Robustness issues in timed models

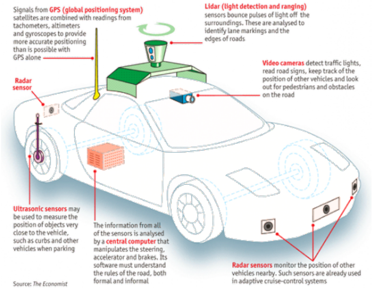
Nicolas Markey

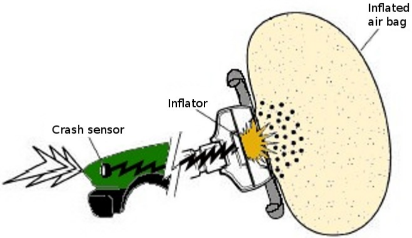
LSV, CNRS & ENS Cachan, France

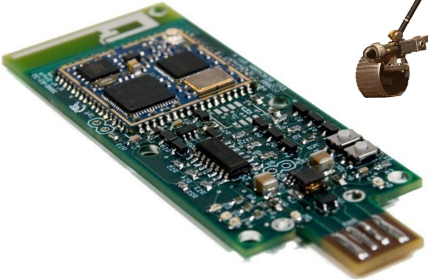
(based on joint works with Patricia Bouyer, Erwin Fang, Pierre-Alain Reynier, Ocan Sankur)
(also starring Martin De Wulf, Laurent Doyen, Jean-François Raskin)

QAPL'14 – Grenoble, France

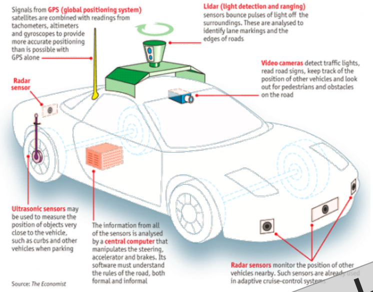
Modelling real-time systems

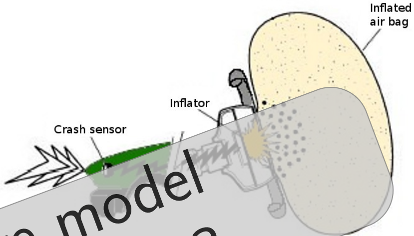


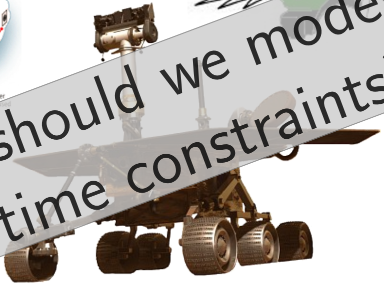


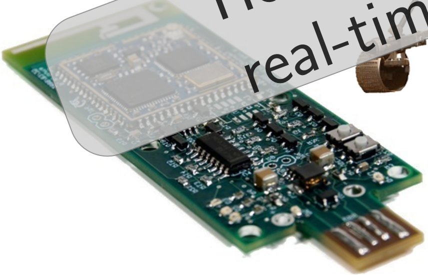


Modelling real-time systems





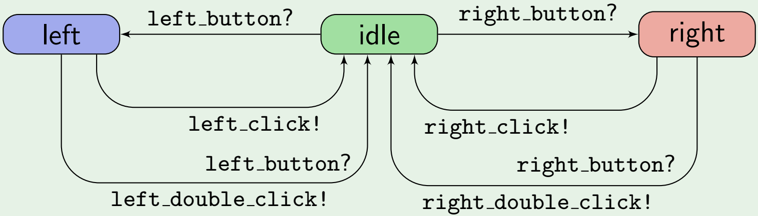




How should we model real-time constraints?

Reasoning about real-time systems

Example (A computer mouse)



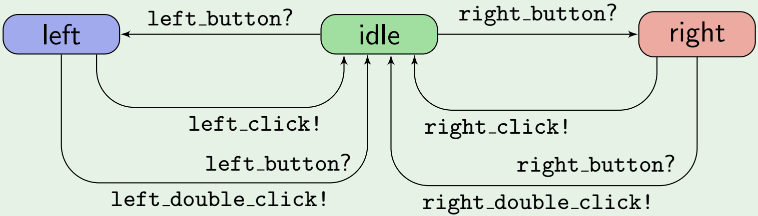
Reasoning about real-time systems

Timed automata [AD90]

A timed automaton is made of

- a transition system,

Example (A computer mouse)



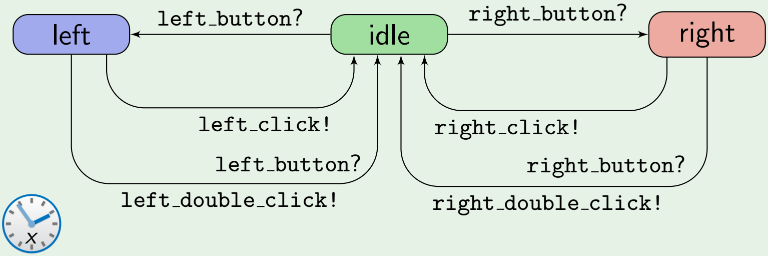
Reasoning about real-time systems

Timed automata [AD90]

A timed automaton is made of

- a transition system,
- a set of clocks,

Example (A computer mouse)



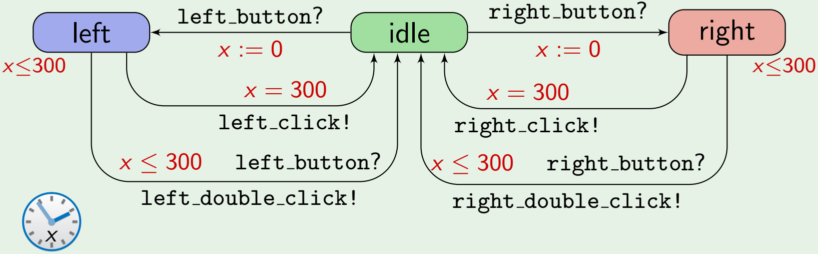
Reasoning about real-time systems

Timed automata [AD90]

A timed automaton is made of

- a transition system,
- a set of clocks,
- timing constraints on states and transitions.

Example (A computer mouse)



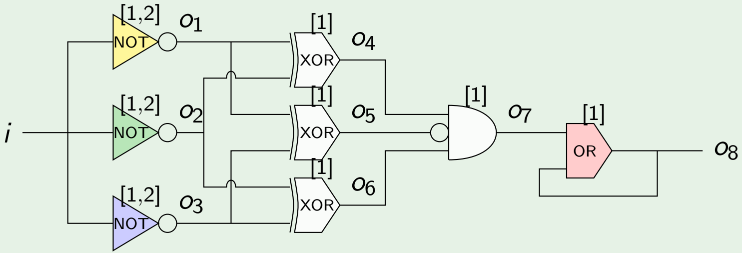
Discrete-time semantics

...because computers are digital!

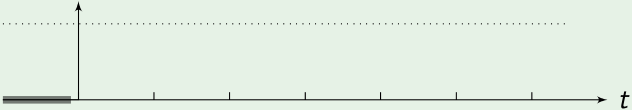
Discrete-time semantics

...because computers are digital!

Example ([Alur91])



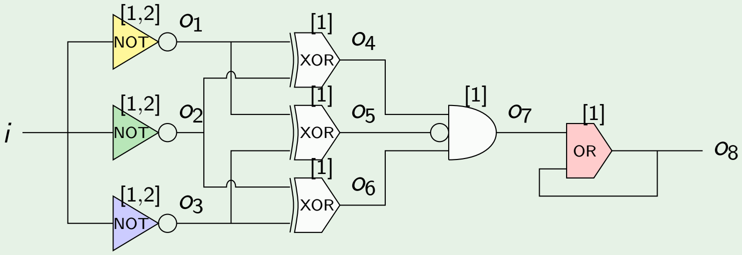
- under discrete-time, the output never changes:



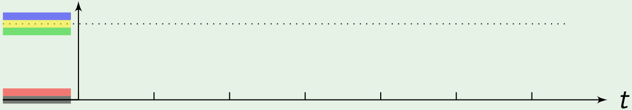
Discrete-time semantics

...because computers are digital!

