Robustness issues in timed models

Nicolas Markey

LSV, CNRS & ENS Cachan, France

(based on joint works with Patricia Bouyer, Erwin Fang, Pierre-Alain Reynier, Ocan Sankur)
(also starring Martin De Wulf, Laurent Doyen, Jean-François Raskin)

QAPL’14 – Grenoble, France
Modelling real-time systems

How should we model real-time constraints?
How should we model real-time constraints?
Reasoning about real-time systems

Example (A computer mouse)

- **idle**
  - left_button?
  - left_click!
  - left_button?
  - left_double_click!

- **right**
  - right_button?
  - right_click!
  - right_button?
  - right_double_click!

- **left**
  - left_button?
Reasoning about real-time systems

Timed automata [AD90]

A timed automaton is made of
- a transition system,

Example (A computer mouse)

- left
- idle
- right

left_button?
left_click!
left_double_click!

right_button?
right_click!
right_double_click!
Reasoning about real-time systems

Timed automata [AD90]

A timed automaton is made of

- a transition system,
- a set of clocks,

Example (A computer mouse)

```
left

left_button?

left_click!

left_button?

left_double_click!

idle

right_button?

right_click!

right_button?

right_double_click!

right
```
Reasoning about real-time systems

Timed automata [AD90]

A **timed automaton** is made of
- a transition system,
- a set of clocks,
- timing constraints on states and transitions.

Example (A computer mouse)

- **idle**
  - $x := 0$
  - $x = 300$
  - left_click!
  - $x \leq 300$
  - left_button?
  - left_double_click!

- **left**
  - $x \leq 300$

- **right**
  - $x \leq 300$

- **right_click**
  - $x = 300$

- **left_click**
  - $x = 300$

- **left_double_click**
  - $x = 300$

- **right_double_click**
  - $x = 300$

[AD90] refers to the paper 'Timed Automata' published in 1990.
Discrete-time semantics

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

...because computers are digital!

Example ([Alur91])

- under discrete-time, the output never changes:
Discrete-time semantics

...because computers are digital!

Example ([Alur91])

- under discrete-time, the output never changes:
Discrete-time semantics

...because computers are digital!

Example ([Alur91])

- under discrete-time, the output never changes:
Discrete-time semantics

...because computers are digital!

Example ([Alur91])

- under discrete-time, the output never changes:
Discrete-time semantics

...because computers are digital!

Example ([Alur91])

- under discrete-time, the output never changes:
Discrete-time semantics

...because computers are digital!

Example ([Alur91])

- under discrete-time, the output never changes:
Discrete-time semantics

...because computers are digital!

Example ([Alur91])

- under continuous-time, the output can change to 1:
Continuous-time semantics

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

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

Example

Theorem ([AD90, ACD93, ...])
Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

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

Example

Theorem ([AD90,ACD93, ...])
Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

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

Example

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

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

Theorem ([AD90, ACD93, ...])

Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

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

Example

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

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

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

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

Theorem ([AD90,ACD93,\ldots])

Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

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

Example

Theorem ([AD90, ACD93, ...])

Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

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

Example

Theorem ([AD90,ACD93, ...])

Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

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

Example

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

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

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

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

\[
y \geq 2
\]

Theorem ([AD90,ACD93, ...])

Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

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

Example

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

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

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

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

Theorem ([AD90, ACD93, ...])

Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

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

Example

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

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

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

Theorem ([AD90,ACD93, ...])

Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

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

Example

Theorem ([AD90, ACD93, ...])
Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

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

Example

Theorem ([AD90,ACD93, ...])

Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

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

Example

Theorem ([AD90,ACD93, ...])
Reachability in timed automata is decidable (as well as many other important properties).
Regions and zones

$x=1\quad y:=0$

$x\leq2,\ x:=0$

$y\geq2,\ y:=0$

$x=0\land y\geq2$
Regions and zones

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

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

\[ x=0 \land y\geq 2 \]
Regions and zones

Zones

Zones are a coarser abstraction:

\[(x \geq 2) \land (0 \leq y \leq 3) \land (x - y \leq 4)\]
Regions and zones

Zones

Zones are a **coarser abstraction**:

\[(x \geq 2) \land (0 \leq y \leq 3) \land (x - y \leq 4)\]

