Robustness for Timed Automata

Claus Thrane
crt@cs.aau.dk

MT-LAB/Aalborg University

Copenhagen October 24, 2011

This talk is based on the contribution from

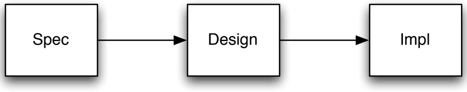
Bouyer, Larsen, Markey, Sankur, and Thrane.
Timed automata can always be made implementable.
In *Proceedings of CONCUR*, 2011.

Larsen, Legay, Traonouez, and Wasowski.
Robust Specification of Real-time Components.
In *Proceedings of FORMATS*, 2011.

Outline

From specification to design and implementation.

- Property & model languages (including relevant operations)
- Verification and refinement procedures

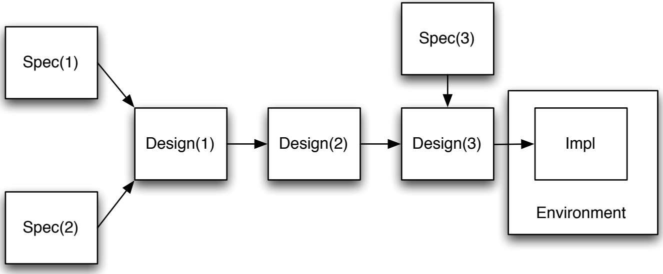


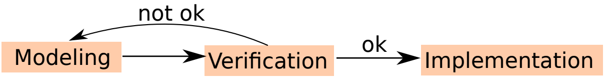
Big Design Up Front

From specification to design and implementation.

- Property & model languages (including relevant operations)
- Verification and refinement procedures

+ approximation of environmental information. Especially for real-time systems using off-the-self hardware.

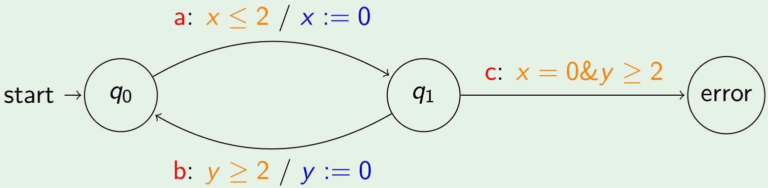




Timed Automata

- Finite automata + Clocks. [Alur and Dill 1994]
- Clocks grow continuously, all at the same rate. They are used to (de)activate the transitions of the automaton and can be reset when taking a transition.

Example $\mathcal{A}$



Timed I/O Automata assumes $\mathbf{Act} = \mathbf{Act}_i \oplus \mathbf{Act}_o$

Semantics

The semantics of TA

Given a TA $\mathcal{A}$, the **the semantics** of $\mathcal{A}$ is a (timed) transition system $[\![\mathcal{A}]\!]$ over *discrete actions* and $\mathbb{R}_{\geq 0}$.

A trace from $[\![\mathcal{A}]\!]$

$$(q_0, (x = 0, y = 0)) \xrightarrow{1.7} (q_0, (x = 1.7, y = 1.7)) \xrightarrow{a} (q_1, (x = 0, y = 1.7)) \xrightarrow{0.5} (q_1, (x = 0.5, y = 2.2)) \xrightarrow{b} (q_0, (x = 0.5, y = 0)) \ldots$$

The semantics of timed automata makes unrealistic assumptions:

- Systems have instant reaction time, $\xrightarrow{a} \xrightarrow{0.00001} \xrightarrow{b}$.
- clocks are infinitely precise. "$x \leq k$".

Semantics

The semantics of TA

Given a TA $\mathcal{A}$, the **the semantics** of $\mathcal{A}$ is a (timed) transition system $[\![\mathcal{A}]\!]$ over *discrete actions* and $\mathbb{R}_{\geq 0}$.

A trace from $[\![\mathcal{A}]\!]$

$$(q_0, (x = 0, y = 0)) \xrightarrow{1.7} (q_0, (x = 1.7, y = 1.7)) \xrightarrow{a} (q_1, (x = 0, y = 1.7)) \xrightarrow{0.5} (q_1, (x = 0.5, y = 2.2)) \xrightarrow{b} (q_0, (x = 0.5, y = 0)) \dots$$

The semantics of timed automata makes unrealistic assumptions:

- Systems have instant reaction time, $\xrightarrow{a}\xrightarrow{0.00001}\xrightarrow{b}$.
- clocks are infinitely precise. "$x \leq k$".

Semantics

The semantics of TA

Given a TA $\mathcal{A}$, the **the semantics** of $\mathcal{A}$ is a (timed) transition system $[\![\mathcal{A}]\!]$ over *discrete actions* and $\mathbb{R}_{\geq 0}$.

