# 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 $\mathcal{A}$, and a target location $\ell$, decide whether some (initial) run of $\mathcal{A}$ visits $\ell$.

▶ PSPACE-complete [Alur & Dill '94].

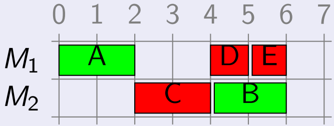
## Applications of Reachability

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: $A \to B, \quad C \to D, E.$

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,

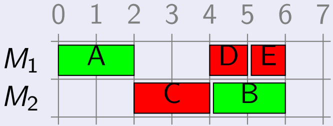
# Motivation: Scheduling

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

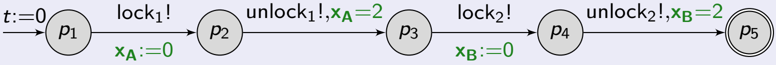
**Goal:** analyse a *greedy* scheduling policy on given scenarios.

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

## Scenario




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




Target location: "all tasks have been completed".

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

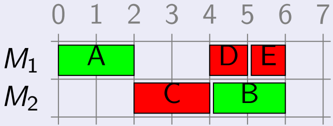
# Motivation: Scheduling

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

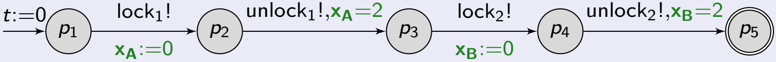
**Goal:** analyse a *greedy* scheduling policy on given scenarios.

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

## Scenario




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

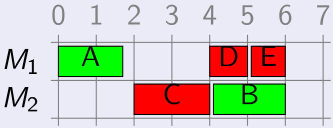



Target location: "all tasks have been completed".

▶ reachability analysis: schedulable in 6 time units.

# Motivation: Robustness in Scheduling

↯ Something happens ↯: duration of $A$ is now 1.999.

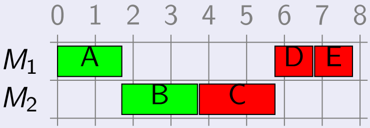



This cannot be an outcome of an algorithm (not greedy).

Best greedy scheduler is ...

# Motivation: Robustness in Scheduling

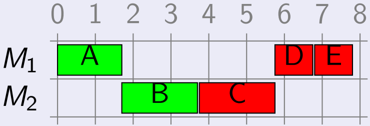
↯ Something happens ↯: duration of $A$ is now 1.999.




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

# 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.

$\rightarrow$ can provide robust analysis, robust controller synthesis...

# Robust Game Semantics

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

## Semantics $\mathcal{G}_\delta(\mathcal{A})$

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

# Robust Game Semantics

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

## Semantics $\mathcal{G}_\delta(\mathcal{A})$

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

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

# Robust Game Semantics

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

## Semantics $\mathcal{G}_\delta(\mathcal{A})$

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 $d' \in [d - \delta, d + \delta]$,

# Robust Game Semantics

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

## Semantics $\mathcal{G}_\delta(\mathcal{A})$

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 $(\ell', (\nu + d')[R \leftarrow 0])$.

# Robust Game Semantics

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

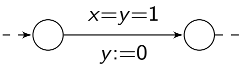
## Semantics $\mathcal{G}_\delta(\mathcal{A})$

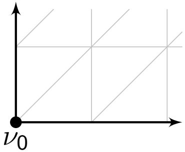
At any state $(\ell, \nu)$,

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 $\delta > 0$,



# Robust Game Semantics

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

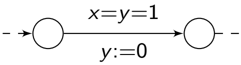
## Semantics $\mathcal{G}_\delta(\mathcal{A})$

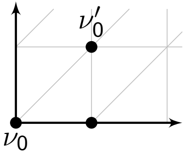
At any state $(\ell, \nu)$,

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 $\delta > 0$,



# Robust Game Semantics

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

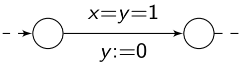
## Semantics $\mathcal{G}_\delta(\mathcal{A})$

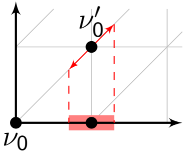
At any state $(\ell, \nu)$,

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 $\delta > 0$,



# 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 $\mathcal{G}_\delta(\mathcal{A})$ for all $\delta \in [0, \delta_0)$?

## Main result

Robust reachability is EXPTIME-complete.