Representation as DBM:

\[
\begin{bmatrix}
0 & x & y \\
0 & -2 & 0 \\
\infty & 0 & 4 \\
3 & \infty & 0 \\
\end{bmatrix}
\equiv
\begin{bmatrix}
0 & x & y \\
0 & -2 & 0 \\
7 & 0 & 4 \\
3 & 1 & 0 \\
\end{bmatrix}
\]
The predecessors of \((\ell_2, x \leq 3 \land y - x \leq 0)\) are computed as

\[
\text{Pre}_{\text{time}} \left( \cap \text{Unreset}_y \left( \begin{array}{c}
\end{array} \right) \right)
\]
Regions and zones

Zones

The predecessors of \((l_2, x \leq 3 \land y - x \leq 0)\) are computed as

\[
\text{Pre}_{\text{time}} \left( \bigcap \text{Unreset}_y \left( \begin{array}{c}
\end{array} \right) \right)
\]

\[= \text{Pre}_{\text{time}} \left( \bigcap \text{Unreset}_y \left( \begin{array}{c}
\end{array} \right) \right)\]

\[\sim \text{efficient implementations}\]
Regions and zones

Zones

The predecessors of \( (\ell_2, x \leq 3 \land y - x \leq 0) \) are computed as

\[
\text{Pre}_{\text{time}} \left( \bigcap \text{Unreset}_y \right)
\]

\[= \text{Pre}_{\text{time}} \left( \bigcap \text{Unreset}_y \right) \]

\[\sim \text{efficient implementations} \]

\[\sim \text{successful applications} \]
Outline of the talk

1. Discrete time vs. dense time

2. From models to implementations

3. Checking robust safety
   - Enlarging clock constraints
   - Shrinking clock constraints

4. Checking robust controllability
   - Parametrized perturbations
   - Permissive strategies

5. Conclusions and future works
Outline of the talk

1. Discrete time vs. dense time

2. From models to implementations

3. Checking robust safety
   - Enlarging clock constraints
   - Shrinking clock constraints

4. Checking robust controllability
   - Parametrized perturbations
   - Permissive strategies

5. Conclusions and future works
Example: Patriot anti-ballistic-missile failure

28 soldiers died.
Example: Patriot anti-ballistic-missile failure

25 February 1991, during Gulf war. 28 soldiers died.

Problem: clock drift

Internal clock incremented by 1/10 every 1/10 s.

\[ x = 0.1, x := 0 \]
\[ \text{clock} += 0.1 \]
From models to implementations

Example: Patriot anti-ballistic-missile failure

28 soldiers died.

Problem: clock drift

Internal clock incremented by 1/10 every 1/10 s.

Clock stored in 24-bit register:

\[
\frac{1}{10} - \left( \frac{1}{10} \right)_\text{24 bit} \approx 10^{-7}
\]

After 100 hours, the total drift was 0.34 seconds.
The incoming missile could not be destroyed.
From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.
From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

Example (Zeno behaviors)

\[ x < 1 \land y < 1 \]
\[ x := 0 \]
\[ y = 1 \]
From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

Example (Converge phenomena)

\[
\begin{align*}
  x &\leq 1 \\
  x &= 1 \\
  x &:= 0 \\
  y &:= 0 \\
  y &= 1 \\
  y &:= 0 \\
  z &> 0 \\
  z &:= 0 \\
  x &\leq 1
\end{align*}
\]
From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

Example (Strict timing constraints)

When $P_1$ and $P_2$ run in parallel (sharing variable $r$), the state where both of them are in is not reachable.
This property is lost when $x_{id} > 2$ is replaced with $x_{id} \geq 2$. 
From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

Parametrized semantics

- parametrized discrete-time semantics:
  Does there exist a time step $\delta$ \textit{(sampling rate)} under which the system behaves correctly?
From models to implementations

**The continuous-time semantics is a mathematical idealization**

- It assumes **zero-delay transitions**;
- It assumes **infinite precision** of the clocks;
- It assumes **immediate communication** between systems.

**Parametrized semantics**

- **Parametrized discrete-time semantics:**
  Does there exists a time step $\delta$ (**sampling rate**) under which the system behaves correctly?