A trace from $[\![\mathcal{A}]\!]$

$$(q_0, (x = 0, y = 0)) \xrightarrow{1.7} (q_0, (x = 1.7, y = 1.7)) \xrightarrow{a} (q_1, (x = 0, y = 1.7)) \xrightarrow{0.5} (q_1, (x = 0.5, y = 2.2)) \xrightarrow{b} (q_0, (x = 0.5, y = 0)) \ldots$$

The semantics of timed automata makes unrealistic assumptions:

- Systems have instant reaction time, $\xrightarrow{a}\xrightarrow{0.00001}\xrightarrow{b}$.
- clocks are infinitely precise. "$x \leq k$".

Interpreting TAs as they would be executed!

Our environment may induce (minor) perturbations in behavior, since:

- Digital clock suffers from drift and finite precision.
- Digital hardware has finite execution speed.

Realistic semantics is considered: e.g. the *Almost*-ASAP [Raskin et.al.] or

Enlarged semantics: $[\![\mathcal{A}_\Delta]\!]$

For a TA $\mathcal{A}$, we relax all constraints into: $x \leq k+\Delta$ and $x \geq k-\Delta$. for arbitrarily small $\Delta > 0$.

in case of I/O invariants and output is enlarged and input is restricted.

Sampling semantics $[\![\mathcal{A}]\!]^{\frac{1}{k}}$

Project $[\![\mathcal{A}]\!]$ to $\frac{1}{k}\mathbb{N}$ for a given positive number k.

Interpreting TAs as they would be executed!

Our environment may induce (minor) perturbations in behavior, since:

- Digital clock suffers from drift and finite precision.
- Digital hardware has finite execution speed.

Realistic semantics is considered: e.g. the *Almost*-ASAP [Raskin et.al.] or

Enlarged semantics: $[\![\mathcal{A}_\Delta]\!]$

For a TA $\mathcal{A}$, we relax all constraints into: $x \leq k{+}\Delta$ and $x \geq k{-}\Delta$. for arbitrarily small $\Delta > 0$.

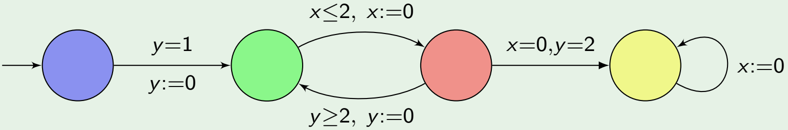
in case of I/O invariants and output is enlarged and input is restricted.

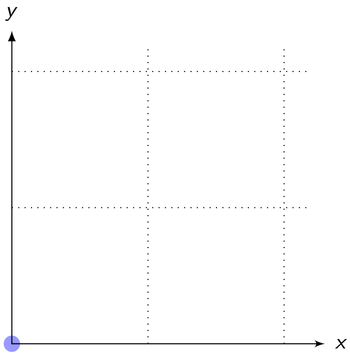
Sampling semantics $[\![\mathcal{A}]\!]^{\frac{1}{k}}$

Project $[\![\mathcal{A}]\!]$ to $\frac{1}{k}\mathbb{N}$ for a given positive number k.

Exact semantics versus Enlarged semantics

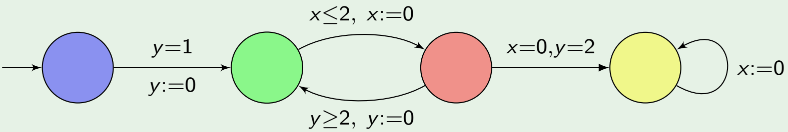
Example [Puri'98: Dynamical properties of timed automata]

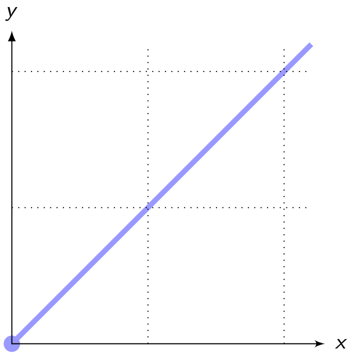




Exact semantics versus Enlarged semantics

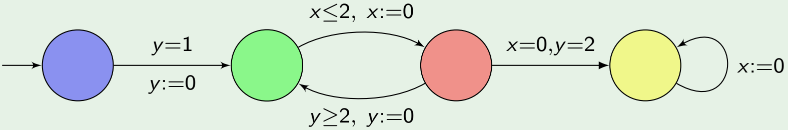
Example [Puri'98: Dynamical properties of timed automata]

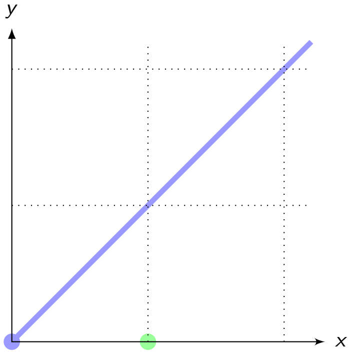




Exact semantics versus Enlarged semantics

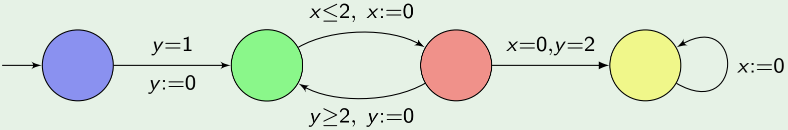
Example [Puri'98: Dynamical properties of timed automata]

