Robustness in Timed Automata: Analysis, Synthesis, Implementation

Ocan Sankur

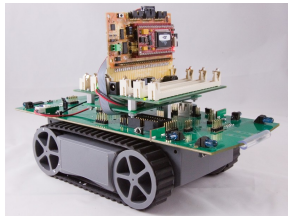
PhD Thesis Defense

LSV, Ecole Normale Supérieure de Cachan

May 24, 2013

Real-Time Systems

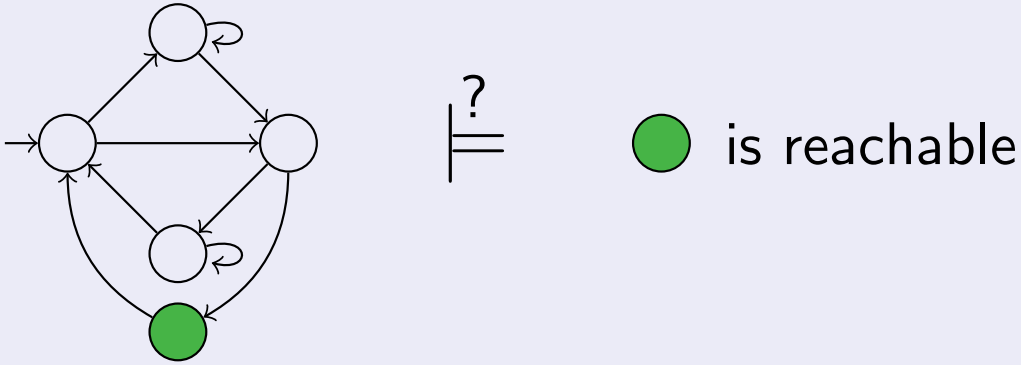
Systems whose behaviors depend on real-time constraints, such as



- Robots,
- Car, train, airplane components,
- Biomedical systems (e.g. insuline pump), ...

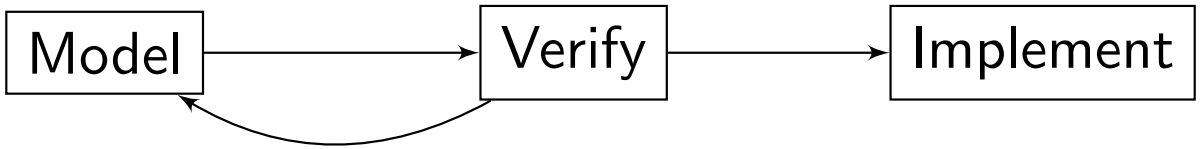
Developing correct real-time systems is difficult: **formal verification**

Model-checking



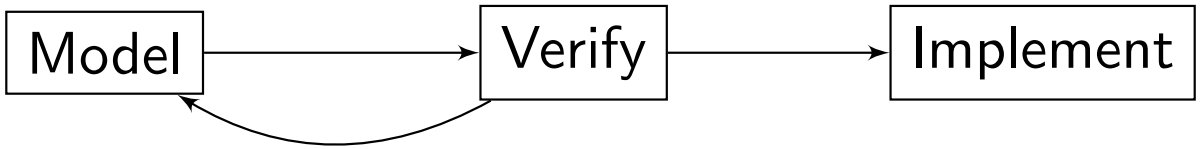
Robustness

Model-checking is often used to validate abstract designs.



Robustness

Model-checking is often used to validate abstract designs.



Model $\approx$ Implementation?

Model:

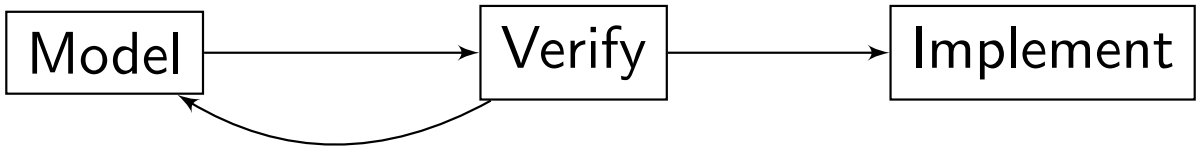
- Abstract, simplified
- Idealized: perfect measurements and timings

Implementation:

- measurement errors,
- unexpected input,
- hardware errors...

Robustness

Model-checking is often used to validate abstract designs.



Model $\approx$ Implementation?

Model:

- Abstract, simplified
- Idealized: perfect measurements and timings

Implementation:

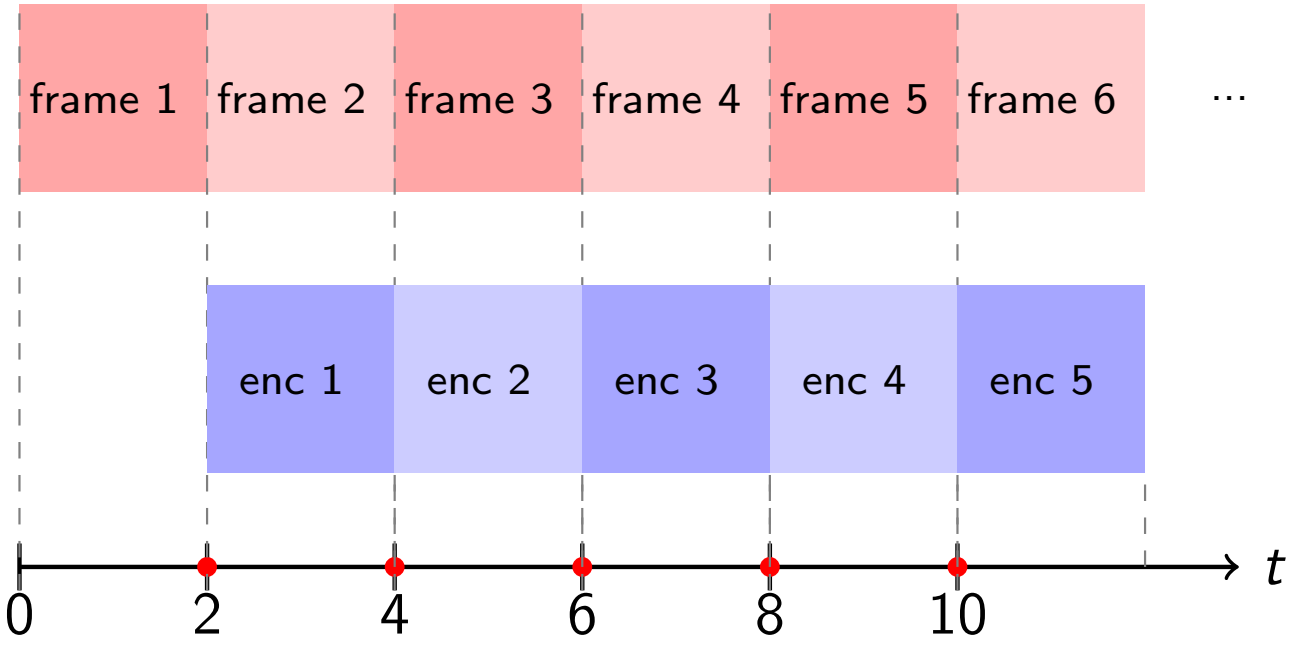
- measurement errors,
- unexpected input,
- hardware errors...

Robustness

The ability of a system to resist to errors upto some bound.

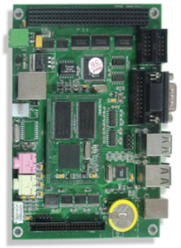
Goal: Add robustness to model-checking of real-time systems.

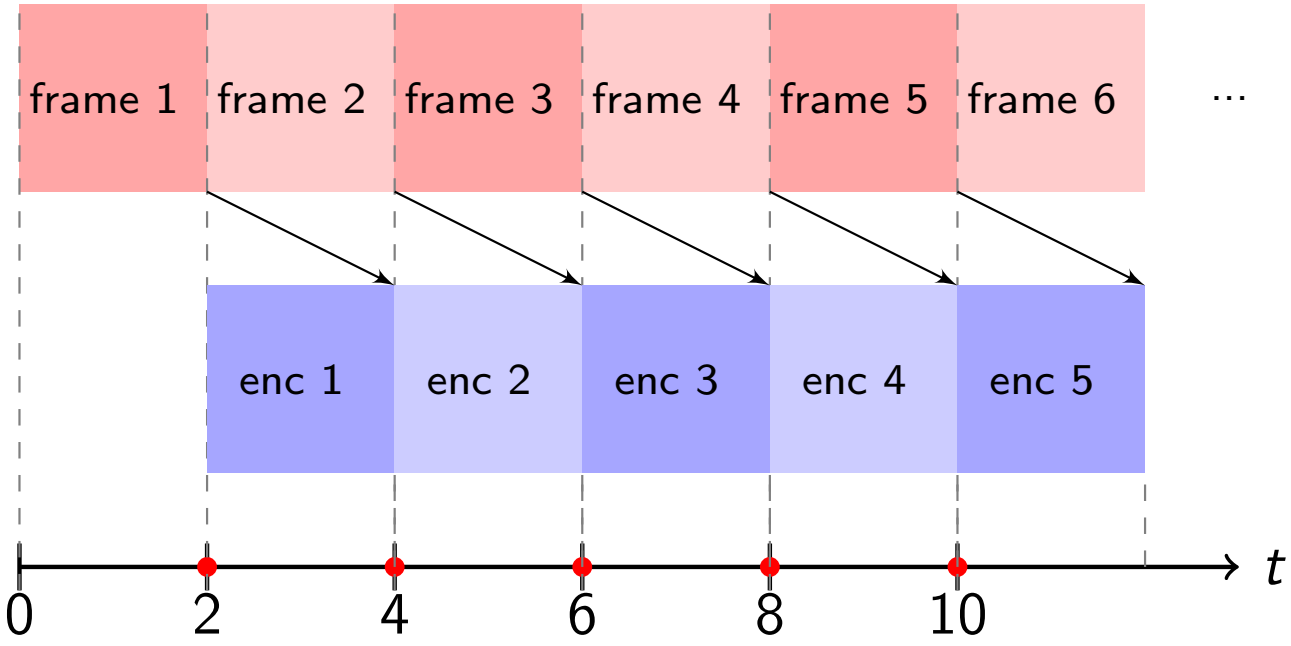
Real-Time System Example: Producer-Consumer



Components are abstracted as periodic events

Real-Time System Example: Producer-Consumer

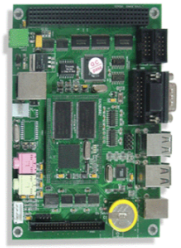


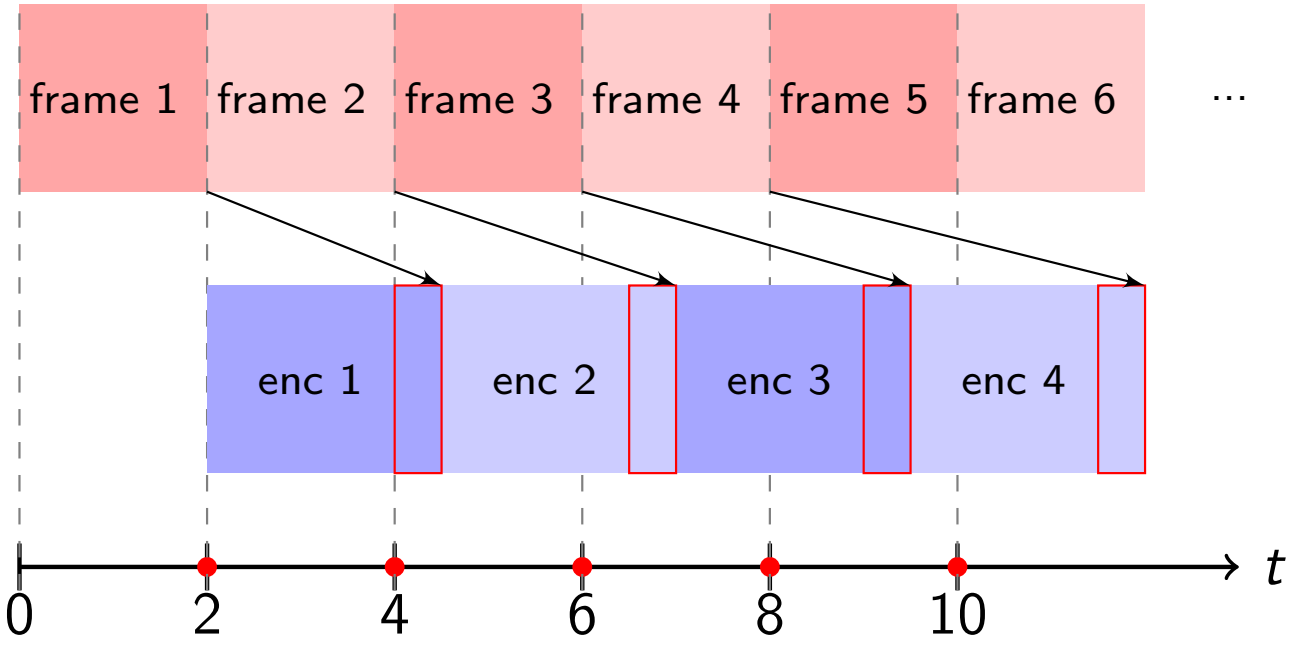


Components are abstracted as periodic events

Property: No buffer overflow.
Model-checking: ✓

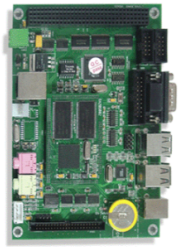
Real-Time System Example: Producer-Consumer