  $\leadsto$ reachability is undecidable [CHR02]
  $\leadsto$ untimed-language inclusion is decidable [AKY10]
From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

Parametrized semantics

- parametrized discrete-time semantics:
  - Does there exist a time step \( \delta \) (sampling rate) under which the system behaves correctly?
  - \( \leadsto \) reachability is undecidable [CHR02]
  - \( \leadsto \) untimed-language inclusion is decidable [AKY10]

- parametrized continuous-time semantics:
  - Does the system behave correctly under continuous-time semantics with imprecisions up to some \( \delta \)?
Outline of the talk

1. Discrete time vs. dense time

2. From models to implementations

3. Checking robust safety
   - Enlarging clock constraints
   - Shrinking clock constraints

4. Checking robust controllability
   - Parametrized perturbations
   - Permissive strategies

5. Conclusions and future works
Enlarged semantics for timed automata

A transition can be taken at any time in \([t - \delta; t + \delta]\).
Enlarged semantics for timed automata

A transition can be taken at any time in \([t - \delta; t + \delta]\).

**Example**

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

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

\[
\begin{align*}
x &= 0 \land y \geq 2
\end{align*}
\]
Enlarged semantics for timed automata

A transition can be taken at any time in \([t - \delta; t + \delta]\).

**Example**

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

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

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

\[
x \leq \delta \land y \geq 2 - \delta
\]
Enlarged semantics for timed automata

A transition can be taken at any time in \([t - \delta; t + \delta]\).

Example

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

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

\[
\begin{align*}
  &x \leq \delta \land y \geq 2 - \delta
\end{align*}
\]
Enlarged semantics for timed automata

A transition can be taken at any time in \([t - \delta; t + \delta]\).

Example

<table>
<thead>
<tr>
<th>(x) ∈ ([1 - \delta, 1 + \delta])</th>
<th>(x \leq 2 + \delta, \ x := 0)</th>
<th>(x \leq \delta \land y \geq 2 - \delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y := 0)</td>
<td></td>
<td>(y \geq 2 - \delta, \ y := 0)</td>
</tr>
</tbody>
</table>
Enlarged semantics for timed automata

A transition can be taken at any time in $[t - \delta; t + \delta]$.

Example

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

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

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

\[ y \geq 2 - \delta, \quad y := 0 \]
Enlarged semantics for timed automata

A transition can be taken at any time in \([t - \delta; t + \delta]\).

Example

\[
\begin{align*}
  y &= 0 & x &\leq 2 + \delta, & x := 0 \\
  y &\geq 2 - \delta, & y := 0 &
\end{align*}
\]
Enlarged semantics for timed automata

A transition can be taken at any time in \([t - \delta; t + \delta]\).

Example

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

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

\[
\begin{align*}
y &\geq 2-\delta, \ y := 0
\end{align*}
\]
Enlarged semantics for timed automata

A transition can be taken at any time in \([t - \delta; t + \delta]\).

Example

\[
x \in [1 - \delta, 1 + \delta] \quad \text{y} := 0
\]

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

\[
y \geq 2 - \delta, \quad y := 0
\]
Enlarged semantics for timed automata

A transition can be taken at any time in $[t - \delta; t + \delta]$.

Example

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

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

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

\[ y \geq 2 - \delta, \quad y := 0 \]
Enlarged semantics for timed automata

A transition can be taken at any time in \([t - \delta; t + \delta]\).

Example

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

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

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

\[
x \leq \delta \land y \geq 2 - \delta
\]
Enlarged semantics for timed automata

A transition can be taken at any time in $[t - \delta; t + \delta]$. 

Example

$x \in [1-\delta, 1+\delta]$

$x \leq 2+\delta, \ x:=0$

$x \leq \delta \land y \geq 2-\delta$

$y := 0$

$y \geq 2-\delta, \ y:=0$
Enlarged semantics for timed automata

A transition can be taken at any time in $[t - \delta; t + \delta]$. 

Example

$x \in [1 - \delta, 1 + \delta]$

$y := 0$

$x \leq 2 + \delta, x := 0$

$y \geq 2 - \delta, y := 0$

$x \leq \delta \land y \geq 2 - \delta$
Enlarged semantics for timed automata

A transition can be taken at any time in \([t - \delta; t + \delta]\).

Example

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

\[
\begin{align*}
x & \leq 2 + \delta, \ x := 0 \\
y & \geq 2 - \delta, \ y := 0
\end{align*}
\]
Enlarged semantics for timed automata

A transition can be taken at any time in \([t - \delta; t + \delta]\).

