Robustness in Timed Automata: A Game-Based Approach

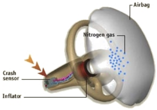
Patricia Bouyer-Decitre

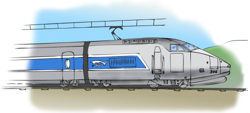
LSV, CNRS & ENS Cachan, France

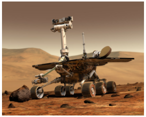
Based on joint works with Nicolas Markey, Pierre-Alain Reynier and Ocan Sankur
Acknowledgment to Nicolas and Ocan for slides

Time-dependent systems

- We are interested in timed systems

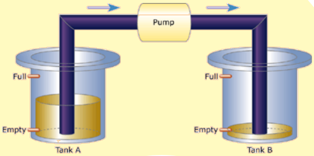






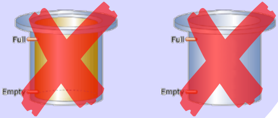
Model-checking and control

system:



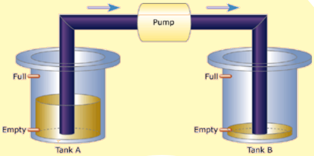
[http://www.embedded.com]

property:



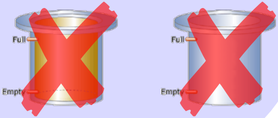
Model-checking and control

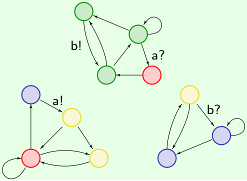
system:



[http://www.embedded.com]

property:



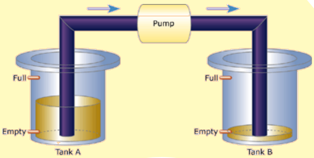


AG(¬`B.overfull`

∧ ¬`B.dried_up`)

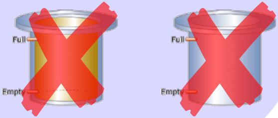
Model-checking and control

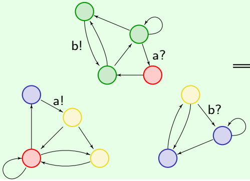
system:



[http://www.embedded.com]

property:



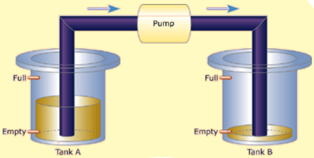


algorithm

$$\mathbf{AG}(\neg\texttt{B.overfull} \wedge \neg\texttt{B.dried\_up})$$

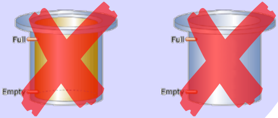
Model-checking and control

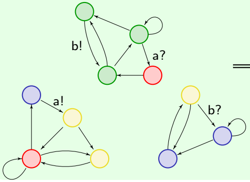
system:



[http://www.embedded.com]

property:





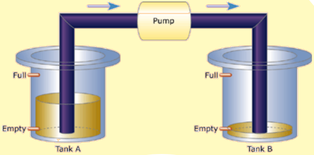
model-checking algorithm

$\mathbf{AG}(\neg\mathtt{B.overfull} \wedge \neg\mathtt{B.dried\_up})$

yes/no

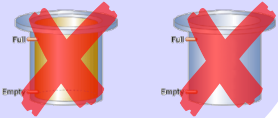
Model-checking and control

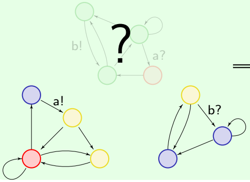
system:



[http://www.embedded.com]

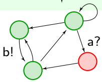
property:





control/synthesis algorithm

$\mathbf{AG}(\neg \texttt{B.overfull} \wedge \neg \texttt{B.dried\_up})$



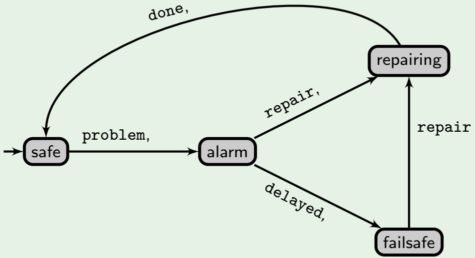
Reasoning about real-time systems

The model of timed automata [AD94]

A timed automaton is made of

- a finite automaton-based structure

Example



[AD94] Alur, Dill: A Theory of Timed Automata. Theor. Comp. Science, 1994.

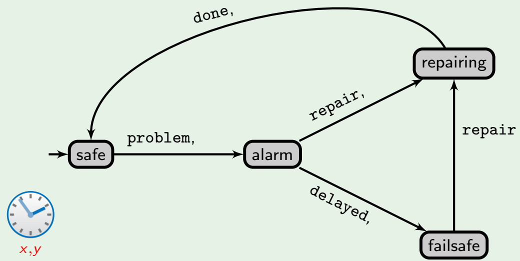
Reasoning about real-time systems

The model of timed automata [AD94]

A timed automaton is made of

- a finite automaton-based structure
- a set of clocks

Example



[AD94] Alur, Dill: A Theory of Timed Automata. Theor. Comp. Science, 1994.

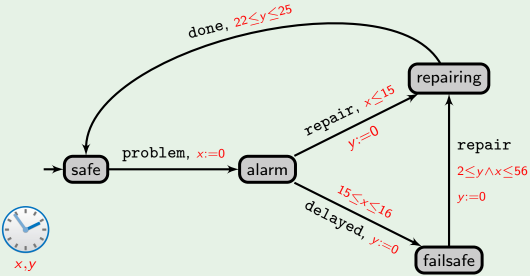
Reasoning about real-time systems

The model of timed automata [AD94]

A timed automaton is made of

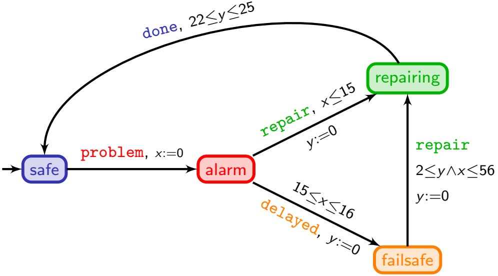
- a finite automaton-based structure
- a set of clocks
- timing constraints on transitions

Example

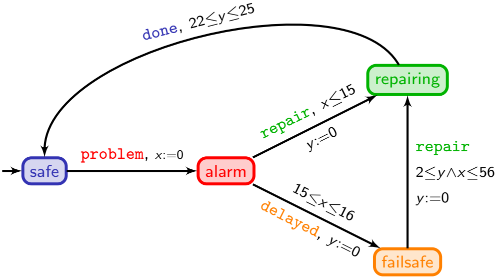


[AD94] Alur, Dill: A Theory of Timed Automata. Theor. Comp. Science, 1994.

Example – cont'd

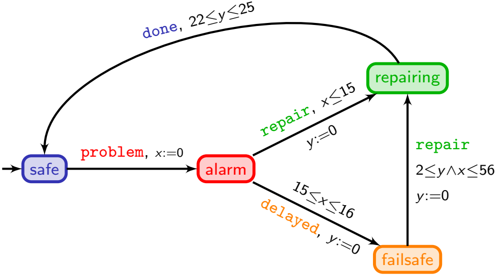


Example – cont'd



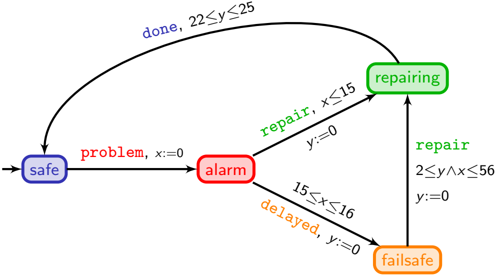
	safe
x	0
y	0

Example – cont'd



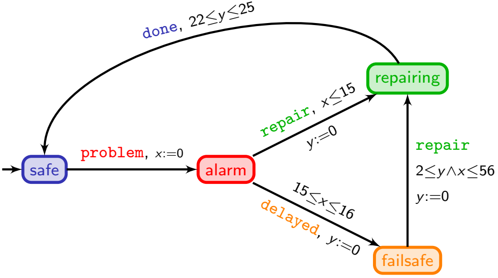
	safe	$\xrightarrow{23}$	safe
x	0		23
y	0		23

Example – cont'd



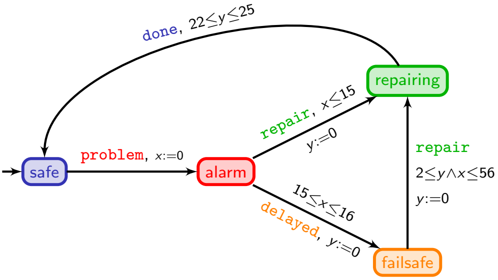
	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm
x	0		23		0
y	0		23		23

Example – cont'd



	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm
x	0		23		0		15.6
y	0		23		23		38.6

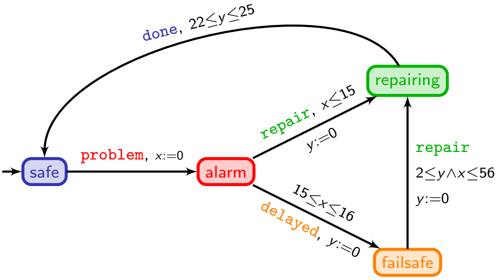
Example – cont'd



	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	$\cdots$
y	0		23		23		38.6		0	

	failsafe
$\cdots$	15.6
	0

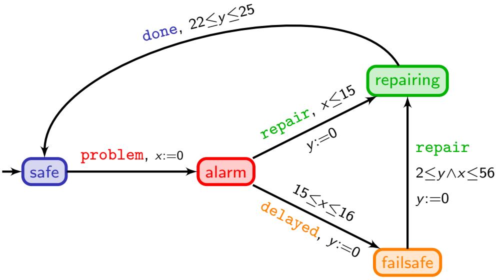
Example – cont'd



	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	⋯
y	0		23		23		38.6		0	

	failsafe	$\xrightarrow{2.3}$	failsafe
⋯	15.6		17.9
	0		2.3

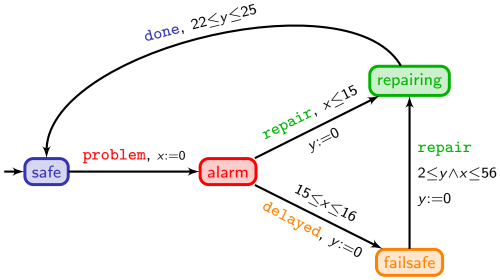
Example – cont'd



	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	···
y	0		23		23		38.6		0	

	failsafe	$\xrightarrow{2.3}$	failsafe	$\xrightarrow{\texttt{repair}}$	repairing
···	15.6		17.9		17.9
	0		2.3		0

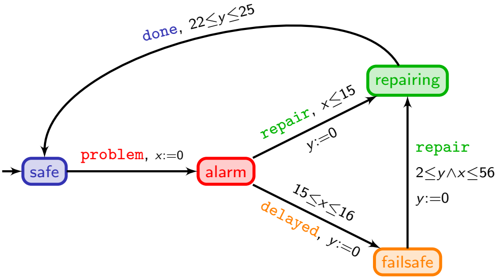
Example – cont'd



	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	$\cdots$
y	0		23		23		38.6		0	

	failsafe	$\xrightarrow{2.3}$	failsafe	$\xrightarrow{\texttt{repair}}$	repairing	$\xrightarrow{22.1}$	repairing
$\cdots$	15.6		17.9		17.9		40
	0		2.3		0		22.1

Example – cont'd



	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	···
y	0		23		23		38.6		0	

	failsafe	$\xrightarrow{2.3}$	failsafe	$\xrightarrow{\texttt{repair}}$	repairing	$\xrightarrow{22.1}$	repairing	$\xrightarrow{\texttt{done}}$	safe
···	15.6		17.9		17.9		40		40
	0		2.3		0		22.1		22.1

Discrete-time semantics

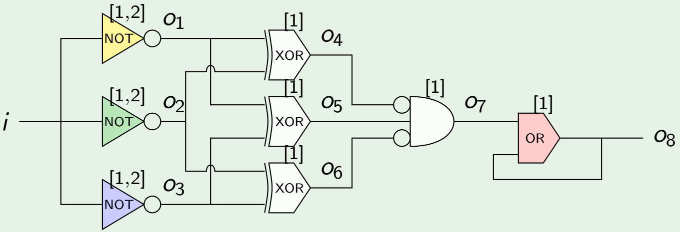
...because computers are digital!