Example ([Alur91])



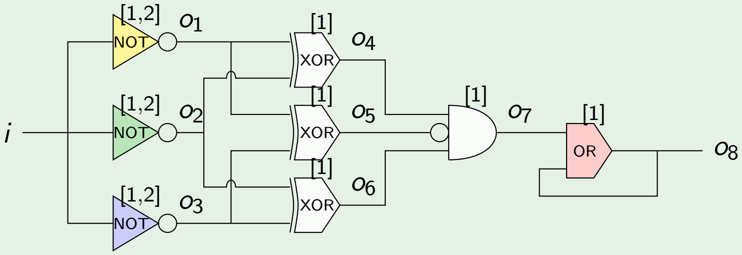
- under discrete-time, the output never changes:



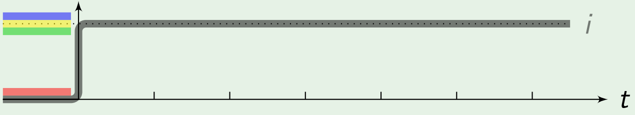
Discrete-time semantics

...because computers are digital!

Example ([Alur91])



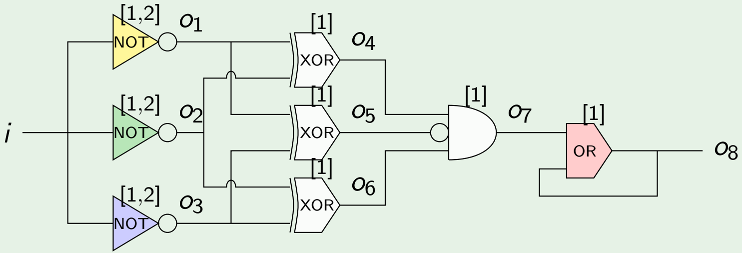
- under discrete-time, the output never changes:



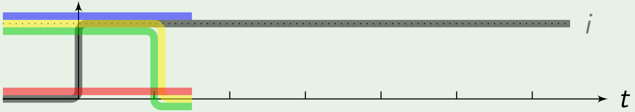
Discrete-time semantics

...because computers are digital!

Example ([Alur91])



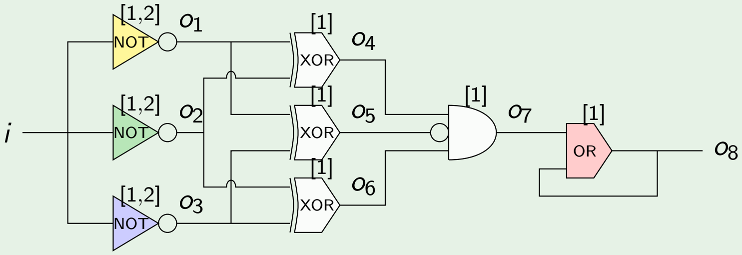
- under discrete-time, the output never changes:



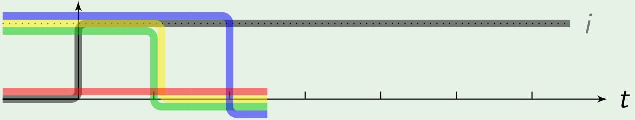
Discrete-time semantics

...because computers are digital!

Example ([Alur91])



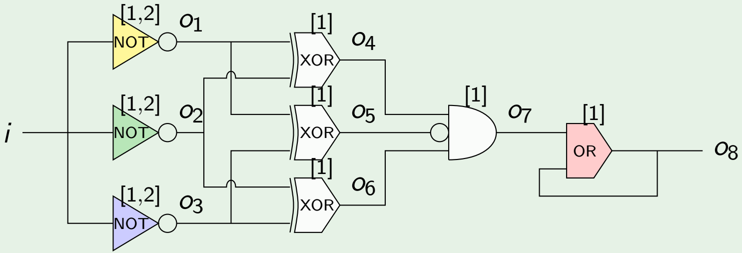
- under discrete-time, the output never changes:



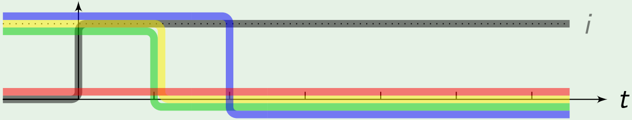
Discrete-time semantics

...because computers are digital!

Example ([Alur91])



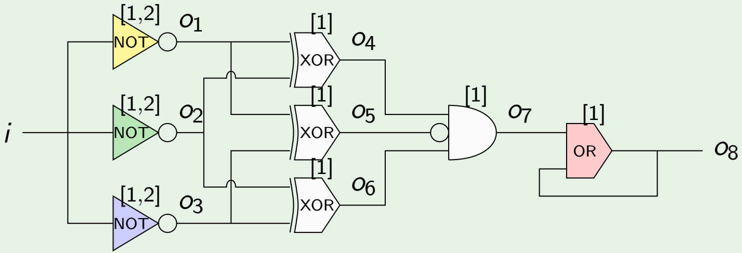
- under discrete-time, the output never changes:



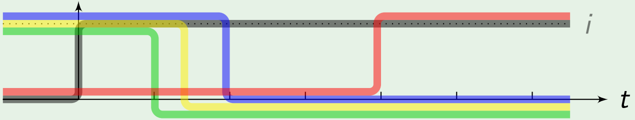
Discrete-time semantics

...because computers are digital!

Example ([Alur91])



- under continuous-time, the output can change to 1:



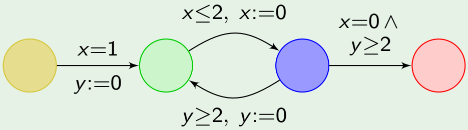
Continuous-time semantics

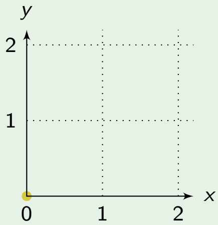
...real-time models for real-time systems!