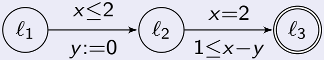
- We provide an upper bound for $\delta_0 > 0$,
- Winning strategies are computed as *parameterized DBMs*, where $\delta$ is the parameter: uniform representation for all $\delta > 0$.

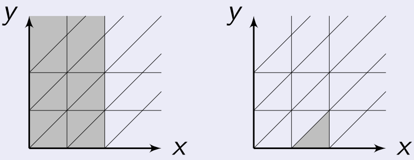
**Previous work:** Chatterjee, Henzinger, Prabhu 2008: for **fixed** $\delta > 0$.

# Robust Game Semantics

## Two challenges

1. **Accumulation of perturbations.**

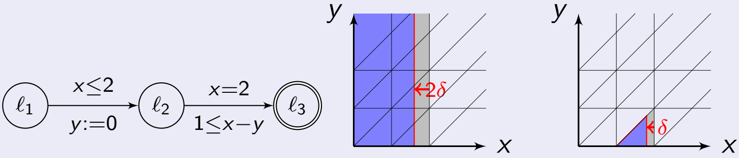


# Robust Game Semantics

## Two challenges

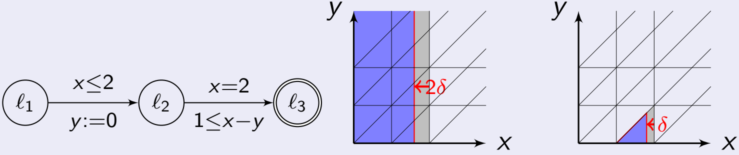
1. **Accumulation of perturbations.**



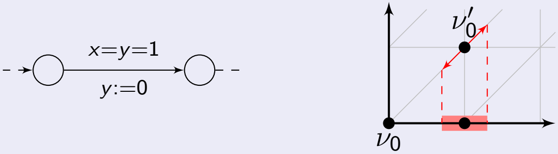
# Robust Game Semantics

## Two challenges

1. **Accumulation of perturbations.**



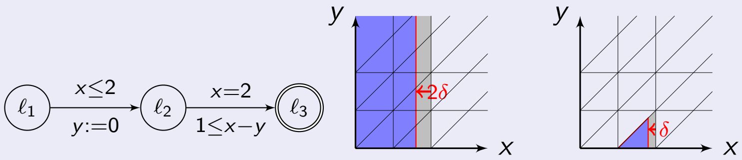

2. **New regions become reachable**



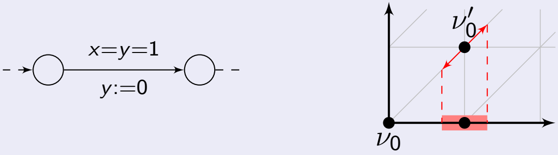
# Robust Game Semantics

## Two challenges

1. **Accumulation of perturbations.**




2. **New regions become reachable**




**Algorithm:**

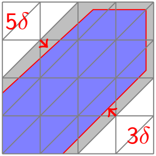
Based on an extension of region construction

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 **shrunk** by $k\delta$ for some $k \in \mathbb{N}$.

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




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

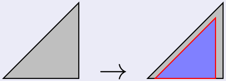
Shrunk zones can be described by a DBM $M$, and an integer matrix $P$. Then, for any $\delta > 0$, $M - \delta P$ describes the above shrunk zone.

# Algorithm overview

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

## Definition

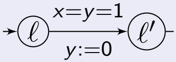
A **shrinking** of a region $r$ is a shrunk region $r - \delta P$, for *some* $P$,

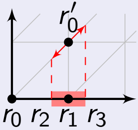


Winning strategies will be described by shrinkings of regions:

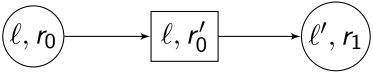
One can win from a region $r \quad \Leftrightarrow \quad$ one can win from a *shrinking* of $r$.