[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

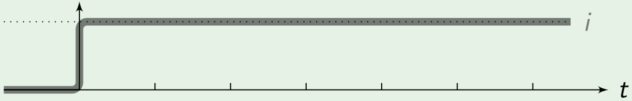
Discrete-time semantics

...because computers are digital!

Example [Alur91]



- under discrete-time, the output is always 0:

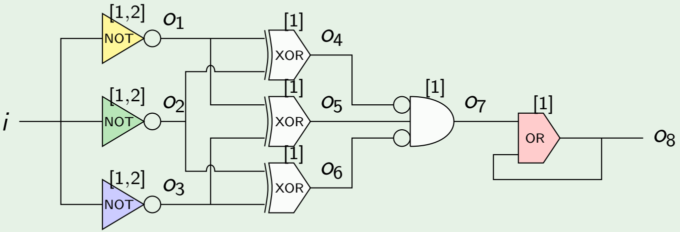


[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

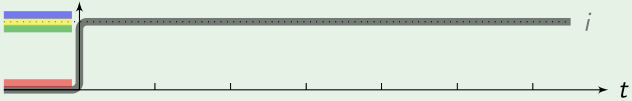
Discrete-time semantics

...because computers are digital!

Example [Alur91]



- under discrete-time, the output is always 0:

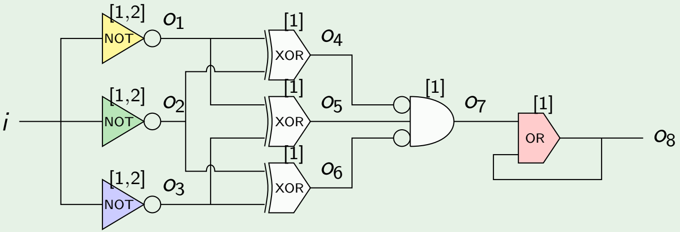


[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

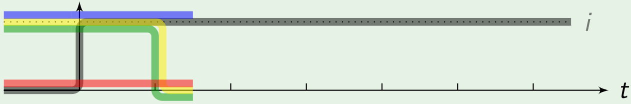
Discrete-time semantics

...because computers are digital!

Example [Alur91]



- under discrete-time, the output is always 0:

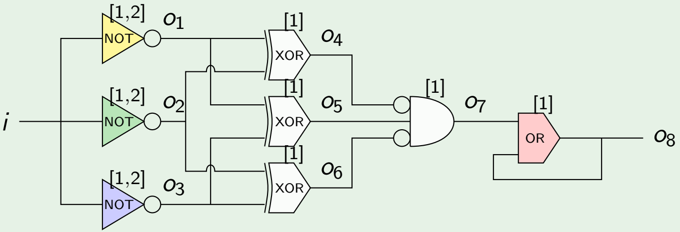


[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

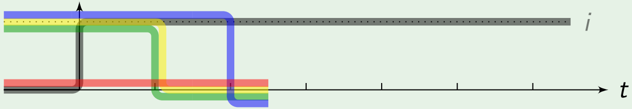
Discrete-time semantics

…because computers are digital!

Example [Alur91]



- under discrete-time, the output is always 0:

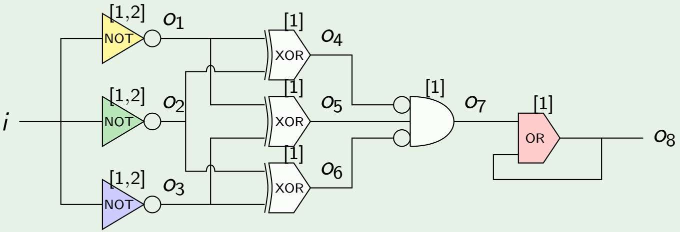


[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

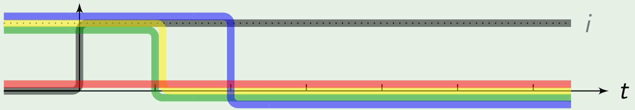
Discrete-time semantics

...because computers are digital!

Example [Alur91]



- under discrete-time, the output is always 0:

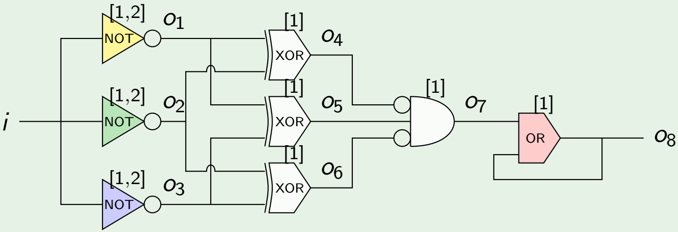


[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

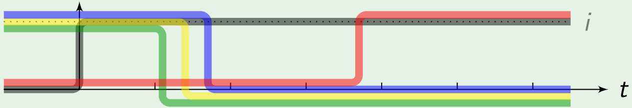
Discrete-time semantics

...because computers are digital!

Example [Alur91]



- under continuous-time, the output can be 1:



[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

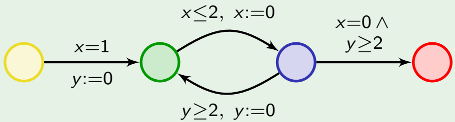
Continuous-time semantics

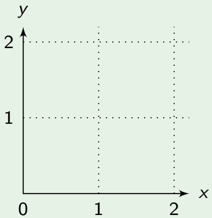
...real-time models for real-time systems!

Continuous-time semantics

...real-time models for real-time systems!

Example

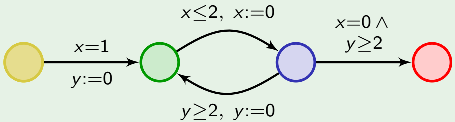


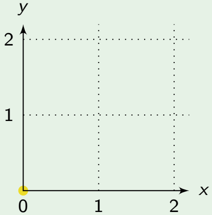


Continuous-time semantics

...real-time models for real-time systems!

Example

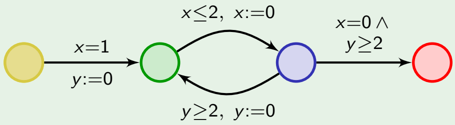


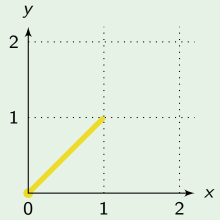


Continuous-time semantics

...real-time models for real-time systems!

Example

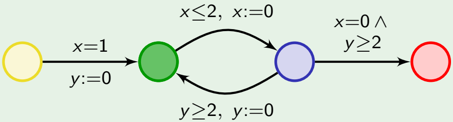


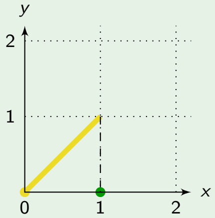


Continuous-time semantics

...real-time models for real-time systems!

Example

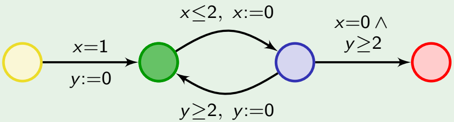


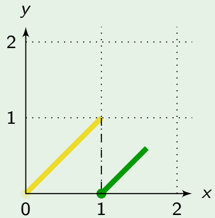


Continuous-time semantics

...real-time models for real-time systems!

Example

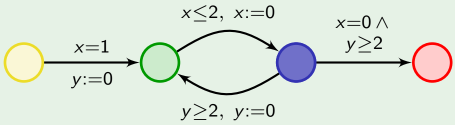


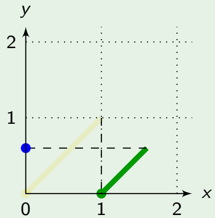


Continuous-time semantics

...real-time models for real-time systems!

Example

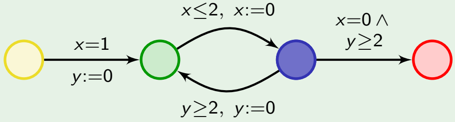


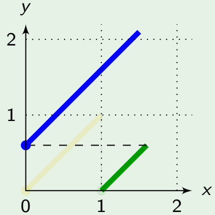


Continuous-time semantics

...real-time models for real-time systems!

Example

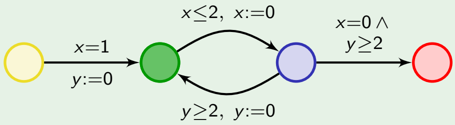


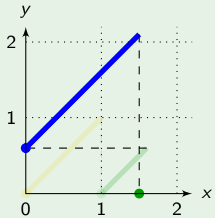


Continuous-time semantics

...real-time models for real-time systems!

Example

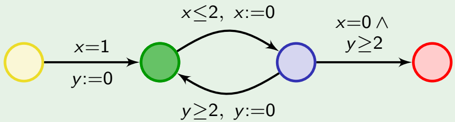


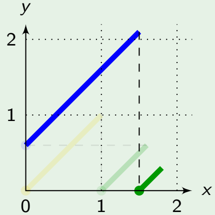


Continuous-time semantics

...real-time models for real-time systems!

Example

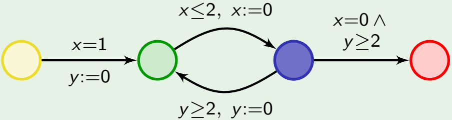


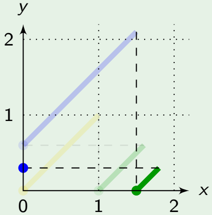


Continuous-time semantics

...real-time models for real-time systems!