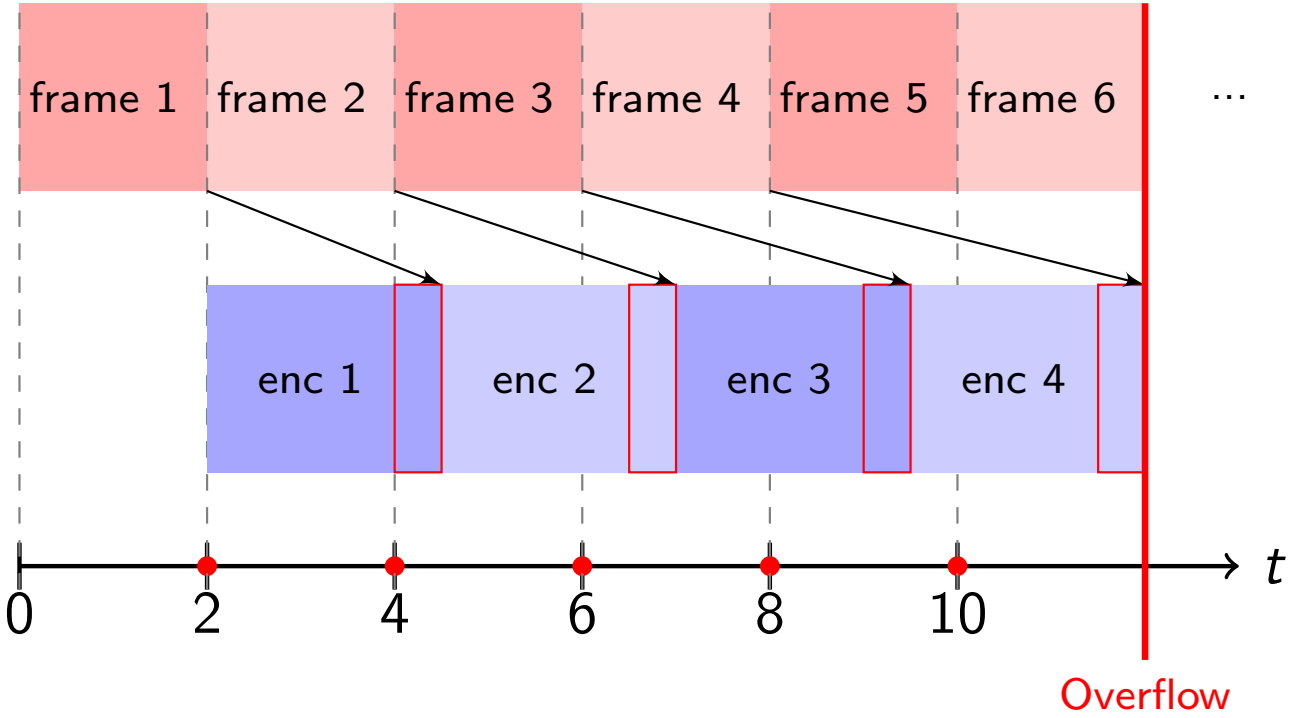




Assume that the implementation of the encoder is slightly slower due to unexpected workload, wrong hardware specification, etc.

Real-Time System Example: Producer-Consumer





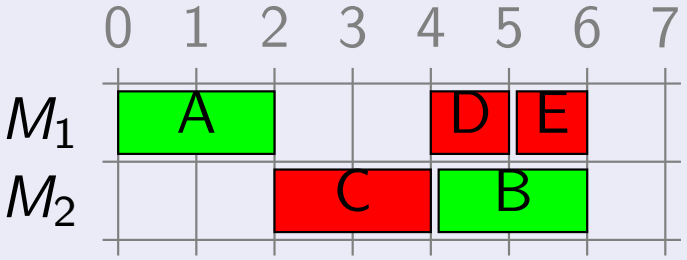
Assume that the implementation of the encoder is slightly slower due to unexpected workload, wrong hardware specification, etc.

Under the slightest **enlargement**, the system is incorrect.

The system is not robust to small increases in execution times.

Real-Time System Example: Scheduling

Scenario

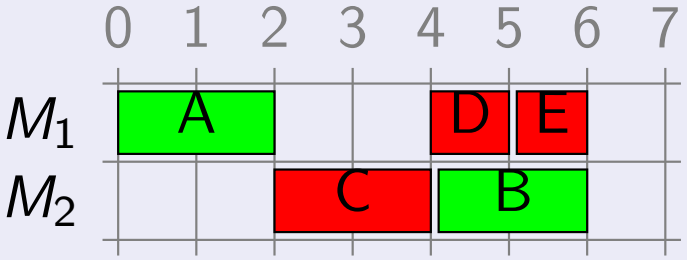


with the constraints: $A \to B, \qquad C \to D, E.$

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,

Real-Time System Example: Scheduling

Scenario



with the constraints: $A \to B, \qquad C \to D, E.$

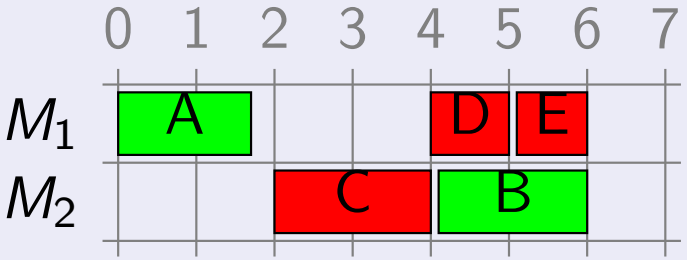
1. A, D, E must be scheduled on machine M_1,
2. B, C must be scheduled on machine M_2,
3. C starts no sooner than 2 time units,

Goal: Analyse a *work-conserving* scheduling policy on a given scenario (*work-conserving:* no machine is idle if a task is waiting for execution)

Property: All tasks terminate in 6 time units
Model-checking: ✓

Real-Time System Example: Scheduling

Scenario

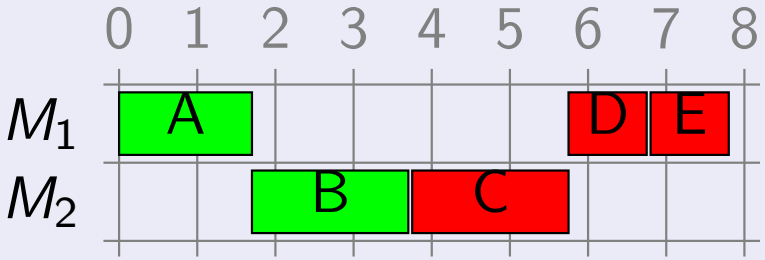


This cannot be an outcome of an algorithm (not work-conserving).

↯ Unexpectedly ↯: duration of A is reduced to 1.999

Real-Time System Example: Scheduling

Scenario



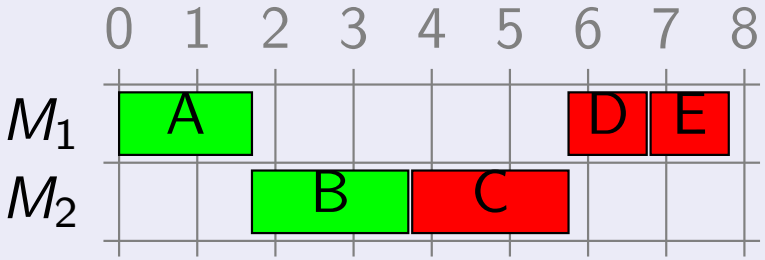
↯ Unexpectedly ↯: duration of A is reduced to 1.999

The best scheduling in this case takes 7.999 time units.

The system is not robust to small decreases in execution times.

Real-Time System Example: Scheduling

Scenario



↯ Unexpectedly ↯: duration of A is reduced to 1.999

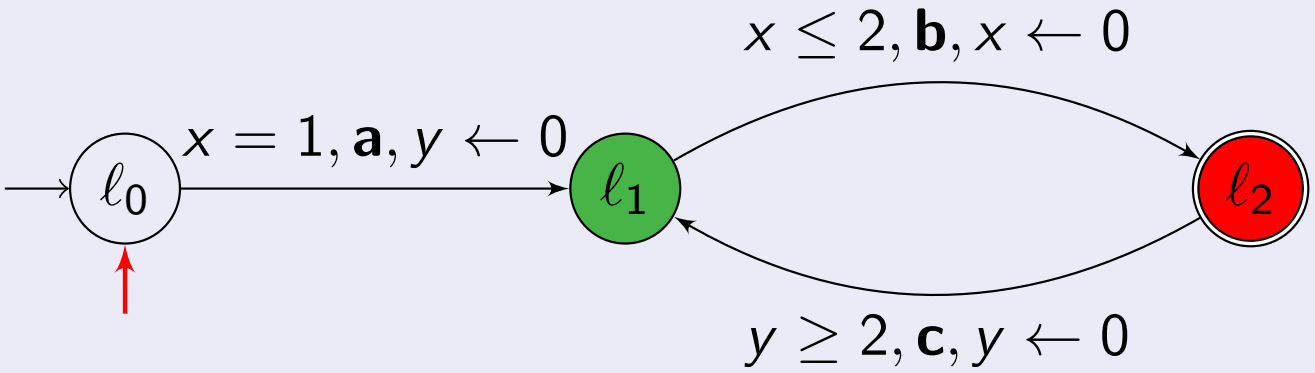
The best scheduling in this case takes 7.999 time units.

The system is not robust to small decreases in execution times.

Next: Timed automata formalism to model real-time systems.

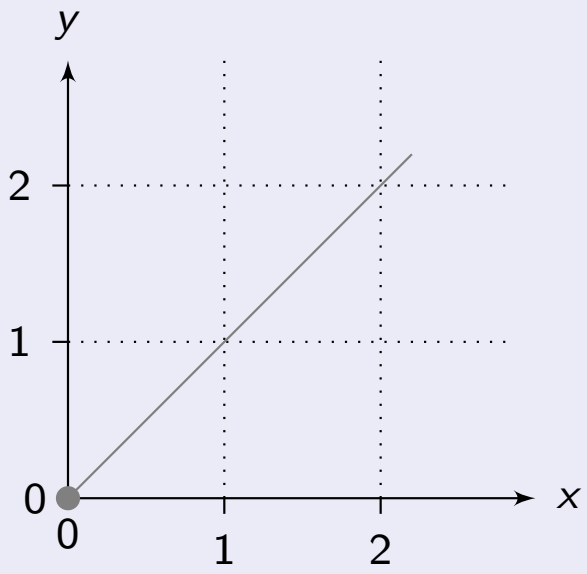
Timed Automata

Timed automata = Finite automata + Analog clocks. [Alur and Dill 1994]