# Construction of the finite turn-based game

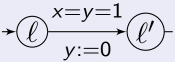


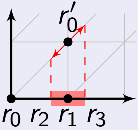




region automaton:

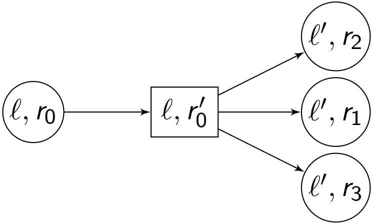


# Construction of the finite turn-based game



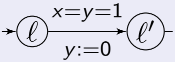



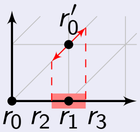

**Extended** region automaton:



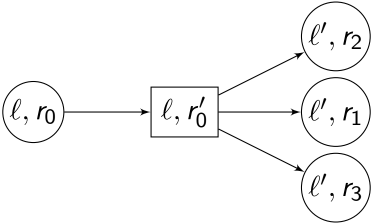

**Idea:** We win from *some* shrinking of $r_0$, if, and only if we win from *some* shrinkings of $r_1, r_2, r_3$.

# Construction of the finite turn-based game



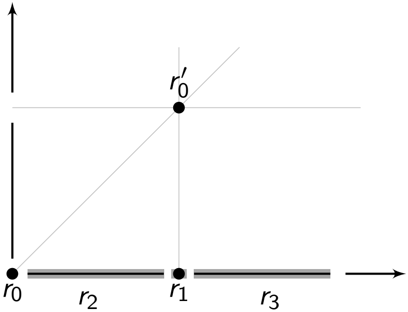




**Extended** region automaton:

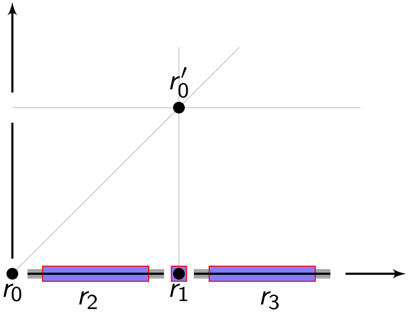



**Idea:** ~~We win from *some* shrinking of $r_0$, if, and only if we win from *some* shrinkings of $r_1, r_2, r_3$.~~ Note quite.

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 $r_1, r_2, r_3$.

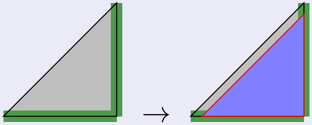



Can these be combined to a winning strategy from $r_0$?

No: we don't have a strategy for valuations around $r_1$.

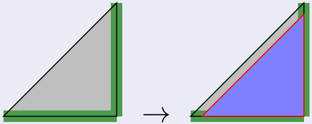
**Solution:** Look for a shrinking of some regions with *constraints*.

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

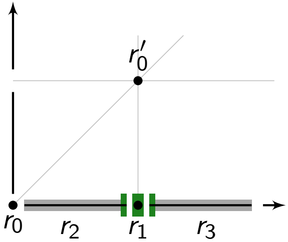


**Solution:** Look for a shrinking of some regions with *constraints*.

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

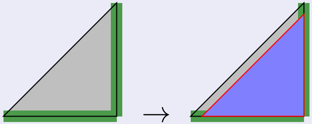


We win from $r_0$ iff we win from shrinkings of **constrained** $r_1, r_2, r_3$.

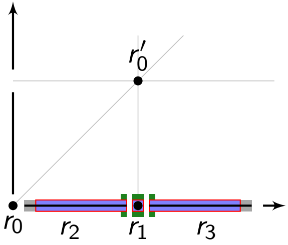


**Solution:** Look for a shrinking of some regions with *constraints*.

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

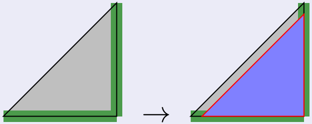



We win from $r_0$ iff we win from shrinkings of **constrained** $r_1, r_2, r_3$.

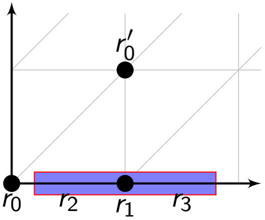


**Solution:** Look for a shrinking of some regions with *constraints*.

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

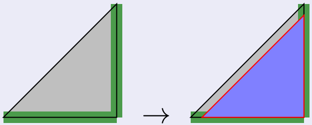


In fact,

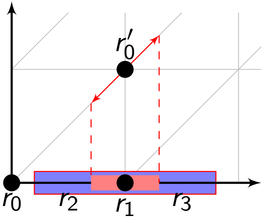


**Solution:** Look for a shrinking of some regions with *constraints*.

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



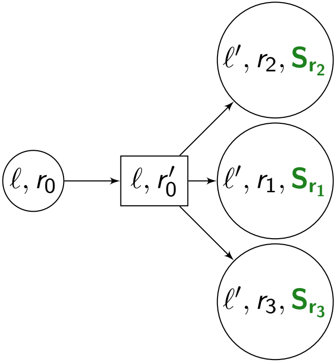
In fact,




OK, we have a strategy for all the points in the red area.