Example

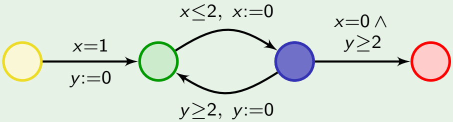


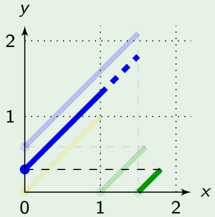


Continuous-time semantics

...real-time models for real-time systems!

Example

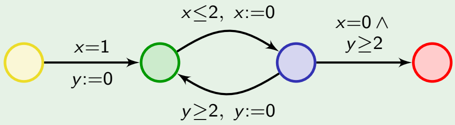


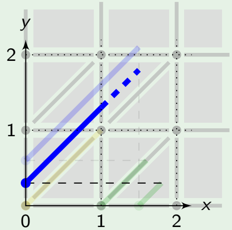


Continuous-time semantics

...real-time models for real-time systems!

Example

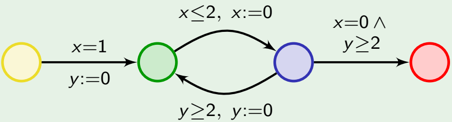


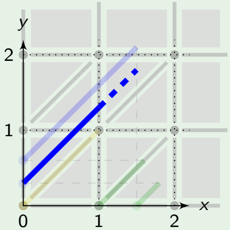


Continuous-time semantics

...real-time models for real-time systems!

Example





Theorem [AD94]

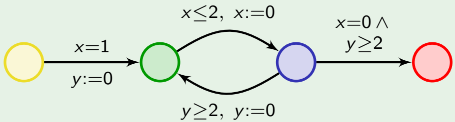
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

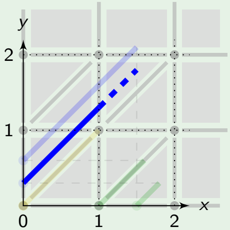
- Technical tool: region abstraction

Continuous-time semantics

...real-time models for real-time systems!

Example





Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

- Technical tool: region abstraction
- Efficient symbolic technics based on zones, implemented in tools

Are we doing the right job?

The continuous-time semantics is
an idealization of a physical system.
It is adequate for abstract design and high-level analysis.

Are we doing the right job?

The continuous-time semantics is
an idealization of a physical system.
It is adequate for abstract design and high-level analysis.

However it suffers from multiple inaccuracies:

- It might not be proper for implementation
  $\rightsquigarrow$ analysis made at the abstract level does not transfer to real world

Are we doing the right job?

**The continuous-time semantics is
an idealization of a physical system.
It is adequate for abstract design and high-level analysis.**

However it suffers from multiple inaccuracies:

- It might not be proper for implementation
  $\rightsquigarrow$ analysis made at the abstract level does not transfer to real world
- It may generate timing anomalies

Are we doing the right job?

The continuous-time semantics is
an idealization of a physical system.
It is adequate for abstract design and high-level analysis.

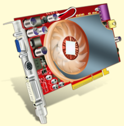
However it suffers from multiple inaccuracies:

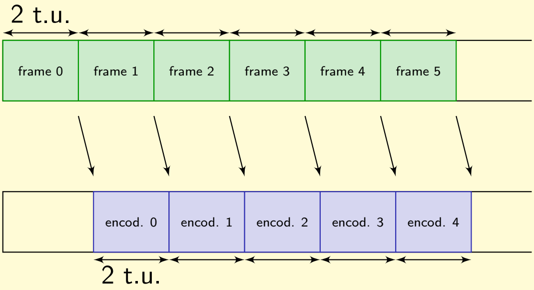
- It might not be proper for implementation
  $\rightsquigarrow$ analysis made at the abstract level does not transfer to real world
- It may generate timing anomalies
- It does not exclude non-realizable behaviours:
  - not only Zeno behaviours
  - many convergence phenomena are hidden

$\rightsquigarrow$ this requires infinite precision and might not be realizable

Example 1: Imprecision on clock values

Frame capture

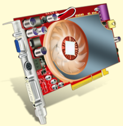


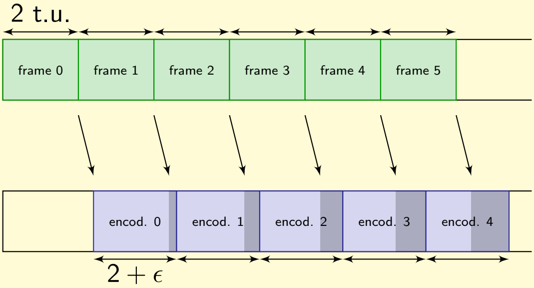


[ACS10] Abdellatif, Combaz, Sifakis. Model-based implementation of real-time applications. *Int. Conf. Embedded Software*, ACM 2010.

Example 1: Imprecision on clock values

Frame capture



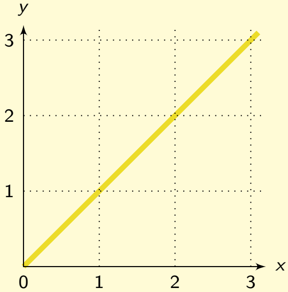


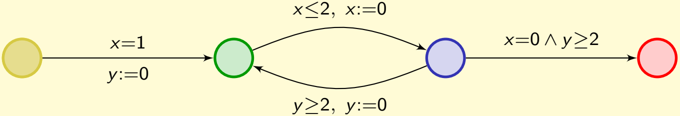
$\rightsquigarrow$ A frame will eventually be skipped

[ACS10] Abdellatif, Combaz, Sifakis. Model-based implementation of real-time applications. *Int. Conf. Embedded Software*, ACM 2010.