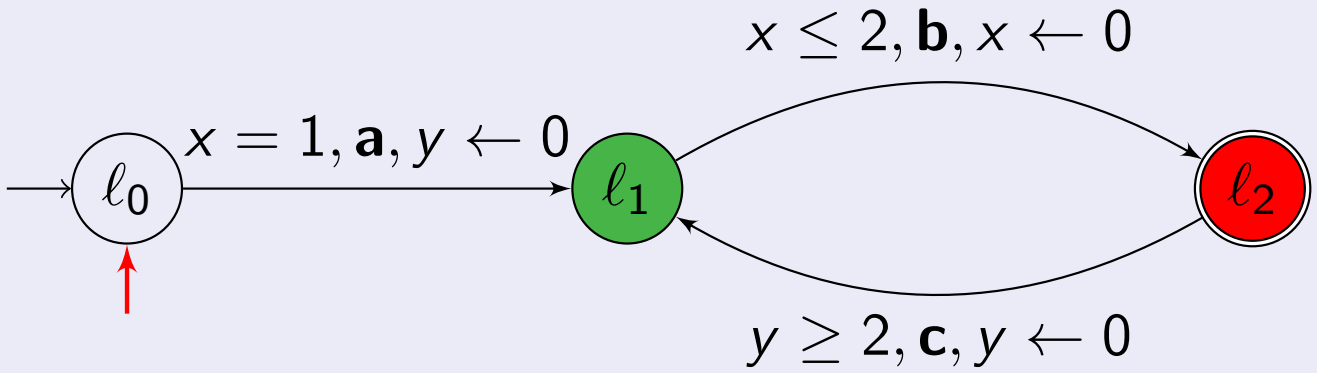
Runs: time delays + discrete actions

$(\ell_0, 0, 0)$



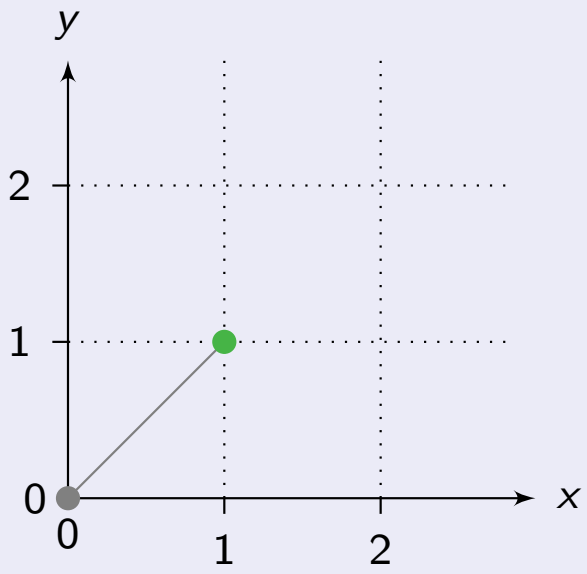
Timed Automata

Timed automata = Finite automata + Analog clocks. [Alur and Dill 1994]



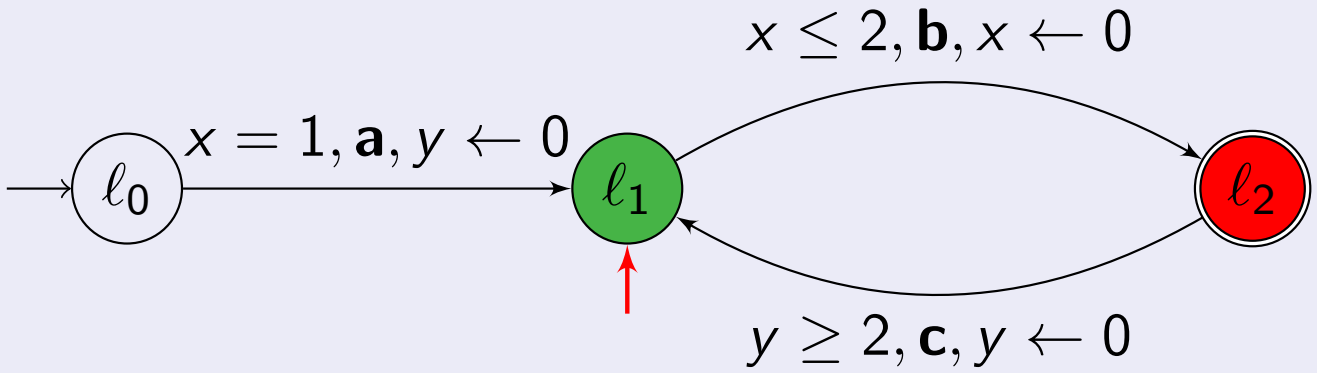
Runs: time delays + discrete actions

$(\ell_0, 0, 0) \xrightarrow{1} (\ell_0, 1, 1)$



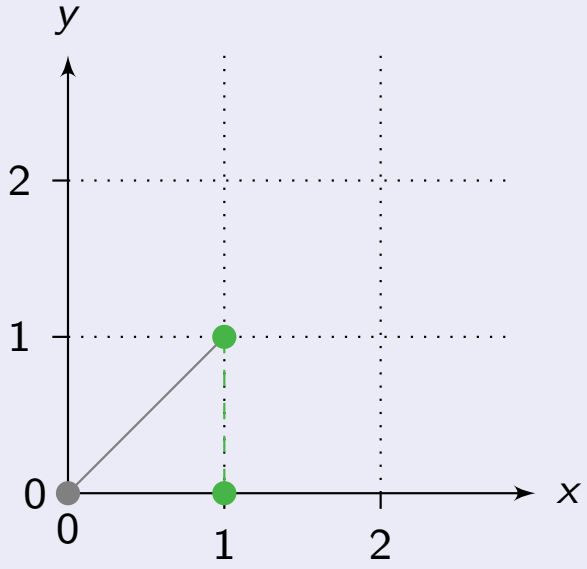
Timed Automata

Timed automata = Finite automata + Analog clocks. [Alur and Dill 1994]



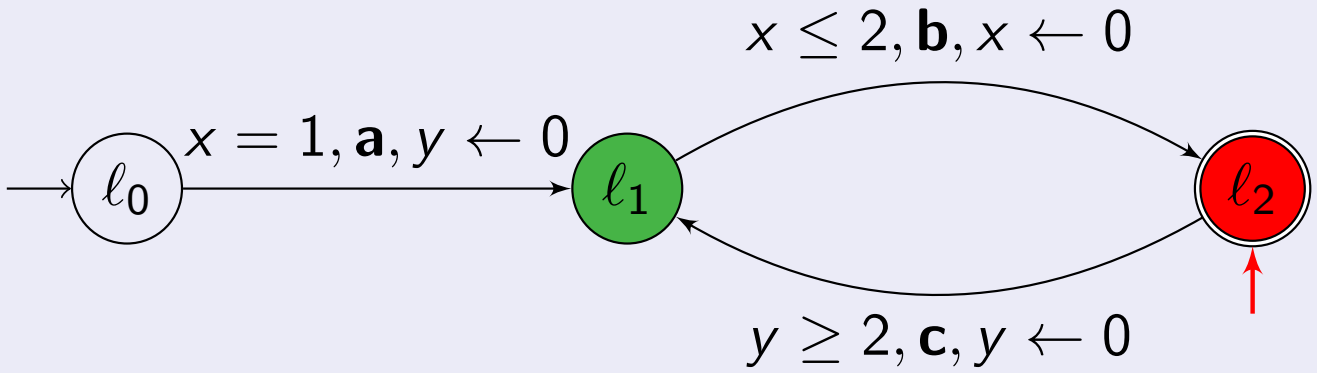
Runs: time delays + discrete actions

$(\ell_0, 0, 0) \xrightarrow{1} (\ell_0, 1, 1) \xrightarrow{a} (\ell_1, 1, 0)$



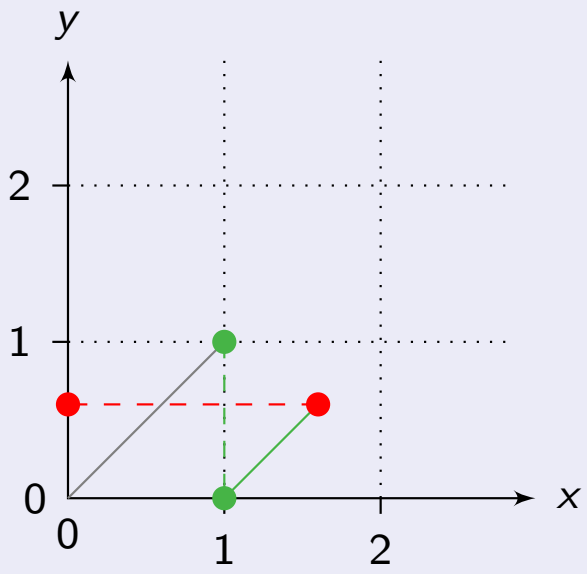
Timed Automata

Timed automata = Finite automata + Analog clocks. [Alur and Dill 1994]



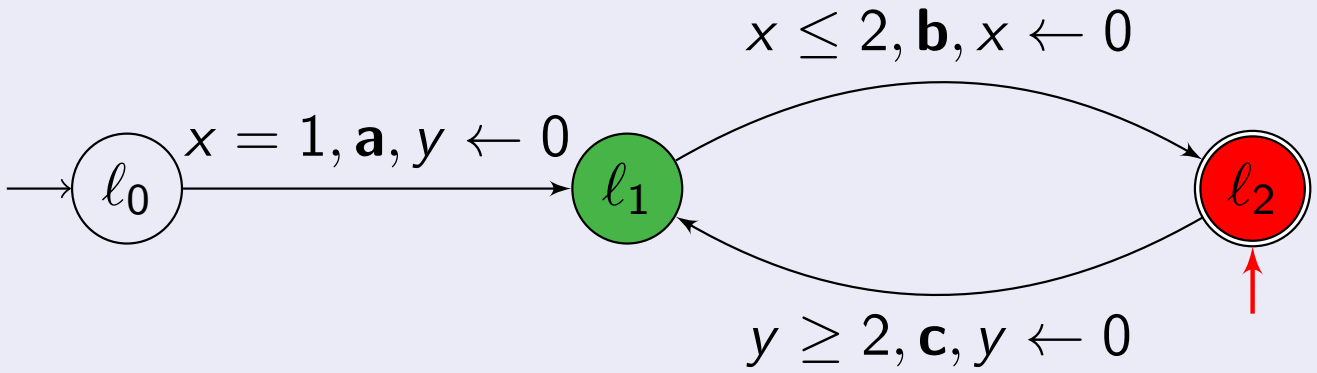
Runs: time delays + discrete actions

$$(\ell_0, 0, 0) \xrightarrow{1} (\ell_0, 1, 1) \xrightarrow{a} (\ell_1, 1, 0)$$
$$\xrightarrow{0.6} (\ell_1, 1.6, 0.6) \xrightarrow{b} (\ell_2, 0, 0.6)$$



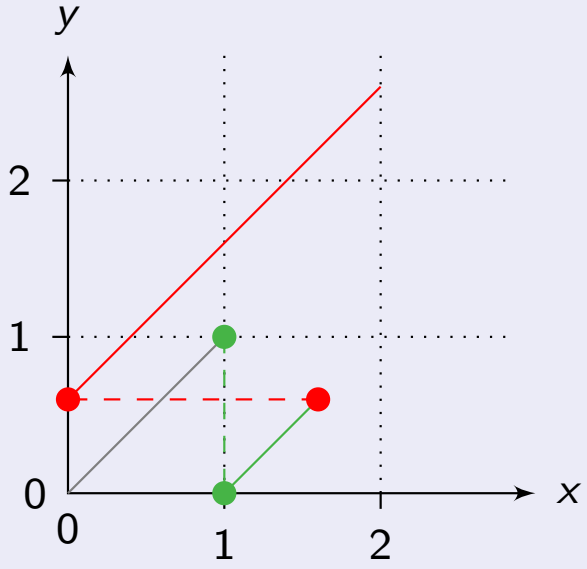
Timed Automata

Timed automata = Finite automata + Analog clocks. [Alur and Dill 1994]



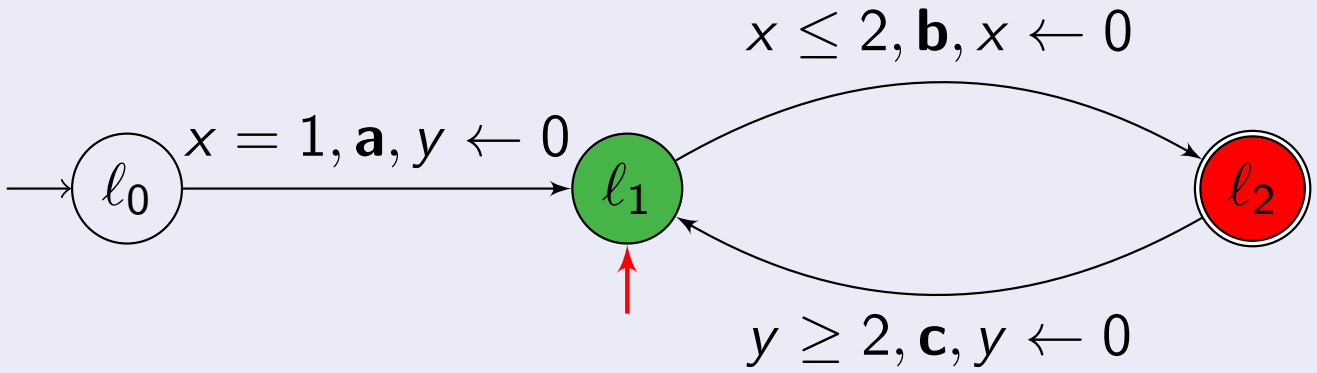
Runs: time delays + discrete actions

$$(\ell_0, 0, 0) \xrightarrow{1} (\ell_0, 1, 1) \xrightarrow{a} (\ell_1, 1, 0)$$
$$\xrightarrow{0.6} (\ell_1, 1.6, 0.6) \xrightarrow{b} (\ell_2, 0, 0.6)$$