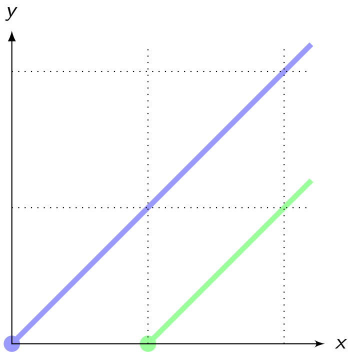




Exact semantics versus Enlarged semantics

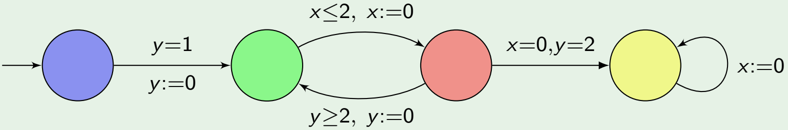
Example [Puri'98: Dynamical properties of timed automata]

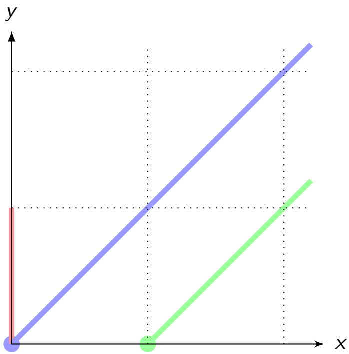




Exact semantics versus Enlarged semantics

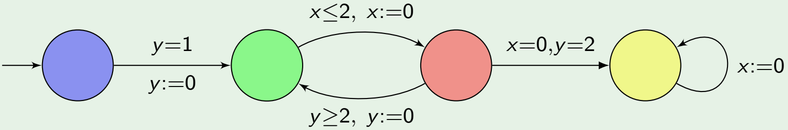
Example [Puri'98: Dynamical properties of timed automata]

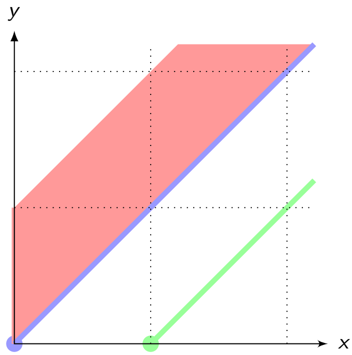




Exact semantics versus Enlarged semantics

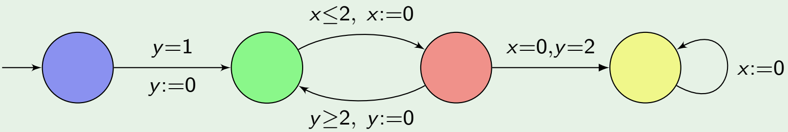
Example [Puri'98: Dynamical properties of timed automata]

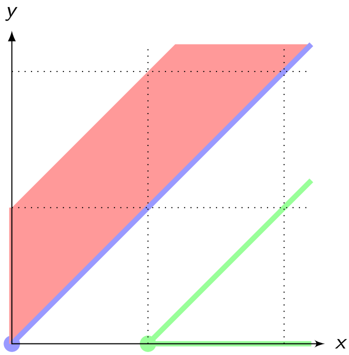




Exact semantics versus Enlarged semantics

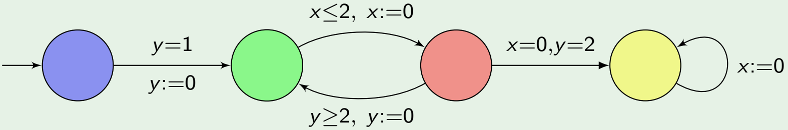
Example [Puri'98: Dynamical properties of timed automata]

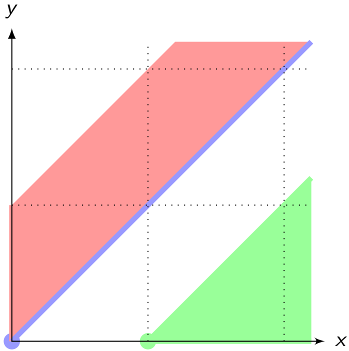




Exact semantics versus Enlarged semantics

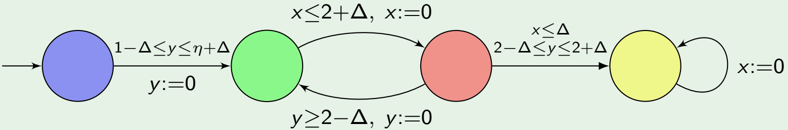
Example [Puri'98: Dynamical properties of timed automata]

