 0$

$x \leq 2, x := 0$

$y \geq 2, y := 0$

Value of clock is converging, even though global time diverges.
Example 5: More complex convergence phenomena
Example 5: More complex convergence phenomena

\[ x = 1, \quad y = 0 \]
\[ x \leq 2, \quad x := 0 \]
\[ y \geq 2, \quad y := 0 \]

Value of clock is converging, even though global time diverges.
Example 5: More complex convergence phenomena

Value of clock $x$ when hitting is converging, even though global time diverges.
An important issue in timed-automata verification

Add robustness to the theory of timed automata!
An important issue in timed-automata verification

Add robustness to the theory of timed automata!

- We need to understand what is the real system behind the mathematical model, and also which implementation we have in mind, if any.
An important issue in timed-automata verification

Add robustness to the theory of timed automata!

- We need to understand what is the real system behind the mathematical model, and also which implementation we have in mind, if any.
- **Aim:** provide frameworks to build robustly correct systems
An important issue in timed-automata verification

Add robustness to the theory of timed automata!

- We need to understand what is the real system behind the mathematical model, and also which implementation we have in mind, if any.
- **Aim:** provide frameworks to build **robustly** correct systems

We focus on perturbations on time measurements and jitter.
Some hints into the robustness of timed automata

Robust model-checking: the worst-case approach

- Compute an overapproximation of the set of perturbed behaviours (Possibly relate with an implementation)
- Prove correctness of this approximation
Some hints into the robustness of timed automata

Robust model-checking: the worst-case approach

- Compute an overapproximation of the set of perturbed behaviours (Possibly relate with an implementation)
- Prove correctness of this approximation

→ Many decidability results!
Often: same complexity as standard verification
Some hints into the robustness of timed automata

Robust model-checking: the worst-case approach

- Compute an overapproximation of the set of perturbed behaviours (Possibly relate with an implementation)
- Prove correctness of this approximation

\[ \leadsto \text{Many decidability results!} \]
Often: same complexity as standard verification

Robust control

- The strategy (partially) dictates the behaviour of the system
- It should tolerate imprecisions in timing measurements
- Simpler case: robust realisability
Some hints into the robustness of timed automata

Robust model-checking: the worst-case approach
- Compute an overapproximation of the set of perturbed behaviours (Possibly relate with an implementation)
- Prove correctness of this approximation

Many decidability results!
Often: same complexity as standard verification

Robust control
- The strategy (partially) dictates the behaviour of the system
- It should tolerate imprecisions in timing measurements
- Simpler case: robust realisability

We propose a game-based approach to robust realisability
Example (Can we ensure an infinite behaviour?)

Strategy: in location \( \bigcirc \) with value \( x \), delay \( \frac{2-x}{2} \)
Realisability

Example (Can we ensure an infinite behaviour?)

Strategy: in location \( \bigcirc \) with value \( x \), delay \( \frac{2-x}{2} \)

- This strategy requires infinite precision
Robust realisability

Example (Can we ensure an infinite behaviour?)

\[ \begin{align*} 
  x &= 1 \\
  y &= 0 \\
  x &\leq 2, \ x := 0 \\
  y &\geq 2, \ y := 0 
\end{align*} \]

Strategy: in location C with value x, delay \( \frac{2-x}{2} \)

- This strategy requires infinite precision
- In practice, when x is close to 2, no additional delay is supported: the run is theoretically infinite, but it is actually blocking
Robust realisability

Example (Can we ensure an infinite behaviour?)

Strategy: in location \( \bigcirc \) with value \( x \), delay \( \frac{2-x}{2} \)

- This strategy requires infinite precision
- In practice, when \( x \) is close to 2, no additional delay is supported: the run is theoretically infinite, but it is actually blocking
- And that is unavoidable
Robust realisability

Idea of robust realisability

~ Synthesize strategies that realise some property while tolerating small timing perturbations

~ Consequence: remove convergence phenomena
Robust realisability via a game semantics

Timed automaton \( A \)

\[
\begin{align*}
\ell & \xrightarrow{g,Y} \ell' \\
\ell & \xrightarrow{e} \ell'
\end{align*}
\]

Game semantics \( G_\delta(A) \) of timed automaton \( A \)...

... between Controller and Perturbator:

Controller suggests a delay \( d \geq \delta \) and a next edge \( e = (\ell \xrightarrow{g,Y} \ell') \) that is available after delay \( d \).

Perturbator then chooses a perturbation \( \epsilon \in [-\delta; +\delta] \).

Next state is \( (\ell', (v + d + \epsilon) [Y \leftarrow 0]) \).

Note: when \( \delta = 0 \), this is the standard semantics of timed automata.