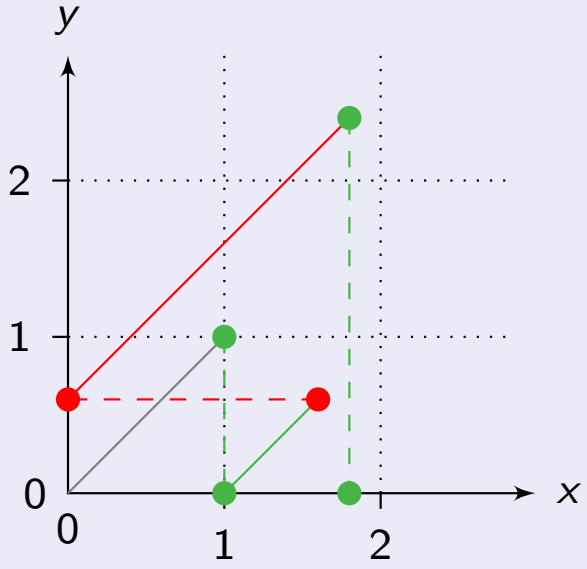
Timed Automata

Timed automata = Finite automata + Analog clocks. [Alur and Dill 1994]



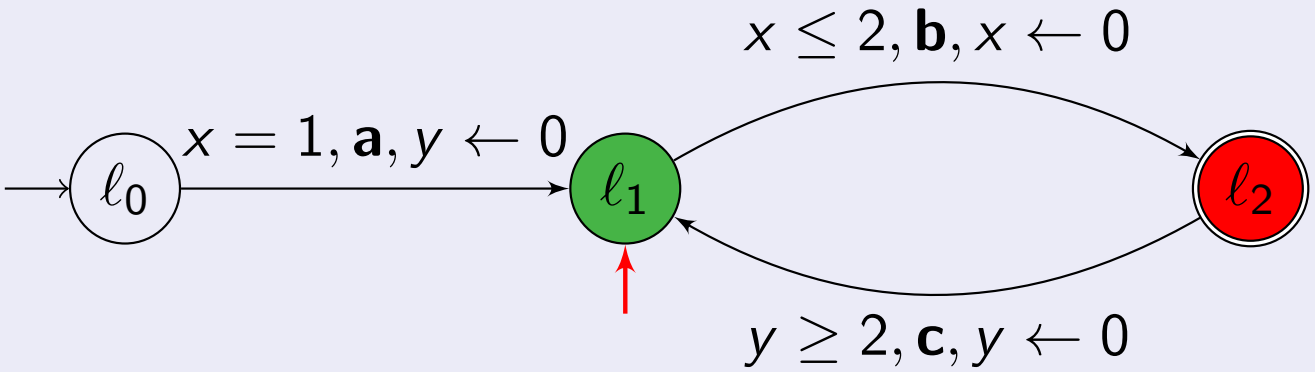
Runs: time delays + discrete actions

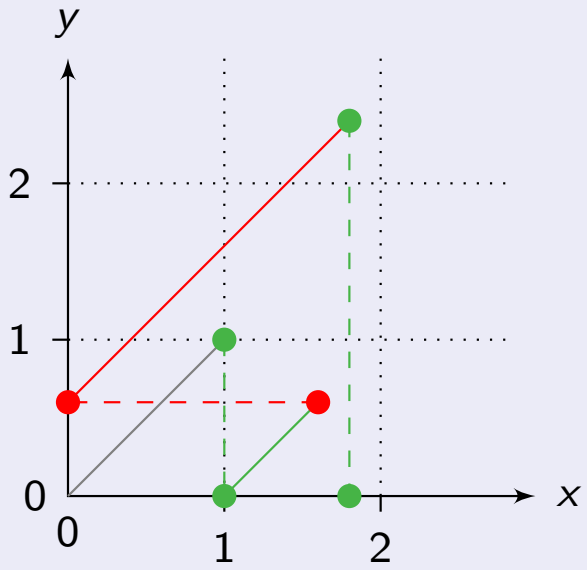
$(\ell_0, 0, 0) \xrightarrow{1} (\ell_0, 1, 1) \xrightarrow{a} (\ell_1, 1, 0)$
$\xrightarrow{0.6} (\ell_1, 1.6, 0.6) \xrightarrow{b} (\ell_2, 0, 0.6) \xrightarrow{1.8}$
$(\ell_2, 1.8, 2.4) \xrightarrow{c} (\ell_1, 1.8, 0)$



Timed Automata

Timed automata = Finite automata + Analog clocks. [Alur and Dill 1994]





Theorem - [Alur & Dill 1994]

Checking the existence of a run reaching a location, or satisfying a Büchi condition is PSPACE-comp.

- Efficient algorithms and tools.

Robustness in Timed Automata

The semantics is idealistic

Convenient for modeling and verification but not realistic:

- Clocks are perfectly continuous and can be read exactly
- Discrete actions are instantaneous
- No lower bounds on time between consecutive actions (infinite frequency)

- How does a timed automaton perform under different assumptions?
- Timed automaton (Design) $\leftrightarrow$ Real-world system (Implementation)?

Robustness in Timed Automata

The semantics is idealistic

Convenient for modeling and verification but not realistic:

- Clocks are perfectly continuous and can be read exactly
- Discrete actions are instantaneous
- No lower bounds on time between consecutive actions (infinite frequency)

- How does a timed automaton perform under different assumptions?
- Timed automaton (Design) $\leftrightarrow$ Real-world system (Implementation)?

In this thesis:

- Study of robustness in different models of perturbations of timings.
- Several methodologies to develop robust systems with timed automata.

Overview

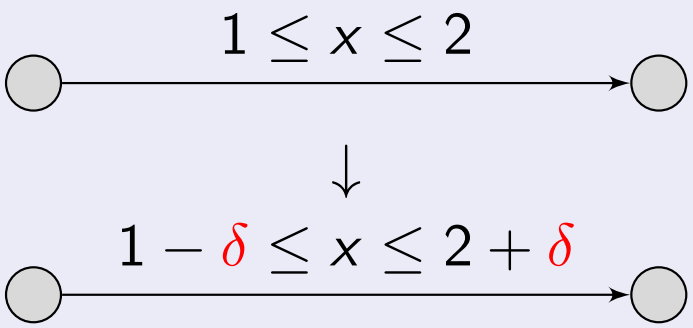
Overview

Perturbations: Guard Enlargement

Enlargement

A measuring error of $\pm\delta$ is added to all guards.

Let $\mathcal{A}_\delta$ the resulting timed automaton.

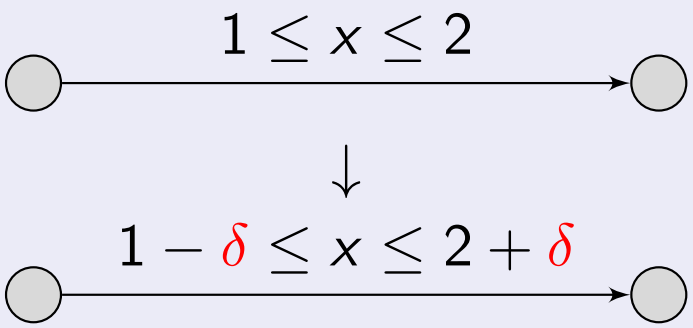


Perturbations: Guard Enlargement

Enlargement

A measuring error of $\pm\delta$ is added to all guards.

Let $\mathcal{A}_\delta$ the resulting timed automaton.



Robust model-checking

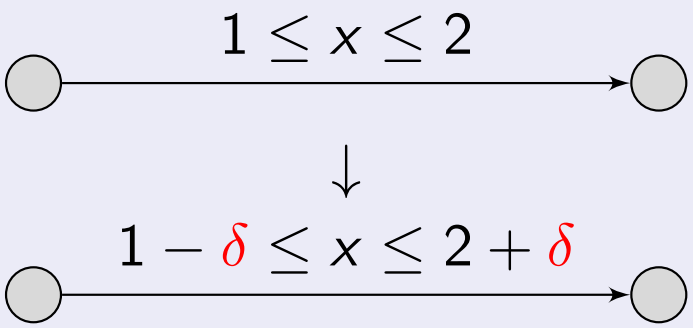
Given a timed automaton $\mathcal{A}$, and a property ϕ, check whether there exists $\delta > 0$ for which (all runs of) $\mathcal{A}_\delta$ satisfies ϕ.

Perturbations: Guard Enlargement

Enlargement

A measuring error of $\pm\delta$ is added to all guards.

Let $\mathcal{A}_\delta$ the resulting timed automaton.



Robust model-checking

Given a timed automaton $\mathcal{A}$, and a property ϕ, check whether there exists $\delta > 0$ for which (all runs of) $\mathcal{A}_\delta$ satisfies ϕ.

Methodology:

1. Design timed automaton $\mathcal{A}$
2. Model-check $\mathcal{A}_\delta$ (δ is a parameter)
3. Implement $\mathcal{A}$ (the implementation is overapproximated by $\mathcal{A}_\delta$).

Perturbations: Guard Enlargement - 2

Has been shown decidable for following properties:

- Safety is PSPACE-c, [Puri 1998], [De Wulf, Doyen, Markey, Raskin 2004]
  symbolic algorithms for flat timed automata, [Jaubert, Reynier 2011]
- Büchi, co-Büchi, LTL is PSPACE-c,
  [Bouyer, Markey, Reynier 2006] [Bouyer, Markey, S. 2011]
- A fragment of MTL is EXPSPACE-c. [Bouyer, Markey, Reynier 2008].

Perturbations: Guard Enlargement - 2

Has been shown decidable for following properties:

- Safety is PSPACE-c, [Puri 1998], [De Wulf, Doyen, Markey, Raskin 2004]
  symbolic algorithms for flat timed automata, [Jaubert, Reynier 2011]
- Büchi, co-Büchi, LTL is PSPACE-c,
  [Bouyer, Markey, Reynier 2006] [Bouyer, Markey, S. 2011]
- A fragment of MTL is EXPSPACE-c. [Bouyer, Markey, Reynier 2008].

Let $L(\mathcal{A})$ denote the **untimed language** of $\mathcal{A}$.

Theorem [S. MFCS 2011]

Checking whether there is $\delta > 0$ such that $L(\mathcal{A}) = L(\mathcal{A}_\delta)$ is in EXPSPACE.

Compare with: Untimed language equivalence is EXPSPACE-complete in timed automata with two clocks. [Brenguier, Göller, S. CONCUR 2012]

Robust Implementation Problem

Robustness analysis checks a given finished design.

Can we rather modify a design to ensure its robustness by construction?

Robust Implementation Problem

Given a timed automaton $\mathcal{A}$, construct $\mathcal{A}'$ such that

- $\mathcal{A}' \sim \mathcal{A}$
- $\mathcal{A}'$ is "robust".

where $\sim$ is timed bisimulation,

Robust Implementation Problem

Given a timed automaton $\mathcal{A}$, construct $\mathcal{A}'$ such that

- $\mathcal{A}' \sim \mathcal{A}$
- $\mathcal{A}' \approx_\epsilon \mathcal{A}'_\delta$, for some δ.

  where $\sim$ is timed bisimulation, and $\approx_\epsilon$ is approx. timed bisimulation.

Robust Implementation Problem

Given a timed automaton $\mathcal{A}$, construct $\mathcal{A}'$ such that

- $\mathcal{A}' \sim \mathcal{A}$
- $\mathcal{A}' \approx_\epsilon \mathcal{A}'_\delta$, for some δ.

  where $\sim$ is timed bisimulation, and $\approx_\epsilon$ is approx. timed bisimulation.

