Robustness in real-time systems

Nicolas Markey

LSV, CNRS & ENS Cachan, France

SIES’11 – June 15, 2011
Verification of (real-time) computerized systems

system:

property:

Always safe

model-checking algorithm

yes/no
Verification of (real-time) computerized systems

system:

property:

Always safe

model-checking algorithm

t ≤ 5

yes/no
Timed automata (AD90)

A timed automaton is made of

- a transition system,
Timed automata

Timed automata (AD90)

A **timed automaton** is made of
- a transition system,
- a set of clocks,

Example

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

Timed automata (AD90)

A **timed automaton** is made of

- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

Example

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

Timed automata (AD90)

A timed automaton is made of
- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

Example

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

Timed automata (AD90)

A timed automaton is made of
- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

Example

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

\( x \)
\( y \)
\( 0 \)
\( 1 \)
\( 2 \)
Timed automata

Timed automata (AD90)

A timed automaton is made of
- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

Example

\[
\begin{align*}
  x &= 1 \\
  y &= 0 \\
  y &\leq 2, \quad x := 0 \\
  y &\geq 2, \quad y := 0 \\
  x &= 0 \land \\
  y &\geq 2
\end{align*}
\]
A **timed automaton** is made of
- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

**Example**

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

Timed automata (AD90)

A timed automaton is made of
- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

Example

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

Timed automata (AD90)

A timed automaton is made of

- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

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*}
Timed automata

Timed automata (AD90)

A timed automaton is made of

- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

Example

A timed automaton with transitions:

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

Graphical representation of the automaton with variable $x$ and $y$. The diagram shows the automaton's states and transitions.
Timed automata

Timed automata (AD90)

A timed automaton is made of
- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

Example

\[
\begin{align*}
&x = 1, \ y := 0 \\
\implies &x \leq 2, \ x := 0 \\
\implies &y \geq 2, \ y := 0 \\
\implies &x = 0 \land y \geq 2
\end{align*}
\]
Timed automata (AD90)

A timed automaton is made of:
- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

Example

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

Timed automata (AD90)

A timed automaton is made of

- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

Example

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

Timed automata (AD90)

A timed automaton is made of
- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

Example

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

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

Timed automata (AD90)

A timed automaton is made of

- a transition system,
- a set of clocks,
- a labelling of transitions with timing informations.

Example

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

Example

Theorem (AD90)

Reachability (and $!$-regular properties) in timed automata can be checked in exponential time (and are PSPACE-complete).
Theorem (AD90)

Reachability (and \( \omega \)-regular properties) in timed automata can be checked in exponential time (and are PSPACE-complete).
Analysing timed automata in practice

- symbolic algorithms (using zones)
- efficient implementations (Uppaal, Kronos, ...)

![Diagram of timed automata analysis](image-url)
Outline of the presentation

1. Introduction – Timed automata
2. Robustness issues in timed automata
3. Several approaches
   - Tube semantics
   - Probabilistic semantics
   - Sampled semantics
4. Enlarged semantics
   - A different approach
   - Checking robustness against enlargement
   - Making timed automata robust
5. Conclusions and perspectives
Outline of the presentation

1. Introduction – Timed automata

2. Robustness issues in timed automata

3. Several approaches
   - Tube semantics
   - Probabilistic semantics
   - Sampled semantics

4. Enlarged semantics
   - A different approach
   - Checking robustness against enlargement
   - Making timed automata robust

5. Conclusions and perspectives
Robustness issues in timed automata

Zeno behaviours

$x < 1 \land y < 1$

$x := 0$

$y = 1$
Robustness issues in timed automata

Zeno behaviours

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

Theorem (AD90)

Checking \( \omega \)-regular properties under non-Zenoness requirement can be done in exponential time.
Robustness issues in timed automata

Convergence phenomena (CHR02)

\[ x \leq 1 \]
\[ y := 0 \]
\[ y = 1 \]
\[ z := 0 \]
\[ z > 0 \]

Diagram:

- Start state: \( x \leq 1 \)
- Transition: \( x := 0 \) to \( x \leq 1 \)
- Transition: \( y := 0 \) to \( y = 1 \)
- Transition: \( z := 0 \) to \( z > 0 \)

Graph:

- Axes: \( x \) and \( y \)
- Points: \( (0, 0) \), \( (1, 1) \)
Robustness issues in timed automata

Convergence phenomena (CHR02)

Diagram:

- Initial state: $x \leq 1$
- Transition: $x = 1$
- Next state: $y := 0$
- Transition: $y = 1$
- Next state: $z := 0$
- Final state: $x \leq 1$

Graph:

- Axes: $x$ and $y$
- Origin: $(0, 0)$
- Points: $(1, 1)$
- Line: $y = x$
Robustness issues in timed automata

Convergence phenomena (CHR02)
Robustness issues in timed automata

Convergence phenomena (CHR02)

\[
x \leq 1
\]

\[
x = 1
\]

\[
x := 0
\]

\[
y := 0
\]

\[
y = 1
\]

\[
 y := 0
\]

\[
z := 0
\]

\[
z > 0
\]

\[
x \leq 1
\]

\[
x \leq 1
\]

\[
x \leq 1
\]

\[
y\]

\[
0 1
\]

\[
x
\]

\[
1
\]
Robustness issues in timed automata

Convergence phenomena (CHR02)

\[ x \leq 1 \]
\[ x := 0 \]
\[ y := 0 \]
\[ y = 1 \]
\[ z := 0 \]
\[ z > 0 \]
\[ x = 1 \]
\[ x \leq 1 \]
\[ x \leq 1 \]
Robustness issues in timed automata

Convergence phenomena (CHR02)

Graphical representation of the system:
- Initial state: \( x \leq 1 \)
- Transition: \( x = 1 \) leads to \( y := 0 \)
- Transition: \( y = 1 \) leads to \( z := 0 \)
- Final state: \( x \leq 1 \)

Graph showing the relationship between \( x \) and \( y \):
Robustness issues in timed automata

Convergence phenomena (CHR02)

$\begin{align*}
  &x \leq 1 \\
  &x := 0 & y := 0 & z := 0 & y = 1 & x \leq 1 \\
  &y := 0 & z > 0 & x = 1 & x := 0 & x \leq 1
\end{align*}$
Robustness issues in timed automata

Convergence phenomena (CHR02)

$\begin{align*}
x &\leq 1 \\
x &= 1 & x &\leq 1 \\
x &:= 0 & x &\leq 1 & y &= 0 \\
y &= 1 & y &:= 0 \\
z &:= 0 & z &> 0 & x &\leq 1 \\
z &= 1 & z &:= 0 & x &\leq 1 \\
y &:= 0 \\
y &= 1 \\
0 &\leq x & x &\leq 1 & 1 &\leq y & y &\leq 1 & 1 &\leq z & z &\leq 1
\end{align*}$
Robustness issues in timed automata

Convergence phenomena (CHR02)

\[ x \leq 1 \quad x := 0 \quad x \leq 1 \quad y := 0 \quad y = 1 \quad z := 0 \quad z > 0 \]

\[ y := 0 \quad y = 1 \quad z := 0 \quad z > 0 \]

\[ x \leq 1 \quad x := 0 \quad x \leq 1 \quad y := 0 \quad y = 1 \quad z := 0 \quad z > 0 \]

\[ 0 \quad 1 \quad 1 \]

\[ x \quad y \]
Robustness issues in timed automata

Theorem (KLL + 97)

When \( P_1 \) and \( P_2 \) run in parallel (sharing variable \( r \)), the state where both of them are in is not reachable. But this property is lost when \( x_{id} > 2 \) is replaced with \( x_{id} \geq 2 \).
Robustness issues in timed automata

Strict timing constraints

\[ P_{id} \]

\[ r := 0 \]
\[ x_{id} := 0 \quad \text{if} \quad x_{id} \leq 2 \]
\[ r := id \]
\[ x_{id} := 0 \]
\[ x_{id} > 2 \]