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

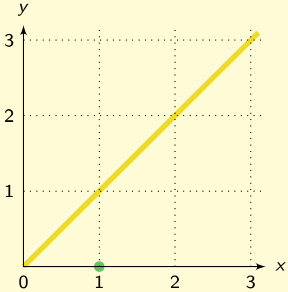


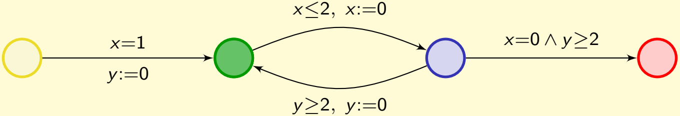


[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

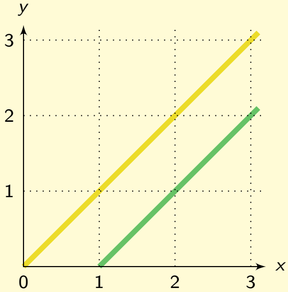


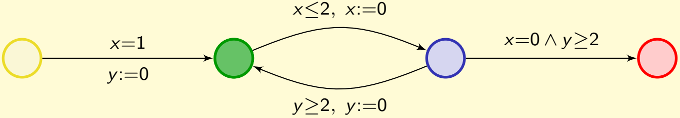


[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

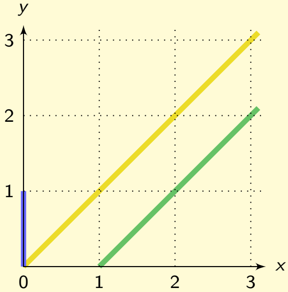


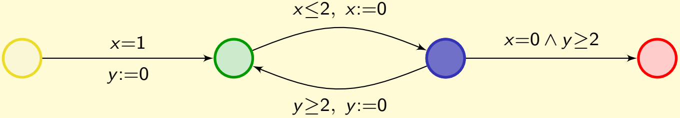


[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

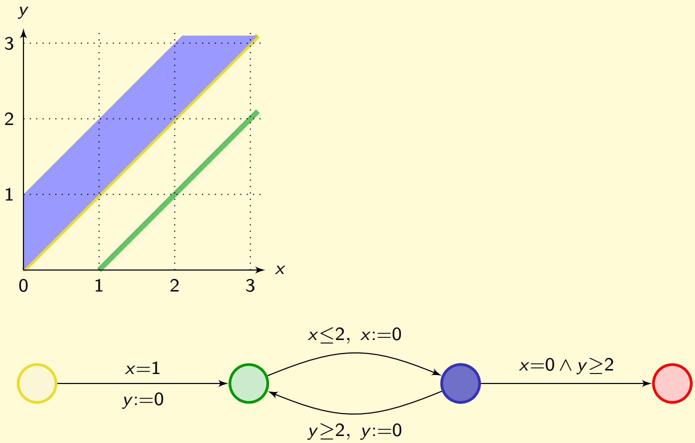




[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

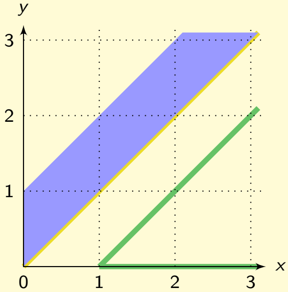
Impact of small timing jitters

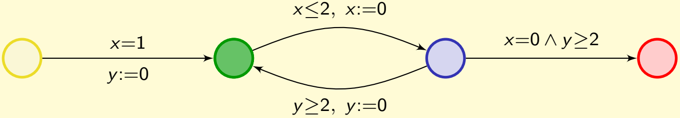


[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

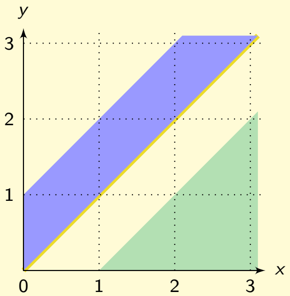


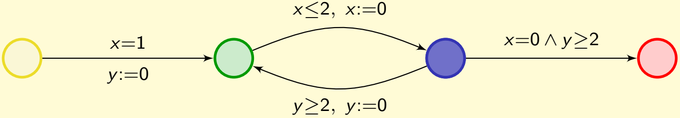


[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

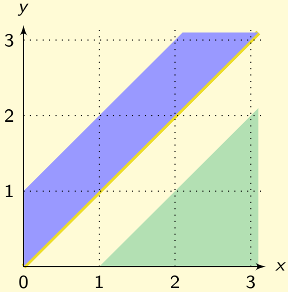


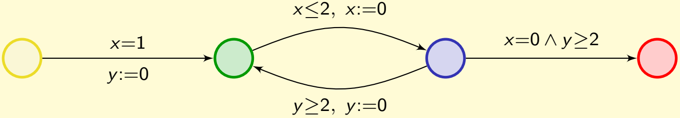


[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

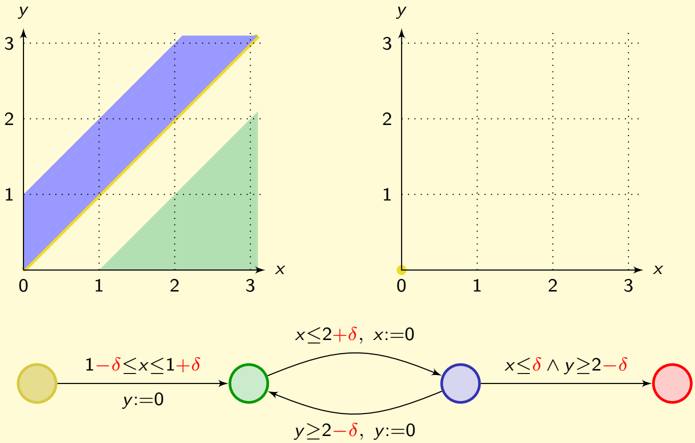




[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

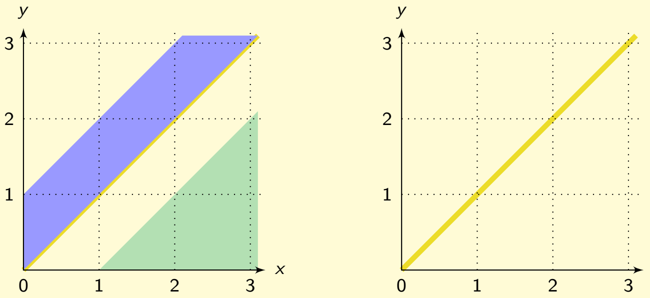
Impact of small timing jitters

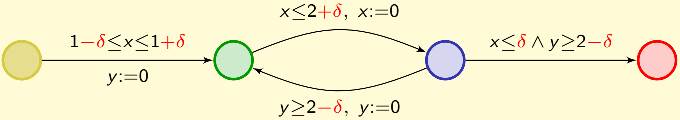


[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters

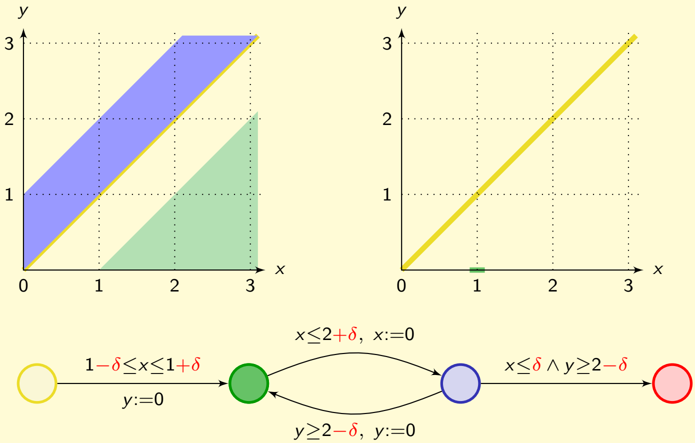




[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

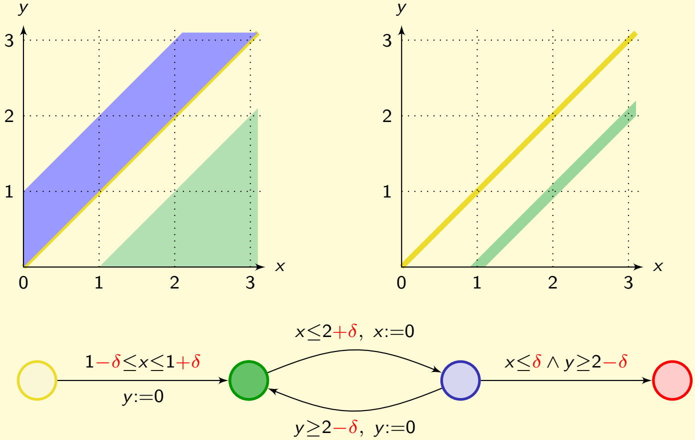
Impact of small timing jitters



[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

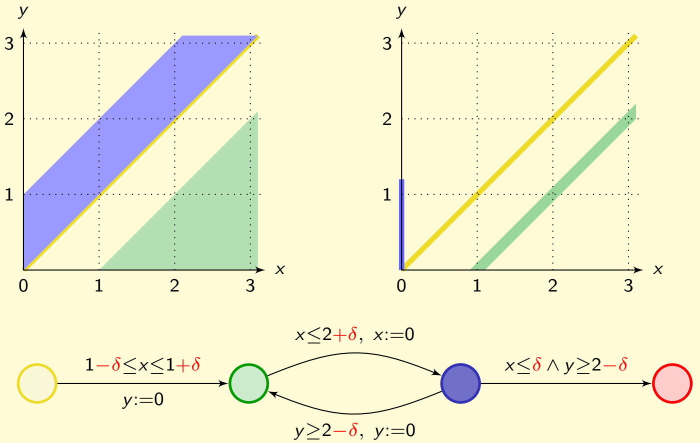
Impact of small timing jitters



[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

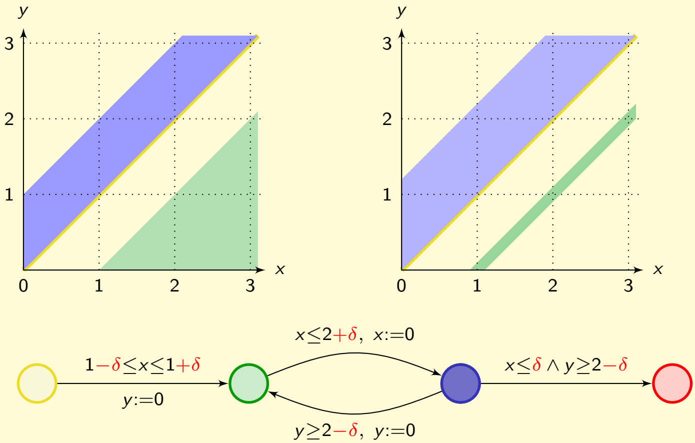
Impact of small timing jitters



[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

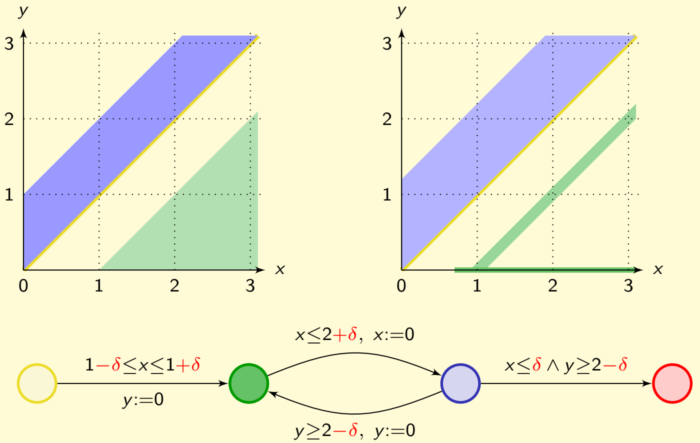
Impact of small timing jitters



[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

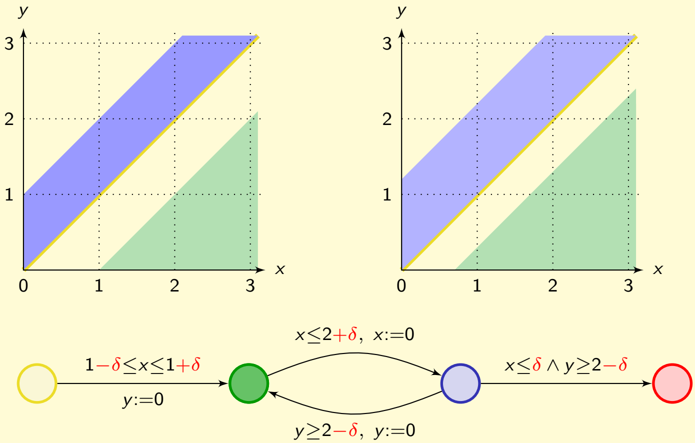
Impact of small timing jitters



[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

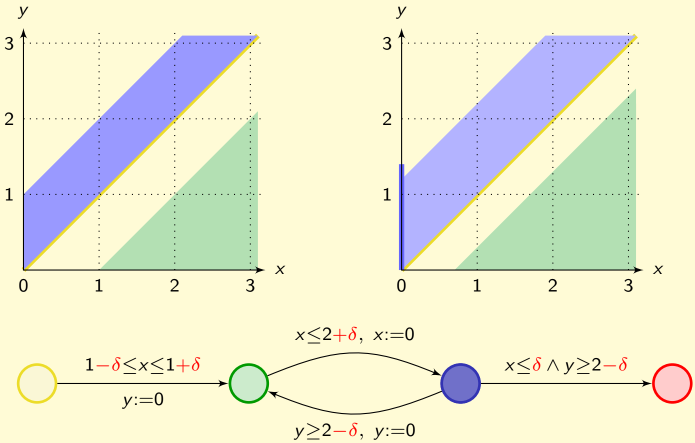
Impact of small timing jitters



[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

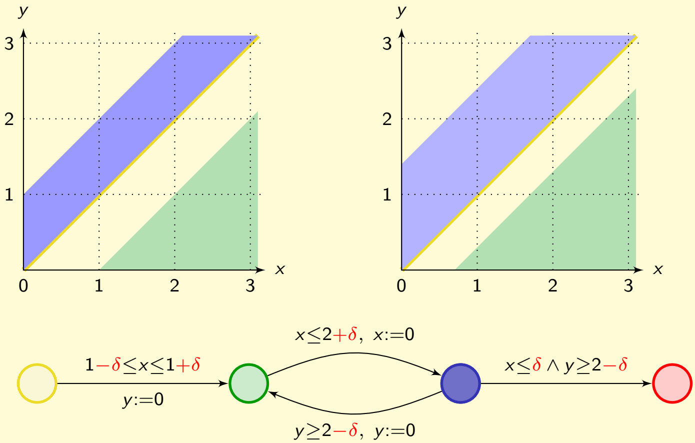
Impact of small timing jitters



[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

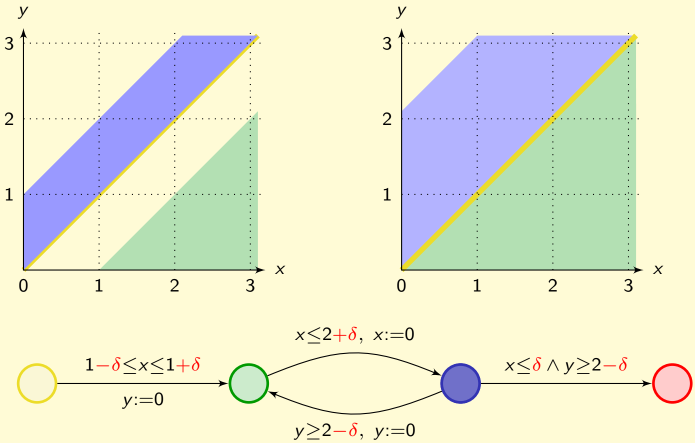
Impact of small timing jitters



[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

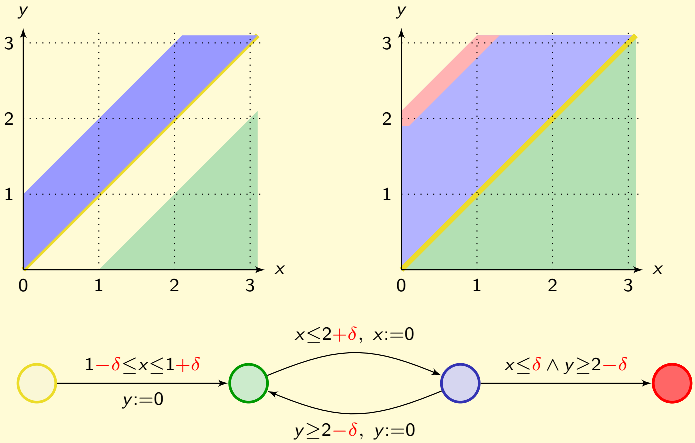
Impact of small timing jitters



[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

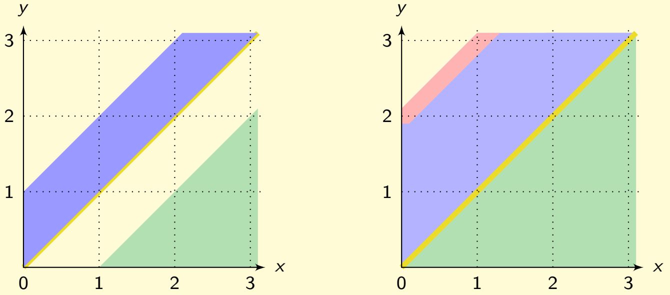
Impact of small timing jitters



[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

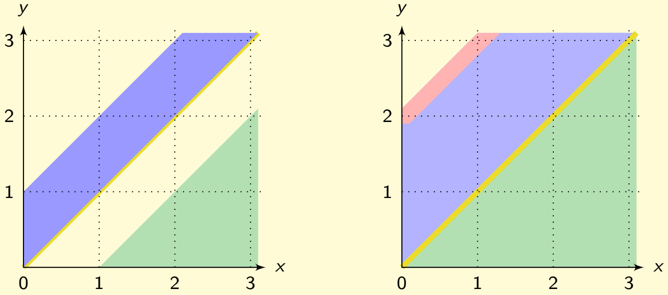
Impact of small timing jitters



[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 2: Does actual analysis transfer to real world?

Impact of small timing jitters



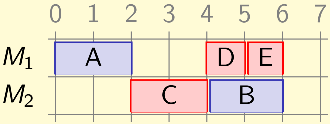
$\rightsquigarrow$ ○ is reachable, however small may be the jitter

[Puri00] Puri. Dynamical properties of timed automata. *Disc. Event Dyn. Syst., 2000.*
[DDMR08] De Wulf, Doyen, Markey, Raskin. Robust safety of timed automata. *FMSD*, 2008.

Example 3: Scheduling and timing anomaly

- Scheduling analysis with timed automata [AAM06]
