Robustness in Timed Automata

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

ImpRo EQualIS

Outline

1. Introduction

2. Robust model-checking and implementation
   - Parameterized enlarged semantics
   - Automatic generation of an implementation
   - Implementation by shrinking

3. Robust realisability
   - Excess semantics
   - Conservative semantics

4. Conclusion

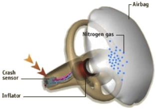
Time-dependent systems

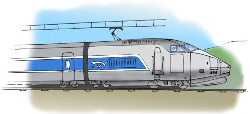
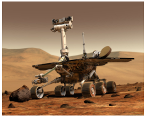
- We are interested in timed systems

Time-dependent systems

- We are interested in timed systems

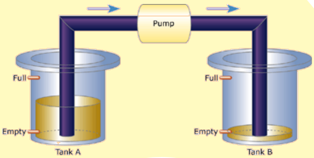




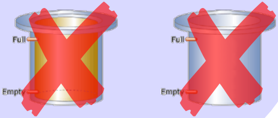
Model-checking and control

system:



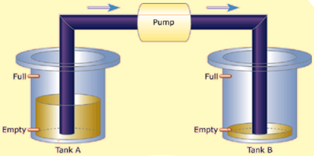
[http://www.embedded.com]

property:



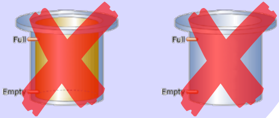
Model-checking and control

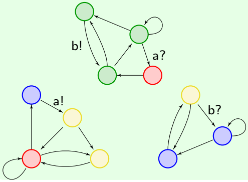
system:



[http://www.embedded.com]

property:

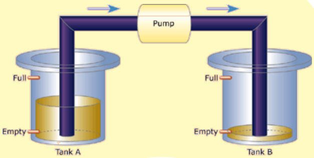




$\mathbf{AG}(\neg \texttt{B.overfull} \wedge \neg \texttt{B.dried\_up})$

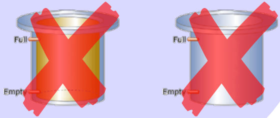
Model-checking and control

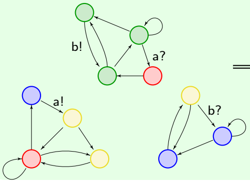
system:



[http://www.embedded.com]

property:



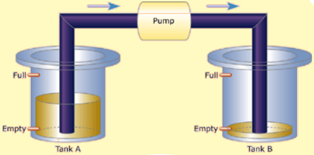


algorithm

AG($\neg$`B.overfull` $\wedge$ $\neg$`B.dried_up`)

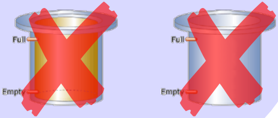
Model-checking and control

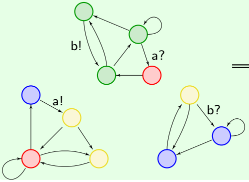
system:



[http://www.embedded.com]

property:





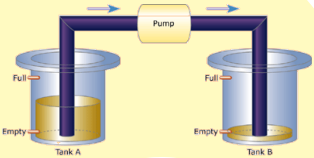
model-checking algorithm

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

yes/no

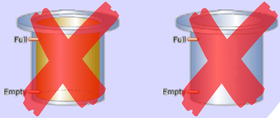
Model-checking and control

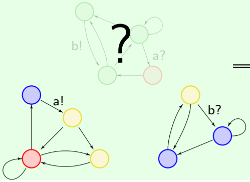
system:



[http://www.embedded.com]

property:

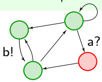




control/synthesis algorithm

AG($\neg$`B.overfull`

$\wedge$ $\neg$`B.dried_up`)



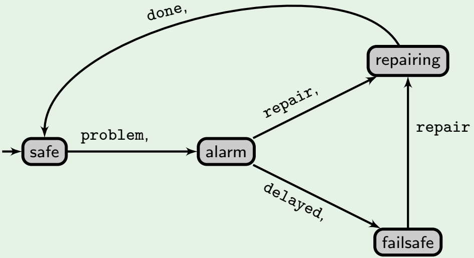
Reasoning about real-time systems

The model of timed automata [AD94]

A timed automaton is made of

- a finite automaton-based structure

Example



[AD9

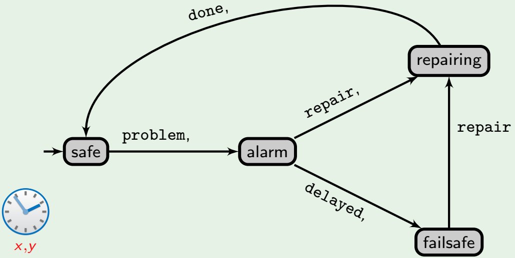
Reasoning about real-time systems

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.

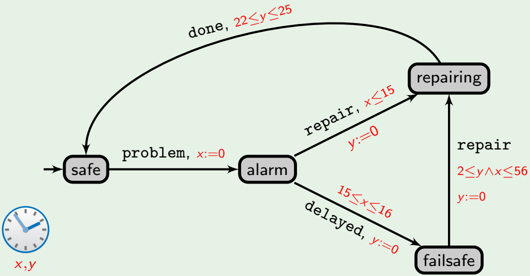
Reasoning about real-time systems

The model of timed automata [AD94]

A timed automaton is made of

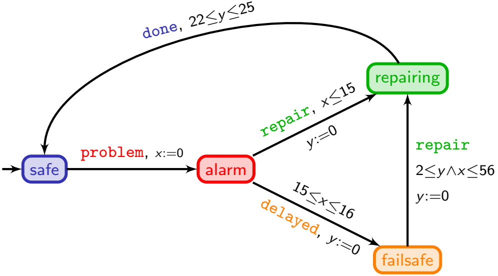
- a finite automaton-based structure
- a set of clocks
- timing constraints on transitions

Example

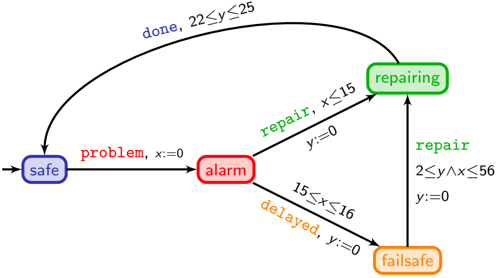


[AD94] Alur, Dill: A Theory of Timed Automata. Theor. Comp. Science, 1994.

Example – cont'd

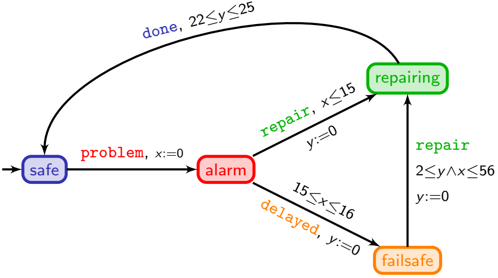


Example – cont'd



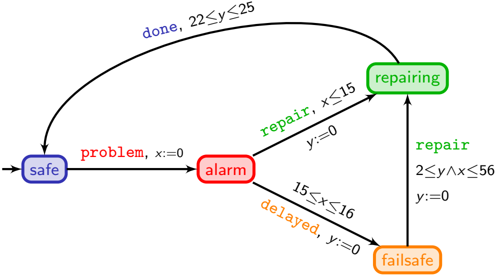
	safe
x	0
y	0

Example – cont'd



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

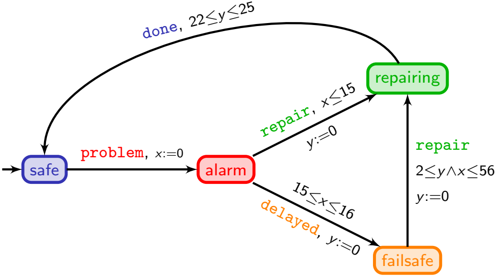
Example – cont'd



$$\text{safe} \xrightarrow{23} \text{safe} \xrightarrow{\texttt{problem}} \text{alarm}$$

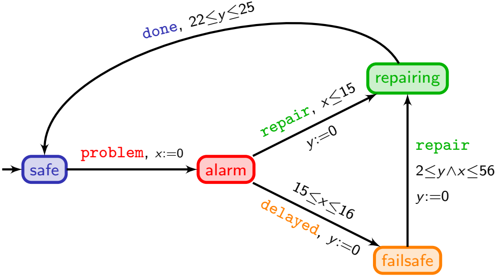
x	0	23	0
y	0	23	23

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

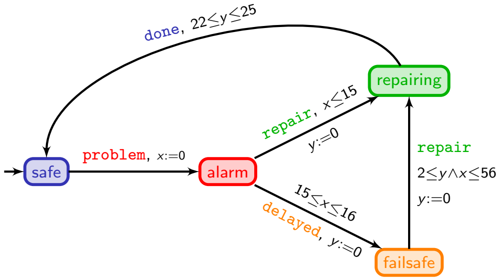
Example – cont'd



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

	failsafe
···	15.6
	0

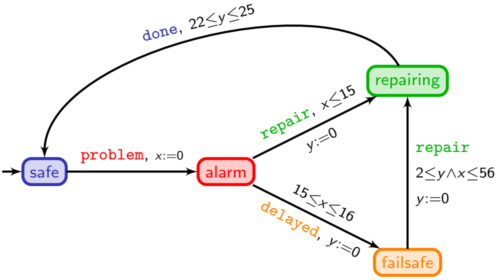
Example – cont'd



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

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

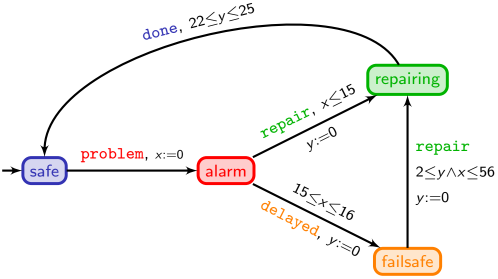
Example – cont'd



	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

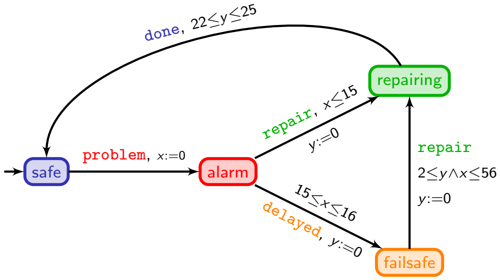
Example – cont'd



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

	failsafe	$\xrightarrow{2.3}$	failsafe	$\xrightarrow{\texttt{repair}}$	repairing	$\xrightarrow{22.1}$	repairing
⋯	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

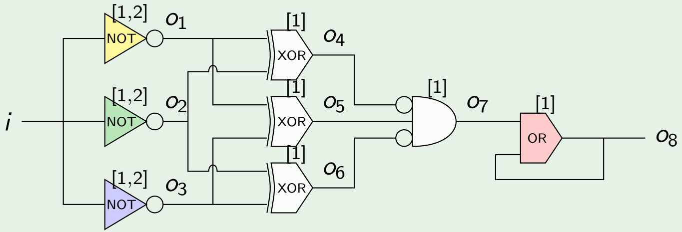
...because computers are digital!

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

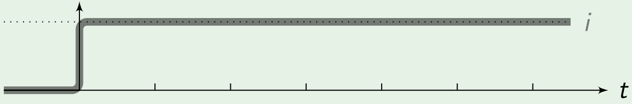
Discrete-time semantics

...because computers are digital!

Example [Alur91]



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

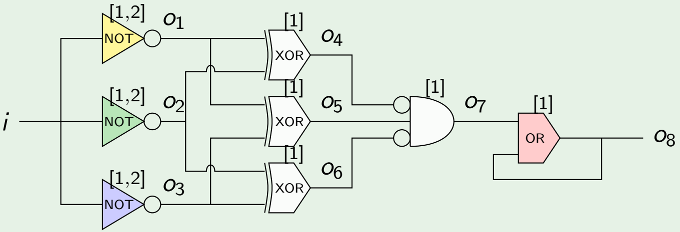


[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

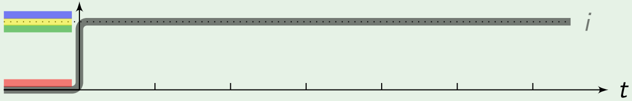
Discrete-time semantics

...because computers are digital!

