Robust Reachability in Timed Automata: A Game-based Approach

Ocan Sankur

LSV, CNRS & ENS de Cachan

Joint with Patricia Bouyer and Nicolas Markey

will be presented at ICALP’12
Reachability in Timed Automata

Reachability problem

Given a timed automaton $A$, and a target location $\ell$, decide whether some (initial) run of $A$ visits $\ell$.

- PSPACE-complete [Alur & Dill ’94].

Applications of Reachability

1. Safety checking — does the system reaches a bad configuration?
2. Checking desired behaviour:
   - The system can terminate correctly.
   - Synthesize controller reaching a given state (resolve non-determinism).
Motivation: Scheduling

Scheduling analysis with timed automata [Abdeddaim, Asarin, Maler 2006]

Goal: analyse a greedy scheduling policy on given scenarios.

greedy: no machine is idle if a task is waiting for execution

Scenario

with the constraints:

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,

\[ A \rightarrow B, \quad C \rightarrow D, E. \]
Motivation: Scheduling

Scheduling analysis with timed automata [Abdeddaim, Asarin, Maler 2006]

Goal: analyse a greedy scheduling policy on given scenarios.

| greedy: no machine is idle if a task is waiting for execution |

Scenario

<table>
<thead>
<tr>
<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>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>

Timed automaton: model $A \rightarrow B$ as:

- $t:=0$
- $\text{lock}_1!$
- $\text{unlock}_1!, x_A=2$
- $\text{lock}_2!$
- $\text{unlock}_2!, x_B=2$

Target location: “all tasks have been completed”.

- Timing analysis: use a clock to measure total elapsed time.
Motivation: Scheduling

Scheduling analysis with timed automata [Abdeddaim, Asarin, Maler 2006]

Goal: analyse a greedy scheduling policy on given scenarios.

greedy: no machine is idle if a task is waiting for execution

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></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></td>
<td></td>
<td></td>
<td>$B$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Timed automaton: model $A \rightarrow B$ as:

$t:=0$ \hspace{1cm} lock_1! \hspace{1cm} unlock_1!, x_A=2 \hspace{1cm} lock_2! \hspace{1cm} unlock_2!, x_B=2$

Target location: “all tasks have been completed”.

reachability analysis: schedulable in 6 time units.
Something happens \( \Rightarrow \): duration of \( A \) is now 1.999.

This cannot be an outcome of an algorithm (not greedy).

Best greedy scheduler is ...
Something happens: duration of $A$ is now 1.999.

Best greedy scheduler is ... which completes in 7.999 time units. Previous analysis did not capture this timing anomaly.
Motivation: Robustness in Scheduling

Something happens: duration of A is now 1.999.

Best greedy scheduler is ... which completes in 7.999 time units.
Previous analysis did not capture this **timing anomaly**.

This work

**Goal:** reachability despite perturbations meanly chosen by the environment.

Model the semantics as a **game** between Controller and Environment.

→ can provide robust analysis, robust controller synthesis...
Robust Game Semantics

Let $\mathcal{A}$ be a timed automaton and $\delta > 0$.

<table>
<thead>
<tr>
<th>Semantics $G_\delta(\mathcal{A})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>At any state $(\ell, \nu)$,</td>
</tr>
</tbody>
</table>
Robust Game Semantics
Let $A$ be a timed automaton and $\delta > 0$.

Semantics $G_\delta(A)$

At any state $(\ell, \nu)$,

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d \models g$. 

For $\delta = 0$, this is the usual semantics.

For $\delta > 0$, $x = y = 1$ $y := 0$ $\nu_0$
Robust Game Semantics

Let $A$ be a timed automaton and $\delta > 0$.

Semantics $G_\delta(A)$

At any state $(\ell, \nu)$,

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d \models g$,

2. Environment chooses $d' \in [d - \delta, d + \delta]$. 
Robust Game Semantics

Let $A$ be a timed automaton and $\delta > 0$.