# Finite game $\mathbf{F}(\mathcal{A})$

Shrinking constraint for region $r$ is represented by a boolean matrix $\mathbf{S_r}$.



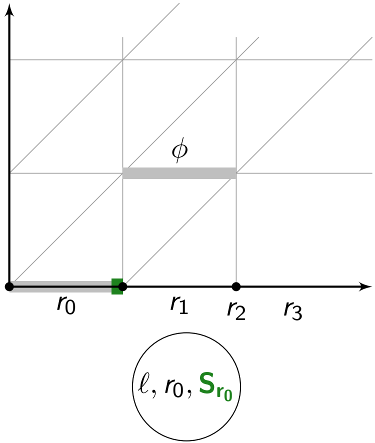

**Theorem**

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

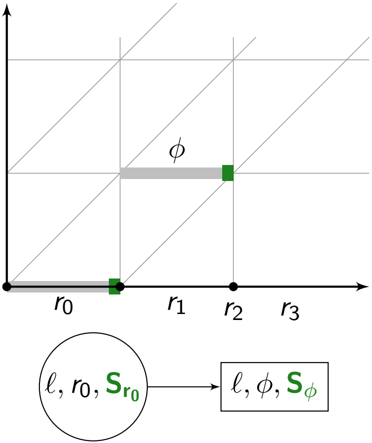
**iff**

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

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



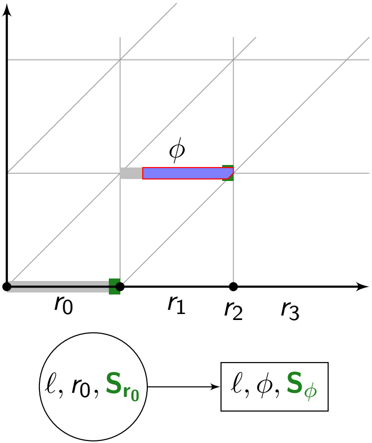
# Details on the definition of $\mathbf{F}(\mathcal{A})$




**$\mathbf{S}_\phi$ is defined such that:**

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

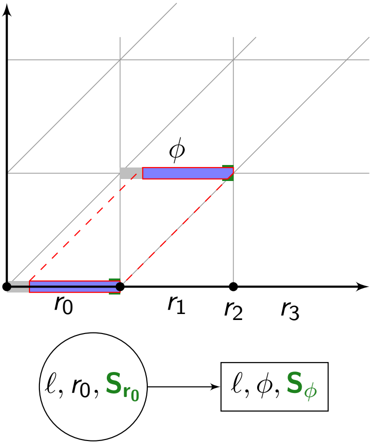
# Details on the definition of $\mathbf{F}(\mathcal{A})$




**$\mathbf{S}_\phi$ is defined such that:**

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

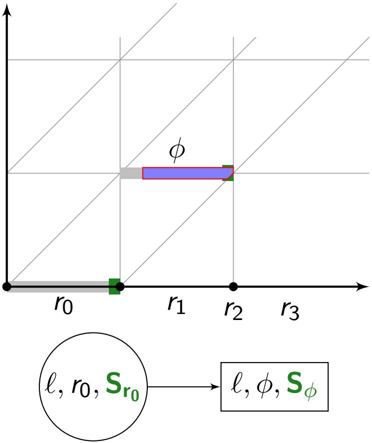
# Details on the definition of $\mathbf{F}(\mathcal{A})$