Example [Alur91]



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

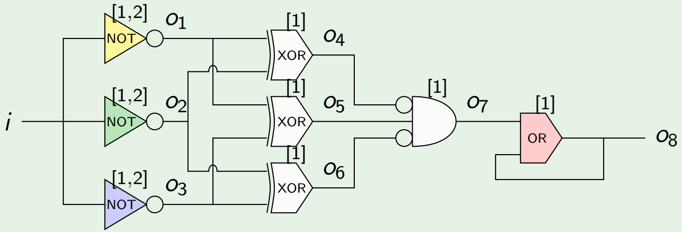


[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

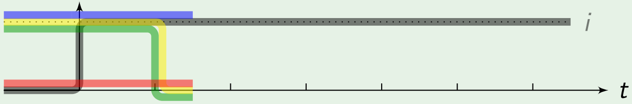
Discrete-time semantics

...because computers are digital!

Example [Alur91]



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

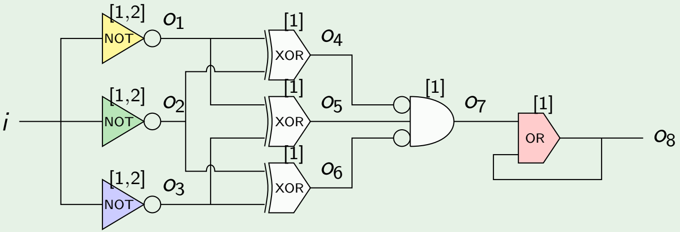


[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

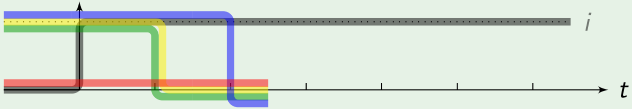
Discrete-time semantics

...because computers are digital!

Example [Alur91]



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

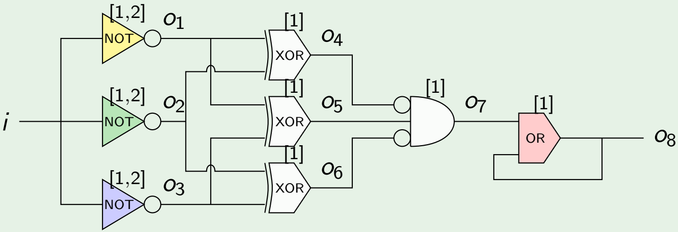


[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

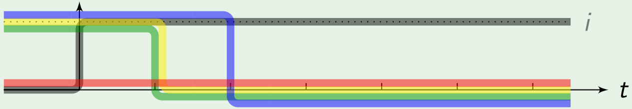
Discrete-time semantics

…because computers are digital!

Example [Alur91]



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

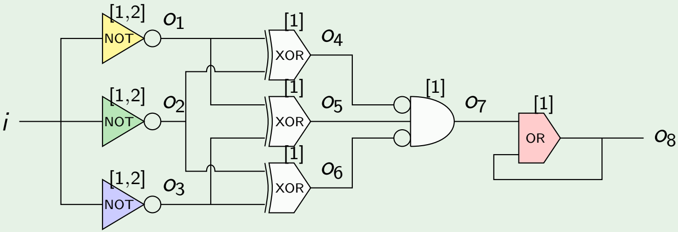


[Alur91] Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

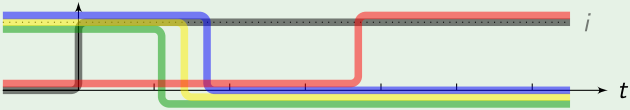
Discrete-time semantics

...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.

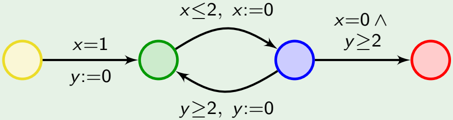
Continuous-time semantics

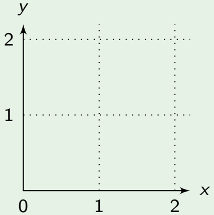
...real-time models for real-time systems!

Continuous-time semantics

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

Example

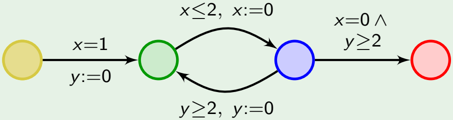


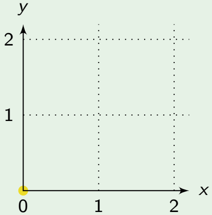


Continuous-time semantics

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

Example

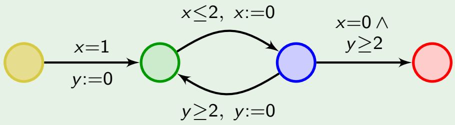


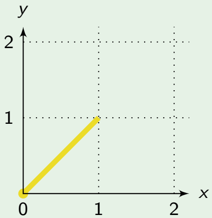


Continuous-time semantics

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

Example

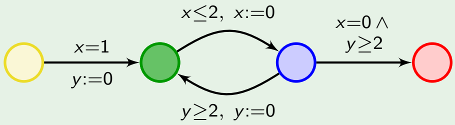


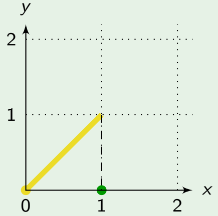


Continuous-time semantics

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

Example

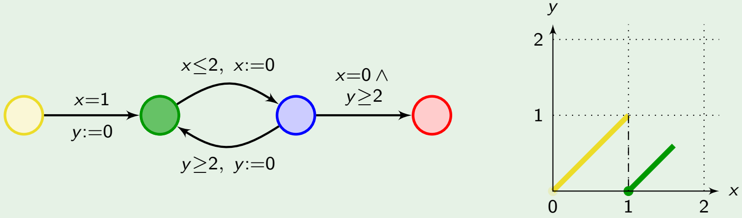




Continuous-time semantics

…real-time models for real-time systems!

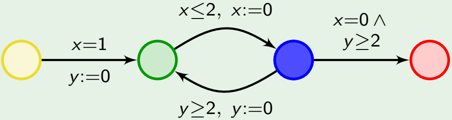
Example

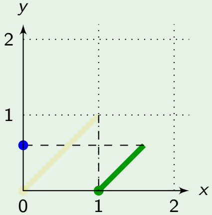


Continuous-time semantics

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

Example

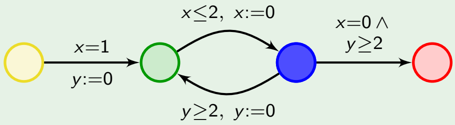


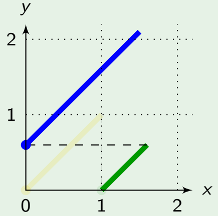


Continuous-time semantics

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

Example

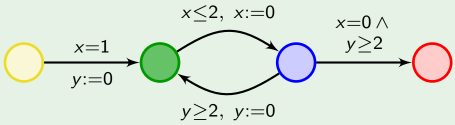


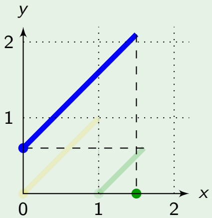


Continuous-time semantics

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

Example

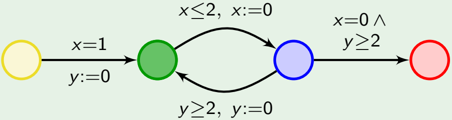


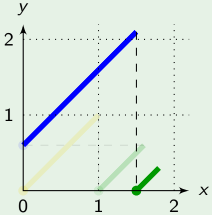


Continuous-time semantics

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

Example

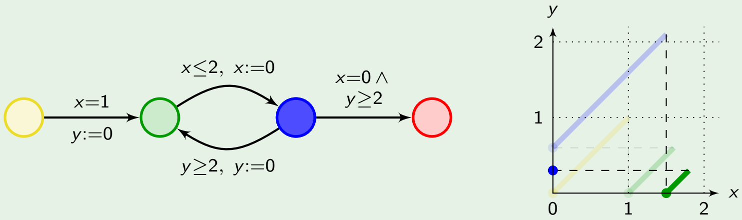




Continuous-time semantics

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

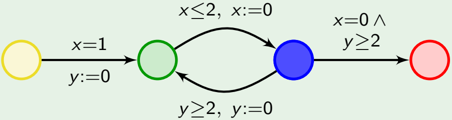
Example

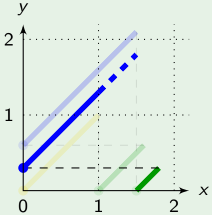


Continuous-time semantics

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

Example

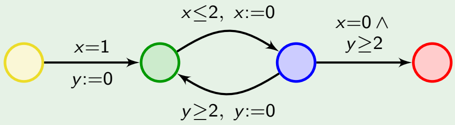


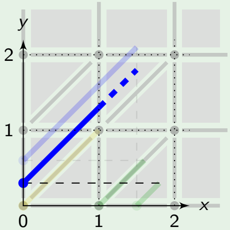


Continuous-time semantics

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

Example

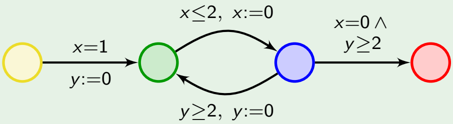


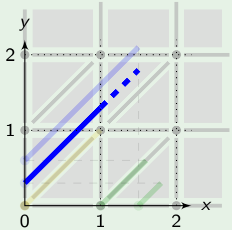


Continuous-time semantics

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

Example





Theorem [AD94]

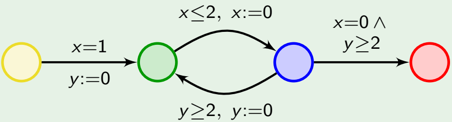
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

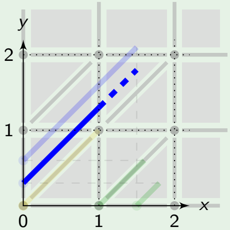
- Technical tool: region abstraction

Continuous-time semantics

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

Example



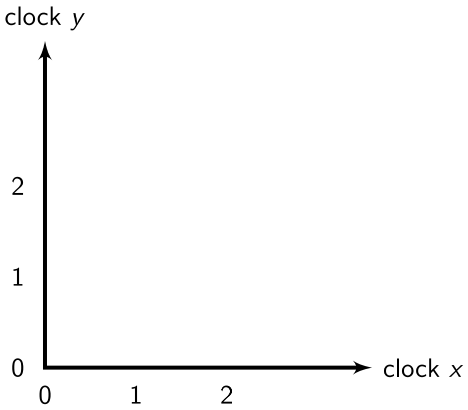


Theorem [AD94]

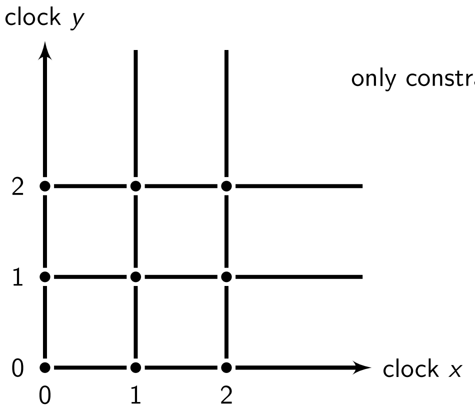
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

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

Technical tool: Region abstraction



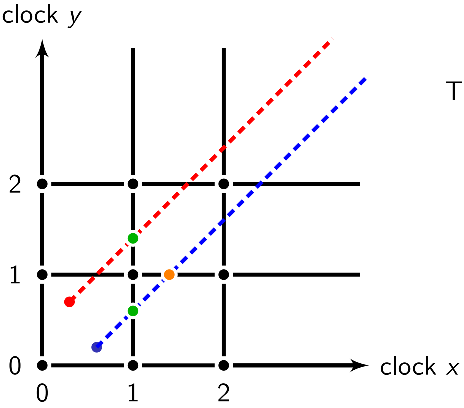
Technical tool: Region abstraction

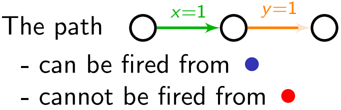


only constraints: $x \sim c$ with $c \in \{0, 1, 2\}$
$y \sim c$ with $c \in \{0, 1, 2\}$