A \( \delta \)-robust strategy for Controller is then a strategy that satisfies the expected property, whatever plays Perturbator.
Robust realisability via a game semantics

Timed automaton $\mathcal{A}$

Game $\mathcal{G}_\delta(\mathcal{A})$

Game semantics $\mathcal{G}_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

... between Controller and Perturbator:
Robust realisability via a game semantics

Timed automaton $\mathcal{A}$

Game $G_\delta(\mathcal{A})$

Game semantics $G_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

... between Controller and Perturbator:
- from $(\ell, v)$, Controller suggests a delay $d \geq \delta$ and a next edge $e = (\ell \xrightarrow{g,Y} \ell')$ that is available after delay $d$
Robust realisability via a game semantics

Timed automaton $\mathcal{A}$

Game $G_\delta(\mathcal{A})$

Game semantics $G_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

... between Controller and Perturbator:
- from $(\ell, v)$, Controller suggests a delay $d \geq \delta$ and a next edge $e = (\ell \xrightarrow{g,Y} \ell')$ that is available after delay $d$
- Perturbator then chooses a perturbation $\epsilon \in [-\delta; +\delta]$
Robust realisability via a game semantics

Timed automaton $\mathcal{A}$

Game $G_\delta(\mathcal{A})$

Game semantics $G_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

... between Controller and Perturbator:

- from $(\ell, v)$, Controller suggests a delay $d \geq \delta$ and a next edge $e = (\ell \xrightarrow{g,Y} \ell')$ that is available after delay $d$
- Perturbator then chooses a perturbation $\epsilon \in [-\delta; +\delta]$
- Next state is $(\ell', (v + d + \epsilon)[Y \leftarrow 0])$
Robust realisability via a game semantics

Timed automaton $\mathcal{A}$

Game $G_\delta(\mathcal{A})$

Game semantics $G_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

... between Controller and Perturbator:

- from $(\ell, v)$, Controller suggests a delay $d \geq \delta$ and a next edge $e = (\ell \xrightarrow{g, Y} \ell')$ that is available after delay $d$
- Perturbator then chooses a perturbation $\epsilon \in [-\delta; +\delta]$
- Next state is $(\ell', (v + d + \epsilon)[Y \leftarrow 0])$

Note: when $\delta = 0$, this is the standard semantics of timed automata.
Robust realisability via a game semantics

Timed automaton $\mathcal{A}$

$\ell \xrightarrow{g,Y,e} \ell'$

Game $\mathcal{G}_\delta(\mathcal{A})$

$\ell, v \xrightarrow{d \geq \delta} e \xrightarrow{-\delta \leq \epsilon \leq \delta} \ell', v'$

with $v'(v + d + \epsilon)[Y \leftarrow 0]$

Game semantics $\mathcal{G}_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

... between Controller and Perturbator:

- from $(\ell, v)$, Controller suggests a delay $d \geq \delta$ and a next edge $e = (\ell \xrightarrow{g,Y} \ell')$ that is available after delay $d$
- Perturbator then chooses a perturbation $\epsilon \in [-\delta; +\delta]$
- Next state is $(\ell', (v + d + \epsilon)[Y \leftarrow 0])$

Note: when $\delta = 0$, this is the standard semantics of timed automata.

A $\delta$-robust strategy for Controller is then a strategy that satisfies the expected property, whatever plays Perturbator.
The excess game semantics

Constraints may not be satisfied after the perturbation:
that is, only $v + d$ should satisfy $g$
The excess game semantics

Constraints may not be satisfied after the perturbation: that is, only $v + d$ should satisfy $g$

Example

The excess game semantics

Constraints may not be satisfied after the perturbation: that is, only $v + d$ should satisfy $g$

Example

[Example diagram]

The excess game semantics

Constraints may not be satisfied after the perturbation: that is, only $v + d$ should satisfy $g$

Example

The excess game semantics

Constraints may not be satisfied after the perturbation:
that is, only $v + d$ should satisfy $g$

Example

$x=y=1$
$y:=0$

The excess game semantics

Constraints may not be satisfied after the perturbation: that is, only $v + d$ should satisfy $g$.

Example

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

\[ \sim \]

Allows simple design of constraints, ensures divergence of time, avoids convergence phenomena.

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a $\delta$-robust strategy that achieves a given goal.
The excess game semantics – Algorithmics

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a $\delta$-robust strategy that achieves a given goal.

Two challenges

Accumulation of perturbations:

$$x \leq 2 \quad y := 0 \quad x = 2 \quad 1 \leq x - y$$

New regions become reachable:

$$x$$

$$y$$

$$x$$

$$y$$
The excess game semantics – Algorithmics

The (parameterized) synthesis problem
Synthesize $\delta > 0$ and a $\delta$-robust strategy that achieves a given goal.

Two challenges

Accumulation of perturbations:
The excess game semantics – Algorithmics

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a $\delta$-robust strategy that achieves a given goal.