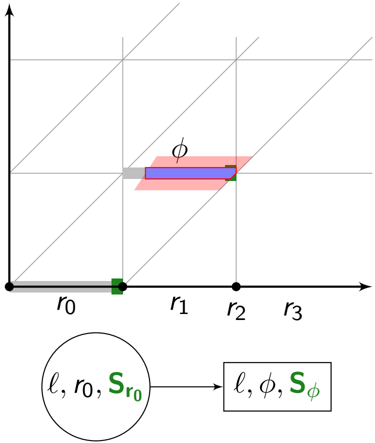

**$\mathbf{S}_\phi$ is defined such that:**

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

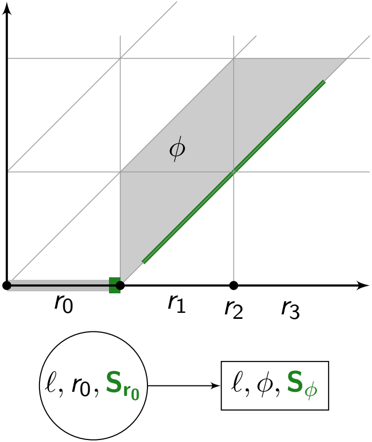
# Details on the definition of $\mathbf{F}(\mathcal{A})$



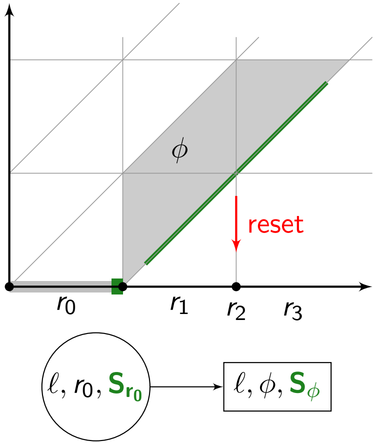
# Details on the definition of $\mathbf{F}(\mathcal{A})$



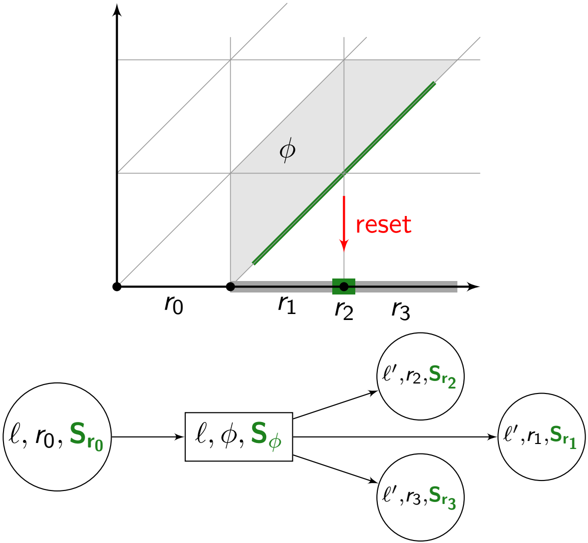
# Details on the definition of $\mathbf{F}(\mathcal{A})$



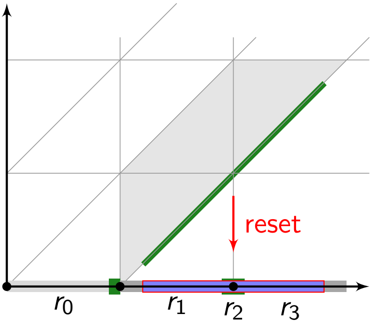
# Details on the definition of $\mathbf{F}(\mathcal{A})$