- "compatibility" between regions and constraints

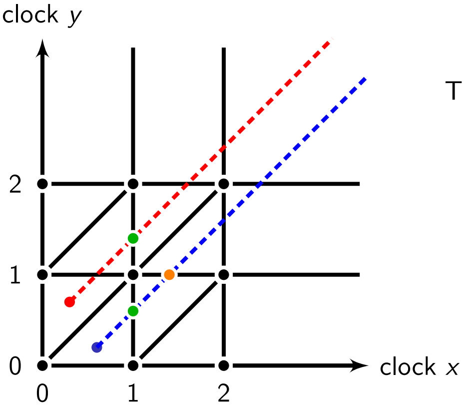
Technical tool: Region abstraction





- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

Technical tool: Region abstraction

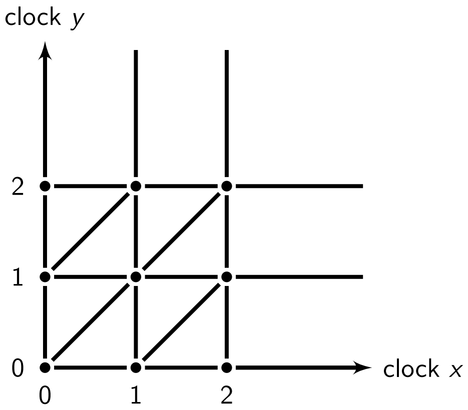


The path 

- can be fired from ●
- cannot be fired from ●

- “compatibility” between regions and constraints
- “compatibility” between regions and time elapsing

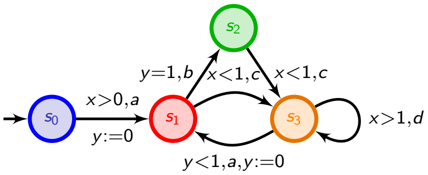
Technical tool: Region abstraction



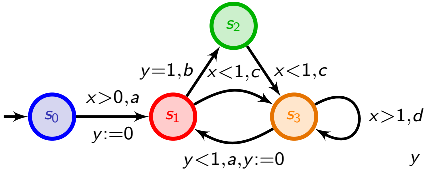
- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

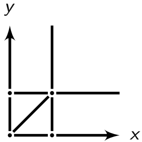
$\rightsquigarrow$ an equivalence of finite index

Technical tool: Region abstraction – An example [AD94]

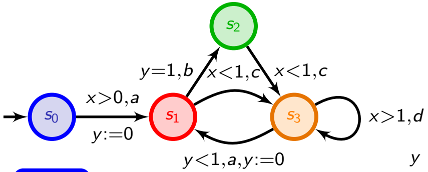


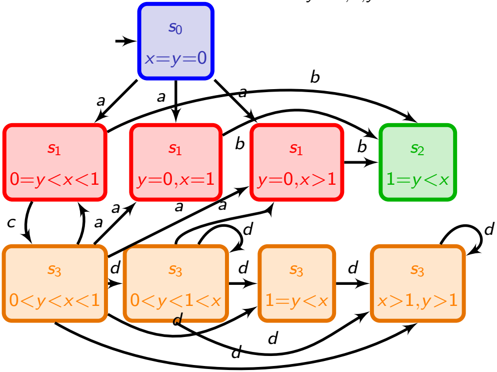
Technical tool: Region abstraction – An example [AD94]

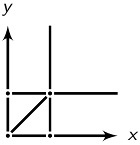




Technical tool: Region abstraction – An example [AD94]







Technical tool: Zones and DBMs

DBM = Difference Bound Matrix

Technical tool: Zones and DBMs

DBM = Difference Bound Matrix

Zones, or DBMs...

... are used to represent sets of states of timed automata:

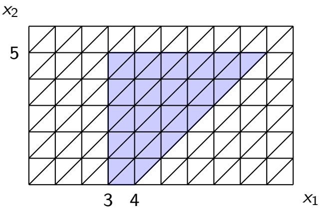
Technical tool: Zones and DBMs

DBM = Difference Bound Matrix

Zones, or DBMs...

... are used to represent sets of states of timed automata:

Zone: $(x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$



Technical tool: Zones and DBMs

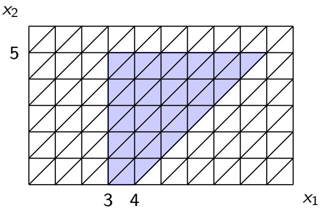
DBM = Difference Bound Matrix

Zones, or DBMs...

... are used to represent sets of states of timed automata:

Zone: $(x_1 \geq 3) \ \wedge \ (x_2 \leq 5) \ \wedge \ (x_1 - x_2 \leq 4)$

DBMs: $$\begin{array}{c c} & \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \end{array} & \begin{pmatrix} \infty & -3 & \infty \\ \infty & \infty & 4 \\ 5 & \infty & \infty \end{pmatrix} \end{array}$$



Technical tool: Zones and DBMs

DBM = Difference Bound Matrix

Zones, or DBMs...

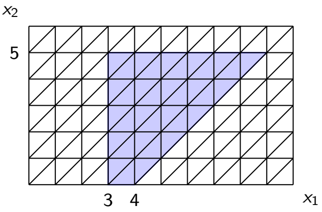
... are used to represent sets of states of timed automata:

Zone: $(x_1 \geq 3) \land (x_2 \leq 5) \land (x_1 - x_2 \leq 4)$

DBMs:
$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} \infty & -3 & \infty \\ \infty & \infty & 4 \\ 5 & \infty & \infty \end{pmatrix} \end{array}$$

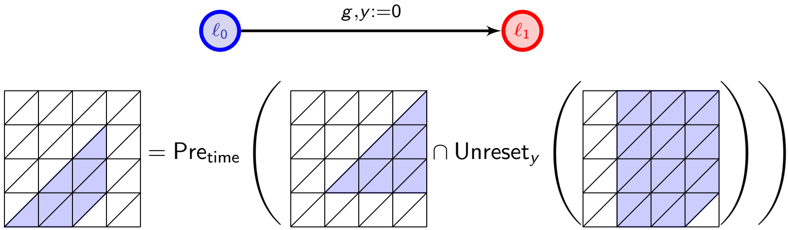
$$|||$$

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix} \end{array}$$



Technical tool: Zones and DBMs

They can be used to compute sets of states in timed automata



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:
  - it assumes zero-delay transitions
  - it assumes infinite precision of the clocks
  - it assumes immediate communication between systems
  - it assumes infinite frequency

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:
  - it assumes zero-delay transitions
  - it assumes infinite precision of the clocks
  - it assumes immediate communication between systems
  - it assumes infinite frequency
- 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:
  - it assumes zero-delay transitions
  - it assumes infinite precision of the clocks
  - it assumes immediate communication between systems
  - it assumes infinite frequency
- 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

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:
  - it assumes zero-delay transitions
  - it assumes infinite precision of the clocks
  - it assumes immediate communication between systems
  - it assumes infinite frequency
- It may generate timing anomalies
- It does not exclude non-realizable behaviours:
  - not only Zeno behaviours
  - many convergence phenomena are hidden

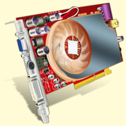
$\rightsquigarrow$ this requires infinite precision and might not be realizable

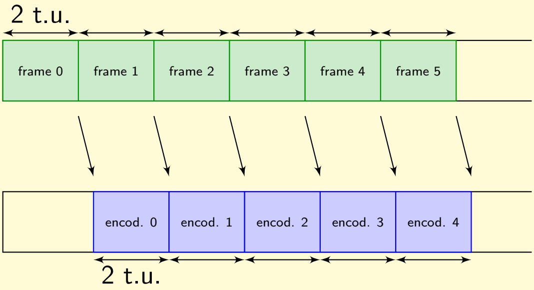
Important questions

- Is the real system correct when it is proven correct on the model?
- Does actual work transfer to real-world systems? To what extent?

Example 1: Imprecision on clock values

Frame capture [ACS10]

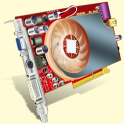


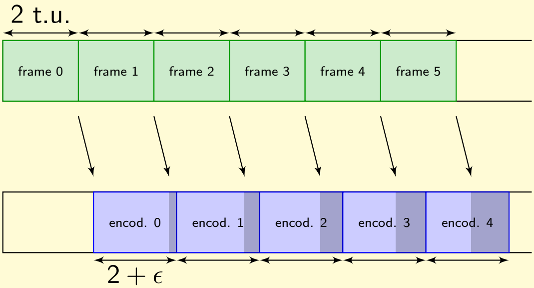


[ACS10] Abdellatif, Combaz, Sifakis. Model-based implementation of real-time applications. *Int. Conf. Embedded Software*, ACM 2010.

Example 1: Imprecision on clock values

Frame capture [ACS10]



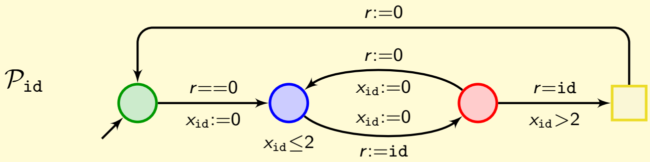


$\rightsquigarrow$ A frame will eventually be skipped

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

Example 2: Strict timing constraints

Mutual exclusion protocol [KLL+97]

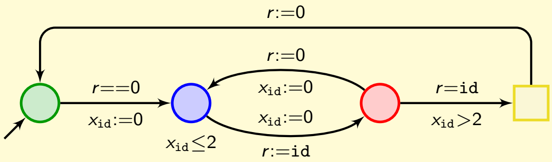


[KLL+97] Kristoffersen, Laroussinie, Larsen, Pettersson, Yi. A compositional proof of a real-time mutual exclusion protocol. *TAPSOFT*, 1997.

Example 2: Strict timing constraints

Mutual exclusion protocol [KLL+97]

$\mathcal{P}_{\mathtt{id}}$

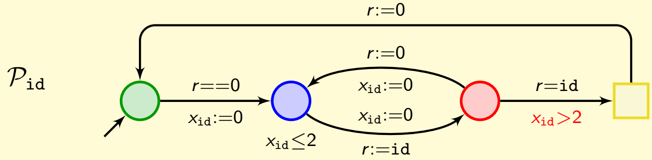


- When $\mathcal{P}_1$ and $\mathcal{P}_2$ run in parallel (sharing variable r), the state where both of them are in □ is not reachable.

[KLL+97] Kristoffersen, Laroussinie, Larsen, Pettersson, Yi. A compositional proof of a real-time mutual exclusion protocol. *TAPSOFT*, 1997.

Example 2: Strict timing constraints

Mutual exclusion protocol [KLL+97]



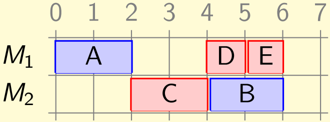
- When $\mathcal{P}_1$ and $\mathcal{P}_2$ run in parallel (sharing variable r), the state where both of them are in □ is not reachable.
- This property is lost when $x_{\mathtt{id}} > 2$ is replaced with $x_{\mathtt{id}} \geq 2$.

[KLL+97] Kristoffersen, Laroussinie, Larsen, Pettersson, Yi. A compositional proof of a real-time mutual exclusion protocol. *TAPSOFT*, 1997.

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



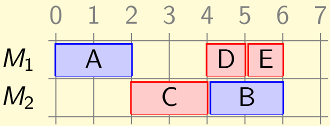
with the dependency constraints: $A \to B$ and $C \to D, E$.