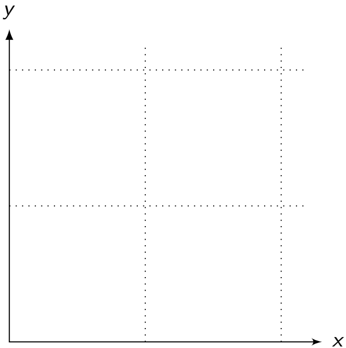




Exact semantics versus Enlarged semantics

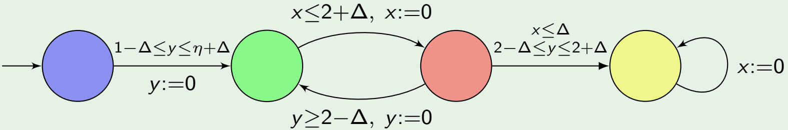
Example [Puri'98: Dynamical properties of timed automata]

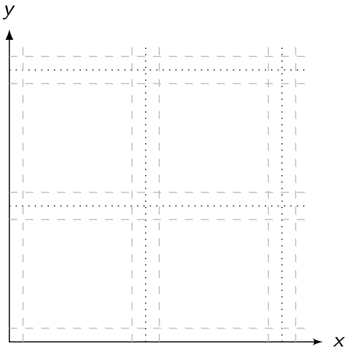




Exact semantics versus Enlarged semantics

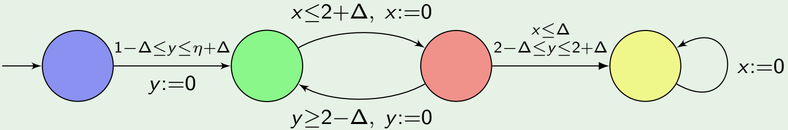
Example [Puri'98: Dynamical properties of timed automata]

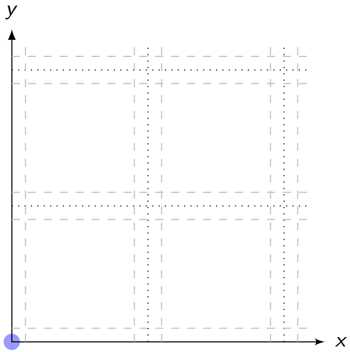




Exact semantics versus Enlarged semantics

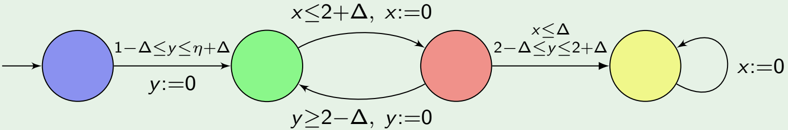
Example [Puri'98: Dynamical properties of timed automata]

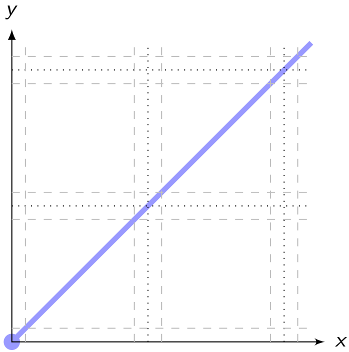




Exact semantics versus Enlarged semantics

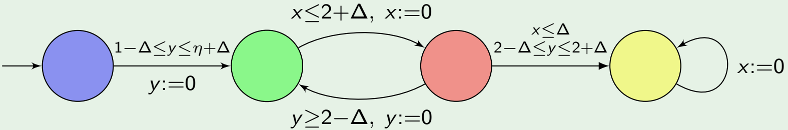
Example [Puri'98: Dynamical properties of timed automata]

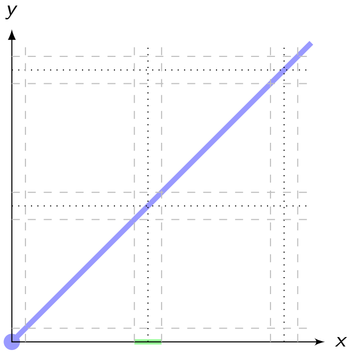




Exact semantics versus Enlarged semantics

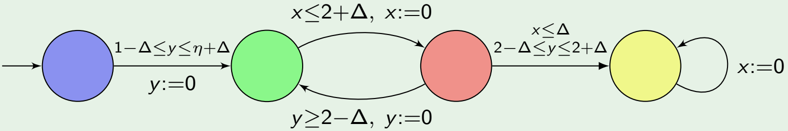
Example [Puri'98: Dynamical properties of timed automata]

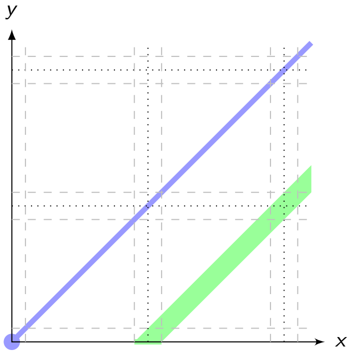




Exact semantics versus Enlarged semantics

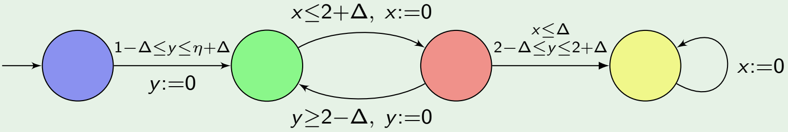
Example [Puri'98: Dynamical properties of timed automata]