$s \approx_\epsilon t$ iff

- $s \xrightarrow{\sigma} s' \Rightarrow t \xrightarrow{\sigma} t'$, and $t \approx_\epsilon t'$,
- $s \xrightarrow{d} s' \Rightarrow \exists d', |d - d'| \leq \epsilon, t \xrightarrow{d'} t'$, $t \approx_\epsilon t'$,
- and symmetrically.

Robust Implementation Problem

Given a timed automaton $\mathcal{A}$, construct $\mathcal{A}'$ such that

- $\mathcal{A}' \sim \mathcal{A}$
- $\mathcal{A}' \approx_\epsilon \mathcal{A}'_\delta$, for some δ.

  where $\sim$ is timed bisimulation, and $\approx_\epsilon$ is approx. timed bisimulation.

Robust implementation methodology:

1. Design timed automaton $\mathcal{A}$,
2. Check the correctness using existing tools,
3. Implement (automatically generated) $\mathcal{A}'$.

Perturbed system is ϵ-bisimilar to the original design: $\mathcal{A}'_\delta \approx_\epsilon \mathcal{A}$.

Approach separates design and implementation: User only concentrates on the exact semantics.

Robust implementation problem: results

Theorem [Bouyer, Larsen, Markey, S., Thrane. CONCUR 2011]

For **any** timed automaton $\mathcal{A}$, and **any** $\epsilon > 0$, there exists $\mathcal{A}'$ and $\delta_0 = O(\epsilon + \frac{1}{|\mathsf{Clocks}|})$ such that

$$\mathcal{A}' \sim \mathcal{A} \quad \text{and} \quad \mathcal{A}' \approx_\epsilon \mathcal{A}'_\delta, \text{ for all } \delta \in [0, \delta_0].$$

"All timed automata are approximately implementable."

Robust implementation problem: results

Theorem [Bouyer, Larsen, Markey, S., Thrane. CONCUR 2011]

For **any** timed automaton $\mathcal{A}$, and **any** $\epsilon > 0$, there exists $\mathcal{A}'$ and $\delta_0 = O(\epsilon + \frac{1}{|\mathsf{Clocks}|})$ such that

$$\mathcal{A}' \sim \mathcal{A} \quad \text{and} \quad \mathcal{A}' \approx_\epsilon \mathcal{A}'_\delta, \text{ for all } \delta \in [0, \delta_0].$$

"All timed automata are approximately implementable."

- $\mathcal{A}'$ is computable in exponential time and $|\mathcal{A}'| = O(2^{|\mathcal{A}|}(\frac{1}{\epsilon})^{|\mathsf{clocks}|})$
- Simpler and possibly smaller version is available: $\mathcal{A} \sim \mathcal{A}'$ and same reachable locations in $\mathcal{A}'$ and $\mathcal{A}'_\delta$.

Robust Controller Synthesis

Previous algorithms concentrated on **worst-case behavior** of a given timed automaton: "Is there a run violating the property in $\mathcal{A}_\delta$?".

Robust Controller Synthesis

Previous algorithms concentrated on **worst-case behavior** of a given timed automaton: "Is there a run violating the property in $\mathcal{A}_\delta$?".

Controller synthesis: Can we construct a controller that chooses delays and edges so that a property is satisfied even in presence of perturbations?

$\rightarrow$ Controller = strategy that observes perturbations and suggests moves accordingly.

Robust Controller Synthesis

A **game** between Controller and Environment parameterized by $\delta > 0$.

Excess Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

Robust Controller Synthesis

A **game** between Controller and Environment parameterized by $\delta > 0$.

Excess Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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

Robust Controller Synthesis

A **game** between Controller and Environment parameterized by $\delta > 0$.

Excess Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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 $\epsilon \in [-\delta, \delta]$,

Robust Controller Synthesis

A **game** between Controller and Environment parameterized by $\delta > 0$.

Excess Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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 $\epsilon \in [-\delta, \delta]$,
3. New state is $(\ell', (\nu + d + \epsilon)[R \leftarrow 0])$.

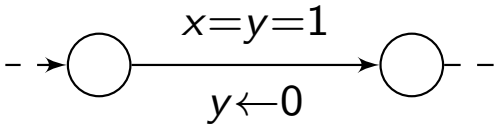
Robust Controller Synthesis

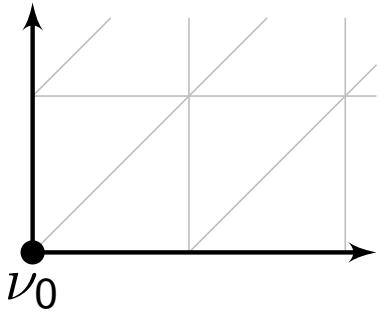
A **game** between Controller and Environment parameterized by $\delta > 0$.

Excess Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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 $\epsilon \in [-\delta, \delta]$,
3. New state is $(\ell', (\nu + d + \epsilon)[R \leftarrow 0])$.





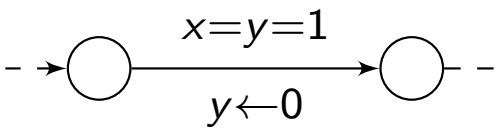
Robust Controller Synthesis

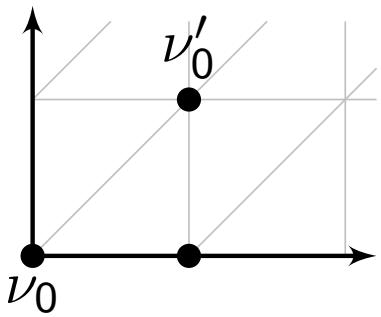
A **game** between Controller and Environment parameterized by $\delta > 0$.

Excess Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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 $\epsilon \in [-\delta, \delta]$,
3. New state is $(\ell', (\nu + d + \epsilon)[R \leftarrow 0])$.





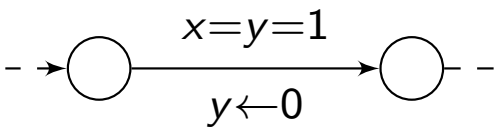
Robust Controller Synthesis

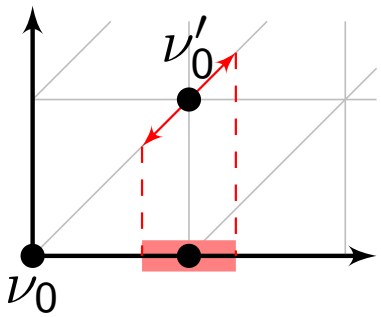
A **game** between Controller and Environment parameterized by $\delta > 0$.

Excess Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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 $\epsilon \in [-\delta, \delta]$,
3. New state is $(\ell', (\nu + d + \epsilon)[R \leftarrow 0])$.





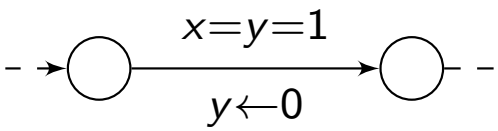
Robust Controller Synthesis

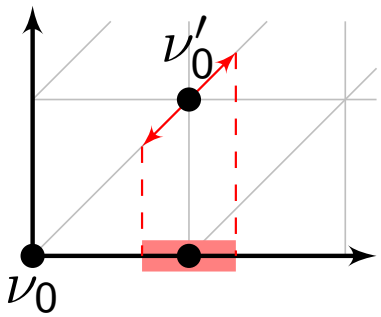
A **game** between Controller and Environment parameterized by $\delta > 0$.

Excess Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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 $\epsilon \in [-\delta, \delta]$,
3. New state is $(\ell', (\nu + d + \epsilon)[R \leftarrow 0])$.





Controller's objective: reaching a given location.
Environment's objective is avoiding the same location.

Robust Controller Synthesis: Results

Parameterized Robust Controller Synthesis

Decide whether for some $\delta > 0$, Controller has a strategy ensuring a reachability objective.

Robust Controller Synthesis: Results

Parameterized Robust Controller Synthesis

Decide whether for some $\delta > 0$, Controller has a strategy ensuring a reachability objective.

Methodology:

- Design a non-deterministic $\mathcal{A}$ describing all possible behaviors.
- Synthesize a controller that achieves the objective despite imprecisions.

Robust Controller Synthesis: Results

Parameterized Robust Controller Synthesis

Decide whether for some $\delta > 0$, Controller has a strategy ensuring a reachability objective.

Methodology:

- Design a non-deterministic $\mathcal{A}$ describing all possible behaviors.
- Synthesize a controller that achieves the objective despite imprecisions.

Theorem [Bouyer, Markey, S. ICALP'12]

Parameterized robust controller synthesis for reachability is EXPTIME-complete for timed automata and turn-based timed games.

Turn-based timed games: Environment determines delays and edges in some locations.

Summary of Guard Enlargement

Three approaches

- Robustness analysis: $L(\mathcal{A}) = L(\mathcal{A}_\delta)$?
- Approximate robust implementation: $\mathcal{A} \approx_\epsilon \mathcal{A}'_\delta$.
- Robust controller synthesis: $G_\delta(\mathcal{A})$
  - Turn-based timed games
  - Undecidability of cost-optimal reachability in weighted timed automata (not presented)

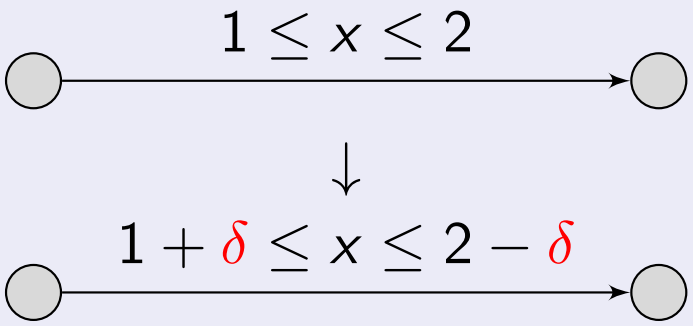
- S. Untimed Language Preservation in Timed Systems. MFCS'11.
- Bouyer, Larsen, Markey, S., Thrane. Timed Automata Can Always Be Made Implementable. CONCUR'11.
- Bouyer, Markey, S. Robust Reachability in Timed Automata: A Game-based Approach. ICALP'12.
- Bouyer, Markey, S. Robust Weighted Timed Automata and Games. Submitted.

Overview

Perturbations: Guard Shrinking

Shrinking

We require the automaton to avoid the borders of the guards by shrinking (= strengthening) them.

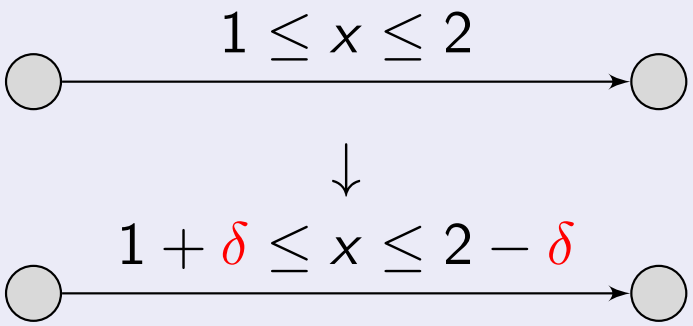


Any equality becomes empty

Perturbations: Guard Shrinking

Shrinking

We require the automaton to avoid the borders of the guards by shrinking (= strengthening) them.

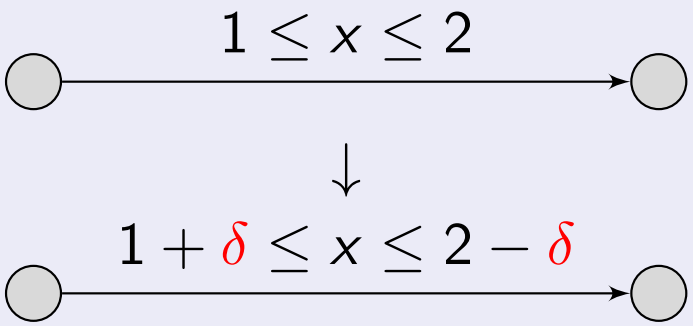


- **Robustness:** Does any significant behavior disappear under shrinking? e.g. liveness
  If yes, then some behaviors require the borders of the guards.

Perturbations: Guard Shrinking

Shrinking

We require the automaton to avoid the borders of the guards by shrinking (= strengthening) them.