**Semantics $G_\delta(A)$**

At any state $(\ell, \nu)$,

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g, R} \ell'$, such that $\nu + d \models g$,
2. Environment chooses $d' \in [d - \delta, d + \delta]$,
3. New state is $(\ell', (\nu + d')[R \leftarrow 0])$.

For $\delta = 0$, this is the usual semantics.

For $\delta > 0$, $x = y = 1$

$\nu_0$
Robust Game Semantics

Let $\mathcal{A}$ be a timed automaton and $\delta > 0$.

**Semantics $G_\delta(\mathcal{A})$**

At any state $(\ell, \nu)$,

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d \models g$,
2. Environment chooses $d' \in [d - \delta, d + \delta]$,
3. New state is $(\ell', (\nu + d')[R \leftarrow 0])$.

For $\delta = 0$, this is the usual semantics.

For $\delta > 0$, 

For $\delta > 0$,
Robust Game Semantics

Let $\mathcal{A}$ be a timed automaton and $\delta > 0$.

**Semantics $G_\delta(\mathcal{A})$**

At any state $(\ell, \nu)$,

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g, R} \ell'$, such that $\nu + d \models g$,
2. Environment chooses $d' \in [d - \delta, d + \delta]$,
3. New state is $(\ell', (\nu + d')[R \leftarrow 0])$.

For $\delta = 0$, this is the usual semantics.

For $\delta > 0$, 

\[
\xymatrix{
& x=y=1 \\
\vee 0 \ar[r] & \nu_0 \ar[r] & \nu'_0
}
\]
Robust Game Semantics

Let $A$ be a timed automaton and $\delta > 0$.

**Semantics $G_\delta(A)$**

At any state $(\ell, \nu)$,

1. **Controller** chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d \models g$.
2. **Environment** chooses $d' \in [d - \delta, d + \delta]$.
3. **New state is** $(\ell', (\nu + d')[R \leftarrow 0])$.

For $\delta = 0$, this is the usual semantics.

For $\delta > 0$, the diagram illustrates the controller's choice of a delay and the new state.
Robust Game Semantics

(Parameterized) Robust Reachability
Given a timed automaton $\mathcal{A}$ and target location $\ell$, Does there exist $\delta_0 > 0$, such that Controller has a strategy reaching $\ell$ in $G_\delta(\mathcal{A})$ for all $\delta \in [0, \delta_0)$?

Main result
Robust reachability is EXPTIME-complete.

- We provide an upper bound for $\delta_0 > 0$,
- Winning strategies are computed as parameterized DBMs, where $\delta$ is the parameter: uniform representation for all $\delta > 0$.

Previous work: Chatterjee, Henzinger, Prabhu 2008: for fixed $\delta > 0$. 
Robust Game Semantics

Two challenges

1. Accumulation of perturbations.

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

\[ x = 2 \]
\[ 1 \leq x - y \]

\[ \nu_0 \]
\[ \nu'_0 \]
Robust Game Semantics

Two challenges

1. Accumulation of perturbations.

\[ \ell_1 \xrightarrow{x \leq 2, y := 0} \ell_2 \xrightarrow{x = 2, 1 \leq x - y} \ell_3 \]

\[ x \leq 2 \]
\[ y := 0 \]
\[ 1 \leq x - y \]
Two challenges

1. **Accumulation of perturbations.**

   - $l_1$ \(\xRightarrow{x \leq 2, y := 0} l_2\)
   - $l_2$ \(\xRightarrow{x = 2, 1 \leq x - y} l_3\)

2. **New regions become reachable**

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

Algorithm:

Based on an extension of region construction

Provides information on the accumulation of perturbations
Two challenges

1. Accumulation of perturbations.

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

2. New regions become reachable

Algorithm:
Based on an extension of region construction
Provides information on the accumulation of perturbations
Data structure to represent winning states

As in previous examples, winning states can be shown to be always zones whose facets are **shrunken** by $k\delta$ for some $k \in \mathbb{N}$.