Example

Theorem ([Pur98, DDMR04])

Parametrized robust safety is decidable.
Extended region automaton

For any location \(\ell\) and any two regions \(r\) and \(r'\), if
- \(\overline{r} \cap \overline{r'} \neq \emptyset\) and
- \((\ell, r')\) belongs to an SCC of \(\mathcal{R}(A)\),
then we add a transition \((\ell, r) \xrightarrow{\gamma} (\ell, r')\).
Extended region automaton

For any location $\ell$ and any two regions $r$ and $r'$, if

- $\overline{r} \cap \overline{r'} \neq \emptyset$ and
- $(\ell, r')$ belongs to an SCC of $\mathcal{R}(A)$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 
For any location $\ell$ and any two regions $r$ and $r'$, if

- $\overline{r} \cap \overline{r'} \neq \emptyset$ and
- $(\ell, r')$ belongs to an SCC of $\mathcal{R}(A)$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 

![Diagram of extended region automaton]
Extended region automaton

For any location \( \ell \) and any two regions \( r \) and \( r' \), if
- \( r \cap r' \neq \emptyset \) and
- \( (\ell, r') \) belongs to an SCC of \( R(A) \),
then we add a transition \((\ell, r) \xrightarrow{\gamma} (\ell, r')\).
Extended region automaton

For any location $\ell$ and any two regions $r$ and $r'$, if

1. $\overline{r} \cap \overline{r'} \neq \emptyset$ and
2. $(\ell, r')$ belongs to an SCC of $R(A)$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 

\begin{tikzpicture}
  \draw[->] (0,0) -- (0,3) node[above] {$y$};
  \draw[->] (0,0) -- (3,0) node[right] {$x$};
  \fill[blue!30] (0,0) -- (3,3) -- (0,3) -- cycle;
  \fill[green!30] (0,0) -- (0,3) -- (3,0) -- cycle;
  \draw[<->,thick] (0,0) -- (3,3);
  \draw[->] (1,1) to[bend left] (2,2);\node at (1.5,1.5) {$\gamma$};
\end{tikzpicture}
Extended region automaton

For any location $\ell$ and any two regions $r$ and $r'$, if

1. $\overline{r} \cap \overline{r'} \neq \emptyset$ and
2. $(\ell, r')$ belongs to an SCC of $R(A)$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 

\begin{center}
\begin{tikzpicture}
\draw[->] (0,0) -- (4,0) node[below] {$x$};
\draw[->] (0,0) -- (0,4) node[left] {$y$};
\draw[thick] (0,0) -- (4,4); \draw[thick] (0,4) -- (4,0);
\draw[dotted] (0,0) -- (4,0) -- (4,4) -- (0,4) -- (0,0);
\fill[blue!20] (0,0) rectangle (3,3);
\fill[green!20] (1,0) rectangle (4,1);
\fill[red!20] (2,0) rectangle (3,2);
\fill[green!20] (0,1) rectangle (3,3);
\fill[blue!20] (0,2) rectangle (3,3);
\fill[red!20] (0,3) rectangle (3,3);
\draw[thick,->, bend angle=45] (1.5,1.5) to (2,2);
\end{tikzpicture}
\end{center}
Shrinking timing constraints

Counteracting guard enlargement

**Shrinking turns constraints** $[a, b]$ **into** $[a + \delta, b - \delta]$.  

In particular, **punctual constraints** become empty.
Shrinking timing constraints

Counteracting guard enlargement

\textbf{Shrinking turns constraints} \([a, b]\) \textbf{into} \([a + \delta, b - \delta]\).

In particular, punctual constraints become empty.

Definition

A timed automaton is shrinkable if, for some \(\delta > 0\), its shrunk automaton (time-abstract) simulates the original automaton.

Theorem ([SBM11])

\textit{Shrinkability is decidable in EXPTIME.}
Shrinking timing constraints

**Counteracting guard enlargement**

Shrinking turns constraints \([a, b]\) into \([a + \delta, b - \delta]\).

In particular, punctual constraints become empty.

**Definition**

A timed automaton is shrinkable if, for some \(\delta > 0\), its shrunk automaton (time-abstract) simulates the original automaton.

**Theorem ([SBM11])**

*Shrinkability is decidable in EXPTIME.*

Main tools: parametrized shrunk DBMs max-plus fixpoint equations
Shrinking timing constraints

Counteracting guard enlargement

**Shrinking turns constraints** $[a, b]$ into $[a + \delta, b - \delta]$.

In particular, punctual constraints become empty.