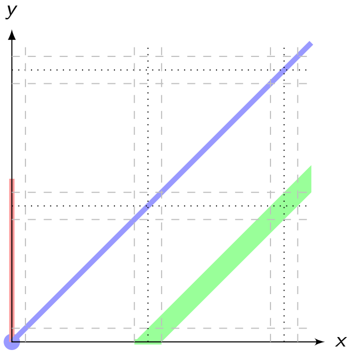




Exact semantics versus Enlarged semantics

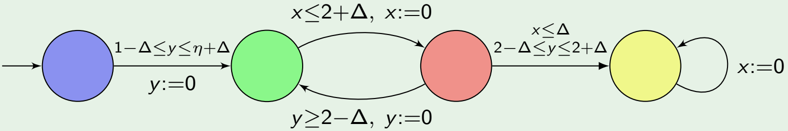
Example [Puri'98: Dynamical properties of timed automata]

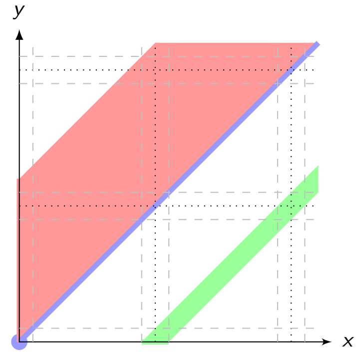




Exact semantics versus Enlarged semantics

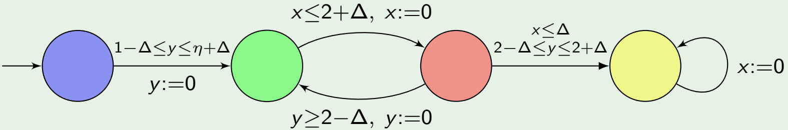
Example [Puri'98: Dynamical properties of timed automata]

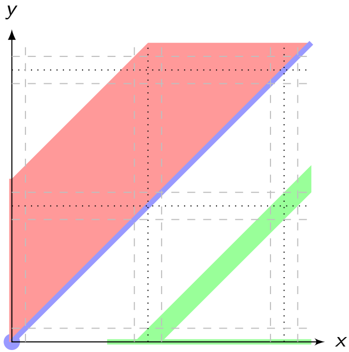




Exact semantics versus Enlarged semantics

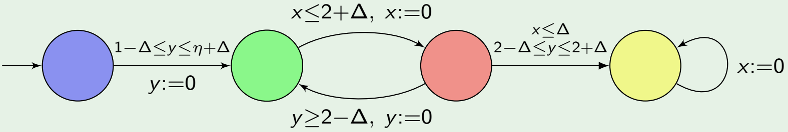
Example [Puri'98: Dynamical properties of timed automata]

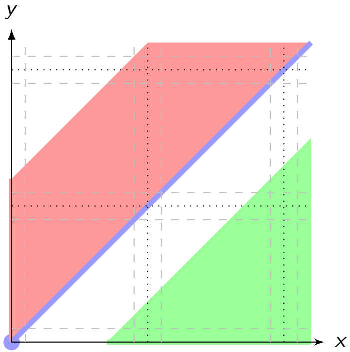




Exact semantics versus Enlarged semantics

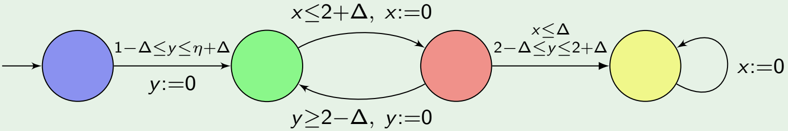
Example [Puri'98: Dynamical properties of timed automata]

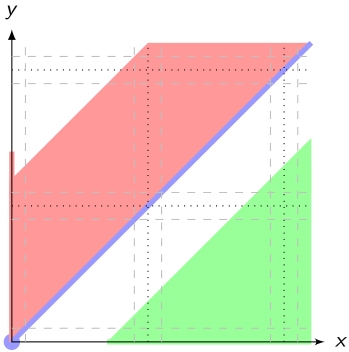




Exact semantics versus Enlarged semantics

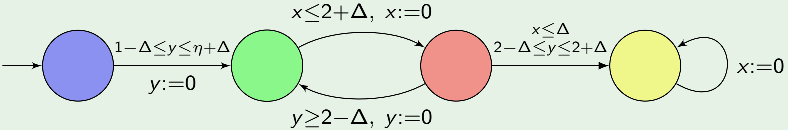
Example [Puri'98: Dynamical properties of timed automata]