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

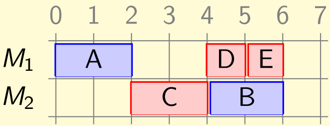
Example of a scenario



$\rightsquigarrow$ Schedulable in 6 time units

Example 3: Scheduling and timing anomaly

Example of a scenario

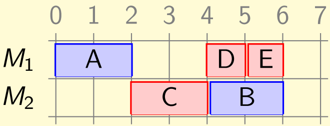


⇝ Schedulable in 6 time units

- Unexpectedly, the duration of A drops to 1.999

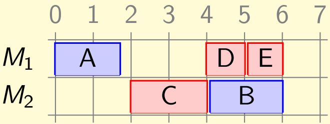
Example 3: Scheduling and timing anomaly

Example of a scenario



$\leadsto$ Schedulable in 6 time units

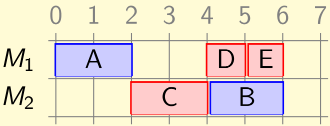
- Unexpectedly, the duration of A drops to 1.999



is not work-conserving

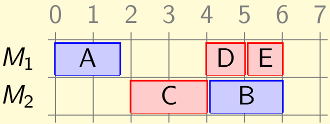
Example 3: Scheduling and timing anomaly

Example of a scenario

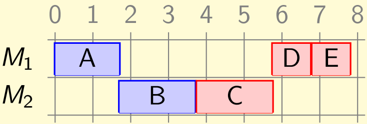


⇝ Schedulable in 6 time units

- Unexpectedly, the duration of A drops to 1.999



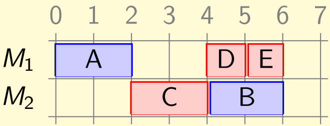
is not work-conserving



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

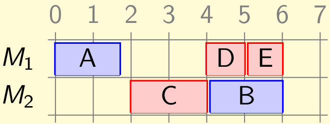
Example 3: Scheduling and timing anomaly

Example of a scenario

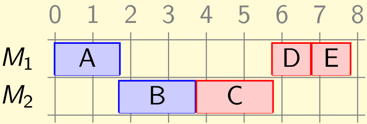


$\rightsquigarrow$ Schedulable in 6 time units

- Unexpectedly, the duration of A drops to 1.999



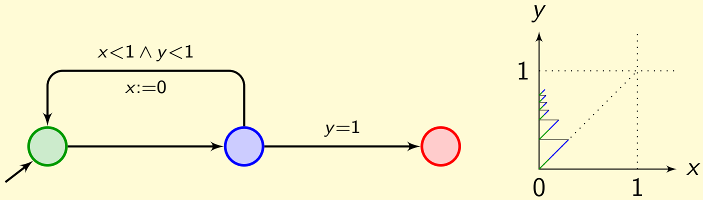
is not work-conserving



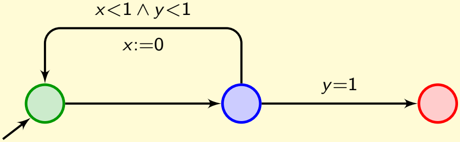
is work-conserving
and completes in 7.999 t.u.

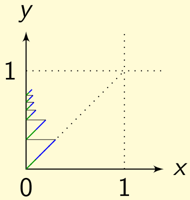
$\rightsquigarrow$ Standard analysis does not capture this **timing anomaly**

Example 4: Zeno behaviours



Example 4: Zeno behaviours

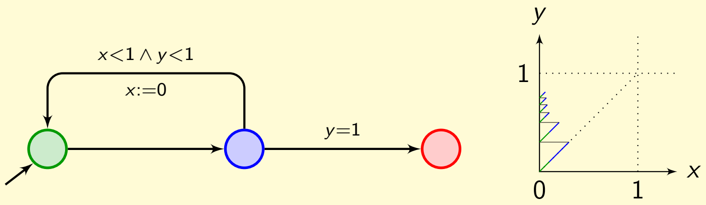




- Those are easy to detect and can be handled; [HS11]

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

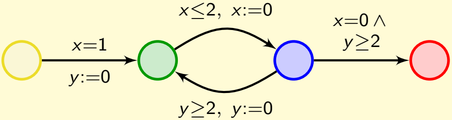
Example 4: Zeno behaviours



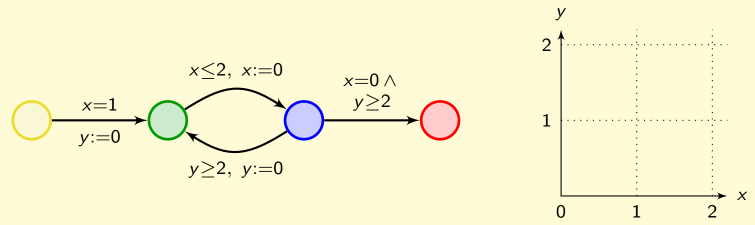
- Those are easy to detect and can be handled; [HS11]
- They are easy to remove by construction.

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

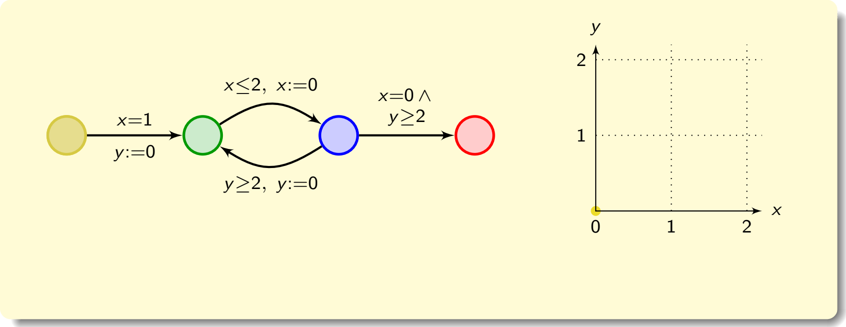
Example 5: More complex convergence phenomena



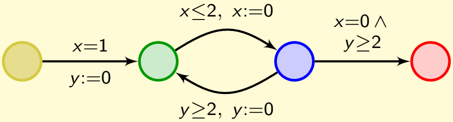
Example 5: More complex convergence phenomena

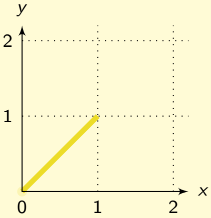


Example 5: More complex convergence phenomena

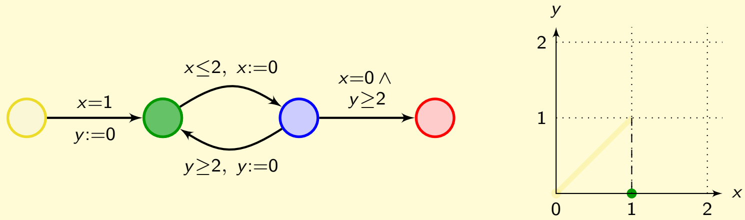


Example 5: More complex convergence phenomena

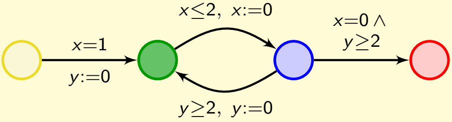


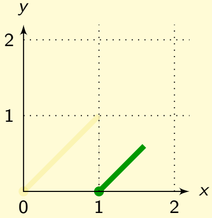


Example 5: More complex convergence phenomena

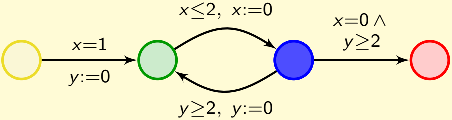


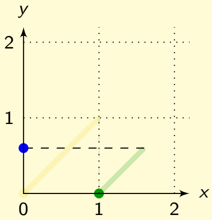
Example 5: More complex convergence phenomena



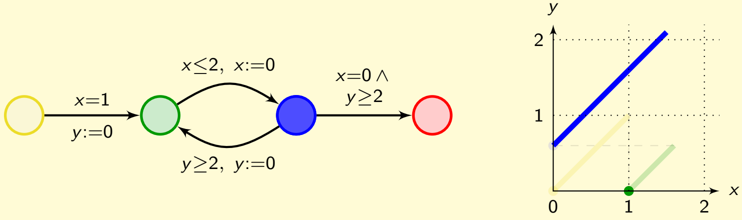


Example 5: More complex convergence phenomena

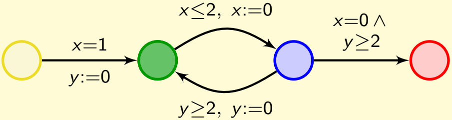


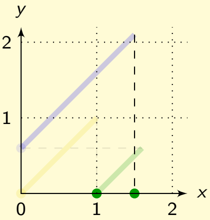


Example 5: More complex convergence phenomena

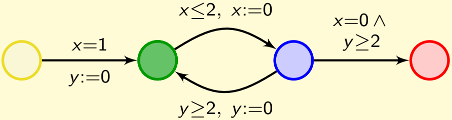


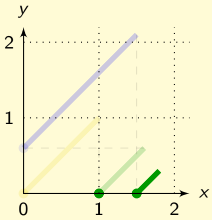
Example 5: More complex convergence phenomena



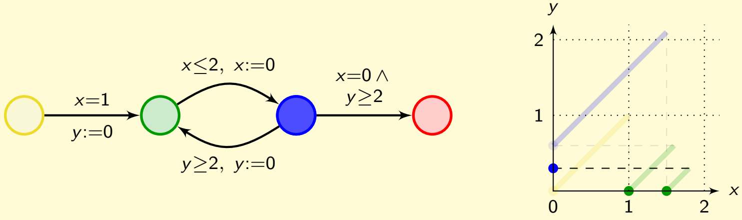


Example 5: More complex convergence phenomena

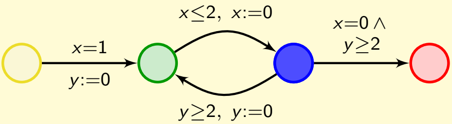


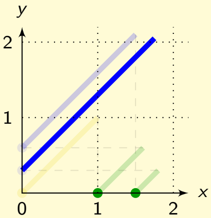


Example 5: More complex convergence phenomena

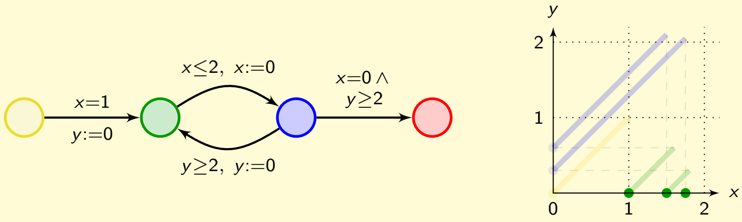


Example 5: More complex convergence phenomena

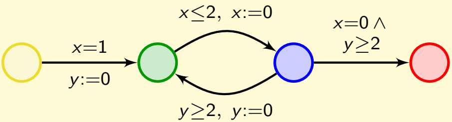


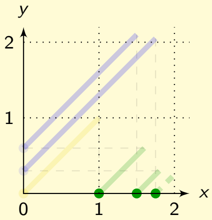


Example 5: More complex convergence phenomena

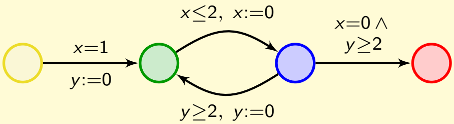


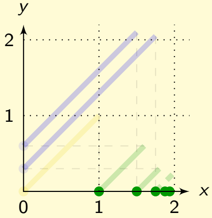
Example 5: More complex convergence phenomena





Example 5: More complex convergence phenomena





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

The goal

Add robustness to the theory of timed automata

The goal

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.

The goal

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 correct systems

The goal

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

The goal

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
  $\rightsquigarrow$ Robustness calls for specific theories for each application areas

The goal

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
  $\rightsquigarrow$ Robustness calls for specific theories for each application areas

Rest of the talk

We present a couple of frameworks that have been developed recently in this context. We focus on perturbations on time measurements and jitter.