Continuous-time semantics

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

Example

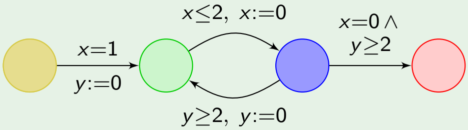


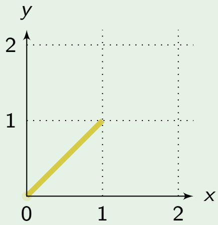


Continuous-time semantics

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

Example

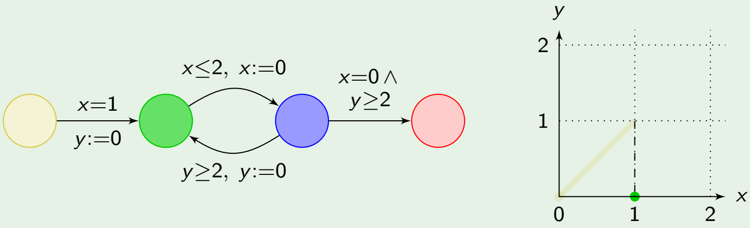




Continuous-time semantics

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

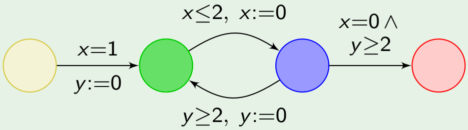
Example

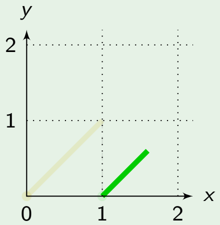


Continuous-time semantics

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

Example

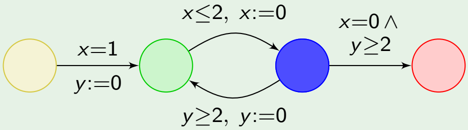


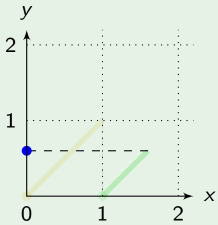


Continuous-time semantics

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

Example

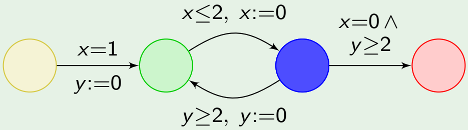


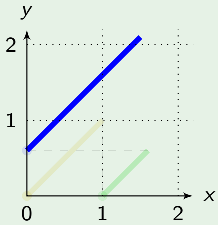


Continuous-time semantics

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

Example

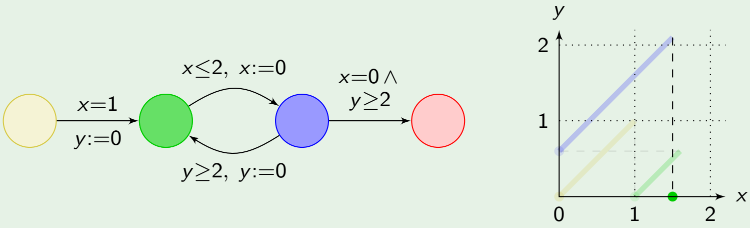




Continuous-time semantics

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

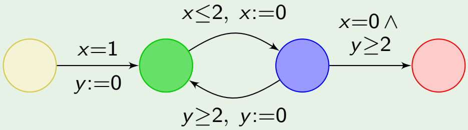
Example

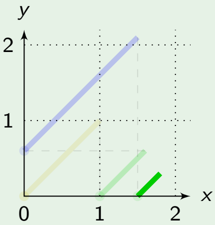


Continuous-time semantics

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

Example

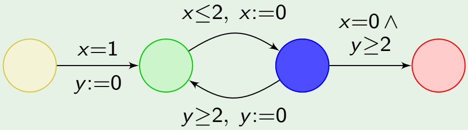


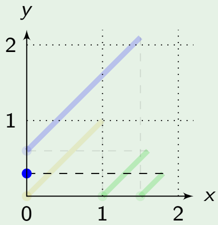


Continuous-time semantics

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

Example

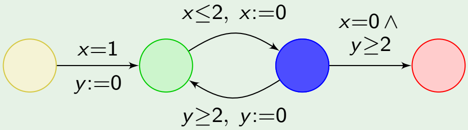


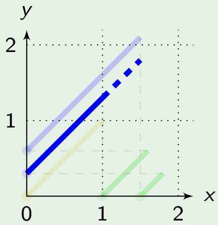


Continuous-time semantics

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

Example

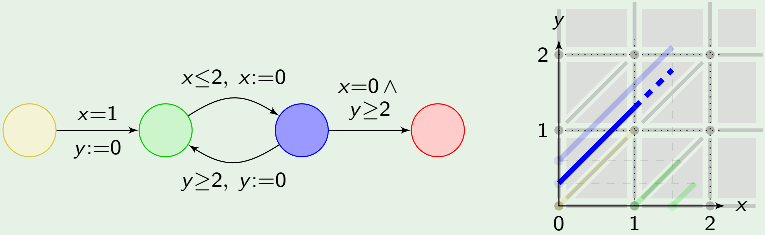




Continuous-time semantics

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

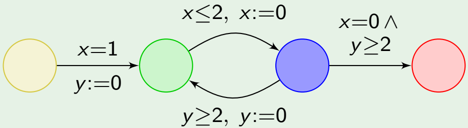
Example

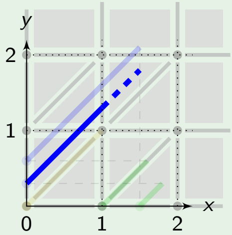


Continuous-time semantics

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

Example

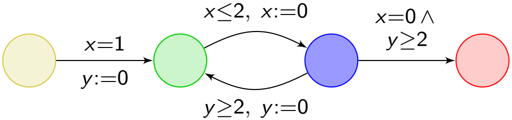




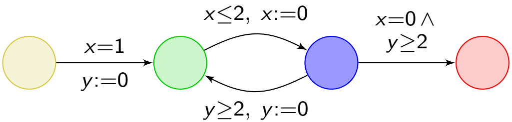
Theorem ([AD90,ACD93, ...])

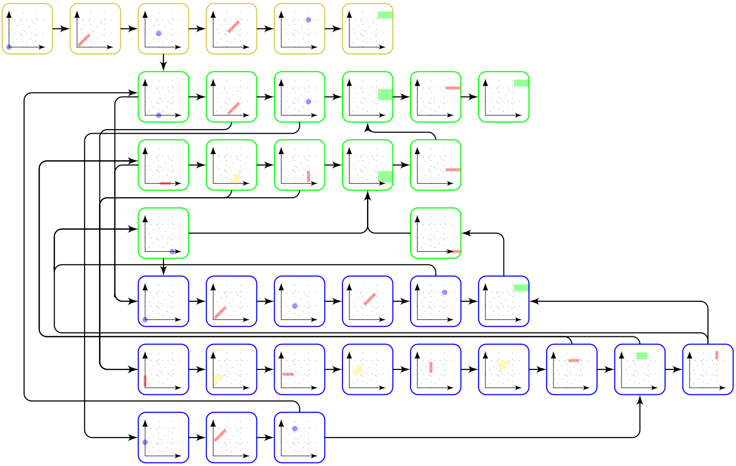
Reachability in timed automata is decidable (as well as many other important properties).

Regions and zones



Regions and zones



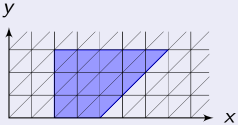


Regions and zones

Zones

Zones are a coarser abstraction:

$$(x \geq 2) \wedge (0 \leq y \leq 3) \wedge (x - y \leq 4)$$

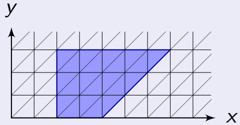


Regions and zones

Zones

Zones are a coarser abstraction:

$$(x \geq 2) \wedge (0 \leq y \leq 3) \wedge (x - y \leq 4)$$

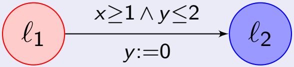


Representation as DBM:

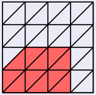
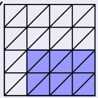
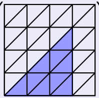
$$\begin{array}{c} \\ 0 \\ x \\ y \end{array}\begin{array}{c} \begin{array}{ccc} 0 & x & y \end{array} \\ \begin{pmatrix} 0 & -2 & 0 \\ +\infty & 0 & 4 \\ 3 & +\infty & 0 \end{pmatrix} \end{array} \equiv \begin{array}{c} \\ 0 \\ x \\ y \end{array}\begin{array}{c} \begin{array}{ccc} 0 & x & y \end{array} \\ \begin{pmatrix} 0 & -2 & 0 \\ 7 & 0 & 4 \\ 3 & 1 & 0 \end{pmatrix} \end{array}$$

Regions and zones

Zones

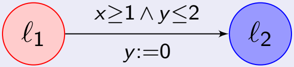


The predecessors of $(\ell_2, x \leq 3 \wedge y - x \leq 0)$ are computed as

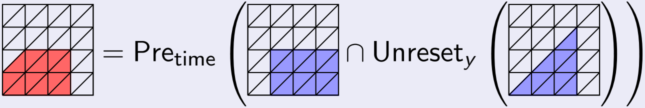
 $= \text{Pre}_{\text{time}} \Big($  $\cap\ \text{Unreset}_y \big($  $\big)\Big)$

Regions and zones

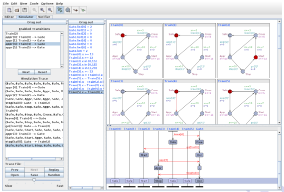
Zones



The predecessors of $(\ell_2, x \leq 3 \wedge y - x \leq 0)$ are computed as

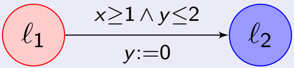


$\rightsquigarrow$ efficient implementations

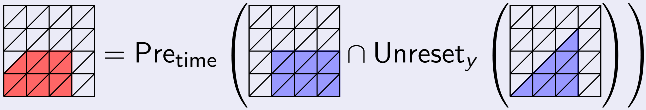


Regions and zones

Zones



The predecessors of $(\ell_2, x \leq 3 \wedge y - x \leq 0)$ are computed as



- $\rightsquigarrow$ efficient implementations
- $\rightsquigarrow$ successful applications

BANG & OLUFSEN

Outline of the talk

1. Discrete time vs. dense time
2. From models to implementations
3. Checking robust safety
   - Enlarging clock constraints
   - Shrinking clock constraints
4. Checking robust controllability
   - Parametrized perturbations
   - Permissive strategies
5. Conclusions and future works

Outline of the talk

From models to implementations

Example: Patriot anti-ballistic-missile failure

25 February 1991, during Gulf war.

28 soldiers died.