**Definition**

A timed automaton is shrinkable if, for some $\delta > 0$, its shrunk automaton (time-abstract) simulates the original automaton.

**Theorem ([SBM11])**

*Shrinkability is decidable in EXPTIME.*

∽ prototype tool:

Shrinking timing constraints

Example

\[ x \leq 2 - k_5 \delta \]
\[ y := 0 \]

\[ 2 - k_1 \delta \leq x \leq 4 - k_2 \delta \]
\[ 2 - k_3 \delta \leq y \leq 4 - k_4 \delta \]

\[ k_5 \] = \max(\[ k_5 \], \[ k_2 + k_3 \])
Shrinking timing constraints

Example

\[ x \leq 2 - k_5 \delta \]
\[ y := 0 \]

\[ 2 - k_1 \delta \leq x \leq 4 - k_2 \delta \]
\[ 2 - k_3 \delta \leq y \leq 4 - k_4 \delta \]

\[ \subseteq \text{Unreset}_y \]

\[ \text{Pre}_{\text{time}} \]

\[ k_3 \delta \]
\[ k_1 \delta \]
\[ k_4 \delta \]
\[ k_2 \delta \]
Shrinking timing constraints

Example

\[ x \leq 2 - k_5 \delta \]
\[ y := 0 \]
\[ 2 - k_1 \delta \leq x \leq 4 - k_2 \delta \]
\[ 2 - k_3 \delta \leq y \leq 4 - k_4 \delta \]

\[ k_5 \delta \subseteq \text{Unreset}_y \]

\[ (k_2 + k_3) \delta \]
Shrinking timing constraints

Example

\[ x \leq 2 - k_5 \delta \]
\[ y := 0 \]
\[ 2 - k_1 \delta \leq x \leq 4 - k_2 \delta \]
\[ 2 - k_3 \delta \leq y \leq 4 - k_4 \delta \]

\[ k_5 \delta \subseteq (k_2 + k_3) \delta \]

\[ k_5 \delta \]

\[ (k_2 + k_3) \delta \]
Shrinking timing constraints

Example

\[ x \leq 2 - k_5 \delta \]
\[ y := 0 \]
\[ 2 - k_1 \delta \leq x \leq 4 - k_2 \delta \]
\[ 2 - k_3 \delta \leq y \leq 4 - k_4 \delta \]

\[ k_5 \delta \subseteq (k_2 + k_3) \delta \]

\[ \sim \]

\[ k_5 = \max(k_5, k_2 + k_3) \]
Outline of the talk

1. Discrete time vs. dense time
2. From models to implementations
3. Checking robust safety
   - Enlarging clock constraints
   - Shrinking clock constraints
4. Checking robust controllability
   - Parametrized perturbations
   - Permissive strategies
5. Conclusions and future works
Game-based approach to robustness

Solving robust reachability

- Player 1 proposes a delay $d$ and a transition $t$;
- transition $t$ is taken after some delay in $[d - \delta, d + \delta]$ chosen by Player 2.
Game-based approach to robustness

Solving robust reachability

- Player 1 proposes a delay $d$ and a transition $t$;
- transition $t$ is taken after some delay in $[d - \delta, d + \delta]$ chosen by Player 2.

Consider a transition with guard $x \leq 3 \land y \geq 1$:

**loose semantics**

**strict semantics**
Solving robust reachability

- Player 1 proposes a delay $d$ and a transition $t$;
- transition $t$ is taken after some delay in $[d - \delta, d + \delta]$ chosen by Player 2.

Theorem ([BMS12,SBMR13])

Robust reachability is EXPTIME-complete in the loose semantics.

Robust reachability and repeated reachability are PSPACE-complete in the strict semantics.
Shrink DBMs for the loose semantics

Extend the region automaton into a 2-player turn-based game

\[ x = y = 1 \quad y := 0 \]
Shrunk DBMs for the loose semantics

Extend the region automaton into a 2-player turn-based game

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

Graphical representation:

- States: \( r_0, r_1, r_2, r_3 \)
- Transitions:
  - \( r_0 \rightarrow r_0' \)
  - \( r_0' \rightarrow r_1, s_1 \)
  - \( r_0' \rightarrow r_2, s_2 \)
  - \( r_0' \rightarrow r_3, s_3 \)
Orbit graphs for the strict semantics

\[ \ell_0 \xrightarrow{1 < x < 2} \ell_1 \xrightarrow{y := 0} \ell_2 \xrightarrow{x \leq 2, x := 0} \ell_0 \]

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

**Definition**

A cycle \( \pi \) is forgetful if its orbit graph is strongly connected.