Theorem (KLL\textsuperscript{+}97)

When \( P_1 \) and \( P_2 \) run in parallel (sharing variable \( r \)), the state where both of them are in \( \square \) is not reachable.
Robustness issues in timed automata

Strict timing constraints

$P_{id}$

\[ \begin{align*}
    x_{id} &:= 0 \\
    x_{id} &:= 0 \\
    x_{id} &:= 0 \\
    r &= 0 \\
    r &= 0 \\
    r &= 0 \\
    r &= 0 \\
    x_{id} &> 2
\end{align*} \]

**Theorem (KLL$^+$97)**

When $P_1$ and $P_2$ run in parallel (sharing variable $r$), the state where both of them are in is not reachable.

But this property is lost when $x_{id} > 2$ is replaced with $x_{id} \geq 2$. 
Robustness issues in timed automata

Imprecision on clock values (ACS10)
Robustness issues in timed automata

Imprecision on clock values (ACS10)

2 t.u.

frame 0  frame 1  frame 2  frame 3  frame 4  frame 5

encod. 0  encod. 1  encod. 2  encod. 3  encod. 4

2 + ε
Outline of the presentation

1. Introduction – Timed automata

2. Robustness issues in timed automata

3. Several approaches
   - Tube semantics
   - Probabilistic semantics
   - Sampled semantics

4. Enlarged semantics
   - A different approach
   - Checking robustness against enlargement
   - Making timed automata robust

5. Conclusions and perspectives
Several solutions have been proposed...

**Tube semantics (GHJ97)**

- discards behaviours that have too strict constraints;
- only consider traces whose neighbouring traces are accepted;
- safety is decidable.
Several solutions have been proposed...

**Tube semantics (GHJ97)**
- discards behaviours that have too strict constraints;
- only consider traces whose neighbouring traces are accepted;
- safety is decidable.

**Probabilistic semantics (BBBBB07)**
- defines a measure on traces;
- discards *unlikely* behaviours;
- safety is decidable.
Several solutions have been proposed...

Sampled semantics (HMP92, AKY10)

- actions are taken only at integer multiples of $\tau$;
- conceptually simpler to handle, but checking safety still takes exponential time;

Samplability

A timed automaton $A$ is samplable if there exists $\tau > 0$ s.t. $A$ exhibits similar (untimed) behaviours under the classical semantics as under the $\tau$-sampled semantics.

Theorem (AKY10)

Samplability is decidable.
Several solutions have been proposed...

Sampled semantics (HMP92,AKY10)

- actions are taken only at integer multiples of $\tau$;
- conceptually simpler to handle, but checking safety still takes exponential time;

Samplability

A timed automaton $A$ is samplable if there exists $\tau > 0$ s.t. $A$ exhibits similar (untimed) behaviours under the classical semantics as under the $\tau$-sampled semantics.
Several solutions have been proposed...

**Sampled semantics (HMP92,AKY10)**

- actions are taken only at integer multiples of $\tau$;
- conceptually simpler to handle, but checking safety still takes exponential time;

**Samplability**

A timed automaton $\mathcal{A}$ is **samplable** if there exists $\tau > 0$ s.t. $\mathcal{A}$ exhibits similar (untimed) behaviours under the classical semantics as under the $\tau$-sampled semantics.

**Theorem (AKY10)**

Samplability is decidable.
Outline of the presentation

1. Introduction – Timed automata
2. Robustness issues in timed automata
3. Several approaches
   - Tube semantics
   - Probabilistic semantics
   - Sampled semantics
4. Enlarged semantics
   - A different approach
   - Checking robustness against enlargement
   - Making timed automata robust
5. Conclusions and perspectives
A different solution...

Enlarged semantics (Pur98)

- clocks evolve at rate in $[1 - \varepsilon, 1 + \varepsilon]$ instead of exactly 1;
- clock constraints $x \in [a, b]$ replaced with $x \in [a - \delta, b + \delta]$;
- contrary to the other approaches, this semantics adds extra behaviours, considering that the classical semantics is too precise.