From models to implementations

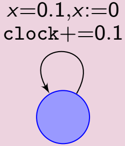
Example: Patriot anti-ballistic-missile failure

25 February 1991, during Gulf war.

28 soldiers died.

Problem: clock drift

Internal clock incremented by 1/10 every 1/10 s.



From models to implementations

Example: Patriot anti-ballistic-missile failure

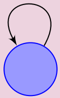
25 February 1991, during Gulf war.
28 soldiers died.

Problem: clock drift

Internal clock incremented by 1/10 every 1/10 s.

Clock stored in 24-bit register:

$$\frac{1}{10} - \left\langle \frac{1}{10} \right\rangle_{24\text{ bit}} \simeq 10^{-7}$$



After 100 hours, the total drift was 0.34 seconds.
The incoming missile could not be destroyed.

From models to implementations

the continuous-time semantics is a mathematical idealization

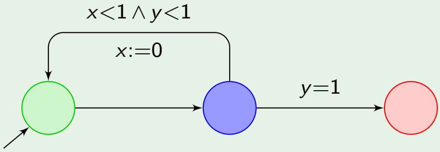
- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

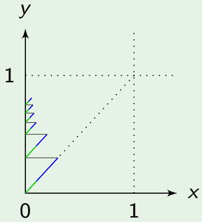
From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

Example (Zeno behaviors)



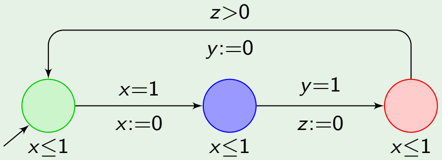


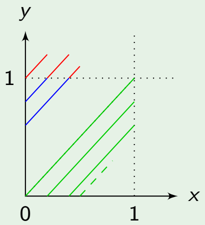
From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

Example (Converge phenomena)



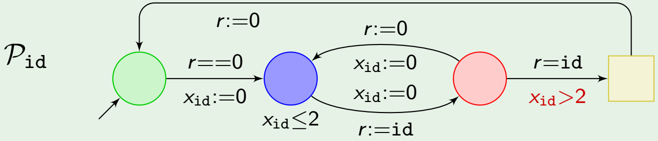


From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

Example (Strict timing constraints)



When P_1 and 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 $x_{\mathtt{id}} \geq 2$.

From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

Parametrized semantics

- parametrized discrete-time semantics:

  Does there exists a time step δ (*sampling rate*) under which the system behaves correctly?

From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

Parametrized semantics

- parametrized discrete-time semantics:

  Does there exists a time step δ (*sampling rate*) under which the system behaves correctly?

  $\rightsquigarrow$ reachability is undecidable [CHR02]

  $\rightsquigarrow$ untimed-language inclusion is decidable [AKY10]

From models to implementations

the continuous-time semantics is a mathematical idealization

- it assumes zero-delay transitions;
- it assumes infinite precision of the clocks;
- it assumes immediate communication between systems.

Parametrized semantics

- parametrized discrete-time semantics:

  Does there exists a time step δ (*sampling rate*) under which the system behaves correctly?

  $\rightsquigarrow$ reachability is undecidable [CHR02]

  $\rightsquigarrow$ untimed-language inclusion is decidable [AKY10]

- parametrized continuous-time semantics:

  Does the system behave correctly under continuous-time semantics with imprecisions up to some δ?

Outline of the talk

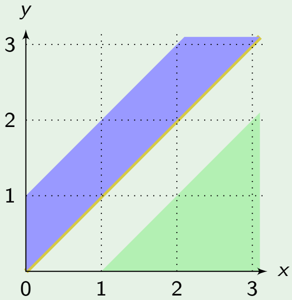
Enlarged semantics for timed automata

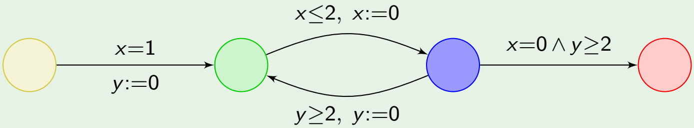
a transition can be taken at any time in $[t - \delta; t + \delta]$.