Outline

1. Introduction

2. Robust model-checking and implementation
   - Parameterized enlarged semantics
   - Automatic generation of an implementation
   - Implementation by shrinking

3. Robust realisability
   - Excess semantics
   - Conservative semantics

4. Conclusion

Robust model-checking approach

Idea

Capture any real (or approximate) behaviours (e.g. the implementation) in the verification process

Robust model-checking approach

Idea

Capture any real (or approximate) behaviours (e.g. the implementation) in the verification process

Due to imprecisions,

$$\text{“standard” correctness of } \mathcal{A} \quad \not\Rightarrow \quad \text{correctness of } \mathcal{A}_{\mathtt{real}}$$

Robust model-checking approach

Idea

Capture any real (or approximate) behaviours (e.g. the implementation) in the verification process

Due to imprecisions,

$$\text{“standard” correctness of } \mathcal{A} \quad \not\Rightarrow \quad \text{correctness of } \mathcal{A}_{\texttt{real}}$$

$\rightsquigarrow$ We aim at proposing frameworks in which we will ensure the correctness of the real behaviour of the system

Robust model-checking approach

Idea

Capture any real (or approximate) behaviours (e.g. the implementation) in the verification process

Due to imprecisions,

$$\text{“standard” correctness of } \mathcal{A} \quad \not\Rightarrow \quad \text{correctness of } \mathcal{A}_{\texttt{real}}$$

$\leadsto$ We aim at proposing frameworks in which we will ensure the correctness of the real behaviour of the system

We describe two such frameworks:

1. either we implement $\mathcal{A}$ and we prove:

$$\text{“robust” correctness of } \mathcal{A} \quad \Rightarrow \quad \text{correctness of } \mathcal{A}_{\texttt{real}}$$

Robust model-checking approach

Idea

Capture any real (or approximate) behaviours (e.g. the implementation) in the verification process

Due to imprecisions,

$$\text{“standard” correctness of } \mathcal{A} \quad \not\Rightarrow \quad \text{correctness of } \mathcal{A}_{\texttt{real}}$$

$\leadsto$ We aim at proposing frameworks in which we will ensure the correctness of the real behaviour of the system

We describe two such frameworks:

1. either we implement $\mathcal{A}$ and we prove:

$$\text{“robust” correctness of } \mathcal{A} \quad \Rightarrow \quad \text{correctness of } \mathcal{A}_{\texttt{real}}$$

2. or we build $\mathcal{A}$ and implement $\mathcal{B}$, and we prove:

$$\begin{aligned} \text{correctness of } \mathcal{A} \quad &\Rightarrow \quad \text{“robust” correctness of } \mathcal{B} \\ &\Rightarrow \quad \text{correctness of } \mathcal{B}_{\texttt{real}} \end{aligned}$$

Outline

1. Introduction

2. Robust model-checking and implementation
   - Parameterized enlarged semantics
   - Automatic generation of an implementation
   - Implementation by shrinking

3. Robust realisability
   - Excess semantics
   - Conservative semantics

4. Conclusion

Parameterized enlarged semantics for timed automata

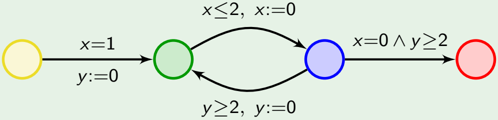
A transition can be taken at any time in $[t - \delta; t + \delta]$

Parameterized enlarged semantics for timed automata

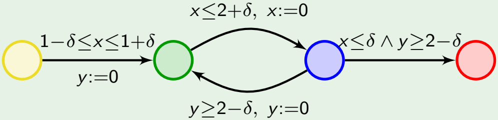
A transition can be taken at any time in $[t - \delta; t + \delta]$

Example

Given a parameter δ,



is transformed into



Parameterized enlarged semantics – Discussion

What is the relevance of this semantics?

- This is a worst-case approach
- This captures approximate behaviours of the system
- One can define program semantics such that for every $\epsilon > 0$:

$$\mathcal{A} \subseteq \mathtt{program}_\epsilon(\mathcal{A}) \subseteq \mathcal{A}_{f(\epsilon)}$$

ϵ: parameters of the semantics

[DDR04] De Wulf, Doyen, Raskin. Almost ASAP semantics: From timed models to timed implementations *HSCC*, 2004.
[SBM11] Sankur, Bouyer, Markey. Shrinking Timed Automata. *FSTTCS*, 2011.

Parameterized enlarged semantics – Discussion

What is the relevance of this semantics?

- This is a worst-case approach
- This captures approximate behaviours of the system
- One can define program semantics such that for every $\epsilon > 0$:

$$\mathcal{A} \subseteq \mathtt{program}_\epsilon(\mathcal{A}) \subseteq \mathcal{A}_{f(\epsilon)}$$

ϵ: parameters of the semantics

Methodology

- Design $\mathcal{A}$
- Verify $\mathcal{A}_\delta$ (better if δ is a parameter)
- Implement $\mathcal{A}$

Parameterized enlarged semantics – Discussion

What is the relevance of this semantics?

- This is a worst-case approach
- This captures approximate behaviours of the system
- One can define program semantics such that for every $\epsilon > 0$:

$$\mathcal{A} \subseteq \mathtt{program}_\epsilon(\mathcal{A}) \subseteq \mathcal{A}_{f(\epsilon)}$$

ϵ: parameters of the semantics

Methodology

- Design $\mathcal{A}$
- Verify $\mathcal{A}_\delta$ (better if δ is a parameter)
- Implement $\mathcal{A}$

$\rightsquigarrow$ This is good for designing systems
with simple timing constraints (e.g. equalities).

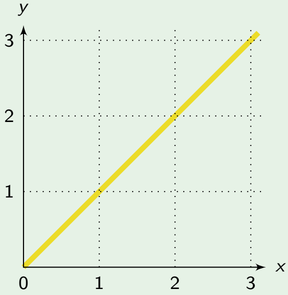
Parameterized enlarged semantics – Algorithmics

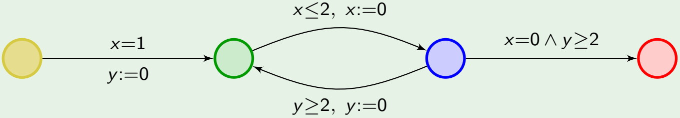
$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Example