Robustness

A timed automaton $A$ is robust if there exist $\delta > 0$ and/or $\varepsilon > 0$ s.t. $A$ exhibits similar (untimed) behaviours under the classical semantics as under the enlarged semantics.

Theorem (Pur98, DDMR04, BMR06, San11)
Robustness is decidable.
A different solution...

**Enlarged semantics (Pur98)**

- clocks evolve at rate in \([1 - \epsilon, 1 + \epsilon]\) instead of exactly 1;
- clock constraints \(x \in [a, b]\) replaced with \(x \in [a - \delta, b + \delta]\);
- contrary to the other approaches, this semantics adds extra behaviours, considering that the classical semantics is too precise.

**Robustness**

A timed automaton \(\mathcal{A}\) is **robust** if there exist \(\epsilon > 0\) and/or \(\delta > 0\) s.t. \(\mathcal{A}\) exhibits similar (untimed) behaviours under the classical semantics as under the enlarged semantics.
A different solution...

Enlarged semantics (Pur98)

- clocks evolve at rate in \([1 - \epsilon, 1 + \epsilon]\) instead of exactly 1;
- clock constraints \(x \in [a, b]\) replaced with \(x \in [a - \delta, b + \delta]\);
- contrary to the other approaches, this semantics adds extra behaviours, considering that the classical semantics is too precise.

Robustness

A timed automaton \(\mathcal{A}\) is robust if there exist \(\epsilon > 0\) and/or \(\delta > 0\) s.t. \(\mathcal{A}\) exhibits similar (untimed) behaviours under the classical semantics as under the enlarged semantics.

Theorem (Pur98,DDMR04,BMR06,San11)

Robustness is decidable.
What happens under the (guard-)enlarged semantics?

Example

\[
\begin{align*}
\text{y} & \geq 2, \quad \text{y} := 0 \\
x & = 0 \wedge \text{y} \geq 2
\end{align*}
\]
What happens under the (guard-)enlarged semantics?

Example

\[ x = 1 \]

\[ y := 0 \]

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

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

\[ x = 0 \land y \geq 2 \]
What happens under the (guard-)enlarged semantics?

Example

\[
\begin{align*}
  x &\in [1-\delta, 1+\delta] \\
  y &:= 0 \\
  x &\leq 2+\delta, \ x:=0 \\
  y &\geq 2-\delta, \ y:=0 \\
  x &\leq \delta \land y \geq 2-\delta
\end{align*}
\]
What happens under the (guard-)enlarged semantics?

Example

\[ x \in [1-\delta, 1+\delta] \]
\[ y := 0 \]
\[ x \leq 2+\delta, \ x := 0 \]
\[ y \geq 2-\delta, \ y := 0 \]
\[ x \leq \delta \land y \geq 2-\delta \]
What happens under the (guard-)enlarged semantics?

Example

\[ x \in [1-\delta, 1+\delta] \]
\[ y := 0 \]
\[ x \leq 2+\delta, \ x:=0 \]
\[ y \geq 2-\delta, \ y:=0 \]
\[ x \leq \delta \land y \geq 2-\delta \]
What happens under the (guard-)enlarged semantics?

Example

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

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

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

\[y \geq 2-\delta, \ y := 0\]
What happens under the (guard-)enlarged semantics?

Example

\[
\begin{align*}
  x &\in [1-\delta, 1+\delta] \quad y := 0 \\
  x \leq 2+\delta, \quad x := 0 &\quad y \geq 2-\delta, \quad y := 0 \\
  x \leq \delta &\land y \geq 2-\delta
\end{align*}
\]
What happens under the (guard-)enlarged semantics?

Example

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

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

\[ x \leq \delta \land y \geq 2-\delta \]
What happens under the (guard-)enlarged semantics?

Example

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

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

\[ x \leq \delta \land y \geq 2 - \delta \]
What happens under the (guard-)enlarged semantics?

Example

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

\[ x \leq 2+\delta, \ x := 0 \]
\[ y \geq 2-\delta, \ y := 0 \]
What happens under the (guard-)enlarged semantics?