These sets will be represented by **shrunken difference-bound matrices (DBMs)**, with parameter $\delta$. [S., Bouyer, Markey, FSTTCS'11]

\begin{center}
\begin{tikzpicture}
\fill[blue!20] (0,0) rectangle (5,5);
\draw[red] (0,0) -- (5,0) -- (5,5) -- (0,5) -- cycle;
\draw[red, very thick] (0,0) -- (3,3);
\draw[red, very thick] (3,0) -- (5,2);
\node at (2.5,2.5) {$5\delta$};
\node at (2.5,1.5) {$3\delta$};
\end{tikzpicture}
\end{center}

Instead of $x - y \leq \alpha \iff$ DBM
we want $x - y \leq \alpha - k\delta \iff$ shrunk DBM

Shrunken zones can be described by a DBM $M$, and an integer matrix $P$. Then, for any $\delta > 0$, $M - \delta P$ describes the above shrunk zone.
Algorithm overview

1. (Forward) Construct an equivalent finite turn-based game, region-based
2. Solve it,
3. (Backward) Construct winning states in $G_\delta(A)$, and deduce $\delta_0$.

Definition

A **shrinking** of a region $r$ is a shrunk region $r - \delta P$, for some $P$,

Winning strategies will be described by shrinkings of regions:

One can win from a region $r \iff$ one can win from a shrinking of $r$. 
Construction of the finite turn-based game

Region automaton:

\[
\ell, r_0 \rightarrow \ell, r'_0 \rightarrow \ell', r_1
\]
Construction of the finite turn-based game

Extended region automaton:

Idea: We win from some shrinking of $r_0$, if, and only if we win from some shrinkings of $r_1, r_2, r_3$. 
Construction of the finite turn-based game

Extended region automaton:

Idea: We win from some shrinking of \( r_0 \), if, and only if we win from some shrinkings of \( r_1, r_2, r_3 \). Note quite.
Assume that we have we can win from some shrinkings of $r_1, r_2, r_3$. 
Assume that we have we can win from some shrinkings of $r_1$, $r_2$, $r_3$.

Can these be combined to a winning strategy from $r_0$? No: we don’t have a strategy for valuations around $r_1$. 
**Solution:** Look for a shrinking of some regions with *constraints*.

A **constrained region** is a region with some of its facets marked. A shrinking of a constrained region **does not shrink** from marked facets.
**Solution:** Look for a shrinking of some regions with *constraints*.

A **constrained region** is a region with some of its facets marked. A shrinking of a constrained region **does not shrink** from marked facets.

We win from $r_0$ iff we win from shrinkings of **constrained** $r_1$, $r_2$, $r_3$. 

[Diagram of constrained regions and shrinkings]
Solution: Look for a shrinking of some regions with constraints.

A constrained region is a region with some of its facets marked. A shrinking of a constrained region does not shrink from marked facets.

We win from \( r_0 \) iff we win from shrinkings of constrained \( r_1, r_2, r_3 \).
**Solution:** Look for a shrinking of some regions with *constraints*.

A **constrained region** is a region with some of its facets marked. A shrinking of a constrained region **does not shrink** from marked facets.

In fact,
Solution: Look for a shrinking of some regions with *constraints*.

A **constrained region** is a region with some of its facets marked. A shrinking of a constrained region **does not shrink** from marked facets.

In fact,

OK, we have a strategy for all the points in the red area.
Finite game $F(\mathcal{A})$

Shrinking constraint for region $r$ is represented by a boolean matrix $S_r$.

Theorem

Controller wins $G_\delta(\mathcal{A})$ for all $\delta \in [0, \delta_0]$ for some $\delta_0 > 0$, iff

Controller wins $F(\mathcal{A})$. 