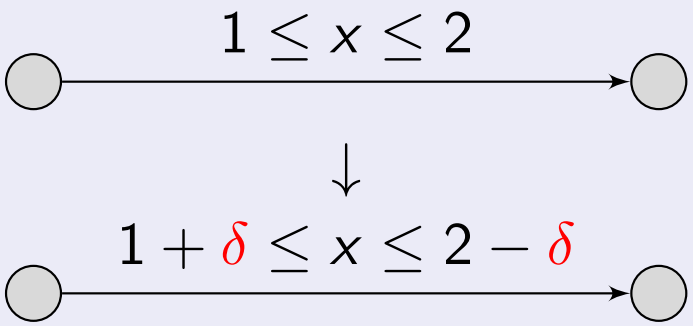
- **Robustness:** Does any significant behavior disappear under shrinking? e.g. liveness
  If yes, then some behaviors require the borders of the guards.
- **Implementation**: If one is concerned about imprecisions by guard enlargement, then

Model	Real-world behavior
$1 \leq x \leq 2$	$1 - \Delta \leq x \leq 2 + \Delta$

Perturbations: Guard Shrinking

Shrinking

We require the automaton to avoid the borders of the guards by shrinking (= strengthening) them.



- **Robustness:** Does any significant behavior disappear under shrinking? e.g. liveness
  If yes, then some behaviors require the borders of the guards.
- **Implementation**: If one is concerned about imprecisions by guard enlargement, then

Model	Real-world behavior
$1 \leq x \leq 2$	$1 - \Delta \leq x \leq 2 + \Delta$
$1 + \delta \leq x \leq 2 - \delta$	$1 + \delta - \Delta \leq x \leq 2 - \delta + \Delta \Rightarrow 1 \leq x \leq 2$

Shrinkability

Problem: Some behaviors can be lost under any shrinking.

We consider a different shrinking parameter for each atomic guard:
$1 \leq x \leq 3 \wedge y \geq 0 \quad \rightarrow \quad 1 + 2\delta \leq x \leq 3 - 5\delta \wedge y \geq 4\delta$.

Rational $\vec{\delta'} \Leftrightarrow \vec{k}\delta$.

For $\delta > 0$, and positive integer vector $\vec{k}$, let $\mathcal{A}_{-\vec{k}\delta}$ denote the automaton $\mathcal{A}$ "shrunk" by $\vec{k}\delta$.

Shrinkability

Problem: Some behaviors can be lost under any shrinking.

We consider a different shrinking parameter for each atomic guard:
$1 \leq x \leq 3 \wedge y \geq 0 \quad \rightarrow \quad 1 + 2\delta \leq x \leq 3 - 5\delta \wedge y \geq 4\delta$.

For $\delta > 0$, and positive integer vector $\vec{k}$, let $\mathcal{A}_{-\vec{k}\delta}$ denote the automaton $\mathcal{A}$ "shrunk" by $\vec{k}\delta$.

Shrinkability

Given a timed automaton $\mathcal{A}$, does there exist positive integers $\vec{k}$ and some $\delta_0 > 0$ such that $\mathcal{A}_{-\vec{k}\delta}$

- is non-blocking — non-blocking-shrinkability
- can time-abstract simulate $\mathcal{A}$ — simulation-shrinkability

for all $\delta \in [0, \delta_0]$?

Shrinkability

Problem: Some behaviors can be lost under any shrinking.

We consider a different shrinking parameter for each atomic guard:
$1 \leq x \leq 3 \wedge y \geq 0 \quad \rightarrow \quad 1 + 2\delta \leq x \leq 3 - 5\delta \wedge y \geq 4\delta$.

For $\delta > 0$, and positive integer vector $\vec{k}$, let $\mathcal{A}_{-\vec{k}\delta}$ denote the automaton $\mathcal{A}$ "shrunk" by $\vec{k}\delta$.

Shrinkability

Given a timed automaton $\mathcal{A}$, does there exist positive integers $\vec{k}$ and some $\delta_0 > 0$ such that $\mathcal{A}_{-\vec{k}\delta}$

- is non-blocking — non-blocking-shrinkability
- can time-abstract simulate some finite automaton $\mathcal{F} \sqsubseteq_{\text{t.a.}} \mathcal{A}$ — simulation-shrinkability

for all $\delta \in [0, \delta_0]$?

Shrinkability: Results

Theorem [S., Bouyer, Markey FSTTCS 2011]

- Non-blocking-shrinkability can be decided in PSPACE
- Simulation-shrinkability can be decided in time pseudo-polynomial in $\mathcal{F}$ and $\mathcal{A}$ (So $\mathcal{A} \sqsubseteq_{\text{t.a.}} \mathcal{A}_{-\vec{k}\delta}$ in EXPTIME)
- Both at the same time, in EXPTIME

Shrinkability: Results

Theorem [S., Bouyer, Markey FSTTCS 2011]

- Non-blocking-shrinkability can be decided in PSPACE
- Simulation-shrinkability can be decided in time pseudo-polynomial in $\mathcal{F}$ and $\mathcal{A}$ (So $\mathcal{A} \sqsubseteq_{\text{t.a.}} \mathcal{A}_{-\vec{k}\delta}$ in EXPTIME)
- Both at the same time, in EXPTIME

Methodology:

1. Design and verify $\mathcal{A}$.
2. Check shrinkability: $\mathcal{A}_{-\vec{k}\delta}$.
3. Implement $\mathcal{A}_{-\vec{k}\delta}$. We have $\mathcal{A} \sqsubseteq_{\text{t.a.}} \mathcal{A}_{-\vec{k}\delta+\Delta} \sqsubseteq \mathcal{A}$.

Shrinkability: Results

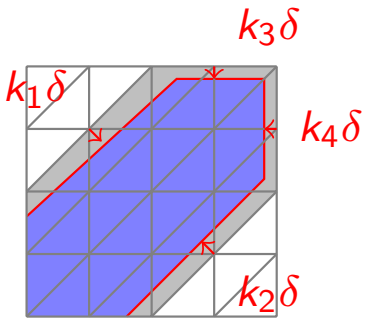
Theorem [S., Bouyer, Markey FSTTCS 2011]

- Non-blocking-shrinkability can be decided in PSPACE
- Simulation-shrinkability can be decided in time pseudo-polynomial in $\mathcal{F}$ and $\mathcal{A}$ (So $\mathcal{A} \sqsubseteq_{\text{t.a.}} \mathcal{A}_{-\vec{k}\delta}$ in EXPTIME)
- Both at the same time, in EXPTIME

Theoretical tools:

- A parameterized extension of difference-bound matrices: shrunk DBMs
- Relations between parameters $\vec{k}$ expressed as max-plus fixpoint equations on natural numbers.

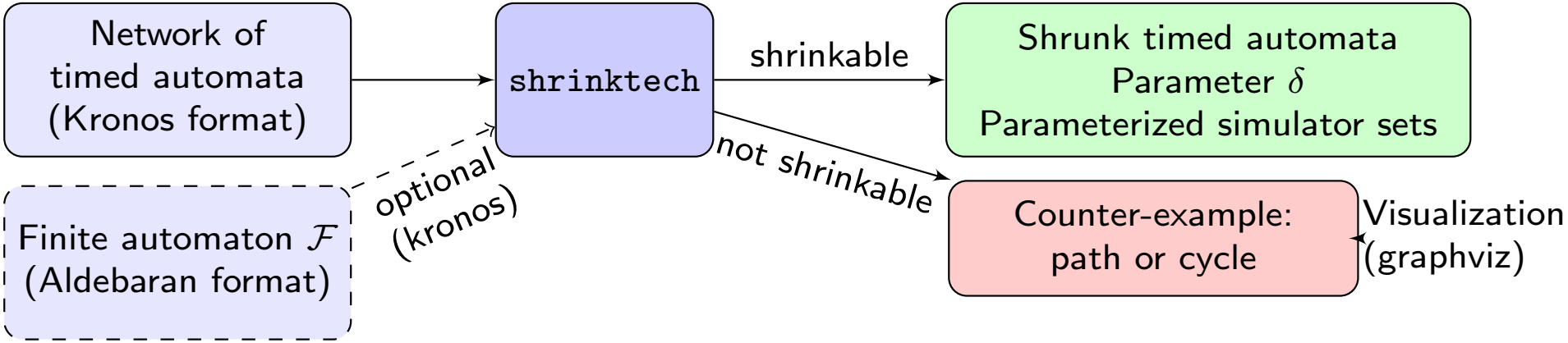
  Proof characterizes equations that have solutions.



$$k_3 = \max(k_1 + k_2, k_3)$$
$$k_2 = \max(k_2, k_1) + k_3$$

Shrinktech: Shrinkability Analysis Tool

An implementation of the simulation-shrinkability algorithm



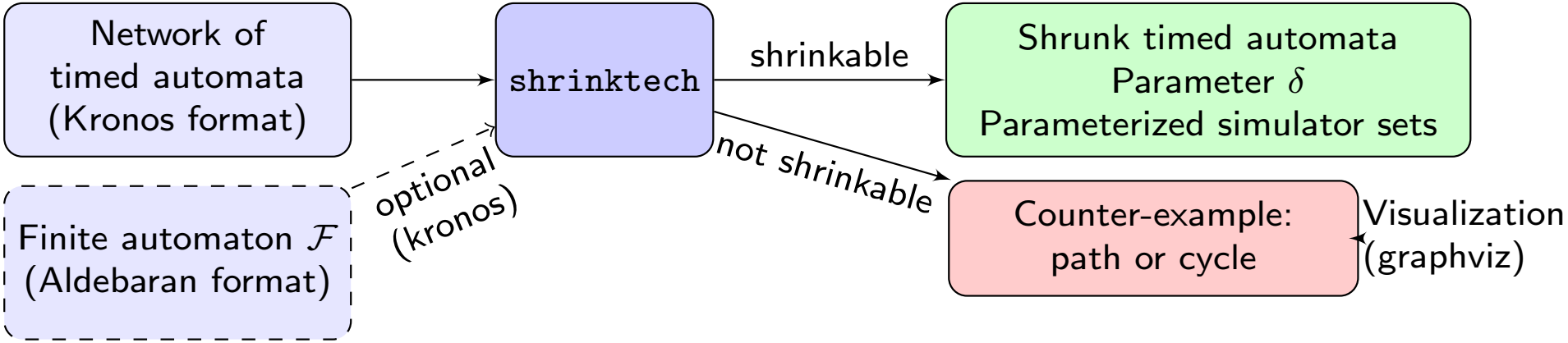
The finite automaton $\mathcal{F}$ can be
– the time-abstract bisimilarity quotient of $\mathcal{A}$ computed by Kronos,
– manually given $\mathcal{F} \sqsubseteq_{\text{t.a.}} \mathcal{A}$.

S. Shrinktech: A Tool For The Robustness Analysis of Timed Automata. CAV 2013.

http://www.lsv.ens-cachan.fr/software/shrinktech

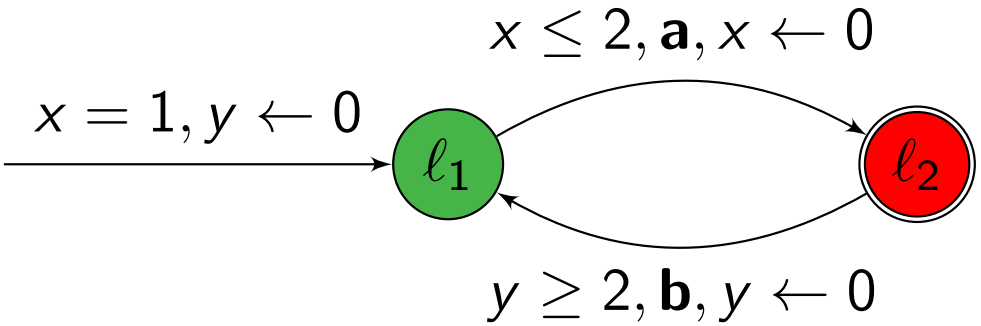
Shrinktech: Shrinkability Analysis Tool

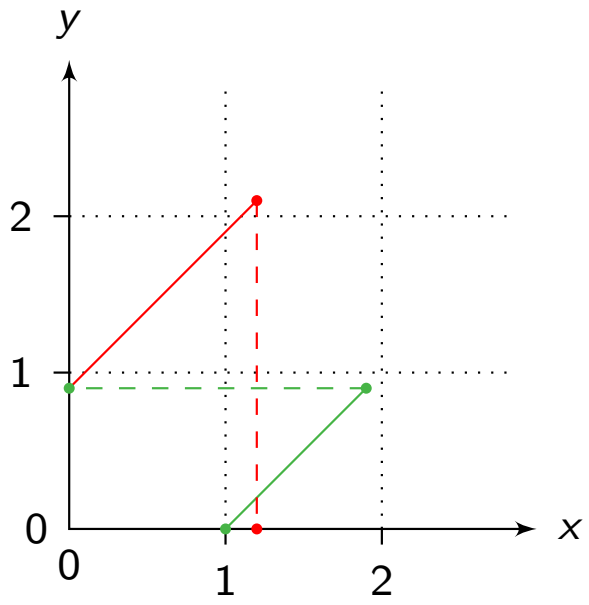
An implementation of the simulation-shrinkability algorithm