Enlarged semantics for timed automata

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

Example

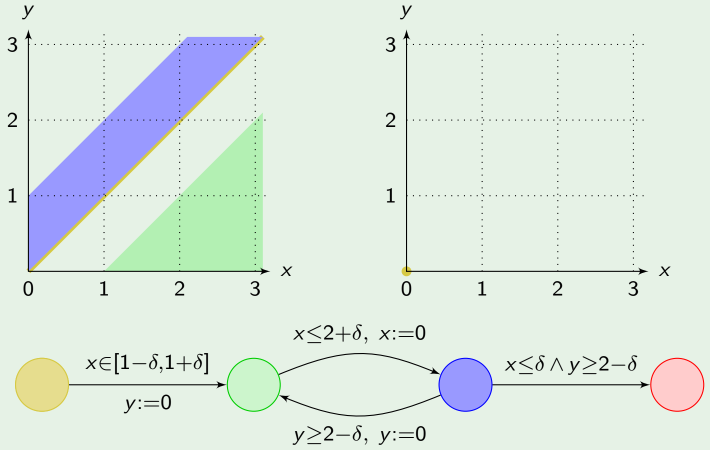




Enlarged semantics for timed automata

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

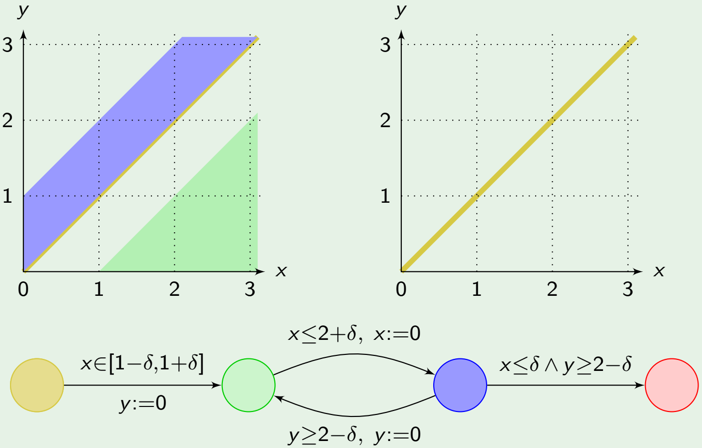
Example



Enlarged semantics for timed automata

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

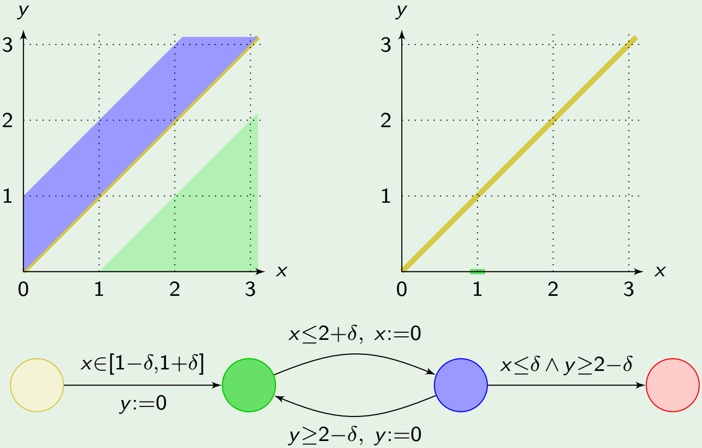
Example



Enlarged semantics for timed automata

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

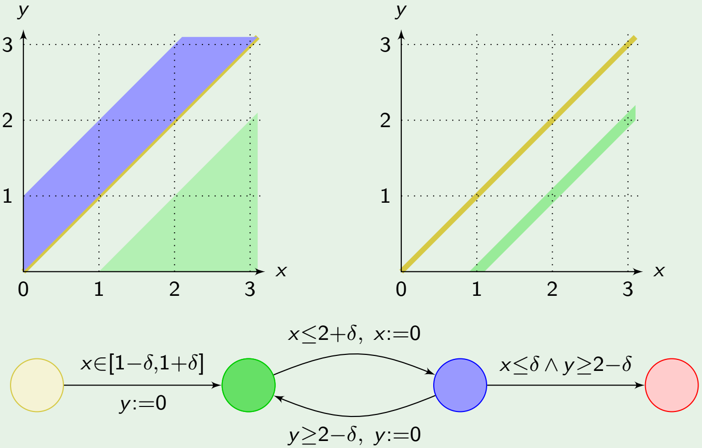
Example



Enlarged semantics for timed automata

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

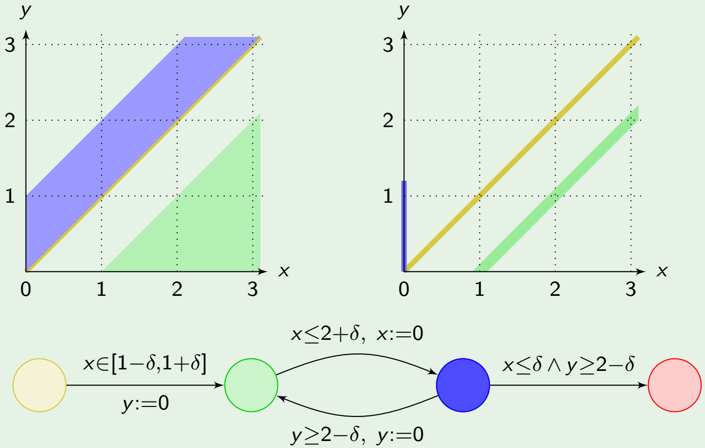
Example



Enlarged semantics for timed automata

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

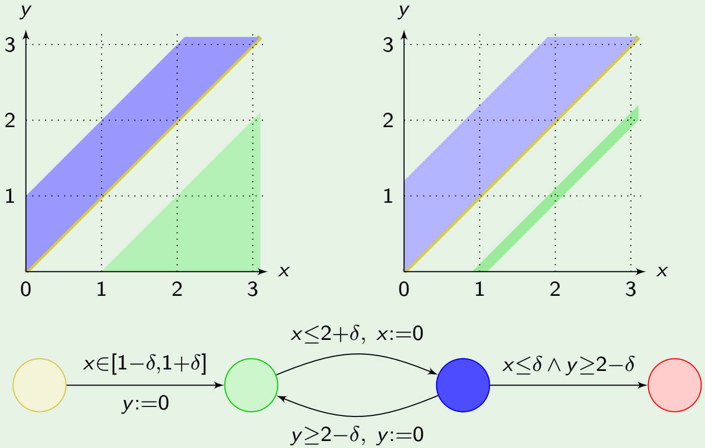
Example



Enlarged semantics for timed automata

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

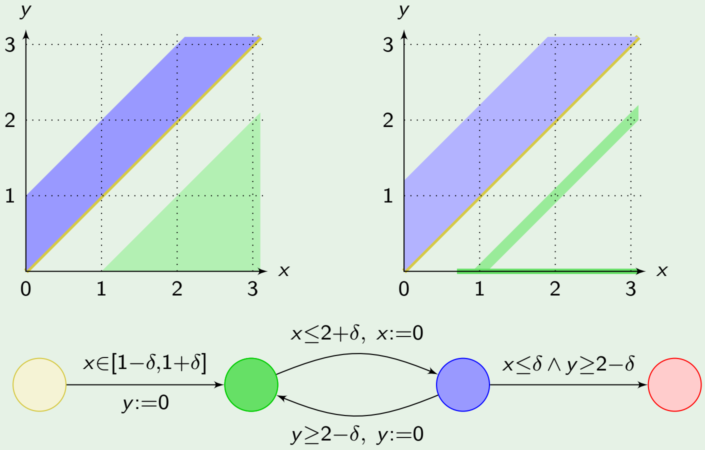
Example



Enlarged semantics for timed automata

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

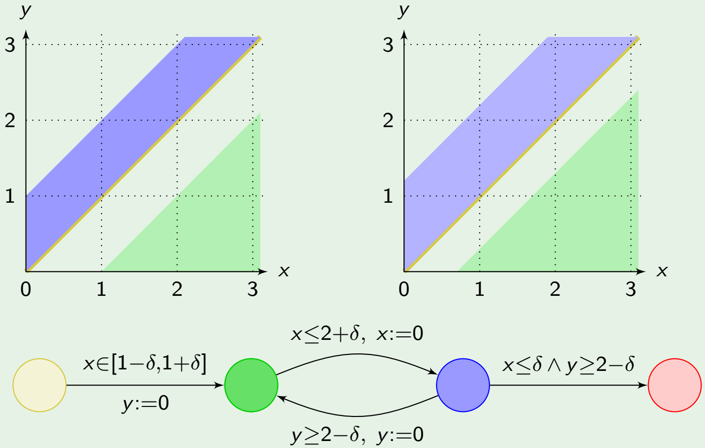
Example



Enlarged semantics for timed automata

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

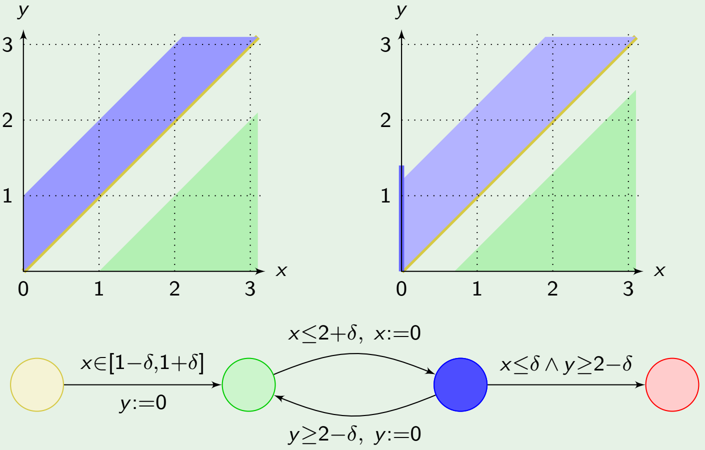
Example



Enlarged semantics for timed automata

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

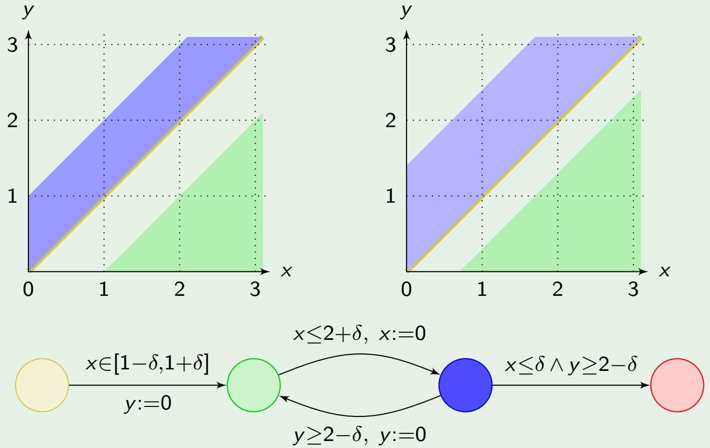
Example



Enlarged semantics for timed automata

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

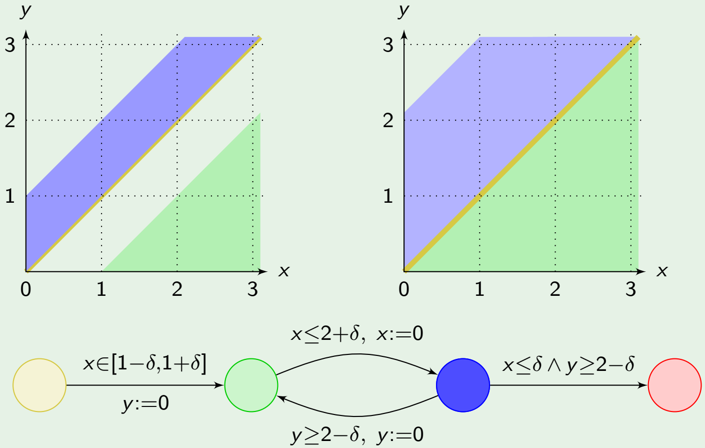
Example



Enlarged semantics for timed automata

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

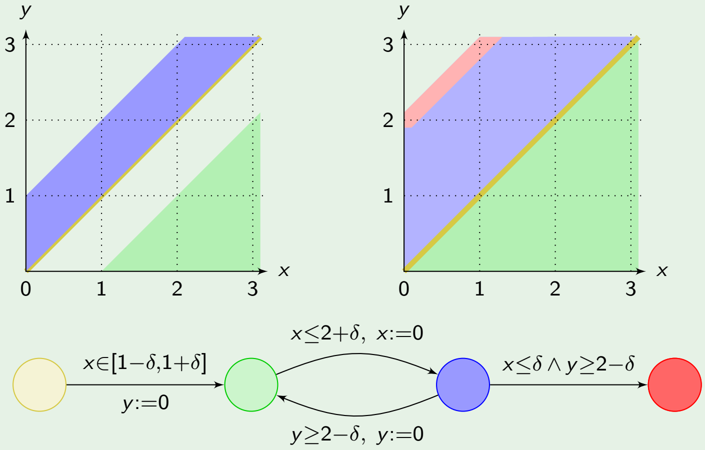
Example



Enlarged semantics for timed automata

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

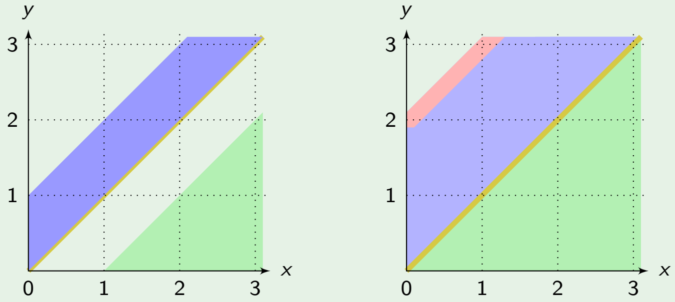
Example



Enlarged semantics for timed automata

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

Example



Theorem ([Pur98,DDMR04])

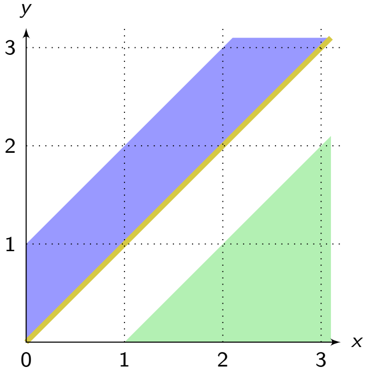
Parametrized robust safety is decidable.

Extended region automaton

For any location ℓ and any two regions r and r', if

- $\overline{r} \cap \overline{r'} \neq \varnothing$ and