Ocan Sankur (LSV)
Details on the definition of $F(\mathcal{A})$

$\ell, r_0, S_{r_0}$
Details on the definition of $F(\mathcal{A})$

$S_\phi$ is defined such that:

Controller wins from *some* shrinking of $(\phi, S_\phi)$ iff
Controller wins from *some* shrinking of $(r_0, S_{r_0})$. 
Details on the definition of $F(A)$

$S_\phi$ is defined such that:

Controller wins from some shrinking of $(\phi, S_\phi)$ iff
Controller wins from some shrinking of $(r_0, S_{r_0})$. 
Details on the definition of $F(\mathcal{A})$

$S_\phi$ is defined such that:

Controller wins from some shrinking of $(\phi, S_\phi)$ iff Controller wins from some shrinking of $(r_0, S_{r_0})$. 
Details on the definition of $F(A)$

Controller wins from some shrinking of $(\phi, S\phi)$ iff Controller wins from some shrinking of $(r_0, S_{r_0})$. 

[Diagram showing a line segment $\ell, r_0, S_{r_0}$ leading to $\ell, \phi, S_{\phi}$]
Details on the definition of $F(\mathcal{A})$
Details on the definition of $F(\mathcal{A})$

Given a region $\phi$ and constraint $S\phi$, one can compute the neighborhood: the union of those regions reached by the slightest perturbation.
Details on the definition of $F(A)$

$\ell, r_0, S_{r_0} \rightarrow \ell, \phi, S_{\phi}$

Controller wins from some shrinking of $(\phi, S_{\phi})$ iff Controller wins from some shrinking of $(r_0, S_{r_0})$. 

reset
Details on the definition of $F(\mathcal{A})$

Controller wins from some shrinking of $(\phi, S\phi)$ iff Controller wins from some shrinking of $(r_0, S_{r_0})$. 

Diagram showing the transitions:
- $l, r_0, S_{r_0}$
- $l, \phi, S_{\phi}$
- $l', r_1, S_{r_1}$
- $l', r_2, S_{r_2}$
- $l', r_3, S_{r_3}$
Constructing a winning strategy from $F(A)$

Each step of the backward propagation gives an upper bound on $\delta$.
Constructing a winning strategy from $F(A)$

Each step of the backward propagation gives an upper bound on $\delta$. 

\[ \ell, r_0, S_{r_0} \rightarrow \ell, \phi, S_\phi \rightarrow \ell', r_1, S_{r_1} \]

\[ \ell', r_2, S_{r_2} \rightarrow \ell', r_3, S_{r_3} \]
Constructing a winning strategy from $F(A)$

Each step of the backward propagation gives an upper bound on $\delta$. 

$\ell, r_0, S_{r_0} \rightarrow \ell, \phi, S_{\phi} \rightarrow \ell', r_2, S_{r_2} \rightarrow \ell', r_1, S_{r_1}$

$\ell', r_3, S_{r_3}$
Constructing a winning strategy from $F(A)$

Each step of the backward propagation gives an upper bound on $\delta$. 

Ocan Sankur (LSV)  
Robust Reachability in Timed Automata  
April 19, 2012 15 / 17
Constructing a winning strategy from $F(\mathcal{A})$

Each step of the backward propagation gives an upper bound on $\delta$. 
EXPTIME-hardness

Usual semantics in TA can encode reachability in linearly bounded Turing machines (PSPACE-complete).

Robust semantics in TA can encode reachability in alternating linearly bounded Turing machines (EXPTIME-complete).

The encoding is similar as in the PSPACE-hardness proofs for TA. **Alternation:** simulated by the perturbating player

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

- Game semantics for robust reachability in timed automata with unknown $\delta$
- Results generalize to two-player timed games $\rightarrow$ (parameterized) robust controller synthesis
- Winning sets are described by parameterized shrunk DBMs
- Uniform representation of strategies for all small $\delta > 0$.

$\rightarrow$ A good tool for reasoning with small parameterized perturbations in timed automata

Future work

- Zone-based algorithm
- Probabilistic semantics
- Safety
Conclusion

- Game semantics for robust reachability in timed automata with unknown $\delta$.
- Results generalize to two-player timed games $\rightarrow$ (parameterized) robust controller synthesis.
- Winning sets are described by parameterized shrunk DBMs.
- Uniform representation of strategies for all small $\delta > 0$.

$\rightarrow$ A good tool for reasoning with small parameterized perturbations in timed automata.

Future work

- Zone-based algorithm
- Probabilistic semantics
- Safety

Thank you!