Two challenges

1. Accumulation of perturbations:

2. New regions become reachable
The excess game semantics – Algorithmics

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a $\delta$-robust strategy that achieves a given goal.

Theorem

The parameterized synthesis problem for reachability properties is decidable and EXPTIME-complete. Furthermore, uniform winning strategies (w.r.t. $\delta$) can be computed.

- Technical tools: a region-based refined game abstraction, shrunk DBMs
- ☻ Extends to two-player games (i.e. to real control problems)

The excess game semantics – Algorithmics

The (parameterized) synthesis problem
Synthesize $\delta > 0$ and a $\delta$-robust strategy that achieves a given goal.

Theorem
The parameterized synthesis problem for reachability properties is decidable and EXPTIME-complete. Furthermore, uniform winning strategies (w.r.t. $\delta$) can be computed.

- Technical tools: a region-based refined game abstraction, shrunk DBMs
- Extends to two-player games (i.e. to real control problems)
- Only valid for reachability properties

The conservative game semantics

Constraints have to be satisfied after the perturbation: that is,

\[ \nu + d + \epsilon \text{ should satisfy } g \text{ for every } \epsilon \in [-\delta; +\delta] \]
The conservative game semantics

Constraints have to be satisfied after the perturbation: that is, $v + d + \epsilon$ should satisfy $g$ for every $\epsilon \in [-\delta; +\delta]$.

Example

\[ 1 < x < 2 \quad y := 0 \]

The conservative game semantics

Constraints have to be satisfied after the perturbation: that is, \( v + d + \epsilon \) should satisfy \( g \) for every \( \epsilon \in [-\delta; +\delta] \)

Example

\[ \begin{align*}
1 \leq x < 2 \\
\text{y := 0}
\end{align*} \]
The conservative game semantics

Constraints have to be satisfied after the perturbation: that is, 
\( v + d + \epsilon \) should satisfy \( g \) for every \( \epsilon \in [-\delta; +\delta] \)

Example

The conservative game semantics

Constraints have to be satisfied after the perturbation: that is,
\( v + d + \epsilon \) should satisfy \( g \) for every \( \epsilon \in [-\delta; +\delta] \)

Example

The conservative game semantics

Constraints have to be satisfied after the perturbation: that is, $v + d + \epsilon$ should satisfy $g$ for every $\epsilon \in [-\delta; +\delta]$. 

Example

$y := 0$

$1 < x < 2$

$\leadsto$ Strongly ensures timing constraints, ensures divergence of time, prevents converging phenomena

The conservative game semantics – Results

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a $\delta$-robust strategy that achieves a given goal.
The conservative game semantics – Results

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a $\delta$-robust strategy that achieves a given goal.

Theorem

The synthesis problem for Büchi properties is decidable and PSPACE-complete. Furthermore, winning strategies with uniform descriptions (w.r.t. $\delta$) can be computed.
Robust realizability

The conservative game semantics – Results

The (parameterized) synthesis problem

Synthesize \( \delta > 0 \) and a \( \delta \)-robust strategy that achieves a given goal.

Theorem

The synthesis problem for Büchi properties is decidable and PSPACE-complete. Furthermore, winning strategies with uniform descriptions (w.r.t. \( \delta \)) can be computed.

- Valid for all \( \omega \)-regular properties
- Same complexity as standard verification!
- Extends partially to timed games [ORS14]

The problem consists in finding cycles that do not become blocked.
The problem consists in finding cycles that do not become blocked.

- Some cycles are converging (non-forgetful)

"There is a constraining half-space"
The problem consists in finding cycles that do not become blocked.

- Some cycles are converging (non-forgetful)
- The other cycles are non-converging (forgetful)
The problem consists in finding cycles that do not become blocked.

- Some cycles are converging (non-forgetful)

  "There is a constraining half-space"

- The other cycles are non-converging (forgetful)

Characterization

There is $\delta > 0$ such that Controller has a $\delta$-robust winning strategy in $G_\delta(A)$ iff $R(A)$ has a reachable aperiodic\(^a\) non-punctual winning lasso.

\(^a\)That is, all its iterations are forgetful.
Non-forgetful cycle $\Rightarrow$ deadlock

Example

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

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

\[ y \geq 2, \ y := 0 \]
Non-forgetful cycle $\Rightarrow$ deadlock

Example

A region cycle:
Non-forgetful cycle $\Rightarrow$ deadlock

Example

A region cycle:

The corresponding orbit graph:
Non-forgetful cycle $\Rightarrow$ deadlock

Example

A region cycle:

The corresponding (folded) orbit graph:
Non-forgetful cycle $\Rightarrow$ deadlock

Example

$$\begin{align*}
x &= 1, \quad y := 0 \\
x &\leq 2, \quad x := 0 \\
y &\geq 2, \quad y := 0
\end{align*}$$