- **Goal:** analyze a *work-conserving* scheduling policy on given scenarios (no machine is idle if a task is waiting for execution)

Example of a scenario



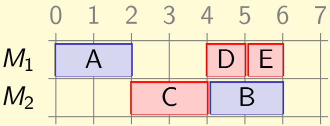
with the dependency constraints: $A \to B$ and $C \to D, E$.

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

[AAM06] Abdeddaim, Asarin, Maler. Scheduling with timed automata. *Theor. Comp. Science*, 2006.

Example 3: Scheduling and timing anomaly

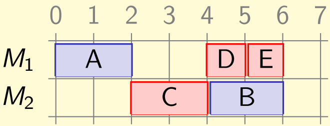
Example of a scenario



$\rightsquigarrow$ Schedulable in 6 time units

Example 3: Scheduling and timing anomaly

Example of a scenario

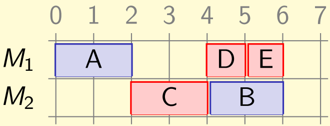


⇝ Schedulable in 6 time units

- Unexpectedly, the duration of A drops to 1.999

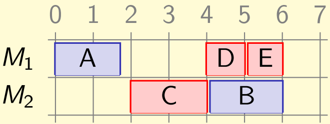
Example 3: Scheduling and timing anomaly

Example of a scenario



$\rightsquigarrow$ Schedulable in 6 time units

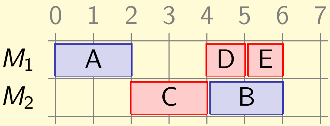
- Unexpectedly, the duration of A drops to 1.999



is not work-conserving

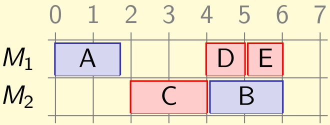
Example 3: Scheduling and timing anomaly

Example of a scenario

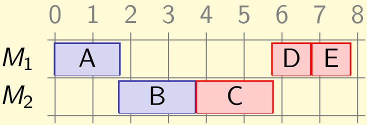


$\leadsto$ Schedulable in 6 time units

- Unexpectedly, the duration of A drops to 1.999



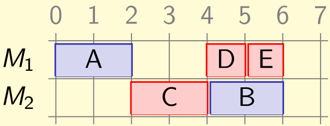
is not work-conserving



is work-conserving
and completes in 7.999 t.u.

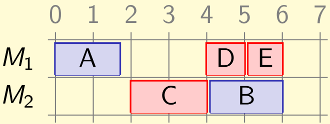
Example 3: Scheduling and timing anomaly

Example of a scenario

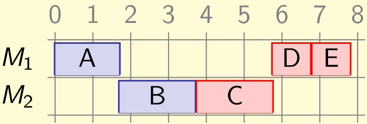


$\rightsquigarrow$ Schedulable in 6 time units

- Unexpectedly, the duration of A drops to 1.999



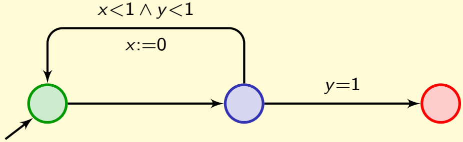
is not work-conserving

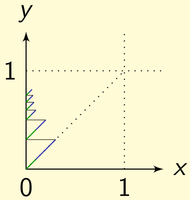


is work-conserving
and completes in 7.999 t.u.

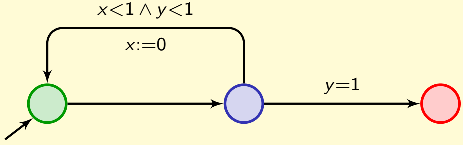
$\rightsquigarrow$ Standard analysis does not capture this **timing anomaly**

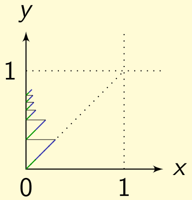
Example 4: Zeno behaviours





Example 4: Zeno behaviours

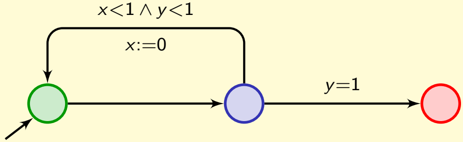


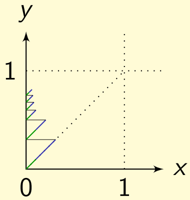


- Those are easy to detect and can be handled; [HS11]

[HS11] Herbreteau, Srivathsan. Coarse abstractions make Zeno behaviours difficult to detect, *Logic. Meth. Comp. Science*, 2011.

Example 4: Zeno behaviours

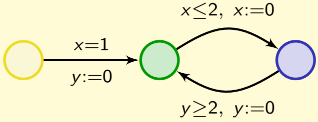




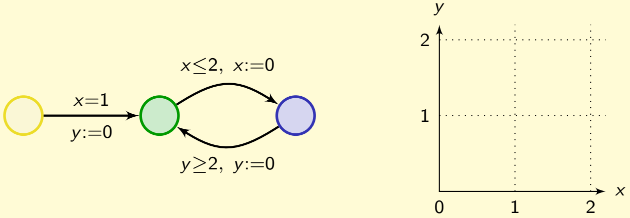
- Those are easy to detect and can be handled; [HS11]
- They are easy to remove by construction.

[HS11] Herbreteau, Srivathsan. Coarse abstractions make Zeno behaviours difficult to detect, *Logic. Meth. Comp. Science*, 2011.

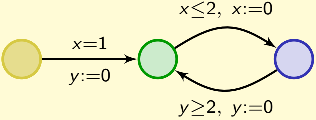
Example 5: More complex convergence phenomena

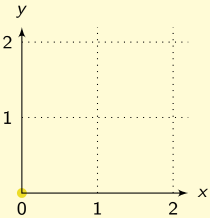


Example 5: More complex convergence phenomena

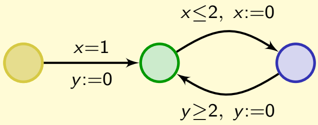


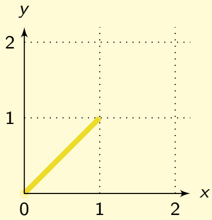
Example 5: More complex convergence phenomena



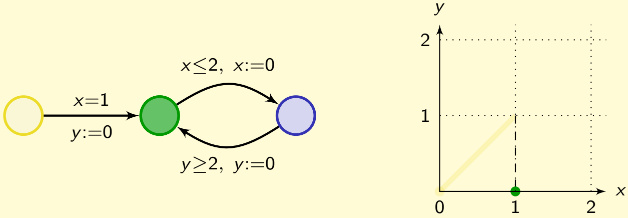


Example 5: More complex convergence phenomena

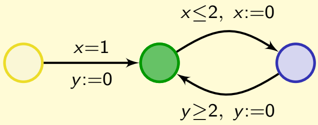


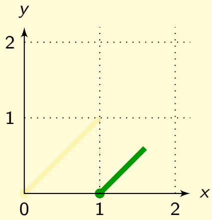


Example 5: More complex convergence phenomena

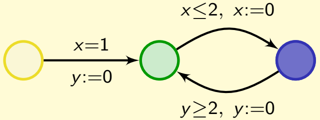


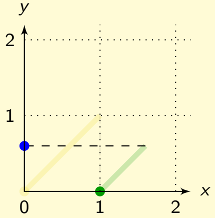
Example 5: More complex convergence phenomena



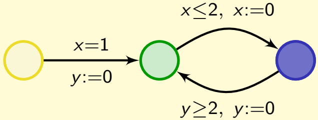


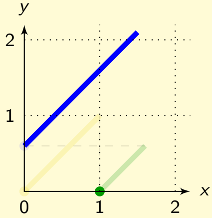
Example 5: More complex convergence phenomena