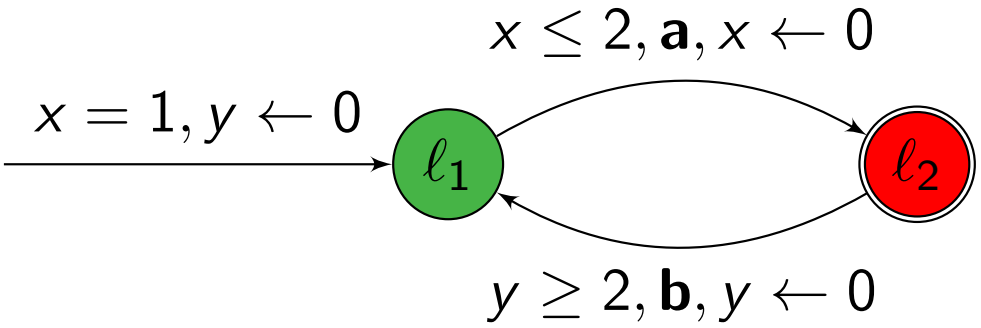
Model	**states**	**trans**	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{F}\rvert$	**time**	**shrinkable**
Lip-Sync Prot.	230	680	5	4484/48049	28s	No
Philips Audio Prot.	446	2097	2	437/2734	46s	Yes
Train Gate Controller	68	199	11	952/8540	34s	No
Fischer's Protocol 3	152	464	3	472/4321	20s	Yes
Fischer's Protocol 4	752	2864	4	4382/65821	310min	Yes
And-Or Circuit	12	20	4	80/497	1.3s	Yes
Flip-Flop Circuit	22	34	5	30/64	0.9s	Yes
Latch Circuit	32	77	7	105/364	1.6s	Yes

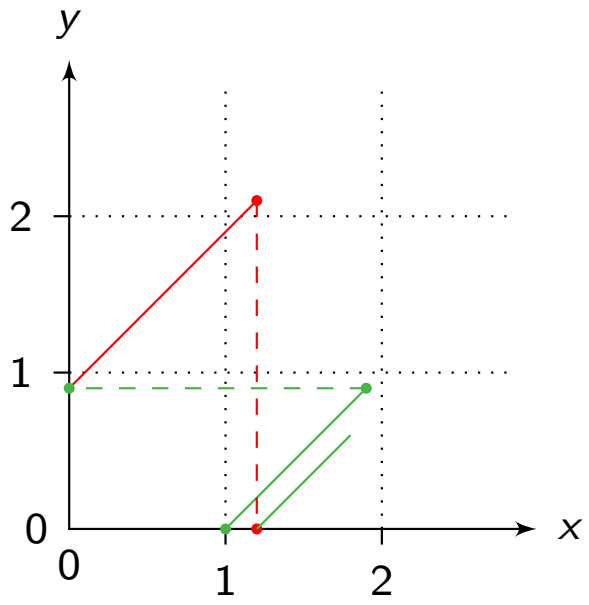
Robust Büchi Acceptance



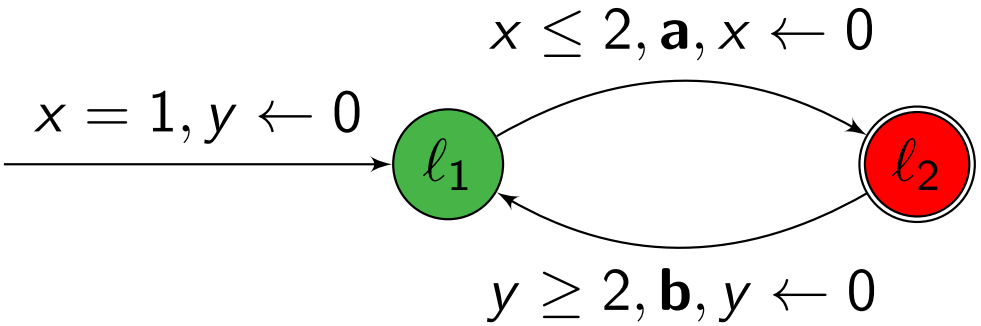


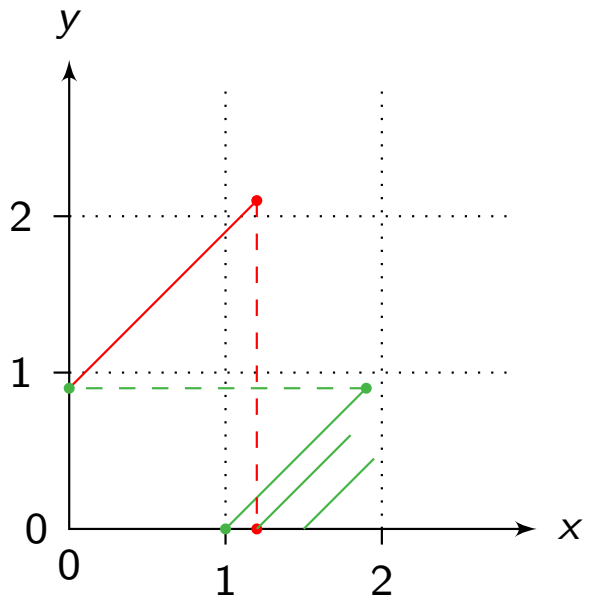
Robust Büchi Acceptance





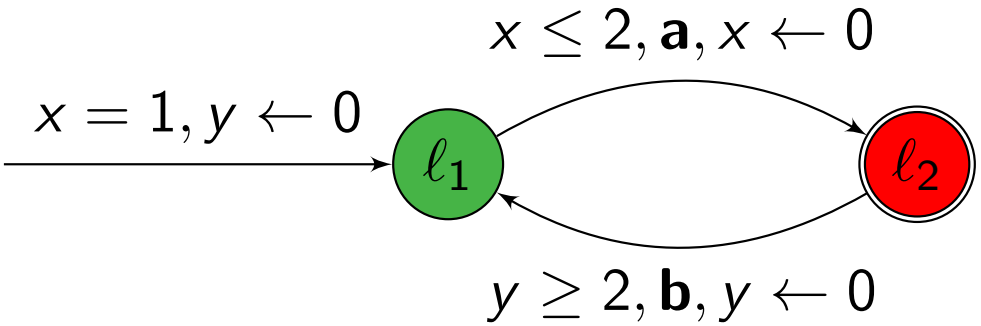
Robust Büchi Acceptance

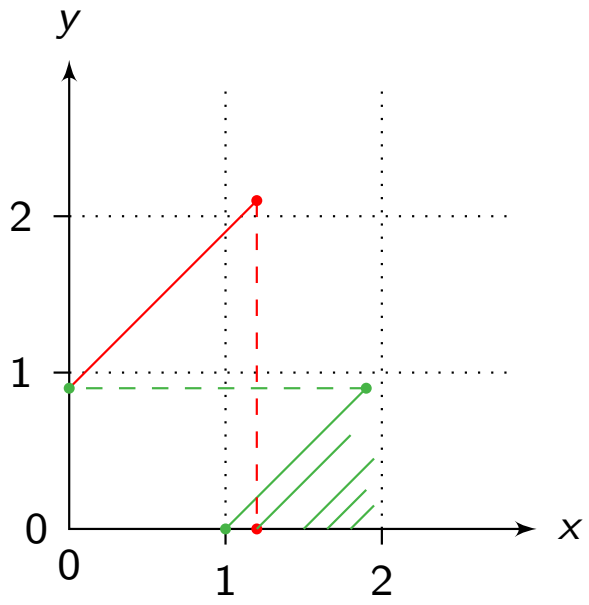




Consecutives values of x at ℓ_1 are nondecreasing, and always $x \leq 2$.

Robust Büchi Acceptance

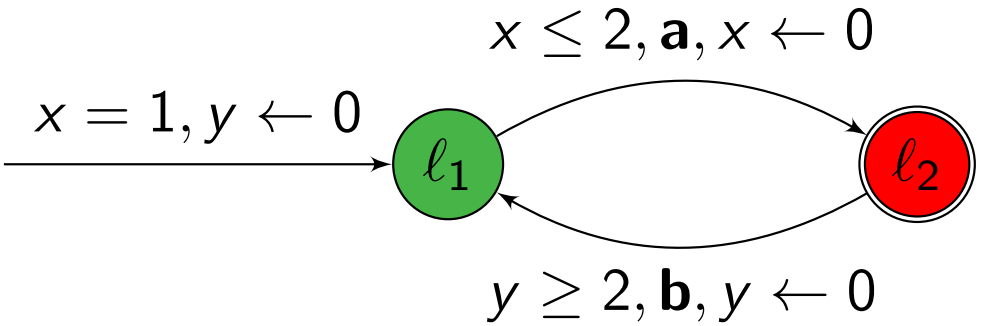


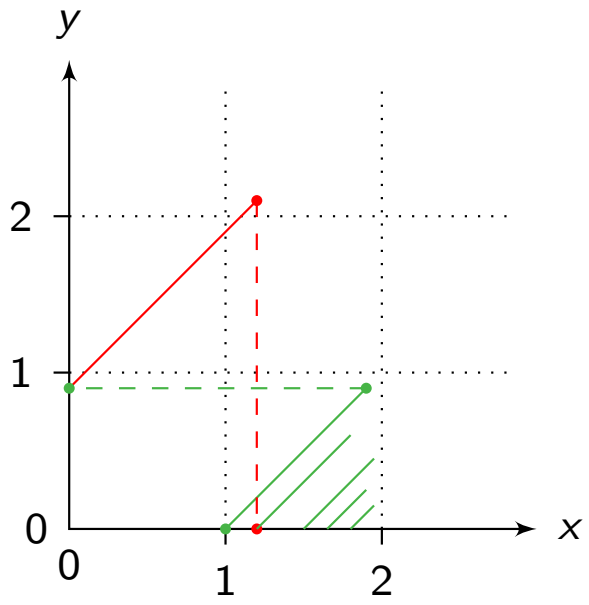


Consecutives values of x at ℓ_1 are nondecreasing, and always $x \leq 2$.

- Along any infinite run, the clock x needs infinite precision.
  A real run would actually be blocking.

Robust Büchi Acceptance





Consecutives values of x at ℓ_1 are nondecreasing, and always $x \leq 2$.

- Along any infinite run, the clock x needs infinite precision.
  A real run would actually be blocking.

How to check if there is an infinite run realizable with "finite precision" delays?

Robust Büchi Acceptance

Some behaviors in timed automata are not realistic,
and may require high precision, and convergence.

Goal: Suggest an alternative notion of Büchi acceptance for timed automata: only accept realizable runs, avoid convergence.

Perturbation Game Semantics

A **game** between Controller and Environment parameterized by $\delta > 0$.

Conservative Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d + \epsilon \models g$ for all $\epsilon \in [-\delta, \delta]$.

Perturbation Game Semantics

A **game** between Controller and Environment parameterized by $\delta > 0$.

Conservative Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d + \epsilon \models g$ for all $\epsilon \in [-\delta, \delta]$.
2. Environment chooses $\epsilon \in [-\delta, \delta]$,

Perturbation Game Semantics

A **game** between Controller and Environment parameterized by $\delta > 0$.

Conservative Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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

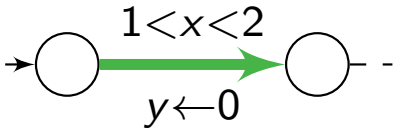
Perturbation Game Semantics

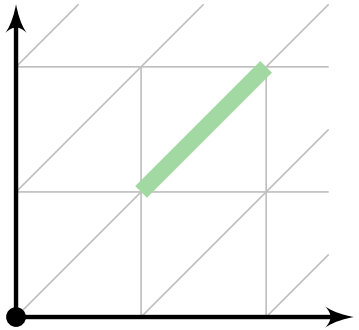
A **game** between Controller and Environment parameterized by $\delta > 0$.