Example

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

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

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

\[ y \geq 2-\delta, \ y := 0 \]
What happens under the (guard-)enlarged semantics?

Example

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

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

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

\[ x \leq \delta \land y \geq 2 - \delta \]
What happens under the (guard-)enlarged semantics?

**Example**

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

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

\[
y \geq 2-\delta, \ y := 0
\]
What happens under the (guard-)enlarged semantics?

Example

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

\[ y := 0 \]

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

\[ y \geq 2-\delta, \ y := 0 \]
Safety checking under the enlarged semantics

**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 $\mathcal{R}(A)$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 
Safety checking under the enlarged semantics

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 $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 

\[ 
\begin{array}{c}
| y |
\hline
| 3 |
| 2 |
| 1 |
| 0 |
\end{array} 
\]  
\[
\begin{array}{c}
| x |
\hline
| 0 |
| 1 |
| 2 |
| 3 |
\end{array} 
\]
Safety checking under the enlarged semantics

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 $\mathcal{R}(A)$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 
Safety checking under the enlarged semantics

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 $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 
Safety checking under the enlarged semantics

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 $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 
Safety checking under the enlarged semantics

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 $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 

---

Diagram:

The diagram illustrates a region with coordinates $x$ and $y$, ranging from 0 to 3. The region is divided into three sections, each shaded with a different color. The transition $\gamma$ is shown as an arrow moving from one shaded section to another.
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 $(\ell, r) \xrightarrow{\gamma} (\ell, r')$. 
Safety checking under the enlarged semantics

Example
Safety checking under the enlarged semantics

Example
Safety checking under the enlarged semantics

Example
Safety checking under the enlarged semantics

Example
Safety checking under the enlarged semantics

**Lemma**

The set of reachable regions in the *extended region automaton* is exactly

\[ \bigcap_{\delta > 0} \text{Reach}(A_{\delta}). \]

(under some technical restrictions)
Safety checking under the enlarged semantics

**Lemma**

The set of reachable regions in the extended region automaton is exactly

\[ \bigcap_{\delta > 0} \text{Reach}(A_{\delta}). \]

(under some technical restrictions)

**Lemma**

For any timed automata \( A \) and for any region \( B \),

\[ \bigcap_{\delta > 0} \text{Reach}_\delta(A) \cap B = \emptyset \quad \text{iff} \quad \exists \delta > 0. \text{Reach}_\delta(A) \cap B = \emptyset. \]
Safety checking under the enlarged semantics

Lemma
The set of reachable regions in the extended region automaton is exactly

$$\bigcap_{\delta > 0} \text{Reach}(A_{\delta}).$$

(under some technical restrictions)

Lemma
For any timed automata $A$ and for any region $B$,

$$\bigcap_{\delta > 0} \text{Reach}_{\delta}(A) \cap B = \emptyset \iff \exists \delta > 0. \text{Reach}_{\delta}(A) \cap B = \emptyset.$$

Theorem
Robust safety in timed automata is decidable in exponential time (and is PSPACE-complete).
Making timed automata robust
Example

This automaton is not robust:

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

But this one is:

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

Robustness is a syntactic criterion.
Making timed automata robust

Example

This automaton is not robust:

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

But this one is:

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

Robustness is a syntactic criterion.
Making timed automata robust

**Example**

This automaton is not robust:

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

But this one is:

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

Robustness is a **syntactic** criterion.
Making timed automata robust

\( \varepsilon \)-bisimilarity

\( \sim \subseteq S \times S \) is an \( \varepsilon \)-bisimulation if

\[
\begin{align*}
s & \overset{a}{\longrightarrow} t \\
\varepsilon & \\
\sim & \\
s' &
\end{align*}
\]

action transitions
Making timed automata robust

$\epsilon$-bisimilarity

$\sim \subseteq S \times S$ is an $\epsilon$-bisimulation if

$\begin{align*}
s \xrightarrow{a} t \\
\_ \quad \_ \\
s' \xrightarrow{a} t'
\end{align*}$

action transitions
Making timed automata robust

\( \varepsilon \)-bisimilarity

\( \sim \subseteq S \times S \) is an \( \varepsilon \)-bisimulation if

\[
\begin{align*}
&\text{action transitions} & &\text{delay transitions} \\
\sim & \text{ implies } a \rightarrow t \sim a \rightarrow t' & &d \rightarrow t \sim \varepsilon \rightarrow t \\
\sim & \text{ implies } \varepsilon \rightarrow s \sim \varepsilon \rightarrow s' & &s \sim \varepsilon \rightarrow t \sim s'
\end{align*}
\]
Making timed automata robust