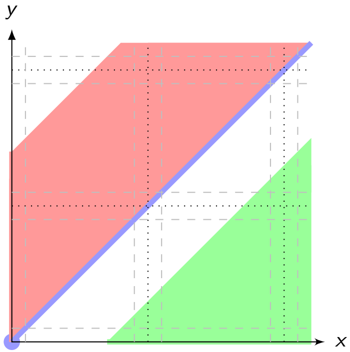




Exact semantics versus Enlarged semantics

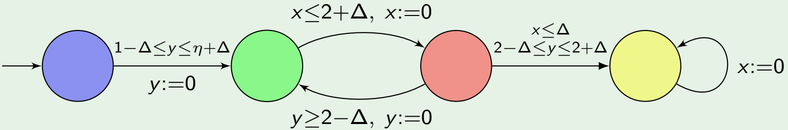
Example [Puri'98: Dynamical properties of timed automata]

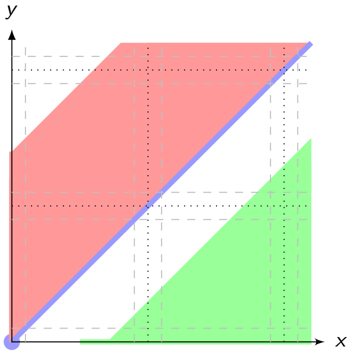




Exact semantics versus Enlarged semantics

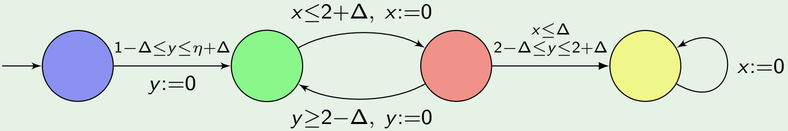
Example [Puri'98: Dynamical properties of timed automata]

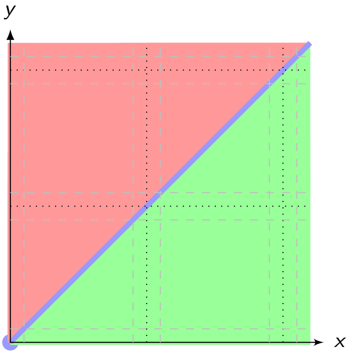




Exact semantics versus Enlarged semantics

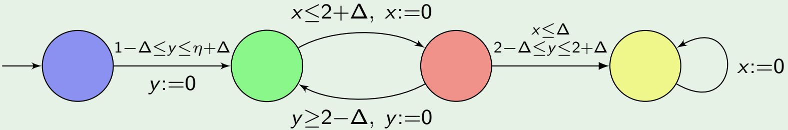
Example [Puri'98: Dynamical properties of timed automata]

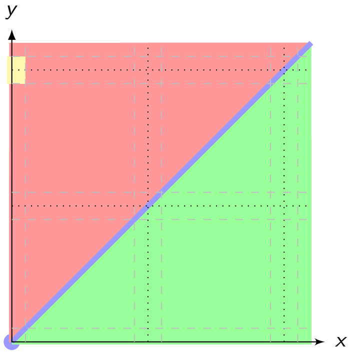




Exact semantics versus Enlarged semantics

Example [Puri'98: Dynamical properties of timed automata]





Specification and Verification with Tools support

Useful analysis implemented in e.g. Uppaal and ECDAR

- refinement relation: $\mathcal{S} \leq \mathcal{T}$
  (defined by an alternating timed simulation).
- satisfaction relation: $\mathcal{I}$ **sat** $\mathcal{S}$ iff $\mathcal{I} \leq \mathcal{S}$
- parallel composition operator: $\mathcal{S} \parallel \mathcal{T}$
- conjunction operator: $\mathcal{S} \wedge \mathcal{T}$
- quotient operator: $\mathcal{S} \setminus\!\!\setminus \mathcal{T}$

and

- Timed CTL model-checking: $\mathcal{A} \models \phi$

and more ...

Model-checking for perturbed systems

- Safety.
- Linear properties.
- Branching properties.
- Implementation verification and Refinement

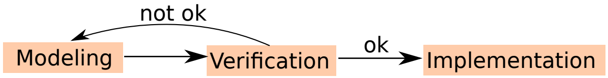
Classical model-checking

Given $\mathcal{A}$ and a property P, does $\mathcal{A}$ satisfy P? If it does, we write $\mathcal{A} \models P$.

Robust model-checking

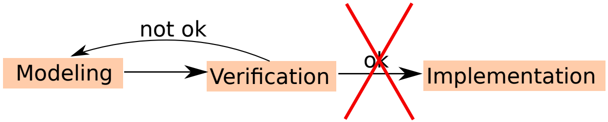
Given $\mathcal{A}$ and a property P, does $[\![\mathcal{A}]\!]_\delta$ satisfy P for some $\delta > 0$?
If it does, we write $\mathcal{A} \models\!\!\!\equiv P$ and say that $\mathcal{A}$ **robustly satisfies** P.

Our problem



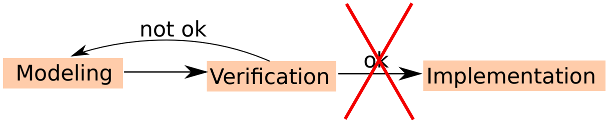
Question: does the classical approach suffice? Does $\mathcal{A} \models P$ imply $\mathcal{A} \mathrel{|}\!\!\equiv P$?