$L_I$ = sum of barycentric coordinates in $I$
Non-forgetful cycle $\Rightarrow$ deadlock

Example

$\lambda_1$, $\lambda_2$, $\lambda_3$

$L_I = \text{sum of barycentric coordinates in } I$
Non-forgetful cycle $\Rightarrow$ deadlock

Example

![Diagram]

\[
\begin{align*}
\lambda'_1 &= p_1 \lambda_1 + p_3 \lambda_3 \\
\lambda'_2 &= (1 - p_1 - p'_1) \lambda_1 + \lambda_2 \\
\lambda'_3 &= p'_1 \lambda_1 + (1 - p_3) \lambda_3
\end{align*}
\]
Non-forgetful cycle $\Rightarrow$ deadlock

Example

The cycle is not forgetful (that is, not strongly connected)
Non-forgetful cycle $\Rightarrow$ deadlock

Example

The cycle is not forgetful (that is, not strongly connected): there is an initial component $I$, and a non-decreasing function $L_I$.

$L_I = \text{sum of barycentric coordinates in } I$
Non-forgetful cycle $\Rightarrow$ deadlock

Example

Perturbator can enforce rapid decrease of $L_f$!
Forgetful cycle $\Rightarrow$ robust strategy

- Idea: target the middle of the regions
Forgetful cycle ⇒ robust strategy

- Idea: target the middle of the regions
  Let $\pi$ be a forgetful (in fact complete) cycle.

preimage by $\pi$

$r$

$s$
Forgetful cycle $\Rightarrow$ robust strategy

- Idea: target the middle of the regions
  Let $\pi$ be a forgetful (in fact complete) cycle.

![Diagram]

- Fact (technical): preimage of $s$ by $\pi$ under $\delta$-perturbations is $r - \delta Q$ ($Q$ fixed) for small $\delta$’s
Forgetful cycle $\Rightarrow$ robust strategy

- Idea: target the middle of the regions
  Let $\pi$ be a forgetful (in fact complete) cycle.
  \[
  \text{preimage by } \pi
  \]

- Fact (technical): preimage of $s$ by $\pi$ under $\delta$-perturbations is $r - \delta Q$ ($Q$ fixed) for small $\delta$’s
- Property of $s$: $s \subseteq r - \delta Q$ for small $\delta$’s
Forgetful cycle $\Rightarrow$ robust strategy

- Idea: target the middle of the regions
  
  Let $\pi$ be a forgetful (in fact complete) cycle.

- Fact (technical): preimage of $s$ by $\pi$ under $\delta$-perturbations is $r - \delta Q$ $(Q \text{ fixed})$ for small $\delta$’s

- Property of $s$: $s \subseteq r - \delta Q$ for small $\delta$’s

  $\Rightarrow$ Robust strategy: enforce $s$ at each cycle
Forgetful cycle $\Rightarrow$ robust strategy

- Idea: target the middle of the regions
  Let $\pi$ be a forgetful (in fact complete) cycle.

- Fact (technical): preimage of $s$ by $\pi$ under $\delta$-perturbations is $r - \delta Q$ ($Q$ fixed) for small $\delta$’s
- Property of $s$: $s \subseteq r - \delta Q$ for small $\delta$’s
  $\Rightarrow$ Robust strategy: enforce $s$ at each cycle

- Technical tool: shrunk DBMs [BMS12]
Conclusion

- **Timed automata**: a nice mathematical model for real-time systems with interesting decidability properties and algorithmics solutions.
- Not always easy to transfer correctness proven in this model to real behaviours of the system.
- **Robustness**: an important issue!
Conclusion

- **Timed automata**: a nice mathematical model for real-time systems with interesting decidability properties and algorithmics solutions.
- Not always easy to transfer correctness proven in this model to real behaviours of the system.
- **Robustness**: an important issue!

- Algorithms for robust realisability
- Extension to richer models seems unfortunately hard [BMS13]
- A quantitative approach to robustness: what if Perturbator plays randomly?

--

Some references:
- Survey at RP’13: "Robustness in Timed Automata" (Bouyer, Markey, Sankur)
- Survey at SiES’11: "Robustness in Real-time Systems" (Markey)
Conclusion

- **Timed automata**: a nice mathematical model for real-time systems with interesting decidability properties and algorithmics solutions.
- Not always easy to transfer correctness proven in this model to real behaviours of the system.
- **Robustness**: an important issue!
- Algorithms for robust realisability
- Extension to richer models seems unfortunately hard [BMS13]
- A quantitative approach to robustness: what if Perturbator plays randomly?

Some references:

- Survey at RP’13: “Robustness in Timed Automata” (Bouyer, Markey, Sankur)
- Survey at SiES’11: “Robustness in Real-time Systems” (Markey)