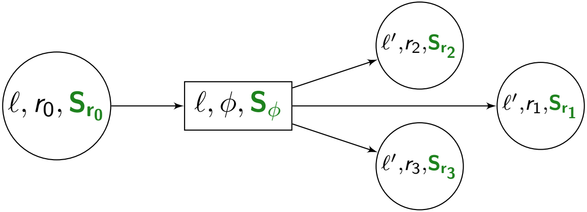


# Details on the definition of $\mathbf{F}(\mathcal{A})$

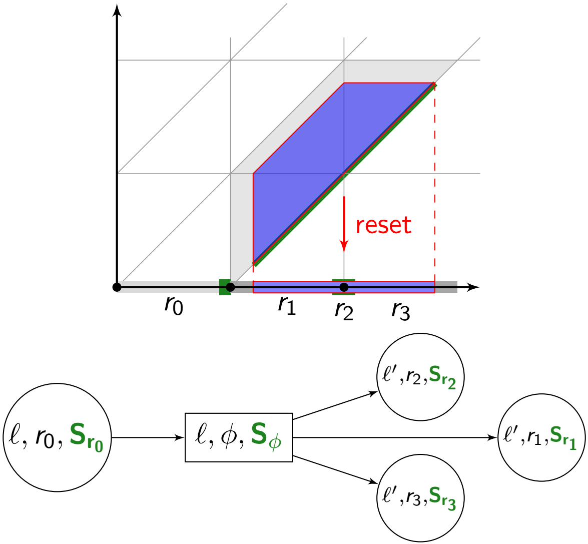


# Constructing a winning strategy from $\mathbf{F}(\mathcal{A})$

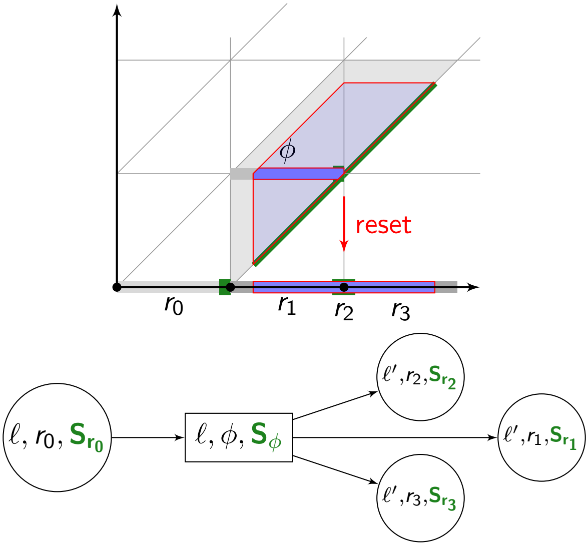


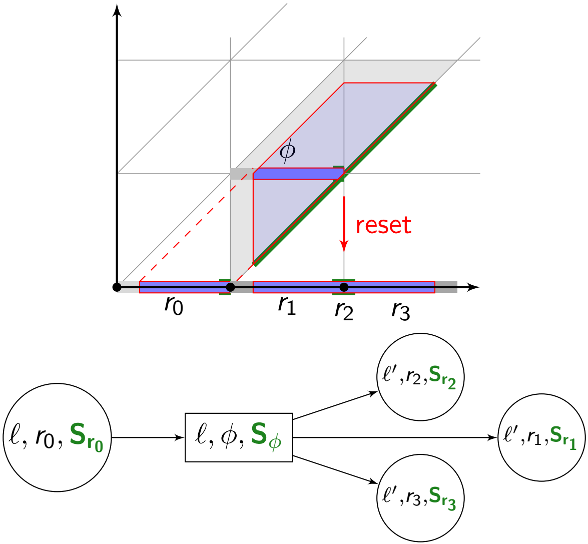
# Constructing a winning strategy from $\mathbf{F}(\mathcal{A})$



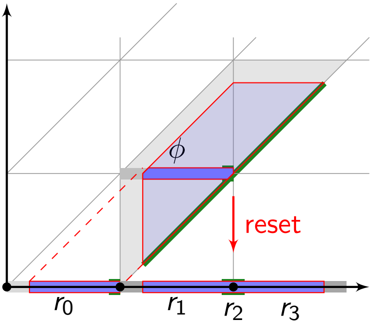
# Constructing a winning strategy from $\mathbf{F}(\mathcal{A})$

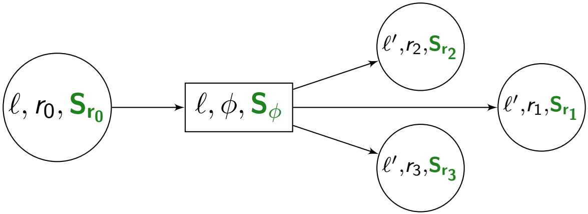


# Constructing a winning strategy from $\mathbf{F}(\mathcal{A})$



# Constructing a winning strategy from $\mathbf{F}(\mathcal{A})$



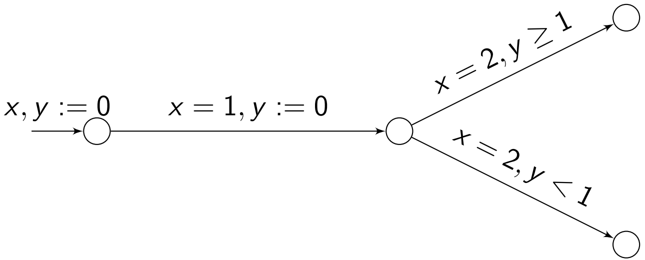

- Each step of the backward propagation gives an upper bound on $\delta$.

# 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



## 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!