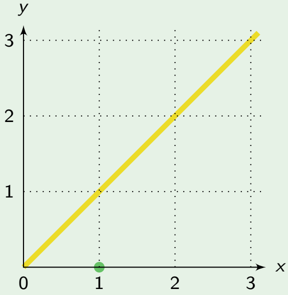


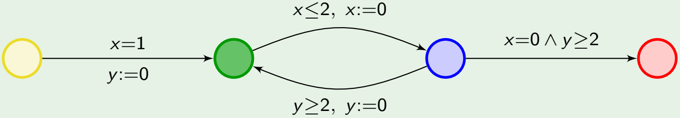


Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Example

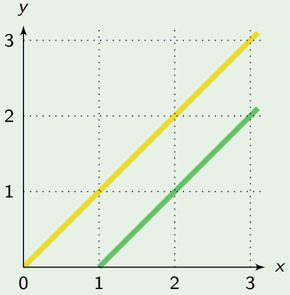


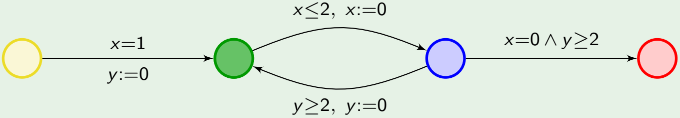


Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Example

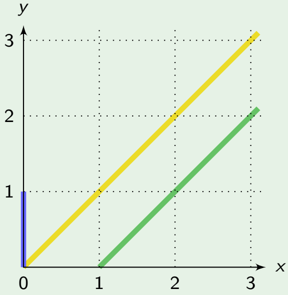


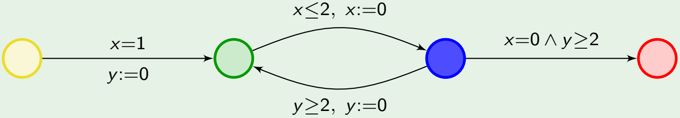


Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Example

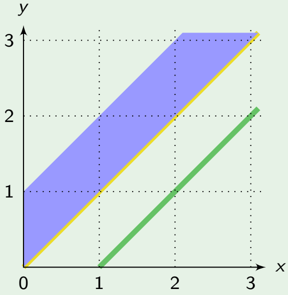


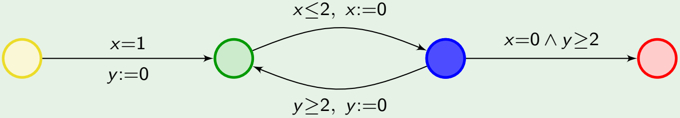


Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Example

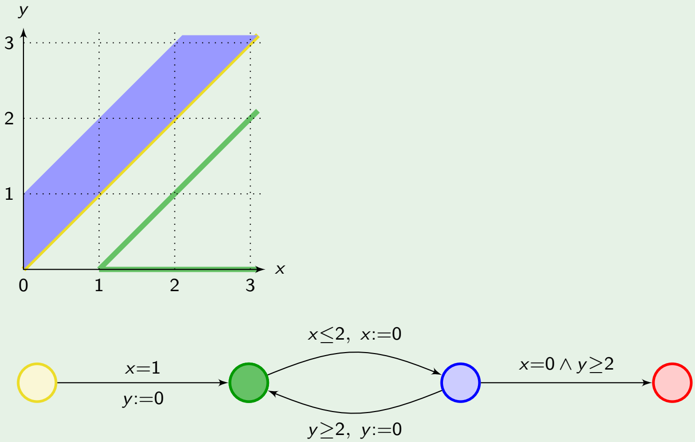




Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

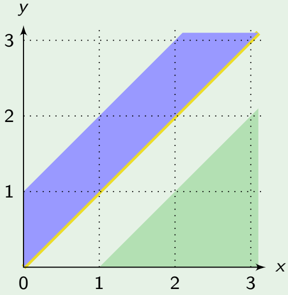
Example

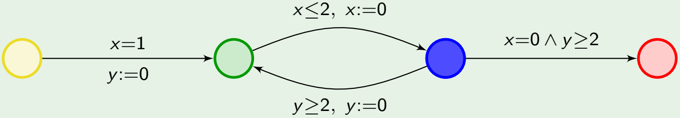


Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Example

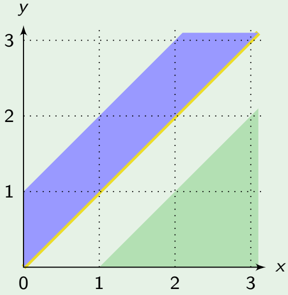


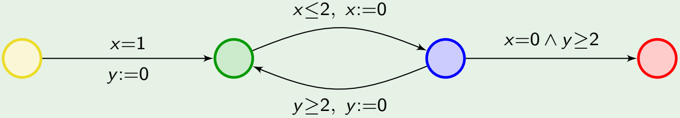


Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Example

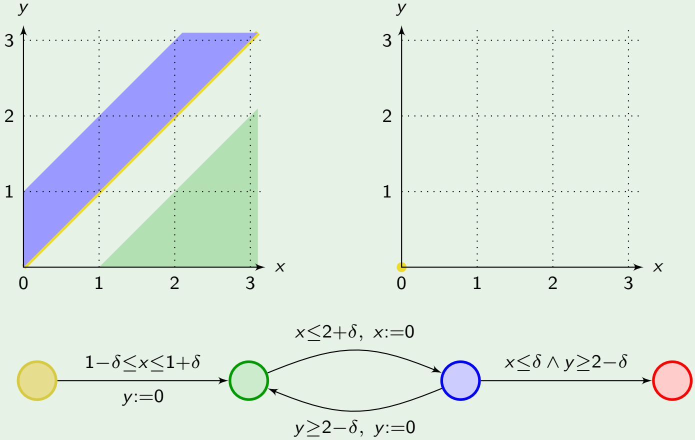




Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

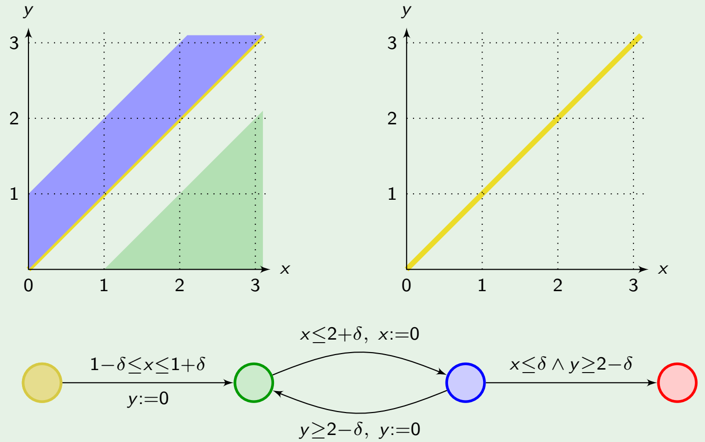
Example



Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

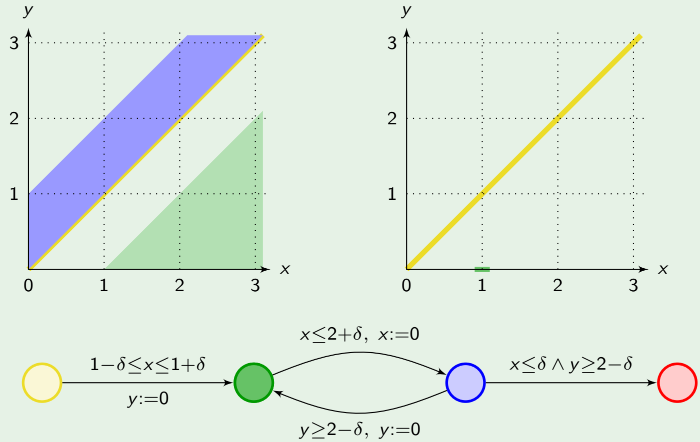
Example



Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

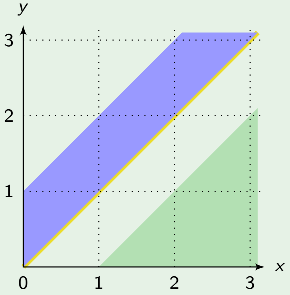
Example

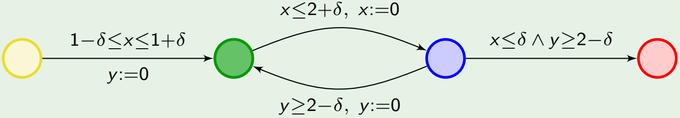


Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Example

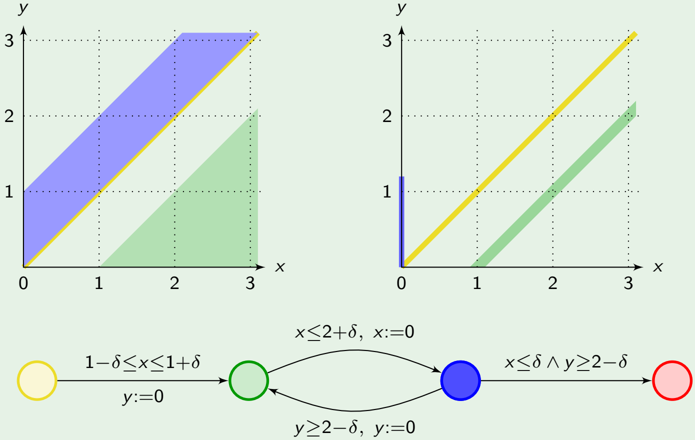




Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

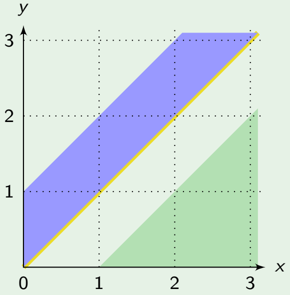
Example

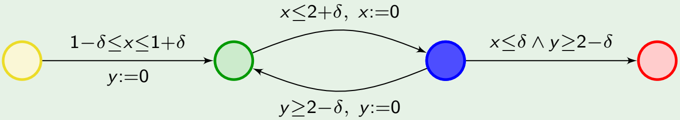


Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Example

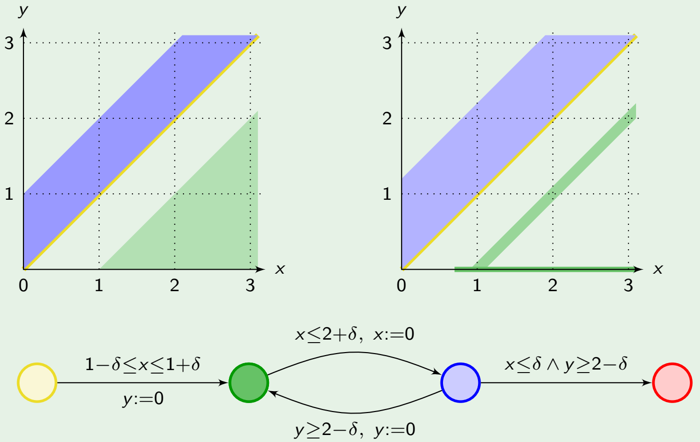




Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

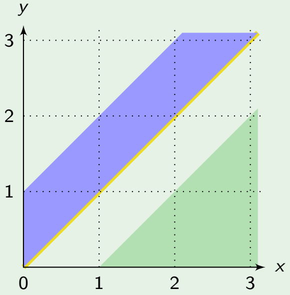
Example

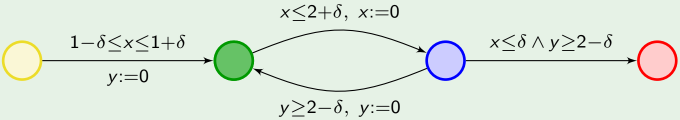


Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Example

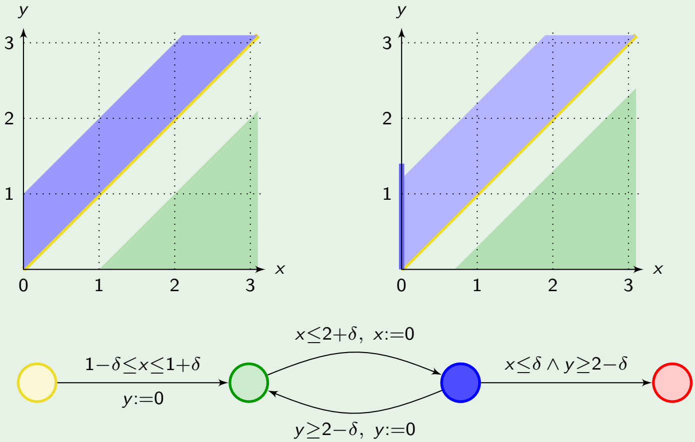




Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

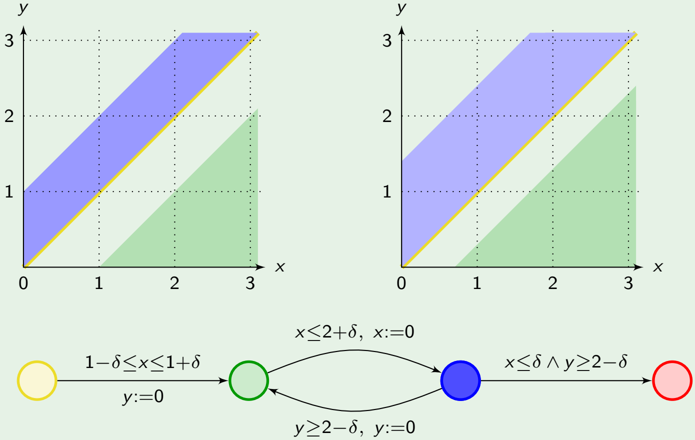
Example



Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

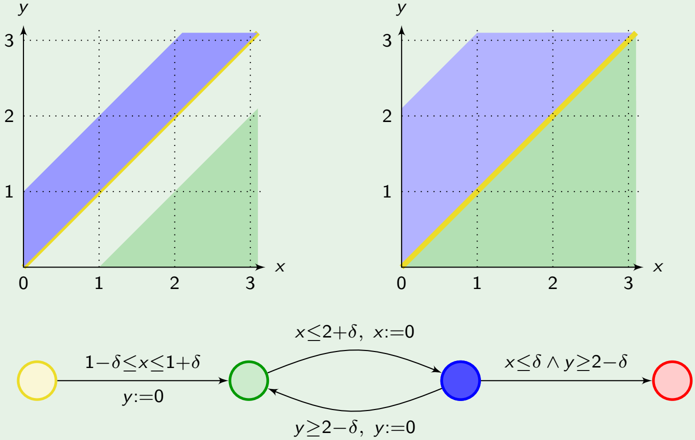
Example



Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

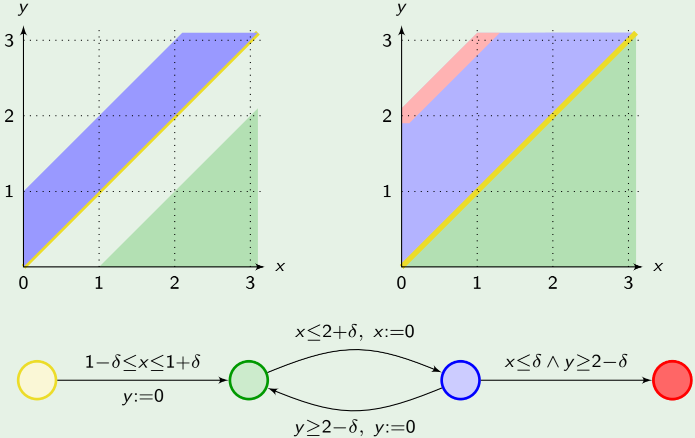
Example



Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

Example



Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

The (parameterized) robust model-checking problem

It asks whether there is some $\delta_0 > 0$ such that for every $0 \leq \delta \leq \delta_0$, $\mathcal{A}_\delta \models \varphi$.

Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

The (parameterized) robust model-checking problem

It asks whether there is some $\delta_0 > 0$ such that for every $0 \leq \delta \leq \delta_0$, $\mathcal{A}_\delta \models \varphi$.

- When δ is small, truth of φ is independent of δ

Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

The (parameterized) robust model-checking problem

It asks whether there is some $\delta_0 > 0$ such that for every $0 \leq \delta \leq \delta_0$, $\mathcal{A}_\delta \models \varphi$.

- When δ is small, truth of φ is independent of δ
- It can be computed using a simple extension of the region automaton

Parameterized enlarged semantics – Algorithmics

$\rightsquigarrow$ It adds extra behaviours, however small may be parameter δ

The (parameterized) robust model-checking problem

It asks whether there is some $\delta_0 > 0$ such that for every $0 \leq \delta \leq \delta_0$, $\mathcal{A}_\delta \models \varphi$.

- When δ is small, truth of φ is independent of δ
- It can be computed using a simple extension of the region automaton

Theorem

Robust model-checking of reachability, Büchi, LTL, CoflatMTL properties is decidable. Complexities are those of standard non robust model-checking problems.

[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.
[BMR06] Bouyer, Markey, Reynier. Robust model-checking of timed automata. *LATIN*, 2006.
[BMR08] Bouyer, Markey, Reynier. Robust analysis of timed automata via channel machines. *FoSSaCS*, 2008.

Outline

1. Introduction

2. Robust model-checking and implementation
   - Parameterized enlarged semantics
   - Automatic generation of an implementation
   - Implementation by shrinking

3. Robust realisability
   - Excess semantics
   - Conservative semantics

4. Conclusion

Automatic generation of an implementation

The (approx.) implementation synthesis problem

Given $\mathcal{A}$, build $\mathcal{A}'$ such that:

- $\mathcal{A}'$ 'identical' (e.g. bisimilar) to $\mathcal{A}$
- $\mathcal{A}'$ is 'robust' (that is, good enough for implementation)

[BLM+11] Bouyer, Larsen, Markey, Sankur, Thrane. Timed automata can always be made implementable. *CONCUR*, 2011.

Automatic generation of an implementation

The (approx.) implementation synthesis problem

Given $\mathcal{A}$, build $\mathcal{A}'$ such that:

- $\mathcal{A}'$ 'identical' (e.g. bisimilar) to $\mathcal{A}$
- $\mathcal{A}'$ is 'robust' (that is, good enough for implementation)

The second condition can be (for instance) read as $\mathcal{A}'$ is approximately the same as $\mathcal{A}'_\delta$, for small enough δ.

[BLM+11] Bouyer, Larsen, Markey, Sankur, Thrane. Timed automata can always be made implementable. *CONCUR*, 2011.

Automatic generation of an implementation

The (approx.) implementation synthesis problem

Given $\mathcal{A}$, build $\mathcal{A}'$ such that:

- $\mathcal{A}'$ 'identical' (e.g. bisimilar) to $\mathcal{A}$
- $\mathcal{A}'$ is 'robust' (that is, good enough for implementation)

The second condition can be (for instance) read as $\mathcal{A}'$ is approximately the same as $\mathcal{A}'_\delta$, for small enough δ.

Theorem

All timed automata are approximately implementable!
(for approx. bisimulation)

- Technical tool: region construction

[BLM+11] Bouyer, Larsen, Markey, Sankur, Thrane. Timed automata can always be made implementable. *CONCUR*, 2011.

Automatic generation of an implementation

Methodology

- Design and verify $\mathcal{A}$
- Implement $\mathcal{A}'$ (automatically generated)

[BLM+11] Bouyer, Larsen, Markey, Sankur, Thrane. Timed automata can always be made implementable. *CONCUR*, 2011.

Automatic generation of an implementation

Methodology

- Design and verify $\mathcal{A}$
- Implement $\mathcal{A}'$ (automatically generated)

- ☺ Separates design and implementation
- ☹ $\mathcal{A}'$ is much bigger than $\mathcal{A}$

[BLM$^+$11] Bouyer, Larsen, Markey, Sankur, Thrane. Timed automata can always be made implementable. *CONCUR*, 2011.