- (ℓ, r') belongs to an SCC of $\mathcal{R}(\mathcal{A})$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$.

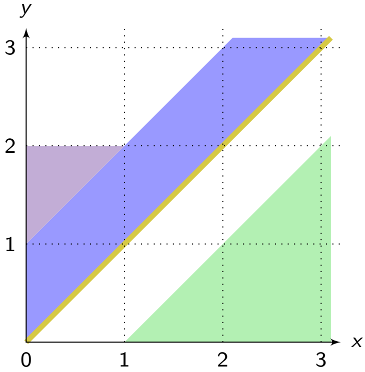


Extended region automaton

For any location ℓ and any two regions r and r', if

- $\overline{r} \cap \overline{r'} \neq \varnothing$ and
- (ℓ, r') belongs to an SCC of $\mathcal{R}(\mathcal{A})$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$.

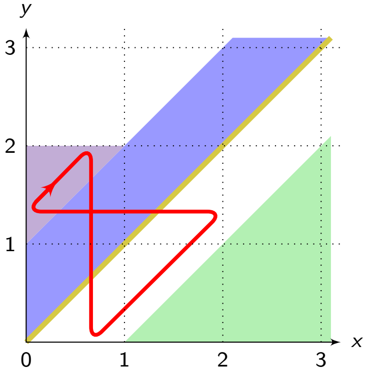


Extended region automaton

For any location ℓ and any two regions r and r', if

- $\overline{r} \cap \overline{r'} \neq \varnothing$ and
- (ℓ, r') belongs to an SCC of $\mathcal{R}(\mathcal{A})$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$.

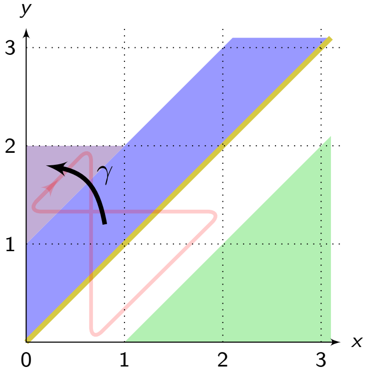


Extended region automaton

For any location ℓ and any two regions r and r', if

- $\overline{r} \cap \overline{r'} \neq \varnothing$ and
- (ℓ, r') belongs to an SCC of $\mathcal{R}(\mathcal{A})$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$.

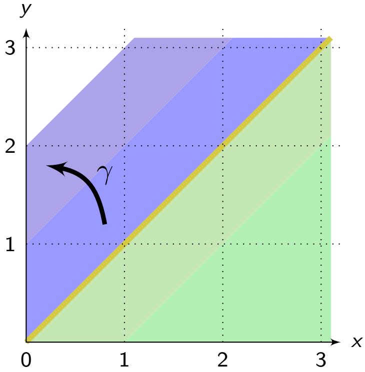


Extended region automaton

For any location ℓ and any two regions r and r', if

- $\overline{r} \cap \overline{r'} \neq \varnothing$ and
- (ℓ, r') belongs to an SCC of $\mathcal{R}(\mathcal{A})$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$.

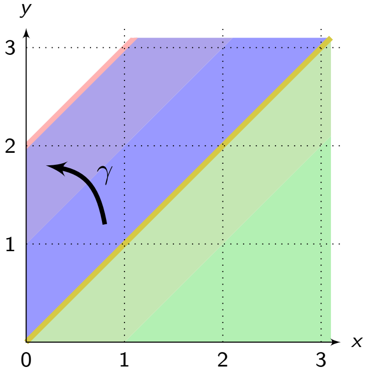


Extended region automaton

For any location ℓ and any two regions r and r', if

- $\overline{r} \cap \overline{r'} \neq \varnothing$ and
- (ℓ, r') belongs to an SCC of $\mathcal{R}(\mathcal{A})$,

then we add a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$.



Shrinking timing constraints

Counteracting guard enlargement

Shrinking turns constraints $[a, b]$ into $[a + \delta, b - \delta]$.

In particular, punctual constraints become empty.

Shrinking timing constraints

Counteracting guard enlargement

Shrinking turns constraints $[a, b]$ into $[a + \delta, b - \delta]$.

In particular, punctual constraints become empty.

Definition

A timed automaton is shrinkable if, for some $\delta > 0$, its shrunk automaton (time-abstract) simulates the original automaton.

Theorem ([SBM11])

Shrinkability is decidable in EXPTIME.

Shrinking timing constraints

Counteracting guard enlargement

Shrinking turns constraints $[a, b]$ into $[a + \delta, b - \delta]$.

In particular, punctual constraints become empty.