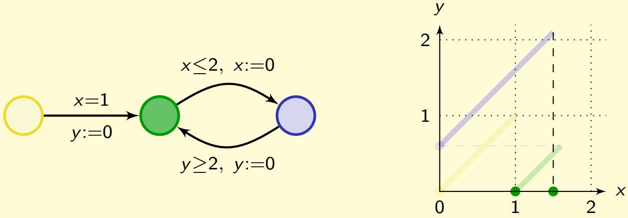


Example 5: More complex convergence phenomena

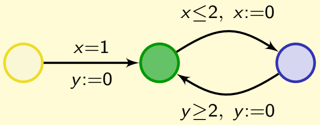


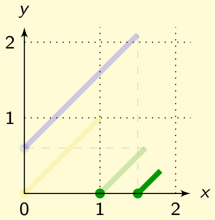


Example 5: More complex convergence phenomena

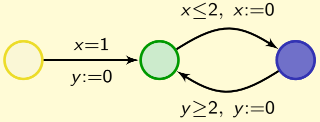


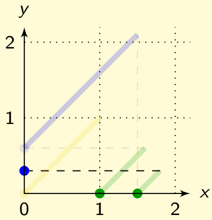
Example 5: More complex convergence phenomena



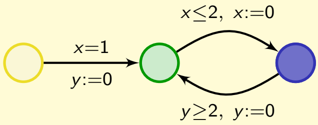


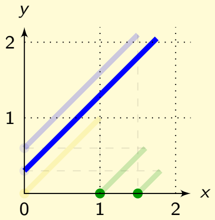
Example 5: More complex convergence phenomena



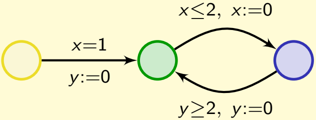


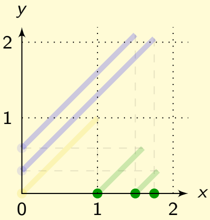
Example 5: More complex convergence phenomena



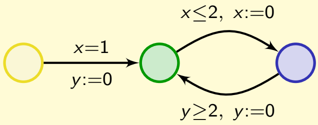


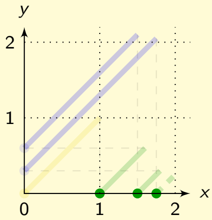
Example 5: More complex convergence phenomena



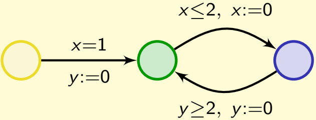


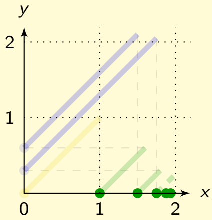
Example 5: More complex convergence phenomena





Example 5: More complex convergence phenomena





$\rightsquigarrow$ Value of clock x when hitting ◯ is converging,
even though global time diverges

An important issue in timed-automata verification

Add robustness to the theory of timed automata!

An important issue in timed-automata verification

Add robustness to the theory of timed automata!

- We need to understand what is the real system behind the mathematical model, and also which implementation we have in mind, if any.

An important issue in timed-automata verification

Add robustness to the theory of timed automata!

- We need to understand what is the real system behind the mathematical model, and also which implementation we have in mind, if any.
- **Aim:** provide frameworks to build robustly correct systems

An important issue in timed-automata verification

Add robustness to the theory of timed automata!

- We need to understand what is the real system behind the mathematical model, and also which implementation we have in mind, if any.
- **Aim:** provide frameworks to build robustly correct systems

We focus on perturbations on time measurements and jitter.

Some hints into the robustness of timed automata

Robust model-checking: the worst-case approach

- Compute an overapproximation of the set of perturbed behaviours (Possibly relate with an implementation)
- Prove correctness of this approximation

Some hints into the robustness of timed automata

Robust model-checking: the worst-case approach

- Compute an overapproximation of the set of perturbed behaviours (Possibly relate with an implementation)
- Prove correctness of this approximation

$\rightsquigarrow$ Many decidability results!

Often: same complexity as standard verification

Some hints into the robustness of timed automata

Robust model-checking: the worst-case approach

- Compute an overapproximation of the set of perturbed behaviours (Possibly relate with an implementation)
- Prove correctness of this approximation

$\rightsquigarrow$ Many decidability results!
Often: same complexity as standard verification

Robust control

- The strategy (partially) dictates the behaviour of the system
- It should tolerate imprecisions in timing measurements
- Simpler case: robust realisability

Some hints into the robustness of timed automata

Robust model-checking: the worst-case approach

- Compute an overapproximation of the set of perturbed behaviours (Possibly relate with an implementation)
- Prove correctness of this approximation

$\rightsquigarrow$ Many decidability results!
Often: same complexity as standard verification

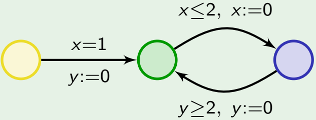
Robust control

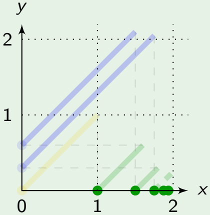
- The strategy (partially) dictates the behaviour of the system
- It should tolerate imprecisions in timing measurements
- Simpler case: robust realisability

We propose a game-based approach to robust realisability

Realisability

Example (Can we ensure an infinite behaviour?)

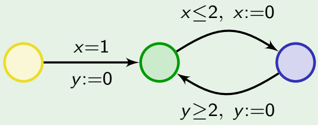


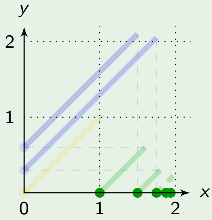


Strategy: in location with value x, delay $\frac{2-x}{2}$

Realisability

Example (Can we ensure an infinite behaviour?)



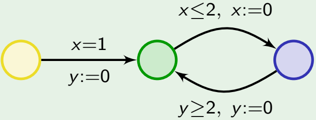


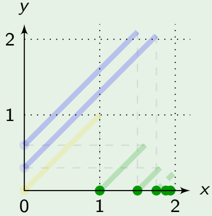
Strategy: in location ◯ with value x, delay $\frac{2-x}{2}$

- This strategy requires infinite precision

Robust realisability

Example (Can we ensure an infinite behaviour?)



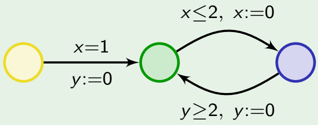


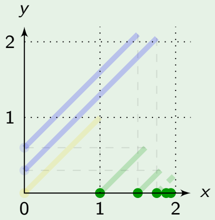
Strategy: in location with value x, delay $\frac{2-x}{2}$

- This strategy requires infinite precision
- In practice, when x is close to 2, no additional delay is supported: the run is theoretically infinite, but it is actually blocking

Robust realisability

Example (Can we ensure an infinite behaviour?)





Strategy: in location ◯ with value x, delay $\frac{2-x}{2}$

- This strategy requires infinite precision
- In practice, when x is close to 2, no additional delay is supported: the run is theoretically infinite, but it is actually blocking
- And that is unavoidable

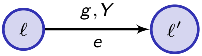
Robust realisability

Idea of robust realisability

- $\rightsquigarrow$ Synthesize strategies that realise some property while tolerating small timing perturbations
- $\rightsquigarrow$ Consequence: remove convergence phenomena

Robust realisability via a game semantics

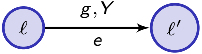
Timed automaton $\mathcal{A}$



Robust realisability via a game semantics

Timed automaton $\mathcal{A}$

Game $\mathcal{G}_\delta(\mathcal{A})$



$\rightsquigarrow$

Game semantics $\mathcal{G}_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

... between Controller and Perturbator:

Robust realisability via a game semantics

Timed automaton $\mathcal{A}$

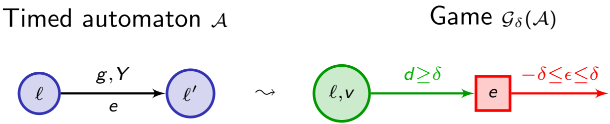
Game $\mathcal{G}_\delta(\mathcal{A})$

Game semantics $\mathcal{G}_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

... between Controller and Perturbator:

- from (ℓ, v), Controller suggests a delay $d \geq \delta$ and a next edge $e = (\ell \xrightarrow{g,Y} \ell')$ that is available after delay d

Robust realisability via a game semantics



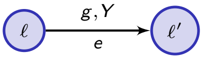
Game semantics $\mathcal{G}_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

... between Controller and Perturbator:

- from (ℓ, v), Controller suggests a delay $d \geq \delta$ and a next edge $e = (\ell \xrightarrow{g,Y} \ell')$ that is available after delay d
- Perturbator then chooses a perturbation $\epsilon \in [-\delta; +\delta]$

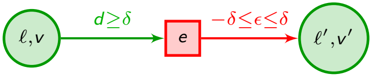
Robust realisability via a game semantics

Timed automaton $\mathcal{A}$



$\rightsquigarrow$

Game $\mathcal{G}_\delta(\mathcal{A})$



with $v'(v+d+\epsilon)[Y \leftarrow 0]$

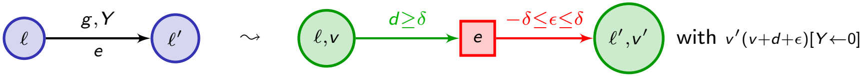
Game semantics $\mathcal{G}_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

... between Controller and Perturbator:

- from (ℓ, v), Controller suggests a delay $d \geq \delta$ and a next edge $e = (\ell \xrightarrow{g,Y} \ell')$ that is available after delay d
- Perturbator then chooses a perturbation $\epsilon \in [-\delta; +\delta]$
- Next state is $(\ell', (v + d + \epsilon)[Y \leftarrow 0])$

Robust realisability via a game semantics

Timed automaton $\mathcal{A}$ Game $\mathcal{G}_\delta(\mathcal{A})$



Game semantics $\mathcal{G}_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

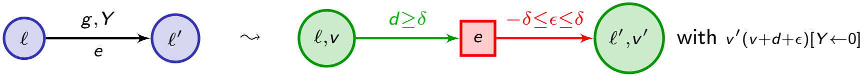
... between Controller and Perturbator:

- from (ℓ, v), Controller suggests a delay $d \geq \delta$ and a next edge $e = (\ell \xrightarrow{g,Y} \ell')$ that is available after delay d
- Perturbator then chooses a perturbation $\epsilon \in [-\delta; +\delta]$
- Next state is $(\ell', (v + d + \epsilon)[Y \leftarrow 0])$

Note: when $\delta = 0$, this is the standard semantics of timed automata.

Robust realisability via a game semantics

Timed automaton $\mathcal{A}$ — Game $\mathcal{G}_\delta(\mathcal{A})$



Game semantics $\mathcal{G}_\delta(\mathcal{A})$ of timed automaton $\mathcal{A}$...

... between Controller and Perturbator:

- from (ℓ, v), Controller suggests a delay $d \geq \delta$ and a next edge $e = (\ell \xrightarrow{g, Y} \ell')$ that is available after delay d
- Perturbator then chooses a perturbation $\epsilon \in [-\delta; +\delta]$
- Next state is $(\ell', (v + d + \epsilon)[Y \leftarrow 0])$

Note: when $\delta = 0$, this is the standard semantics of timed automata.