Outline

1. Introduction

2. Robust model-checking and implementation
   - Parameterized enlarged semantics
   - Automatic generation of an implementation
   - Implementation by shrinking

3. Robust realisability
   - Excess semantics
   - Conservative semantics

4. Conclusion

Parameterized shrunk semantics for timed automata

A constraint $[a, b]$ **is shrunk to** $[a + k\delta; b - h\delta]$

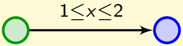
[SBM11] Sankur, Bouyer, Markey. Shrinking Timed Automata. *FSTTCS*, 2011.

Parameterized shrunk semantics for timed automata

A constraint $[a, b]$ is shrunk to $[a + k\delta; b - h\delta]$

Why should we do that?

Abstract model



Real-world model

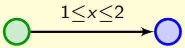
[SBM11] Sankur, Bouyer, Markey. Shrinking Timed Automata. *FSTTCS*, 2011.

Parameterized shrunk semantics for timed automata

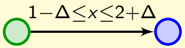
A constraint $[a, b]$ **is shrunk to** $[a + k\delta; b - h\delta]$

Why should we do that?

Abstract model



Real-world model



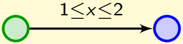
[SBM11] Sankur, Bouyer, Markey. Shrinking Timed Automata. *FSTTCS*, 2011.

Parameterized shrunk semantics for timed automata

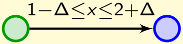
A constraint $[a, b]$ **is shrunk to** $[a + k\delta; b - h\delta]$

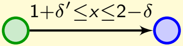
Why should we do that?

Abstract model



Real-world model





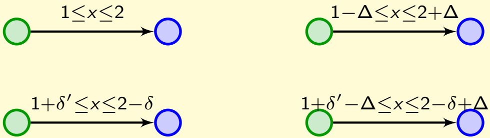
[SBM11] Sankur, Bouyer, Markey. Shrinking Timed Automata. *FSTTCS*, 2011.

Parameterized shrunk semantics for timed automata

A constraint $[a, b]$ is shrunk to $[a + k\delta; b - h\delta]$

Why should we do that?

Abstract model　　　　Real-world model



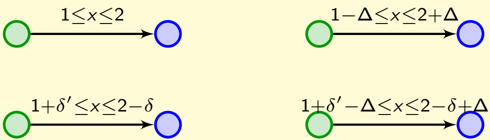
[SBM11] Sankur, Bouyer, Markey. Shrinking Timed Automata. *FSTTCS*, 2011.

Parameterized shrunk semantics for timed automata

A constraint $[a, b]$ **is shrunk to** $[a + k\delta; b - h\delta]$

Why should we do that?

Abstract model | Real-world model



It is fine as soon as $[1 + \delta' - \Delta; 2 - \delta + \Delta] \subseteq [1; 2]$,
which is the case when $\delta, \delta' \geq \Delta$.

[SBM11] Sankur, Bouyer, Markey. Shrinking Timed Automata. *FSTTCS*, 2011.

Parameterized shrunk semantics for timed automata

A constraint $[a, b]$ **is shrunk to** $[a + k\delta; b - h\delta]$

Summary of the approach

$\rightsquigarrow$ Shrink the clock constraints in the model, to prevent additional behaviours in the implementation

- If $\mathcal{B} = \mathcal{A}_{-\mathbf{k}\delta}$, then

$$\mathcal{B} \subseteq \texttt{program}_\epsilon(\mathcal{B}) \subseteq \mathcal{B}_{f(\epsilon)} = \mathcal{A}_{-\mathbf{k}\delta+f(\epsilon)} \subseteq \mathcal{A}$$

[SBM11] Sankur, Bouyer, Markey. Shrinking Timed Automata. *FSTTCS*, 2011.

Parameterized shrunk semantics – Discussion

What is the relevance of that approach?

Anticipate imprecisions to prevent additional behaviours in the real-world

Parameterized shrunk semantics – Discussion

What is the relevance of that approach?

Anticipate imprecisions to prevent additional behaviours in the real-world

Methodology

- Design and verify $\mathcal{A}$
- Implement $\mathcal{A}_{-\mathbf{k}\delta}$ (parameters are $\mathbf{k}$ and δ)

Parameterized shrunk semantics – Discussion

What is the relevance of that approach?

Anticipate imprecisions to prevent additional behaviours in the real-world

Methodology

- Design and verify $\mathcal{A}$
- Implement $\mathcal{A}_{-\mathbf{k}\delta}$ (parameters are **k** and δ)

$\leadsto$ This is good for designing systems with strong/hard timing constraints

Parameterized shrunk semantics – Discussion

What is the relevance of that approach?

Anticipate imprecisions to prevent additional behaviours in the real-world

Methodology

- Design and verify $\mathcal{A}$
- Implement $\mathcal{A}_{-\mathbf{k}\delta}$ (parameters are **k** and δ)

$\leadsto$ This is good for designing systems with strong/hard timing constraints

⚠ Problem

Make sure that no important behaviours are lost in $\mathcal{A}_{-\mathbf{k}\delta}$!!

Parameterized shrunk semantics – Algorithmics

The (parameterized) shrinkability problem

Find parameters $\mathbf{k}$ and δ such that:

- $\mathcal{A} \sqsubseteq_{\mathtt{t.a.}} \mathcal{A}_{-\mathbf{k}\delta}$ (or $\mathcal{F} \sqsubseteq_{\mathtt{t.a.}} \mathcal{A}_{-\mathbf{k}\delta}$ for some finite automaton $\mathcal{F}$)
  [shrinkability w.r.t. untimed simulation]
- $\mathcal{A}_{-\mathbf{k}\delta}$ is non-blocking whenever $\mathcal{A}$ is non-blocking
  [shrinkability w.r.t. non-blockingness]

Parameterized shrunk semantics – Algorithmics

The (parameterized) shrinkability problem

Find parameters $\mathbf{k}$ and δ such that:

- $\mathcal{A} \sqsubseteq_{\mathtt{t.a.}} \mathcal{A}_{-\mathbf{k}\delta}$ (or $\mathcal{F} \sqsubseteq_{\mathtt{t.a.}} \mathcal{A}_{-\mathbf{k}\delta}$ for some finite automaton $\mathcal{F}$)
  [shrinkability w.r.t. untimed simulation]
- $\mathcal{A}_{-\mathbf{k}\delta}$ is non-blocking whenever $\mathcal{A}$ is non-blocking
  [shrinkability w.r.t. non-blockingness]

Theorem

Parameterized shrinkability can be decided (in exponential time).

- Challenge: take care of the accumulation of perturbations
- Technical tools: parameterized shrunk DBM, max-plus equations
- Tool Shrinktech developed by Ocan Sankur [San13]

  `http://www.lsv.ens-cachan.fr/Software/shrinktech/`

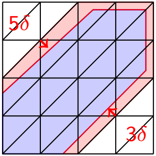
[San13] Sankur. Shrinktech: A tool for the robustness analysis of timed automata. *CAV*, 2013.

Technical tools: Shrunk DBMs and max-plus equations

Shrunk DBMs with parameter δ...

... are used to represent sets of states of shrunk automata:

DBM: $x - y \leq \alpha$
Shrunk DBM: $x - y \leq \alpha - 5\delta$

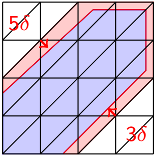


Technical tools: Shrunk DBMs and max-plus equations

Shrunk DBMs with parameter δ...

... are used to represent sets of states of shrunk automata:

DBM:	$x - y \leq \alpha$
Shrunk DBM:	$x - y \leq \alpha - 5\delta$
Parameterized shrunk DBM:	$x - y \leq \alpha - k\delta$

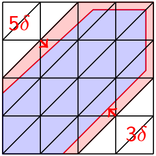


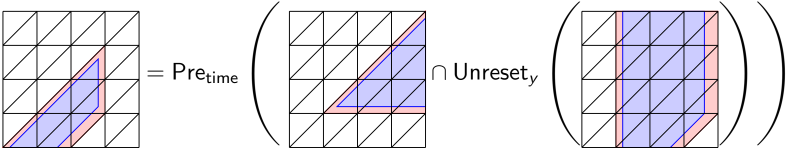
Technical tools: Shrunk DBMs and max-plus equations

Shrunk DBMs with parameter δ...

... are used to represent sets of states of shrunk automata:

DBM:	$x - y \leq \alpha$
Shrunk DBM:	$x - y \leq \alpha - 5\delta$
Parameterized shrunk DBM:	$x - y \leq \alpha - k\delta$



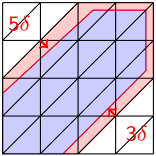


Technical tools: Shrunk DBMs and max-plus equations

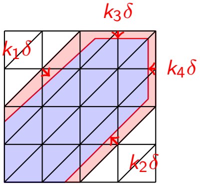
Shrunk DBMs with parameter δ...

... are used to represent sets of states of shrunk automata:

DBM:	$x - y \leq \alpha$
Shrunk DBM:	$x - y \leq \alpha - 5\delta$
Parameterized shrunk DBM:	$x - y \leq \alpha - k\delta$

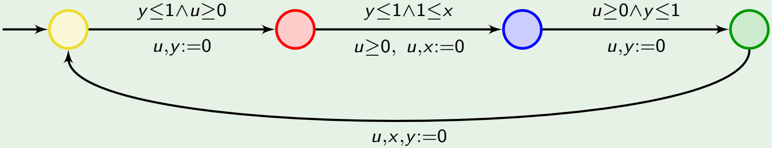


... to max-plus equations

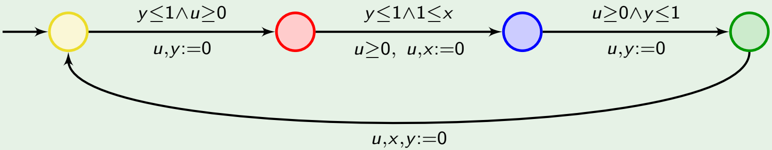


$$\rightsquigarrow \quad \begin{aligned} k_3 &= \max(k_1 + k_2, k_3) \\ k_2 &= \max(k_2, k_1) + k_3 \end{aligned}$$

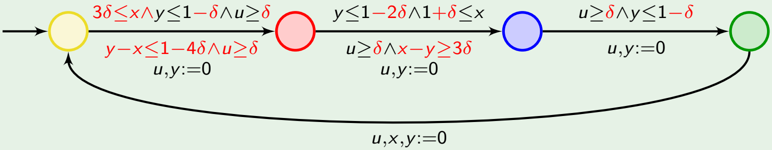
Example


$y{\leq}1{\wedge}u{\geq}0$
$u,y{:=}0$
$y{\leq}1{\wedge}1{\leq}x$
$u{\geq}0,\ u,x{:=}0$
$u{\geq}0{\wedge}y{\leq}1$
$u,y{:=}0$
$u,x,y{:=}0$

Example



The largest shrunk automaton which is correct w.r.t. untimed simulation and non-blockingness is:



Outline

1. Introduction

2. Robust model-checking and implementation
   - Parameterized enlarged semantics
   - Automatic generation of an implementation
   - Implementation by shrinking

3. Robust realisability
   - Excess semantics
   - Conservative semantics

4. Conclusion

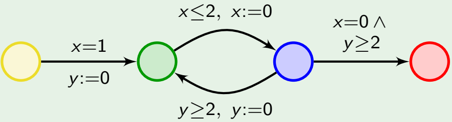
Robust realisability

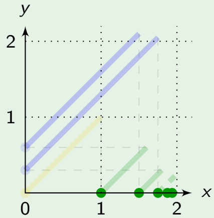
Here, a strategy in a timed automaton is a way to resolve (time and action) non-determinism

Robust realisability

Here, a strategy in a timed automaton is a way to resolve (time and action) non-determinism

Example



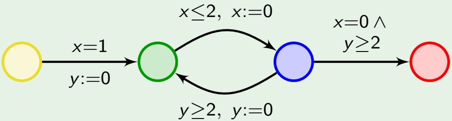


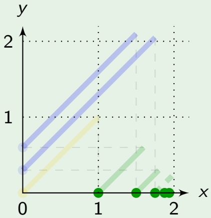
Strategy: in location with value x, delay $\frac{2-x}{2}$

Robust realisability

Here, a strategy in a timed automaton is a way to resolve (time and action) non-determinism

Example





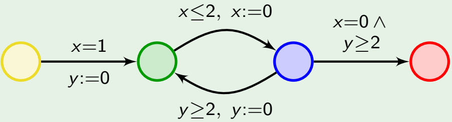
Strategy: in location with value x, delay $\frac{2-x}{2}$

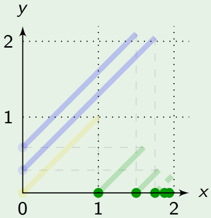
- This strategy requires infinite precision

Robust realisability

Here, a strategy in a timed automaton is a way to resolve (time and action) non-determinism

Example





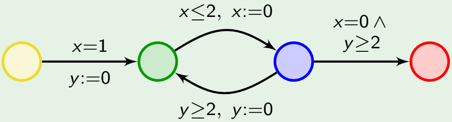
Strategy: in location with value x, delay $\frac{2-x}{2}$

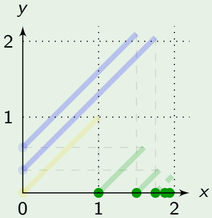
- This strategy requires infinite precision
- In practice, when x is close to 2, no additional delay is supported: the run is theoretically infinite, but it is actually blocking

Robust realisability

Here, a strategy in a timed automaton is a way to resolve (time and action) non-determinism

Example





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

Here, a strategy in a timed automaton is a way to resolve (time and action) non-determinism

Idea of robust realisability

Synthesize strategies that realise some property, even under perturbations: strategies should adapt to previous imprecisions imprecisions

$\rightsquigarrow$ develop a theory of robust strategies that tolerate errors/imprecisions and avoid convergence

Game semantics of a timed automaton

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

Game semantics of a timed automaton

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.