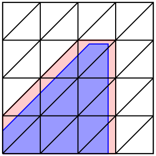
Definition

A timed automaton is shrinkable if, for some $\delta > 0$, its shrunk automaton (time-abstract) simulates the original automaton.

Theorem ([SBM11])

Shrinkability is decidable in EXPTIME.

Main tools: parametrized shrunk DBMs
max-plus fixpoint equations



Shrinking timing constraints

Counteracting guard enlargement

Shrinking turns constraints $[a, b]$ into $[a+\delta, b-\delta]$.

In particular, punctual constraints become empty.

Definition

A timed automaton is shrinkable if, for some $\delta > 0$, its shrunk automaton (time-abstract) simulates the original automaton.

Theorem ([SBM11])

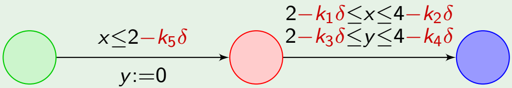
Shrinkability is decidable in EXPTIME.

$\rightsquigarrow$ prototype tool:

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

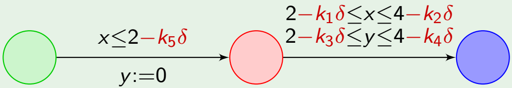
Shrinking timing constraints

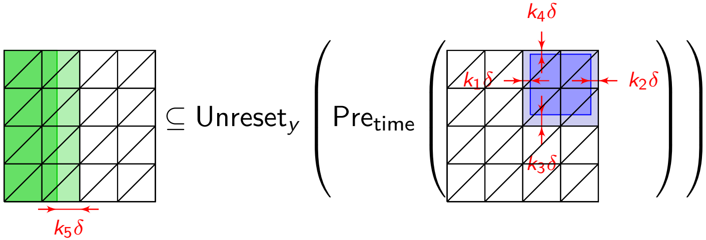
Example



Shrinking timing constraints

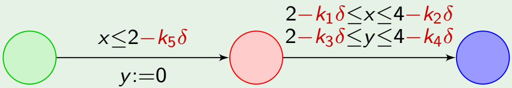
Example

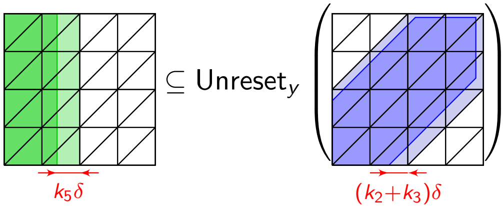




Shrinking timing constraints

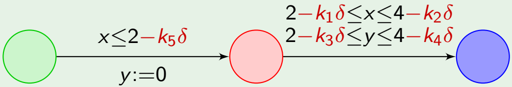
Example

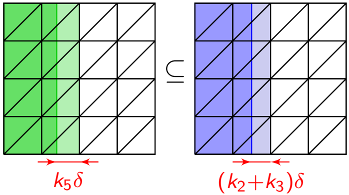




Shrinking timing constraints

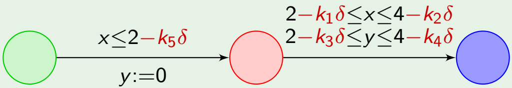
Example

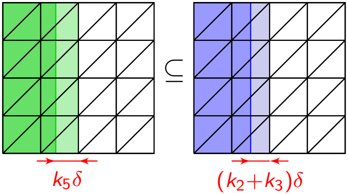




Shrinking timing constraints

Example





$$\rightsquigarrow \qquad k_5 = \max(k_5, k_2 + k_3)$$

Outline of the talk

Game-based approach to robustness

Solving robust reachability

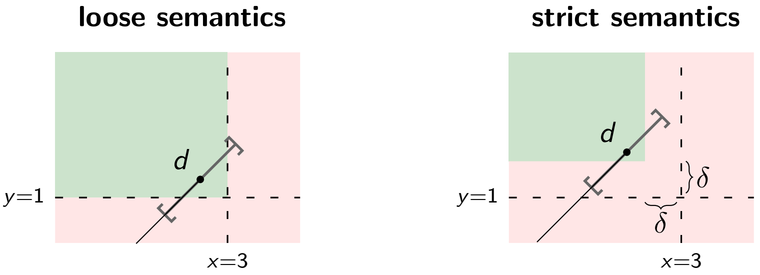
- Player 1 proposes a delay d and a transition t;
- transition t is taken after some delay in $[d-\delta, d+\delta]$ chosen by Player 2.

Game-based approach to robustness

Solving robust reachability

- Player 1 proposes a delay d and a transition t;
- transition t is taken after some delay in $[d-\delta, d+\delta]$ chosen by Player 2.

Consider a transition with guard $x \leq 3 \wedge y \geq 1$:



Game-based approach to robustness

Solving robust reachability

- Player 1 proposes a delay d and a transition t;
- transition t is taken after some delay in $[d-\delta, d+\delta]$ chosen by Player 2.

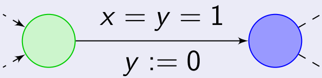
Theorem ([BMS12,SBMR13])

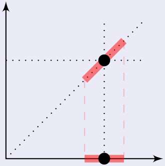
Robust reachability is EXPTIME*-complete in the loose semantics.*

Robust reachability and repeated reachability are PSPACE*-complete in the strict semantics.*

Shrunk DBMs for the loose semantics

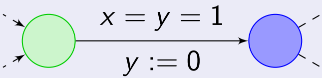
Extend the region automaton into a 2-player turn-based game

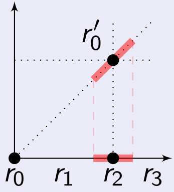


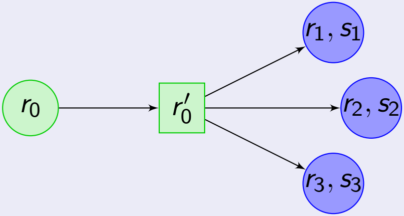


Shrunk DBMs for the loose semantics

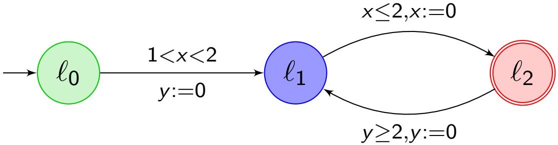
Extend the region automaton into a 2-player turn-based game

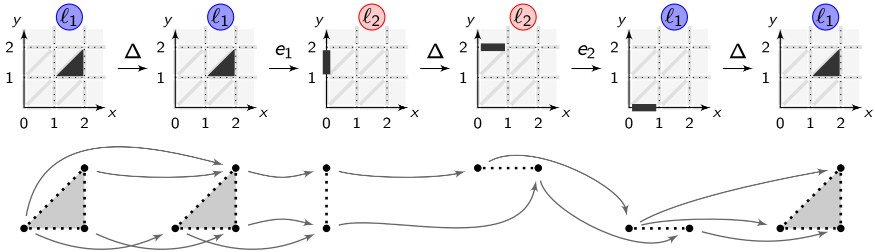




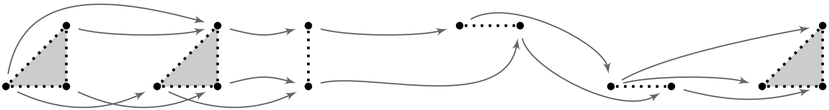


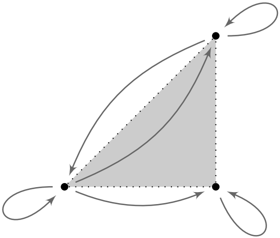
Orbit graphs for the strict semantics



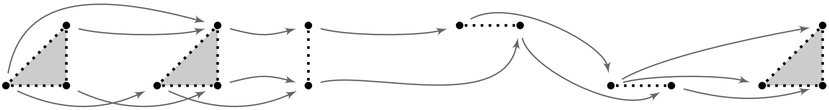


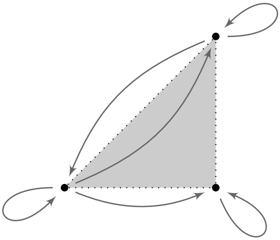
Orbit graphs for the strict semantics





Orbit graphs for the strict semantics



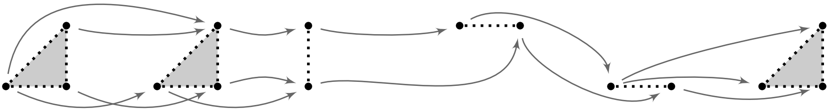


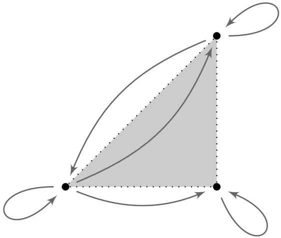
Definition

A cycle π is forgetful if its orbit graph is strongly connected.