Conservative Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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





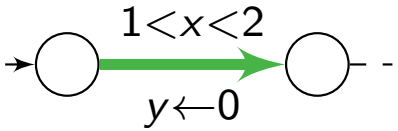
Perturbation Game Semantics

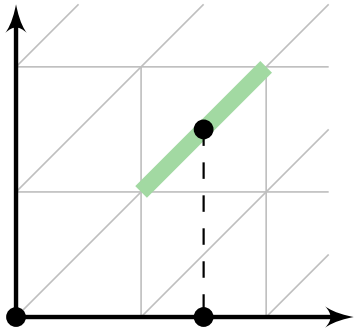
A **game** between Controller and Environment parameterized by $\delta > 0$.

Conservative Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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





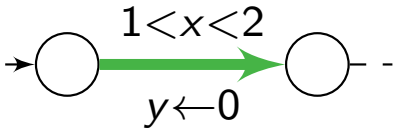
Perturbation Game Semantics

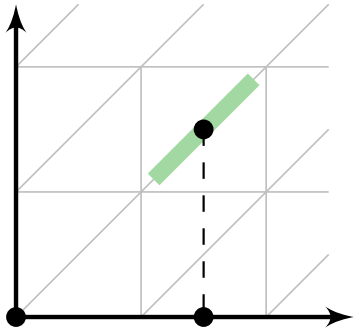
A **game** between Controller and Environment parameterized by $\delta > 0$.

Conservative Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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





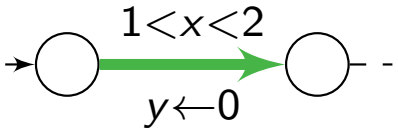
Perturbation Game Semantics

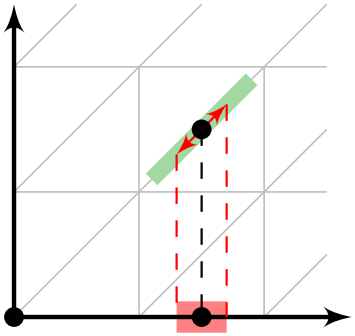
A **game** between Controller and Environment parameterized by $\delta > 0$.

Conservative Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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





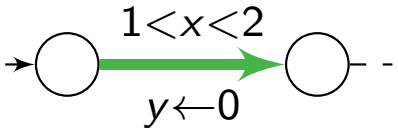
Perturbation Game Semantics

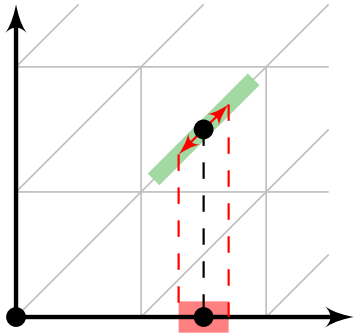
A **game** between Controller and Environment parameterized by $\delta > 0$.

Conservative Game Semantics $G_\delta(\mathcal{A})$

At any state (ℓ, ν),

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





Controller's objective: ensuring a Büchi condition
Environment's objective: the complement

Result: Parameterized Robust Controller Synthesis

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

Parameterized Robust Controller Synthesis

Decide whether for some $\delta > 0$, Controller has a strategy ensuring the Büchi condition.

Such an infinite run is then realizable despite imprecisions

Result: Parameterized Robust Controller Synthesis

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

Parameterized Robust Controller Synthesis

Decide whether for some $\delta > 0$, Controller has a strategy ensuring the Büchi condition.

Theorem [S., Bouyer, Markey, Reynier. Submitted].

Parameterized robust controller synthesis for Büchi objectives is PSPACE-complete on timed automata

The problem consists in finding cycles that do not become blocked (= aperiodicity)

Result: Parameterized Robust Controller Synthesis

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

Parameterized Robust Controller Synthesis

Decide whether for some $\delta > 0$, Controller has a strategy ensuring the Büchi condition.

Theorem [S., Bouyer, Markey, Reynier. Submitted].

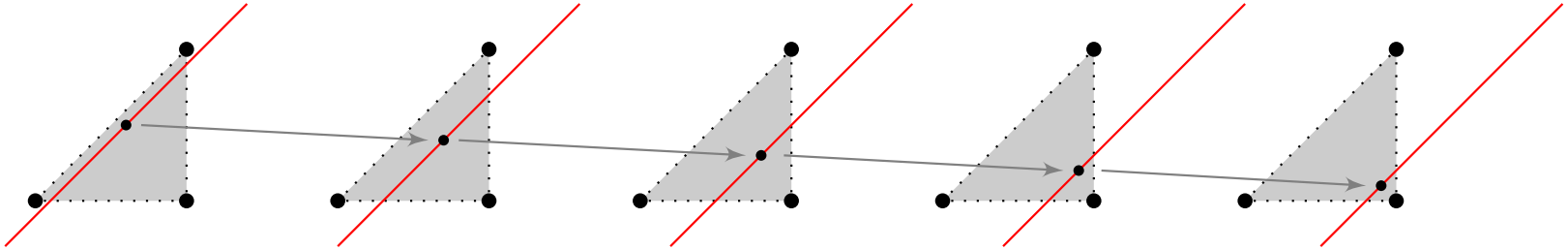
Parameterized robust controller synthesis for Büchi objectives is PSPACE-complete on timed automata

Robustly controllable $\Leftrightarrow$ there exists an "aperiodic" lasso.

The problem consists in finding cycles that do not become blocked (= aperiodicity)

Aperiodic vs Non-aperiodic [Asarin, Basset 2011]

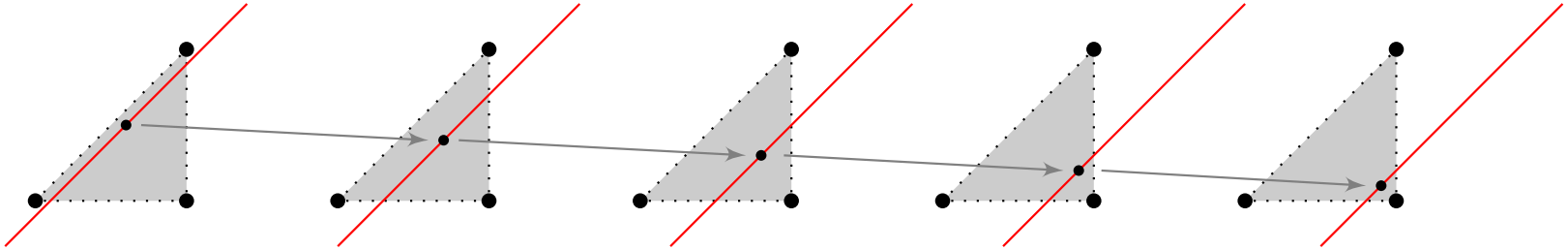
Non-aperiodic cycle:



At each iteration, the only reachable states are in the bottom half-space.

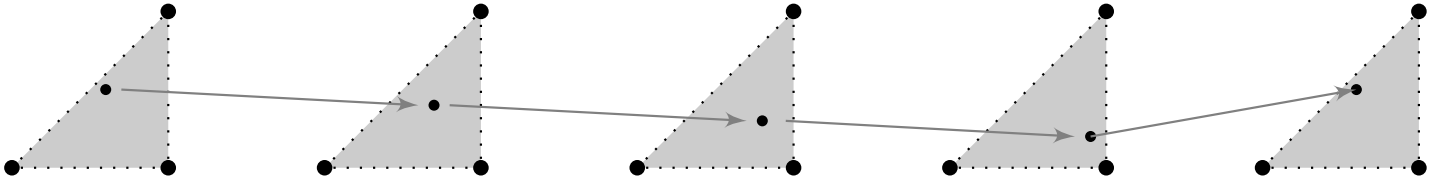
Aperiodic vs Non-aperiodic [Asarin, Basset 2011]

Non-aperiodic cycle:



At each iteration, the only reachable states are in the bottom half-space.

Aperiodic cycle:

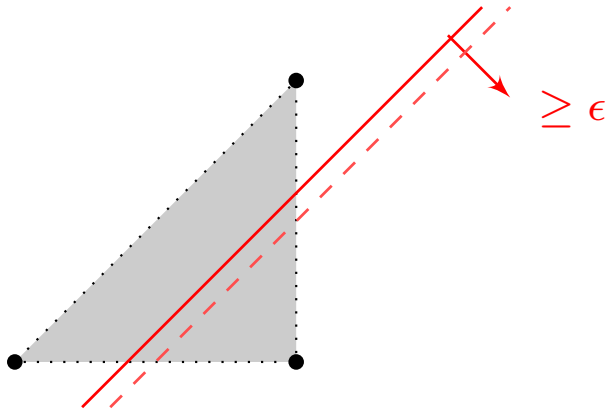


No such constraining half-spaces.

Non-aperiodic Cycles in the Game Semantics

Lemma

Environment has a strategy ensuring a distance of at least ϵ from the half-space, along any non-aperiodic cycle.

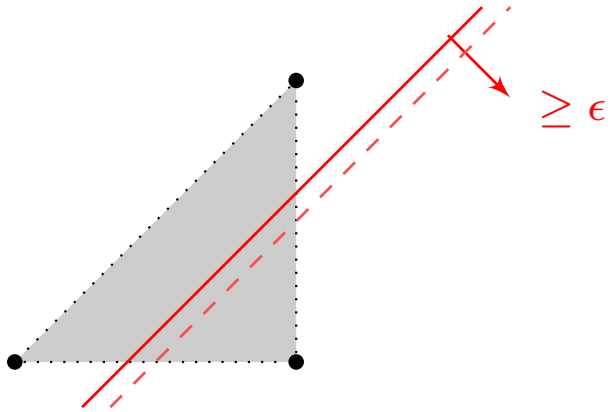


No infinite iteration of such a cycle is possible $\Rightarrow$ One cannot satisfy Büchi.

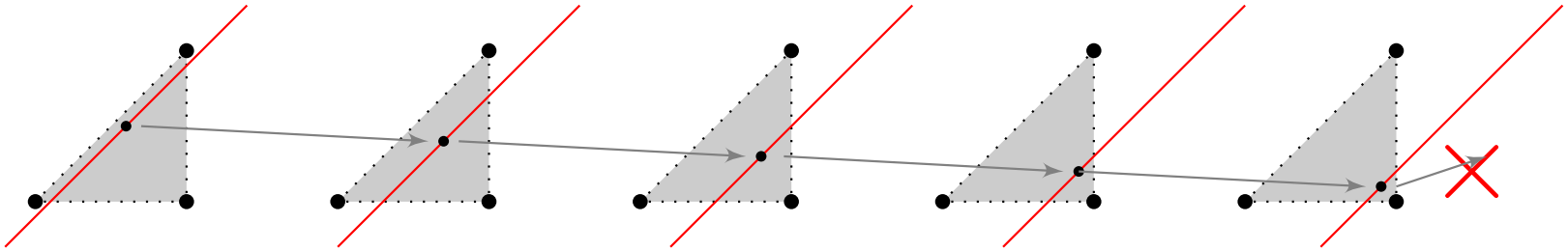
Non-aperiodic Cycles in the Game Semantics

Lemma

Environment has a strategy ensuring a distance of at least ϵ from the half-space, along any non-aperiodic cycle.



No infinite iteration of such a cycle is possible $\Rightarrow$ One cannot satisfy Büchi.



Summary of Shrinking

- Shrinkability analysis: a new notion for robustness and implementability.
- Software tool and experimental results.
- Robust Büchi acceptance.
- Perturbation game semantics: Decidable cost-optimal reachability for weighted timed automata, but undecidable for weighted timed games (not presented).

- S., Bouyer, Markey. Shrinking Timed Automata. FSTTCS 2011.
- S. `Shrinktech`: A Tool for the Robustness Analysis of Timed Automata. CAV 2013.
- S., Bouyer, Markey, Reynier. Robust Controller Synthesis for Timed Automata. Submitted.
- Bouyer, Markey, S. Robust Weighted Timed Automata and Games. Submitted.

Conclusion

- Several perturbation models:
  - Enlargement: syntactic, game semantics.
  - Shrinking: syntactic, game semantics.
  - Sampling (not presented).
- Several methodologies:
  - Robustness analysis
  - Robust controller synthesis
  - Robust implementation
- Software tool for shrinkability analysis.
- Parameter synthesis.

Perspectives

- Same or close computational complexity as the classical setting

  $+$

  Extensions of some techniques from the exact case: shrunk DBMs, regions with shrinking constraints, orbit graphs

  $=$

  Symbolic algorithms?
- Robust controller synthesis on timed games (by giving Environment more power)
- Probabilistic perturbation models:
  - Almost-sure reachability and safety
  - Quantifying mean-time to failure
- Compositional robustness