Game semantics of a timed automaton

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.

Outline

1. Introduction

2. Robust model-checking and implementation
   - Parameterized enlarged semantics
   - Automatic generation of an implementation
   - Implementation by shrinking

3. Robust realisability
   - Excess semantics
   - Conservative semantics

4. Conclusion

The excess game semantics

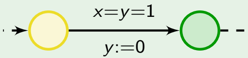
Constraints may not be satisfied after the perturbation: that is, only $v + d$ should satisfy g

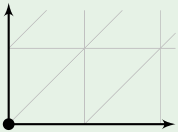
[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP*, 2012.

The excess game semantics

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

Example



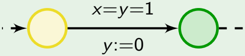


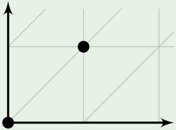
[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP*, 2012.

The excess game semantics

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

Example



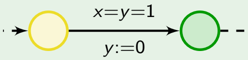


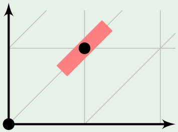
[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP*, 2012.

The excess game semantics

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

Example



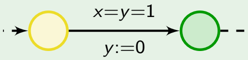


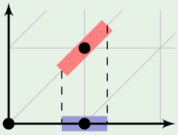
[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP*, 2012.

The excess game semantics

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

Example



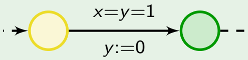


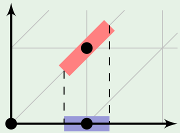
[BMS12] Bouyer, Markey, Sankur. Robust reachability in timed automata: A game-based approach. *ICALP*, 2012.

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*, 2012.

The excess game semantics – Algorithmics

The (parameterized) synthesis problem

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

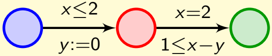
The excess game semantics – Algorithmics

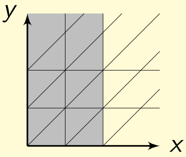
The (parameterized) synthesis problem

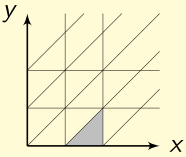
Synthesize $\delta > 0$ and a δ-robust strategy that achieves a given goal.

Two challenges

1. Accumulation of perturbations:







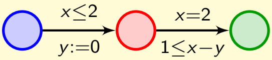
The excess game semantics – Algorithmics

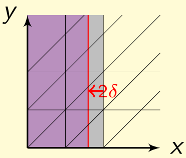
The (parameterized) synthesis problem

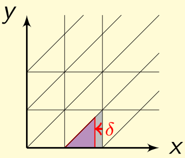
Synthesize $\delta > 0$ and a δ-robust strategy that achieves a given goal.

Two challenges

1. Accumulation of perturbations:







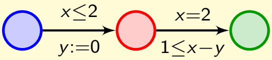
The excess game semantics – Algorithmics

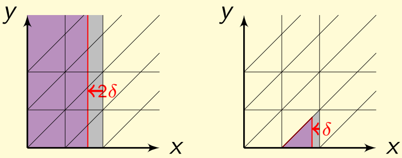
The (parameterized) synthesis problem

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

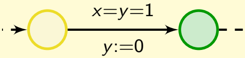
Two challenges

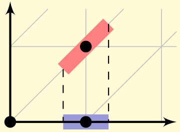
1. Accumulation of perturbations:





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

Outline

1. Introduction

2. Robust model-checking and implementation
   - Parameterized enlarged semantics
   - Automatic generation of an implementation
   - Implementation by shrinking

3. Robust realisability
   - Excess semantics
   - Conservative semantics

4. Conclusion

The conservative game semantics

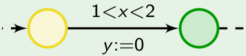
Constraints have to be satisfied after the perturbation: that is, $v + d + \epsilon$ should satisfy g for every $\epsilon \in [-\delta; +\delta]$

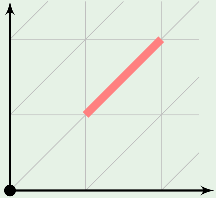
[SBMR13] Sankur, Bouyer, Markey, Reynier. Robust Controller Synthesis in Timed Automata. *CONCUR'13*.

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



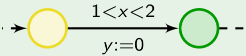


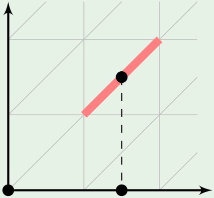
[SBMR13] Sankur, Bouyer, Markey, Reynier. Robust Controller Synthesis in Timed Automata. *CONCUR'13.*

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





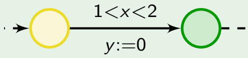
$\rightsquigarrow$ Strongly ensures timing constraints, ensures divergence of time, prevents converging phenomena

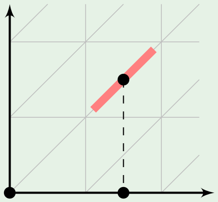
[SBMR13] Sankur, Bouyer, Markey, Reynier. Robust Controller Synthesis in Timed Automata. *CONCUR'13.*

The conservative game semantics

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

Example





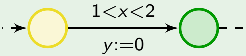
$\rightsquigarrow$ Strongly ensures timing constraints, ensures divergence of time, prevents converging phenomena

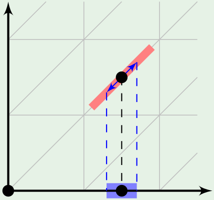
[SBMR13] Sankur, Bouyer, Markey, Reynier. Robust Controller Synthesis in Timed Automata. *CONCUR'13*.

The conservative game semantics

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 – Algorithmics

The (parameterized) synthesis problem

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

The conservative game semantics – Algorithmics

The (parameterized) synthesis problem

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

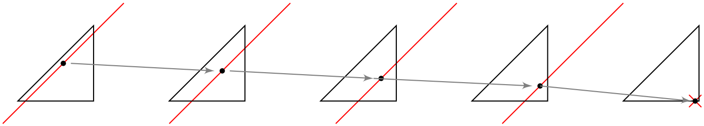
Theorem

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

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

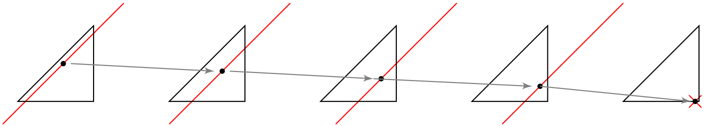
The problem consists in finding cycles that do not become blocked.

- A converging phenomena:

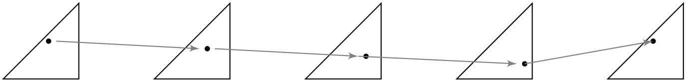


The problem consists in finding cycles that do not become blocked.

- A converging phenomena:



- No convergence:

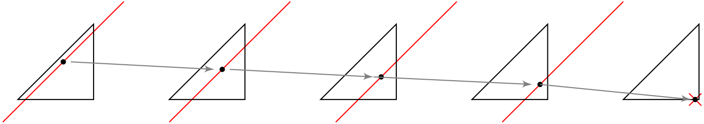


No such constraining half-spaces.

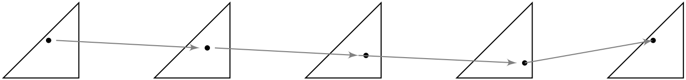
[AB11] Asarin, Basset. Thin and Thick Timed Regular Languages. *FORMATS*, 2011.

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

- A converging phenomena:



- No convergence:



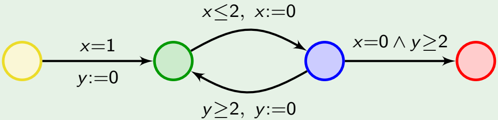
No such constraining half-spaces.

Tools for solving the synthesis problem

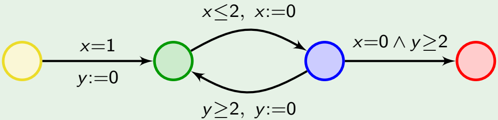
- Orbit graphs, forgetful cycles [AB11]
- Forgetful (that is, strongly connected) orbit graph $\Leftrightarrow$ no convergence phenomena
  $\rightsquigarrow$ strong relation with thick automata.

[AB11] Asarin, Basset. Thin and Thick Timed Regular Languages. *FORMATS*, 2011.

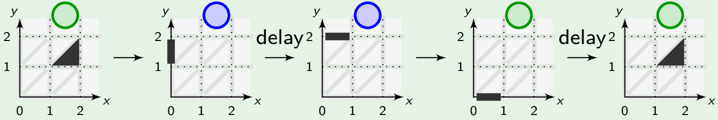
Example



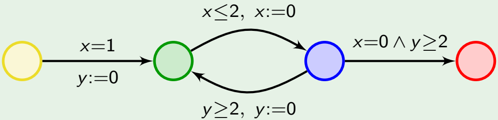
Example



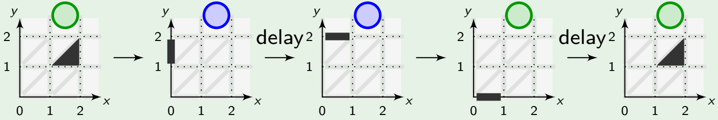
A region cycle:



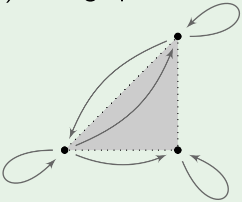
Example



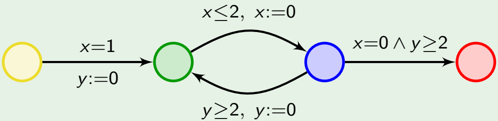
A region cycle:



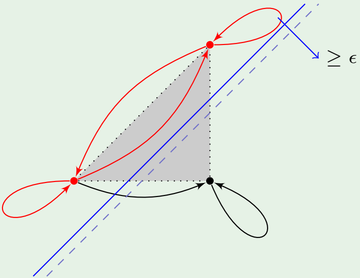
The corresponding (folded) orbit graph:



Example



The cycle is not forgetful (that is, not strongly connected), Perturbator can enforce convergence:



Outline

1. Introduction

2. Robust model-checking and implementation
   - Parameterized enlarged semantics
   - Automatic generation of an implementation
   - Implementation by shrinking

3. Robust realisability
   - Excess semantics
   - Conservative semantics

4. Conclusion

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.
- We have shown several frameworks for robustness that can be used to ensure correctness in the real-world..

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.
- We have shown several frameworks for robustness that can be used to ensure correctness in the real-world..

- Extension of these works to richer models seems unfortunately hard [BMS13]
- A quantitative approach to robustness: Perturbator plays randomly
- Symbolic algorithms?

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.
- We have shown several frameworks for robustness that can be used to ensure correctness in the real-world..

- Extension of these works to richer models seems unfortunately hard [BMS13]
- A quantitative approach to robustness: Perturbator plays randomly
- Symbolic algorithms?

- This list of possible approaches is not exhaustive:
  - tube acceptance [GHJ97]
  - sampling approach [KP05,BLM+11]
  - probabilistic approach [BBB+08,BBJM12]
  - ...