Our problem



Question: does the classical approach suffice? Does $\mathcal{A} \models P$ imply $\mathcal{A} \models\!\!\!\equiv P$?
No! There exists automata $\mathcal{A}$ such that $\text{Reach}([\![\mathcal{A}]\!]) \subsetneq \text{Reach}([\![\mathcal{A}]\!]_\delta)$ for any $\delta > 0$. (previous slide).

Our problem



Question: does the classical approach suffice? Does $\mathcal{A} \models P$ imply $\mathcal{A} \models\!\!\!\models P$?
No! There exists automata $\mathcal{A}$ such that $\mathrm{Reach}(\llbracket \mathcal{A} \rrbracket) \subsetneq \mathrm{Reach}(\llbracket \mathcal{A} \rrbracket_\delta)$ for any $\delta > 0$. (previous slide).

So now what?

- Give up?
- Consider only timed automata, which are already **robust**?
- Can we impose **robustness**?

Given a Spec/Design $\mathcal{A}$

Approximate: Strengthen the model. Such that all behavior is (almost) good.

Exact: Synthesize *an* implementation, forcing good behavior.

Preserving behavior

- Equivalent sets of reachable locations
- Language inclusion/equivalence
- Simulation and Refinement (alternating simulation)
- Bisimulation
- .. and so on

We are looking to relate behaviors

Given a timed (I/O) automaton $\mathcal{A}$:

$$[\![\mathcal{A}]\!] \; \mathcal{R} \; [\![\mathcal{A}_\Delta]\!]$$

Notions considered

It's application specific! Given $\Delta > 0$, a timed automaton $\mathcal{A}$ is

safety-robust [Puri'98]

if $\mathcal{A}$ has the same set of reachable locations as $\mathcal{A}_\Delta$

Δ-robust consistent

if there exists and implementation $\mathcal{I}$ s.t. $\mathcal{I}_\Delta \leq \mathcal{A}$

timed-action (strong timed) bisimulaton-robust

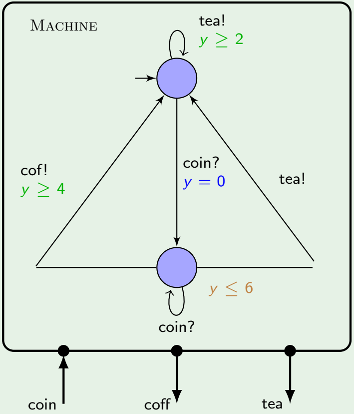
if $\mathcal{A} \approx_\epsilon \mathcal{A}_\Delta$ (resp. $\mathcal{A} \sim_\epsilon \mathcal{A}_\Delta$) for some $\epsilon > 0$

All of these have very natural game characterizations!

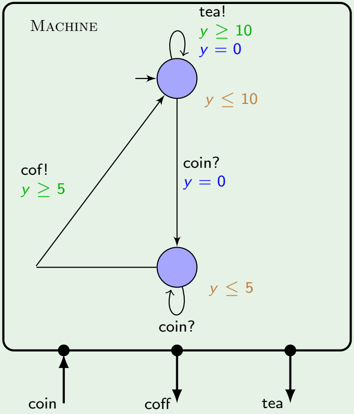
Refinement ($\leq$)

And implementation $\mathcal{I}$ is deterministic, input-enabled, and is output urgency and allows independent progress.

A specification



An impl



1. The big picture
   - Timed Automata
   - Specifications
   - Our problem
2. Defining Robustness
3. Addressing the robustness problem
   - Robust analysis and model-checking
   - Robust synthesis
   - Model Transformations
4. Conclusion

Background: Robust model-checking

Robust model-checking algorithms for:

- Reachability properties,
  [Puri'98], [De Wulf, Doyen, Markey, Raskin '04].
- LTL properties,
  [Bouyer, Markey, Reynier '06].
- a fragment of MTL
  [Bouyer, Markey, Reynier '08].

Finding robust implementations: Robust timed games

A TIOA $\mathcal{A}$ defines a timed game. Let f be a strategy for output:

- we build a TIOA $\mathcal{A}_f$, that represents the syntactic outcome of f
- $\lceil \mathcal{A}_f \rceil^o_\Delta$ is the perturbation of the outcome for player o.

Δ-robust strategy

f is a Δ-robust winning strategy for a condition W iff

$$\mathsf{Runs}(\lceil \mathcal{A}_f \rceil^o_\Delta) \subseteq W$$

Solving robust timed games

A syntactic transformation:

$$\mathcal{A} \longrightarrow \mathcal{A}^{\Delta}_{\mathrm{rob}}$$

Theorem

If

$\exists f$, winning strategy in the robust game $(\mathcal{A}^{\Delta}_{\mathrm{rob}}, W)$,

then

$\exists f'$, Δ-robust winning strategy in the game $(\mathcal{A}, W)$.

and f' can be obtained from f.

