Robustness of 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)

Quantitative models: expressiveness, analysis, and new applications
January 2014
Reasoning about real-time systems

Timed automata [AD90]

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

Example (A computer mouse)

- **idle**
  - left_button?
  - left_click!
  - left_double_click!
  - right_button?
  - right_click!
  - right_double_click!

- **left**
  - left_button?
  - left_click!
  - left_double_click!

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

Timed automata [AD90]

A timed automaton is made of
- a transition system,

Example (A computer mouse)

- left
  - left_button?
  - left_click!
  - left_double_click!

- idle
  - left_button?
  - right_button?
  - right_click!

- right
  - right_button?
  - 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?

idle

left_click!
left_button?
left_double_click!

right → right_button?

right_click!
right_button?
right_double_click!
```

idle

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)

```plaintext

left
  x = 300
  left_button?
  x ≤ 300
  x := 0
  left_double_click!

idle
  x = 300
  left_button?
  x ≤ 300

right
  x = 300
  right_button?
  x ≤ 300
  right_double_click!

right_button?
  x := 0
```

idle
  x := 0

left_button?
  x := 0
```

right_double_click!
```

left_double_click!
```
Discrete-time semantics

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

...because computers are digital!

Example ([BS91])

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

...because computers are digital!

Example ([BS91])

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

...because computers are digital!

Example ([BS91])

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

...because computers are digital!

Example ([BS91])

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

...because computers are digital!

Example ([BS91])

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

...because computers are digital!

Example ([BS91])

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

...because computers are digital!

Example ([BS91])

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

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

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

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

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

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

Let's consider the following system:

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

Theorem ([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
dd{x} &\leq 2, \ x := 0 \\
y &\geq 2, \ y := 0
dd{x} &= 0 \land \
y &\geq 2
\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

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

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

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

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

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

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

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

\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

\[
\begin{align*}
\text{Example} & : \quad x=1 & \implies y:=0 \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{\textcolor{green}{$x \leq 2$, $x:=0$}} \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{\textcolor{blue}{$y \geq 2$, $y:=0$}} \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{\textcolor{red}{$x=0 \land y \geq 2$}} \\
\end{align*}
\]

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

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

Reachability in timed automata is decidable (as well as many other important properties).
Regions and zones

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

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

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

\[ y \geq 2, \quad y = 0 \]
Regions and zones

- $x=1, \ y:=0$
- $x \leq 2, \ x:=0$
- $y \geq 2, \ y:=0$
- $x=0 \land y \geq 2$
Zones are a coarser abstraction:

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

Zones are a coarser abstraction:

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

Representation as DBM:

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

\[
\text{Pre}_{\text{time}} \left( \bigcap \text{Unreset}_y \right)
\]
The predecessors of \((\ell_2, x \leq 3 \land y - x \leq 0)\) are computed as

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

\(\sim\) efficient implementations
Continuous-time semantics

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.
Continuous-time semantics

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)

\[ x < 1 \land y < 1 \]
\[ x := 0 \]
\[ y = 1 \]
Continuous-time semantics

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)
Continuous-time semantics

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 $\square$ is not reachable.

This property is lost when $x_{id} > 2$ is replaced with $x_{id} \geq 2$. 
Continuous-time semantics

- 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?
Continuous-time semantics

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]
Continuous-time semantics

- 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. Introduction

2. Several approaches to robustness issues in timed automata
   - Enlarging timing constraints
   - Shrinking timing constraints
   - Game-based approach

3. Extensions to richer settings
   - Robustness in weighted timed automata
   - Synthesizing robust strategies

4. Conclusion
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

\[
\begin{align*}
x &= 1 \\
y &:= 0 \\
x &\leq 2, \ x := 0 \\
y &\geq 2, \ y := 0 \\
x &= 0 \land y \geq 2
\end{align*}
\]
Enlarged semantics for timed automata

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

**Example**

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

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

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

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

Example

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

![Diagram](image_url)
For any location \( \ell \) and any two regions \( r \) and \( r' \), if
\[
\overline{r} \cap \overline{r'} \neq \emptyset \quad \text{and} \\
(\ell, r') \text{ belongs to an SCC of } R(A),
\]
then we add a transition \((\ell, r) \rightarrow (\ell, r')\).
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 $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 
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 $R(A)$,

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

![Diagram of a 3x3 grid with regions and transitions labeled with $\gamma$]
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 $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 

\[\begin{array}{c}
\gamma \\
\end{array}\]
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 $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 
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

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.
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 tool: parametrized shrunk DBMs:
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 \(\text{EXPTIME}\).

\(\leadsto\) prototype tool:

http://www.lsv.ens-cachan.fr/Software/shrinktech/
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**
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.

Theorem ([BMS12,SBMR13])

Robust reachability is EXPTIME-complete in the loose semantics.

Robust reachability and repeated reachability are PSPACE-complete in the strict semantics.
Robustness in weighted timed automata

Imprecisions could make verification easier

Hardness/undecidability proofs in weighted timed automata need arbitrary precision.

These proofs do not carry on in a robust setting!
Robustness in weighted timed automata

Imprecisions could make verification easier

Hardness/undecidability proofs in weighted timed automata need arbitrary precision.

These proofs do not carry on in a robust setting!

Sadly, this is not the case of the game-based approach:

**Theorem**

Robust optimal reachability is PSPACE-complete for the strict semantics.

Robust optimal reachability is undecidable for the loose semantics.

Robust optimal reachability games are undecidable under both semantics.
<table>
<thead>
<tr>
<th>Permissive strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissive strategies can propose several moves rather than a single one.</td>
</tr>
</tbody>
</table>
Synthesizing robust strategies

**Permissive strategies**

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

In the untimed setting... [BDMR09, BMOU11]
Synthesizing robust strategies

Permissive strategies

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

In the untimed setting... [BDMR09, BMOU11]
Synthesizing robust 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 timed setting...

Permissive strategies propose intervals of delays.

Our setting:

the penalty assigned to interval \([a, b]\) is \(1/(b - a)\).
Permissive strategies

Permissive strategies can propose _several moves_ rather than a single one.

In the timed setting...

- From $l_0$: $a, x \geq 2$ (sad face)
- From $l_0$: $a, x < 2$
- From $l_1$: $b, x \leq 1$ (happy face)
- From $l_1$: $b, x := 0$
- From $l_2$: $a, x \leq 2$
- From $l_2$: $a, x \leq 2$
Permissive strategies

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

In the timed setting...

Possible (memoryless) strategy:
- in $\ell_0$, play $(a, [0, 2))$;
- 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 $\ell_0$, 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 robust strategies

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

In the timed setting...

Theorem
Optimal-penalty strategies are computable for one-clock timed games.
Conclusion and challenges

Conclusions

Robustness issues identified long ago...

Several relevant approaches, but no decent tool support.
Conclusion and challenges

Conclusions

Robustness issues identified long ago...

Several relevant approaches, but no decent tool support.

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.