A cycle \( \pi \) is aperiodic if \( \pi^k \) is forgetful, for all \( k \).
Orbit graphs for the strict semantics

Definition

A cycle $\pi$ is forgetful if its orbit graph is strongly connected.

A cycle $\pi$ is aperiodic if $\pi^k$ is forgetful, for all $k$. 
Orbit graphs for the strict semantics

Definition

A cycle $\pi$ is forgetful if its orbit graph is strongly connected.

A cycle $\pi$ is aperiodic if $\pi^k$ is forgetful, for all $k$. 
Orbit graphs for the strict semantics

**Definition**
A cycle $\pi$ is forgetful if its orbit graph is strongly connected.

A cycle $\pi$ is aperiodic if $\pi^k$ is forgetful, for all $k$.

**Theorem**
The automaton is robustly controllable if, and only if, it has a reachable aperiodic cycle.
Permissive strategies can propose several moves rather than a single one.
Permissive strategies can propose several moves rather than a single one.
Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the untimed setting... [BDMR09, BMOU11]
Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the untimed setting... [BDMR09, BMOU11]

Diagram showing the relationships between nodes a, b, c, and d with various edges and weights.
Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the timed setting...

Permissive strategies propose intervals of delays.

Our setting:

the penalty assigned to interval $[a, b]$ is $1/(b - a)$. 
Permissive strategies can propose several moves rather than a single one.

In the timed setting...

Possible (memoryless) strategy:
Permissive strategies can propose several moves rather than a single one.

In the timed setting...

**Possible (memoryless) strategy:**

- **in** $\ell_0$, play $(a, [0, 2))$;
- **in** $\ell_1$:
  - if $x \leq 1$, play $(b, [0, 1 - x])$;
  - otherwise, play $(a, [0, 2 - x])$;
- **in** $\ell_2$, play $(b, [0, +\infty))$
Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the timed setting...

Possible (memoryless) strategy:

- in ℓ₀, play (a, [0, 2));
- in ℓ₁:
  - if x ≤ 1, play (b, [0, 1 − x]);
  - otherwise, play (a, [0, 2 − x]);
- in ℓ₂, play (b, [0, +∞))

\[\text{penalty} = +\infty\]
Permissive strategies can propose several moves rather than a single one.

In the timed setting...

Possible (memoryless) strategy:

- in $\ell_0$, play $(a, [0, 1])$;
- in $\ell_1$:
  - if $x = 0$, play $(b, [0, 1])$;
  - otherwise, play $(a, [0, 2 - x])$;
- in $\ell_2$, play $(b, [0, +\infty ))$
Synthesizing permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the timed setting...

Possible (memoryless) strategy:
- in $\ell_0$, play $(a, [0, 1])$;
- in $\ell_1$:
  - if $x = 0$, play $(b, [0, 1])$;
  - otherwise, play $(a, [0, 2 - x])$;
- in $\ell_2$, play $(b, [0, +\infty )$)

\[ \sim \text{ penalty} = 1 \]
Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the timed setting...

Theorem

For one-clock timed games:

- Memoryless optimal-penalty strategies exist.
- They can be computed in polynomial time.
Outline of the talk

1. Discrete time vs. dense time
2. From models to implementations
3. Checking robust safety
   - Enlarging clock constraints
   - Shrinking clock constraints
4. Checking robust controllability
   - Parametrized perturbations
   - Permissive strategies
5. Conclusions and future works
Conclusion and challenges

Conclusions

Robustness issues identified long ago...
Several attempts, but no satisfactory solution yet!
Conclusion and challenges

Conclusions

Robustness issues identified long ago...

Several attempts, but no satisfactory solution yet!

Challenges and open questions

- symbolic algorithms;
- measuring robustness, using distances between automata;
  - link between “syntactic distance” and “semantic distance”
- probabilistic approach to robustness;
  - evaluate expected time before a new state is visited.
- investigate robustness in weighted timed automata;
  - energy constraints;
  - imprecision on cost rates;
- synthesis of robust strategies.