A δ-robust strategy for Controller is then a strategy that satisfies the expected property, whatever plays Perturbator.

The excess game semantics

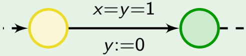
Constraints may not be satisfied after the perturbation:
that is, only $v + d$ should satisfy g

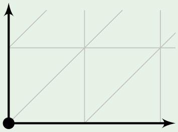
[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP'12*.

The excess game semantics

Constraints may not be satisfied after the perturbation: that is, only $v + d$ should satisfy g

Example



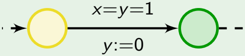


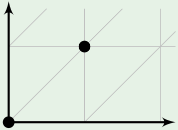
[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP'12*.

The excess game semantics

Constraints may not be satisfied after the perturbation: that is, only $v + d$ should satisfy g

Example



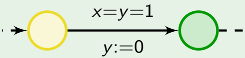


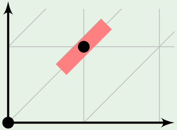
[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP'12*.

The excess game semantics

Constraints may not be satisfied after the perturbation: that is, only $v + d$ should satisfy g

Example



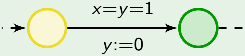


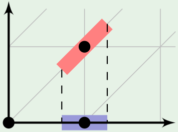
[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP'12.*

The excess game semantics

Constraints may not be satisfied after the perturbation: that is, only $v + d$ should satisfy g

Example



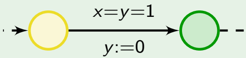


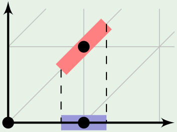
[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP'12*.

The excess game semantics

Constraints may not be satisfied after the perturbation: that is, only $v + d$ should satisfy g

Example





$\rightsquigarrow$ Allows simple design of constraints, ensures divergence of time, avoids convergence phenomena

[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP'12*.

The excess game semantics – Algorithmics

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a δ-robust strategy that achieves a given goal.

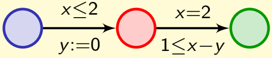
The excess game semantics – Algorithmics

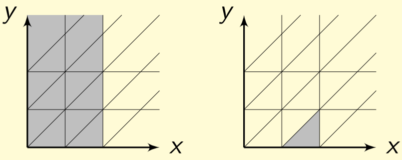
The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a δ-robust strategy that achieves a given goal.

Two challenges

1. Accumulation of perturbations:





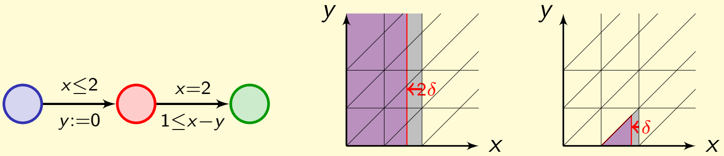
The excess game semantics – Algorithmics

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a δ-robust strategy that achieves a given goal.

Two challenges

1. Accumulation of perturbations:



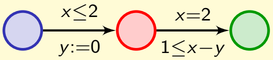
The excess game semantics – Algorithmics

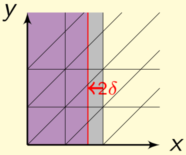
The (parameterized) synthesis problem

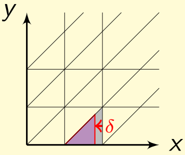
Synthesize $\delta > 0$ and a δ-robust strategy that achieves a given goal.

Two challenges

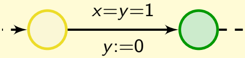
1. Accumulation of perturbations:

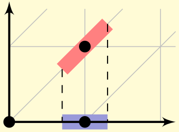






2. New regions become reachable





The excess game semantics – Algorithmics

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a δ-robust strategy that achieves a given goal.

Theorem

The parameterized synthesis problem for reachability properties is decidable and EXPTIME-complete. Furthermore, uniform winning strategies (w.r.t. δ) can be computed.

- Technical tools: a region-based refined game abstraction, shrunk DBMs
- ☺ Extends to two-player games (i.e. to real control problems)

[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP'12.*

The excess game semantics – Algorithmics

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a δ-robust strategy that achieves a given goal.

Theorem

The parameterized synthesis problem for reachability properties is decidable and EXPTIME-complete. Furthermore, uniform winning strategies (w.r.t. δ) can be computed.

- Technical tools: a region-based refined game abstraction, shrunk DBMs
- ☺ Extends to two-player games (i.e. to real control problems)
- ☹ Only valid for reachability properties

[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP'12.*

The conservative game semantics

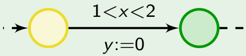
Constraints have to be satisfied after the perturbation: that is,
$v + d + \epsilon$ **should satisfy** g **for every** $\epsilon \in [-\delta; +\delta]$

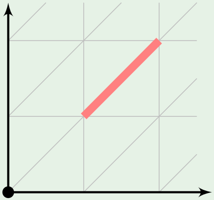
[SBMR13] Sankur, Bouyer, Markey, Reynier. Robust Controller Synthesis in Timed Automata. *CONCUR'13*.

The conservative game semantics

Constraints have to be satisfied after the perturbation: that is, $v + d + \epsilon$ should satisfy g for every $\epsilon \in [-\delta; +\delta]$

Example



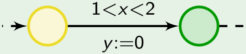


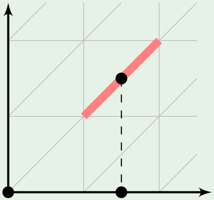
[SBMR13] Sankur, Bouyer, Markey, Reynier. Robust Controller Synthesis in Timed Automata. *CONCUR'13*.

The conservative game semantics

Constraints have to be satisfied after the perturbation: that is, $v + d + \epsilon$ should satisfy g for every $\epsilon \in [-\delta; +\delta]$

Example



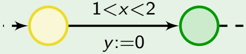


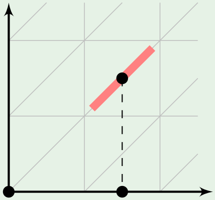
[SBMR13] Sankur, Bouyer, Markey, Reynier. Robust Controller Synthesis in Timed Automata. *CONCUR'13*.

The conservative game semantics

Constraints have to be satisfied after the perturbation: that is, $v + d + \epsilon$ should satisfy g for every $\epsilon \in [-\delta; +\delta]$

Example



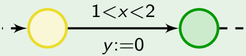


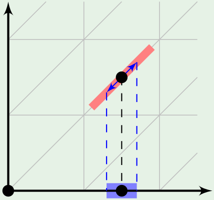
[SBMR13] Sankur, Bouyer, Markey, Reynier. Robust Controller Synthesis in Timed Automata. *CONCUR'13.*

The conservative game semantics

Constraints have to be satisfied after the perturbation: that is, $v + d + \epsilon$ should satisfy g for every $\epsilon \in [-\delta; +\delta]$

Example



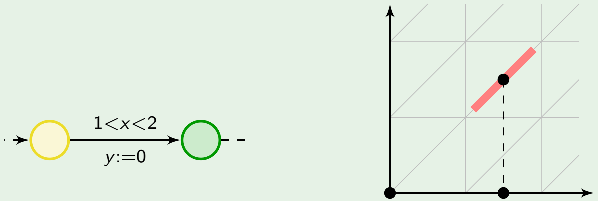


[SBMR13] Sankur, Bouyer, Markey, Reynier. Robust Controller Synthesis in Timed Automata. *CONCUR'13*.

The conservative game semantics

Constraints have to be satisfied after the perturbation: that is, $v + d + \epsilon$ should satisfy g for every $\epsilon \in [-\delta; +\delta]$

Example



⤳ Strongly ensures timing constraints, ensures divergence of time, prevents converging phenomena

[SBMR13] Sankur, Bouyer, Markey, Reynier. Robust Controller Synthesis in Timed Automata. *CONCUR'13*.

The conservative game semantics – Results

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a δ-robust strategy that achieves a given goal.

The conservative game semantics – Results

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a δ-robust strategy that achieves a given goal.

Theorem

The synthesis problem for Büchi properties is decidable and PSPACE-complete. Furthermore, winning strategies with uniform descriptions (w.r.t. δ) can be computed.

The conservative game semantics – Results

The (parameterized) synthesis problem

Synthesize $\delta > 0$ and a δ-robust strategy that achieves a given goal.

Theorem

The synthesis problem for Büchi properties is decidable and PSPACE-complete. Furthermore, winning strategies with uniform descriptions (w.r.t. δ) can be computed.

- ☺ Valid for all ω-regular properties
- ☺ Same complexity as standard verification!
- ☺ Extends partially to timed games [ORS14]

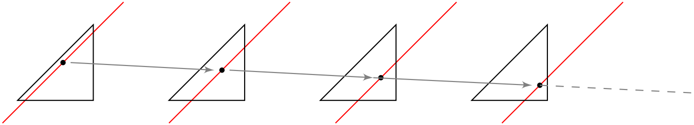
[ORS14] Oualhadj, Reynier, Sankur. Robust strategies in timed games.

The problem consists in finding cycles that do not become blocked.

[AB11] Asarin, Basset. Thin and Thick Timed Regular Languages. *FORMATS'11*.

The problem consists in finding cycles that do not become blocked.

- Some cycles are converging (non-forgetful)

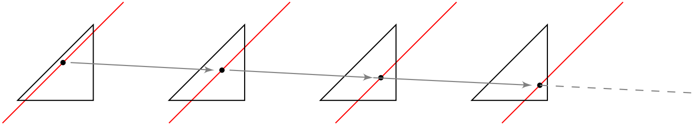


"There is a constraining half-space"

[AB11] Asarin, Basset. Thin and Thick Timed Regular Languages. *FORMATS'11*.

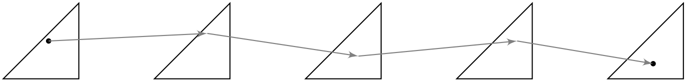
The problem consists in finding cycles that do not become blocked.

- Some cycles are converging (non-forgetful)



"There is a constraining half-space"

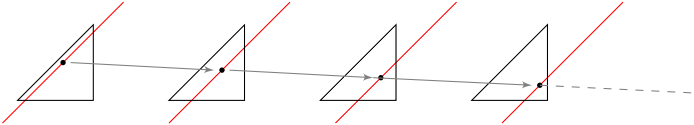
- The other cycles are non-converging (forgetful)



[AB11] Asarin, Basset. Thin and Thick Timed Regular Languages. *FORMATS'11*.

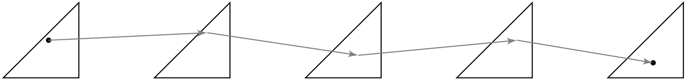
The problem consists in finding cycles that do not become blocked.

- Some cycles are converging (non-forgetful)



"There is a constraining half-space"

- The other cycles are non-converging (forgetful)



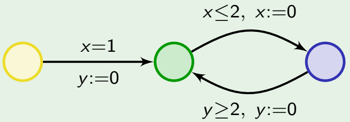
Characterization

There is $\delta > 0$ such that Controller has a δ-robust winning strategy in $\mathcal{G}_\delta(\mathcal{A})$ iff $R(\mathcal{A})$ has a reachable aperiodic[a] non-punctual winning lasso.