A cycle π is aperiodic if π^k is forgetful, for all k.

Orbit graphs for the strict semantics





Definition

A cycle π is forgetful if its orbit graph is strongly connected.

A cycle π is aperiodic if π^k is forgetful, for all k.

Theorem

The automaton is robustly controllable if, and only if, it has a reachable aperiodic cycle.

Synthesizing permissive strategies

Permissive strategies

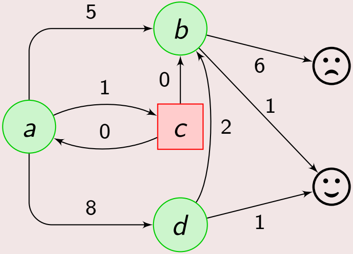
Permissive strategies can propose several moves rather than a single one.

Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the untimed setting... [BDMR09, BMOU11]

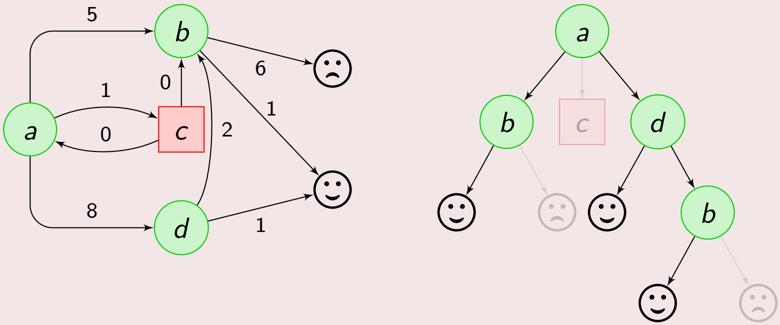


Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the untimed setting... [BDMR09, BMOU11]

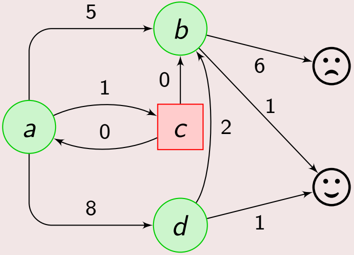


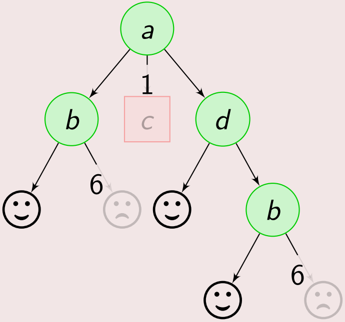
Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the untimed setting... [BDMR09, BMOU11]





Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the timed setting...

Permissive strategies propose intervals of delays.

Our setting:

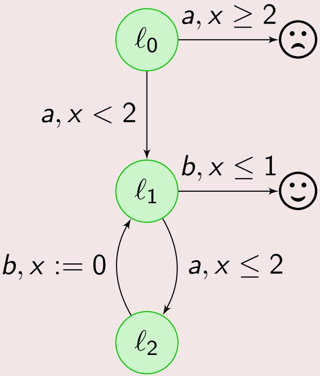
the penalty assigned to interval $[a, b]$ is $1/(b - a)$.

Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the timed setting...

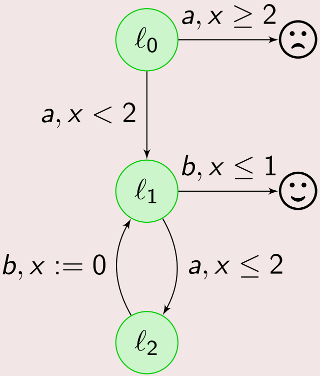


Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the timed setting...



Possible (memoryless) strategy:

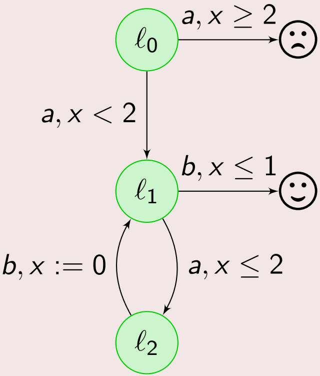
- in ℓ_0, play $(a, [0, 2))$;
- in ℓ_1:
  - if $x \leq 1$, play $(b, [0, 1 - x])$;
  - otherwise, play $(a, [0, 2 - x])$;
- in ℓ_2, play $(b, [0, +\infty))$

Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the timed setting...



Possible (memoryless) strategy:

- in ℓ_0, play $(a, [0, 2))$;
- in ℓ_1:
  - if $x \leq 1$, play $(b, [0, 1 - x])$;
  - otherwise, play $(a, [0, 2 - x])$;
- in ℓ_2, play $(b, [0, +\infty))$

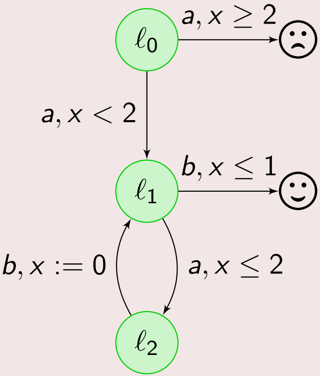
$\rightsquigarrow$ **penalty** $= +\infty$

Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the timed setting...



Possible (memoryless) strategy:

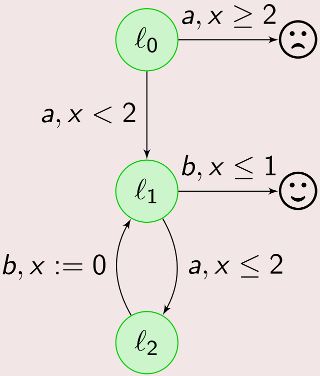
- in ℓ_0, play $(a, [0, 1])$;
- in ℓ_1:
  - if $x = 0$, play $(b, [0, 1])$;
  - otherwise, play $(a, [0, 2 - x])$;
- in ℓ_2, play $(b, [0, +\infty))$

Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the timed setting...



Possible (memoryless) strategy:

- in ℓ_0, play $(a, [0, 1])$;
- in ℓ_1:
  - if $x = 0$, play $(b, [0, 1])$;
  - otherwise, play $(a, [0, 2 - x])$;
- in ℓ_2, play $(b, [0, +\infty))$

$\rightsquigarrow$ **penalty = 1**

Synthesizing permissive strategies

Permissive strategies

Permissive strategies can propose several moves rather than a single one.

In the timed setting...

Theorem

For one-clock timed games:

- *Memoryless optimal-penalty strategies exist.*
- *They can be computed in polynomial time.*

Outline of the talk

1. Discrete time vs. dense time
2. From models to implementations
3. Checking robust safety
   - Enlarging clock constraints
   - Shrinking clock constraints
4. Checking robust controllability
   - Parametrized perturbations
   - Permissive strategies
5. Conclusions and future works

Conclusion and challenges

Conclusions

Robustness issues identified long ago...

Several attempts, but no satisfactory solution yet!

Conclusion and challenges

Conclusions

Robustness issues identified long ago...

Several attempts, but no satisfactory solution yet!

Challenges and open questions

- symbolic algorithms;
- measuring robustness, using distances between automata;
  $\rightsquigarrow$ link between "syntactic distance" and "semantic distance"
- probabilistic approach to robustness;
  $\rightsquigarrow$ evaluate expected time before a new state is visited.
- investigate robustness in weighted timed automata;
  $\rightsquigarrow$ energy constraints;
  $\rightsquigarrow$ imprecision on cost rates;
- synthesis of robust strategies.