Robust consistency game

Safety objective: Output must avoid the set of inconsistent states $\mathrm{err}_{\Delta}^{S}$

Solve the game $(\mathcal{S}, WS^{o}(\mathrm{err}_{\Delta}^{S}))$:

1. determine a robust strategy f,
2. build from f an implementation $\mathcal{I}_f$.

Theorem

$\mathcal{I}_f$ is a robust implementation of $\mathcal{S}$

Composition of robust implementations

Independent implementation is also possible in the robust case:

Property

If

$$\mathcal{I}\ \mathbf{sat}_\Delta\ \mathcal{S} \quad \text{and} \quad \mathcal{J}\ \mathbf{sat}_\Delta\ \mathcal{T},$$

then

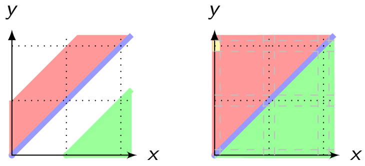
$$\mathcal{I} \parallel \mathcal{J}\ \mathbf{sat}_\Delta\ \mathcal{S} \parallel \mathcal{T}.$$

Using Approximation

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

- $[\![\mathcal{A}]\!]$ has the same behaviour as $[\![\mathcal{A}']\!]$,
- $\mathcal{A}'$ is robust, i.e. $[\![\mathcal{A}']\!]$ has approximately the same behaviour as $[\![\mathcal{A}']\!]_\delta$, for some $\delta > 0$.

Notice that in the former example, $[\![\mathcal{A}]\!]_\delta$ doesn't respect the region automaton.

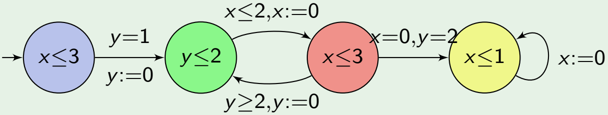


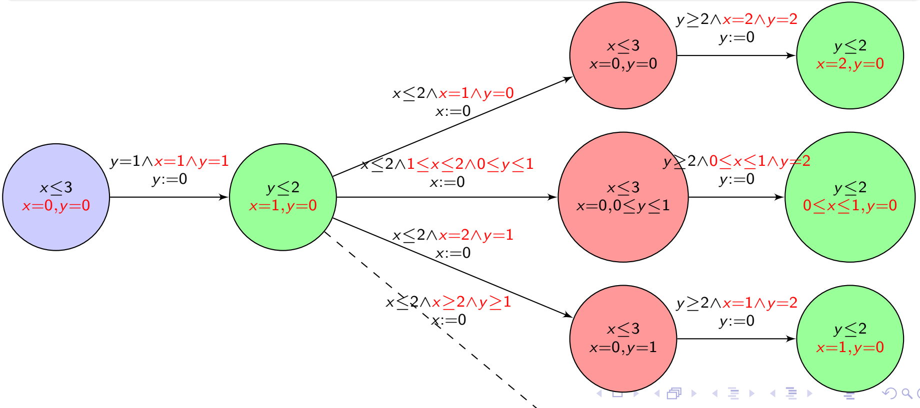
(a) $[\![\mathcal{A}]\!]$ (b) $[\![\mathcal{A}]\!]_\delta$

Basic idea: Enforce the region automaton: encoding regions in locations + strengthening guards to make behaviour compatible with the region automaton.

Automaton made robust

Example $\mathcal{A}$





Arbitrary close approximations

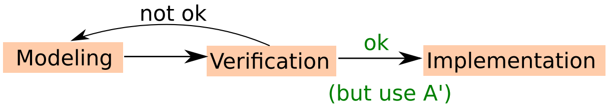
Given $\mathcal{A}$ and granularity η, our construction gives a **mixed**[1] timed automaton $\widetilde{\mathcal{A}_\eta}$.

Theorem

For any $\mathcal{A}$,

- $[\![\mathcal{A}]\!] \approx_\epsilon [\![\overline{\mathcal{A}_\eta}]\!]$ for all $\epsilon > 0$,
- $[\![\overline{\mathcal{A}_\eta}]\!] \approx_{\eta+2\delta} [\![\overline{\mathcal{A}_\eta}]\!]_\delta$, for any $\delta > 0$.

- $[\![\widetilde{\mathcal{A}_\eta}]\!]$ preserves all timed branching properties.
- $[\![(\widetilde{\mathcal{A}_\eta})_\Delta]\!]$ satisfies **almost** the same timed branching properties (in TCTL)



[1] that uses both open and closed guards such as $x \geq 0$ and $x < 2$

Additional properties

- $[\![\widetilde{\mathcal{A}}]\!]$ is big, but not too big.
- $\approx_\epsilon$ is sufficient.
- $\approx_\epsilon$ is *stronger* that safety and untimed CTL.
- We can do the same for the sampled semantics.

Conclusion

- Two approaches to the robustness question.
  - If an robust implementation exist, we can compute it.
  - Otherwise, we can always find a approximation of the *spec*, in which any implementation is robust.
- There is a lot of tool support which can be reused!