$\varepsilon$-bisimilarity

$\sim \subseteq S \times S$ is an $\varepsilon$-bisimulation if

\[ s \xrightarrow{a} t \quad \sim \quad s' \xrightarrow{a} t' \]

\[ \varepsilon \quad \sim \quad \varepsilon \]

\[ s \xrightarrow{d} t \quad \sim \quad s' \xrightarrow{d'} t' \quad |d' - d| \leq \varepsilon \]

action transitions  
delay transitions
Making timed automata robust

$\varepsilon$-bisimilarity

$\sim \subseteq S \times S$ is an $\varepsilon$-bisimulation if

\[
\begin{align*}
&s \xrightarrow{a} t \\
&s' \xrightarrow{a} t' \\
&\sim \\
&s \xrightarrow{\varepsilon} t \\
&s' \xrightarrow{\varepsilon} t' \\
&s \xrightarrow{d} t \\
&s' \xrightarrow{d'} t' \quad |d' - d| \leq \varepsilon
\end{align*}
\]

action transitions delay transitions

Quantitative notion of robustness

A timed automaton $\mathcal{A}$ is $\varepsilon$-robust if there exists $\delta > 0$ s.t. $\mathcal{A}$ and its $\delta$-enlarged semantics $\mathcal{A}_\delta$ are $\varepsilon$-bisimilar.
Making timed automata robust

$\epsilon$-bisimilarity

$\sim \subseteq S \times S$ is an $\epsilon$-bisimulation if

- Action transitions:
  - $s \xrightarrow{a} t$
  - $s' \xrightarrow{a} t'$

- Delay transitions:
  - $s \xrightarrow{d} t$
  - $s' \xrightarrow{d'} t'$
  - $|d' - d| \leq \epsilon$

Theorem (BFL$^+$11)

Given a timed automaton $\mathcal{A}$ and $\epsilon > 0$, we can build a timed automaton $\mathcal{A}'$ s.t.

- $\mathcal{A}$ and $\mathcal{A}'$ are 0-bisimilar;
- $\mathcal{A}'$ is $\epsilon$-robust.
Outline of the presentation

1. Introduction – Timed automata
2. Robustness issues in timed automata
3. Several approaches
   - Tube semantics
   - Probabilistic semantics
   - Sampled semantics
4. Enlarged semantics
   - A different approach
   - Checking robustness against enlargement
   - Making timed automata robust
5. Conclusions and perspectives
Conclusions and perspectives

Robustness is an important issue in timed systems

- timed automata are governed by a mathematical semantics;
- this raises important robustness issues:
  - time-convergent behaviours;
  - strict timing constraints...
- several approaches:
  - ignoring isolated traces;
  - considering surrounding runs.

Perspectives

- develop the quantitative approach to robustness;
- probabilistic (as opposed to worst-case) enlargement;
- shrinking timed automata (to counteract enlargement);
- robust controller synthesis;
- robustness in priced timed automata (with energy constraints).
Conclusions and perspectives

Robustness is an important issue in timed systems
- timed automata are governed by a mathematical semantics;
- this raises important robustness issues:
  - time-convergent behaviours;
  - strict timing constraints...
- several approaches:
  - ignoring isolated traces;
  - considering surrounding runs.

Perspectives
- develop the quantitative approach to robustness;
- probabilistic (as opposed to worst-case) enlargement;
- shrinking timed automata (to counteract enlargement);
- robust controller synthesis;
- robustness in priced timed automata (with energy constraints).