[a] That is, all its iterations are forgetful.

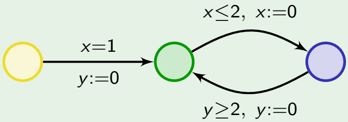
Non-forgetful cycle ⇒ deadlock

Example

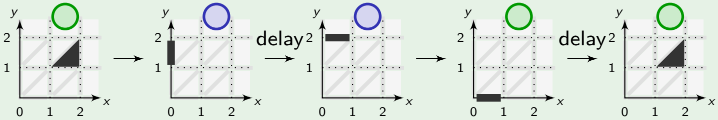


Non-forgetful cycle ⇒ deadlock

Example

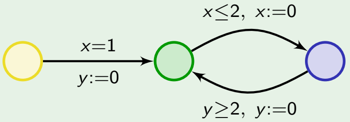


A region cycle:

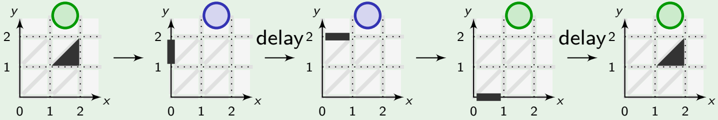


Non-forgetful cycle ⇒ deadlock

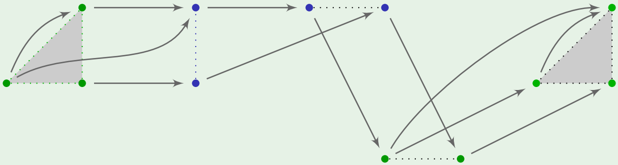
Example



A region cycle:

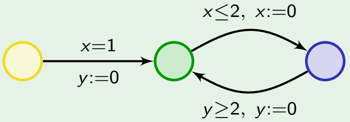


The corresponding orbit graph:

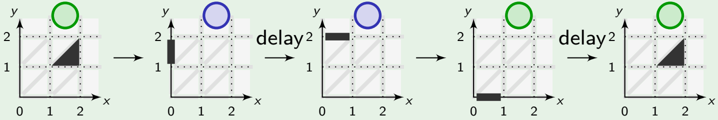


Non-forgetful cycle ⇒ deadlock

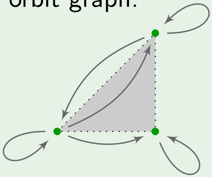
Example



A region cycle:

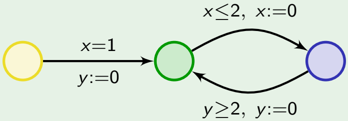


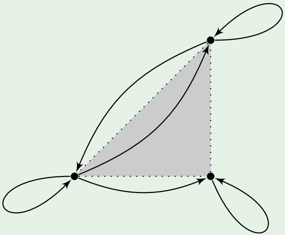
The corresponding (folded) orbit graph:



Non-forgetful cycle ⇒ deadlock

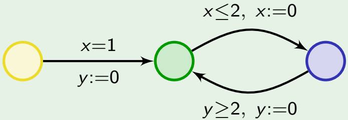
Example

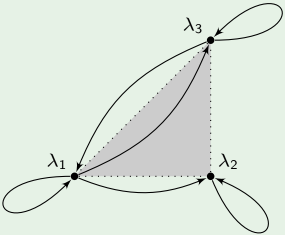




Non-forgetful cycle ⇒ deadlock

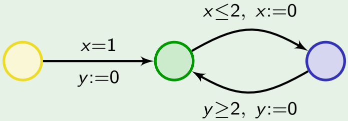
Example

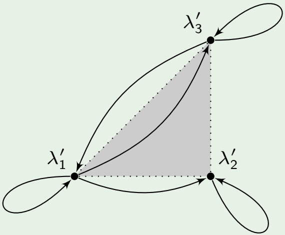




Non-forgetful cycle $\Rightarrow$ deadlock

Example

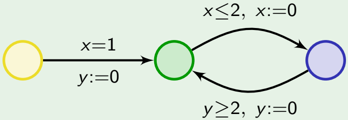




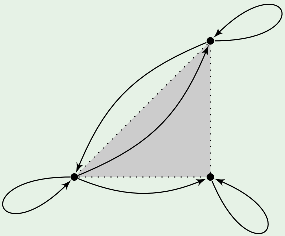
with
$$\begin{cases} \lambda'_1 = p_1\lambda_1 + p_3\lambda_3 \\ \lambda'_2 = (1 - p_1 - p'_1)\lambda_1 + \lambda_2 \\ \lambda'_3 = p'_1\lambda_1 + (1 - p_3)\lambda_3 \end{cases}$$

Non-forgetful cycle $\Rightarrow$ deadlock

Example

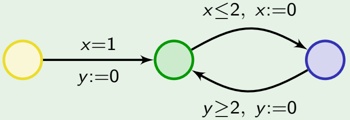


The cycle is not forgetful (that is, not strongly connected)

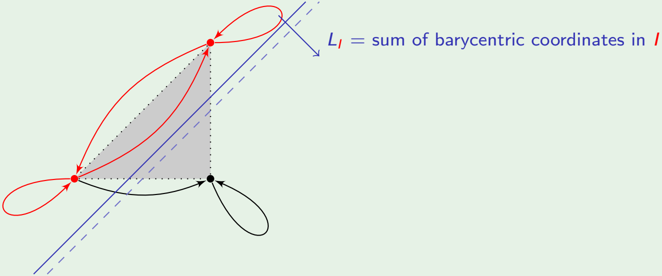


Non-forgetful cycle ⇒ deadlock

Example

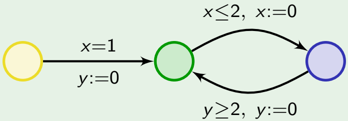


The cycle is not forgetful (that is, not strongly connected): there is an initial component I, and a non-decreasing function L_I.

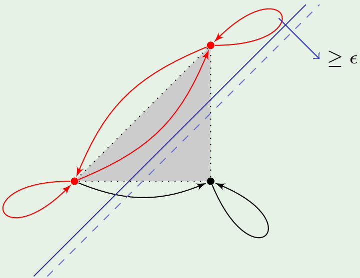


Non-forgetful cycle ⇒ deadlock

Example



Perturbator can enforce rapid decrease of L_l!

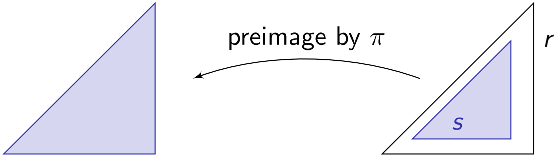


Forgetful cycle $\Rightarrow$ robust strategy

- Idea: target the middle of the regions

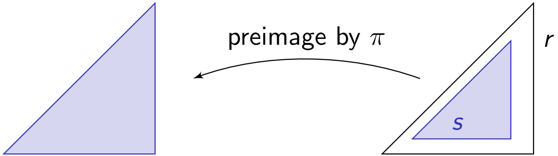
Forgetful cycle $\Rightarrow$ robust strategy

- Idea: target the middle of the regions
  Let π be a forgetful (in fact complete) cycle.



Forgetful cycle $\Rightarrow$ robust strategy

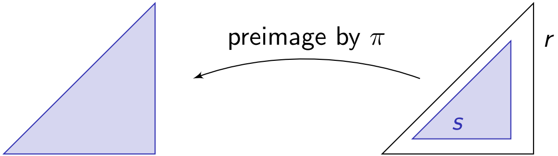
- Idea: target the middle of the regions
  Let π be a forgetful (in fact complete) cycle.



  - Fact (technical): preimage of s by π under δ-perturbations is $r - \delta Q$ (Q fixed) for small δ's

Forgetful cycle $\Rightarrow$ robust strategy

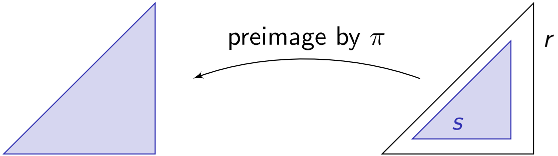
- Idea: target the middle of the regions
  Let π be a forgetful (in fact complete) cycle.



  - Fact (technical): preimage of s by π under δ-perturbations is $r - \delta Q$ (Q fixed) for small δ's
  - Property of s: $s \subseteq r - \delta Q$ for small δ's

Forgetful cycle ⇒ robust strategy

- Idea: target the middle of the regions
  Let π be a forgetful (in fact complete) cycle.

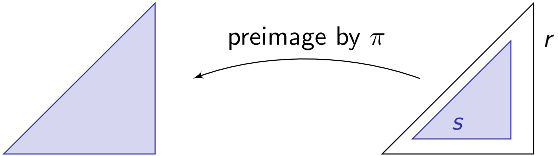


  - Fact (technical): preimage of s by π under δ-perturbations is $r - \delta Q$ (Q fixed) for small δ's
  - Property of s: $s \subseteq r - \delta Q$ for small δ's

$\Rightarrow$ Robust strategy: enforce s at each cycle

Forgetful cycle $\Rightarrow$ robust strategy

- Idea: target the middle of the regions
  Let π be a forgetful (in fact complete) cycle.



  - Fact (technical): preimage of s by π under δ-perturbations is $r - \delta Q$ (Q fixed) for small δ's
  - Property of s: $s \subseteq r - \delta Q$ for small δ's

  $\Rightarrow$ Robust strategy: enforce s at each cycle
- Technical tool: shrunk DBMs [BMS12]

[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP'12*.

Conclusion

- Timed automata: a nice mathematical model for real-time systems with interesting decidability properties and algorithmics solutions.
- Not always easy to transfer correctness proven in this model to real behaviours of the system.
- Robustness: an important issue!

Conclusion

- Timed automata: a nice mathematical model for real-time systems with interesting decidability properties and algorithmics solutions.
- Not always easy to transfer correctness proven in this model to real behaviours of the system.
- Robustness: an important issue!

- Algorithms for robust realisability
- Extension to richer models seems unfortunately hard [BMS13]
- A quantitative approach to robustness: what if Perturbator plays randomly?

Conclusion

- Timed automata: a nice mathematical model for real-time systems with interesting decidability properties and algorithmics solutions.
- Not always easy to transfer correctness proven in this model to real behaviours of the system.
- Robustness: an important issue!

- Algorithms for robust realisability
- Extension to richer models seems unfortunately hard [BMS13]
- A quantitative approach to robustness: what if Perturbator plays randomly?

Some references:

- PhD thesis of Ocan Sankur: "Robustness in Timed Automata: Analysis, Synthesis, Implementation" (2013)
- Survey at RP'13: "Robustness in Timed Automata" (Bouyer, Markey, Sankur)
- Survey at SiES'11: "Robustness in Real-time Systems" (Markey)