Robust Model-Checking of Linear-Time Properties in Timed Automata

Patricia Bouyer, Nicolas Markey, Pierre-Alain Reynier

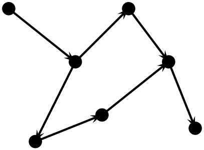
LSV – CNRS & ENS de Cachan – France

1. Context
2. Robust model-checking of pure-safety properties
3. Robust model-checking of LTL
4. Conclusion

Model-checking

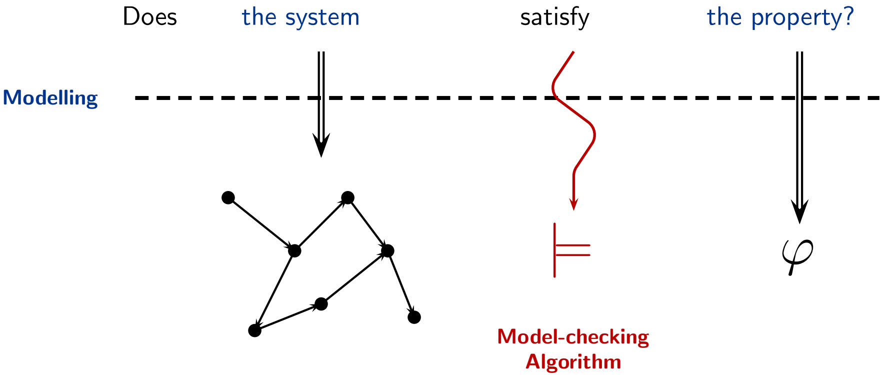
Does the system satisfy the property?

Modelling



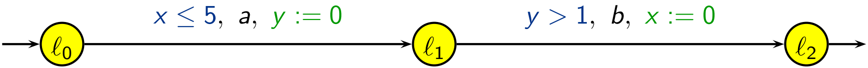
φ

Model-checking



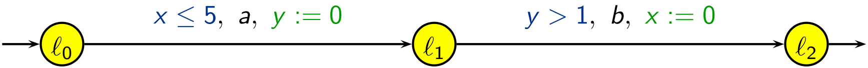
Timed automata (example)

x, y : clocks



Timed automata (example)

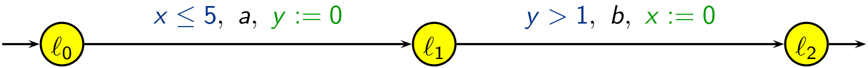
x, y : clocks



	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

Timed automata (example)

x, y : clocks

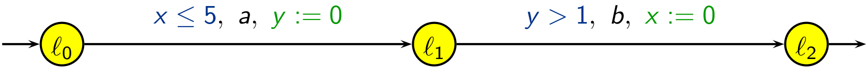


	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

(clock) valuation

Timed automata (example)

x, y : clocks



	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

(clock) valuation

➜ timed word $(a, 4.1)(b, 5.5)$

Implementatiblity of a timed automaton

- **"Implementing" a timed automaton assumes perfect hardware**

Infinitely punctual: exact synchronization of communications
Infinitely precise: clocks increase at the same rate
Infinitely fast: a timed automaton might have to perform actions faster and faster

Implementatiblity of a timed automaton

- **“Implementing” a timed automaton assumes perfect hardware**

Infinitely punctual: exact synchronization of communications
Infinitely precise: clocks increase at the same rate
Infinitely fast: a timed automaton might have to perform actions faster and faster

- **In practice, a processor is digital and imprecise**

Implementatiblity of a timed automaton

- **"Implementing" a timed automaton assumes perfect hardware**

Infinitely punctual: exact synchronization of communications
Infinitely precise: clocks increase at the same rate
Infinitely fast: a timed automaton might have to perform actions faster and faster

- **In practice, a processor is digital and imprecise**

➥ Even if we prove that a timed automaton is correct, it may happen that it cannot be correctly implemented.

From implementability to robustness

- **A design point-of-view** **[Altisen, Tripakis – FORMATS'05]**
  - integrate architecture in the system $\rightsquigarrow$ very general
  - defaults: - correctness depends on the architecture
    - faster is not always better

From implementability to robustness

- **A design point-of-view** **[Altisen, Tripakis – FORMATS'05]**
  - integrate architecture in the system $\rightsquigarrow$ very general
  - defaults: - correctness depends on the architecture
    - faster is not always better

- **A semantical point-of-view** **[De Wulf, Doyen, Raskin – HSCC'04]** **[De Wulf, Doyen, Markey, Raskin – FORMATS'04]**
  - enlarge guards: $g = [a, b] \quad \rightsquigarrow \quad g^{\Delta} = [a - \Delta, b + \Delta]$
  - a simple model of architecture

From implementability to robustness

- **A design point-of-view** **[Altisen, Tripakis – FORMATS'05]**
  - integrate architecture in the system $\rightsquigarrow$ very general
  - defaults: - correctness depends on the architecture
    - faster is not always better

- **A semantical point-of-view** **[De Wulf, Doyen, Raskin – HSCC'04]** **[De Wulf, Doyen, Markey, Raskin – FORMATS'04]**
  - enlarge guards: $g = [a, b] \quad \rightsquigarrow \quad g^\Delta = [a - \Delta, b + \Delta]$
  - a simple model of architecture

From implementability to robustness [DDR04]

- A timed automaton $\mathcal{A}$ is implementable w.r.t. property φ iff there exists Δ s.t. $\mathcal{A}^\Delta$ satisfies φ (for some properties φ).
- If $\mathcal{A}^{\Delta_0}$ satisfies φ, then for every $0 < \Delta < \Delta_0$, $\mathcal{A}^\Delta$ satisfies φ.

➥ "Faster is better"

Case of pure-safety properties

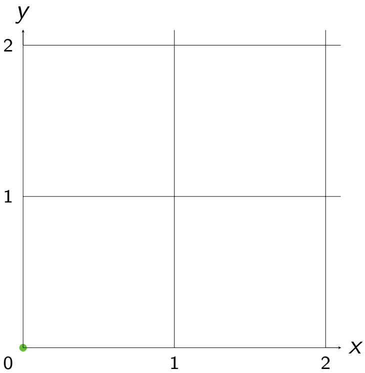
[Puri – FTRTFT'98]
[De Wulf, Doyen, Markey, Raskin – FORMATS'04]

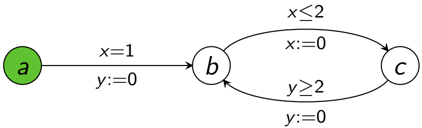
Theorem

Given a timed automaton $\mathcal{A}$ and a set of bad states Bad, we can decide whether there exists $\Delta > 0$ s.t. $\mathtt{Reach}(\mathcal{A}^\Delta) \cap \mathsf{Bad} = \emptyset$.

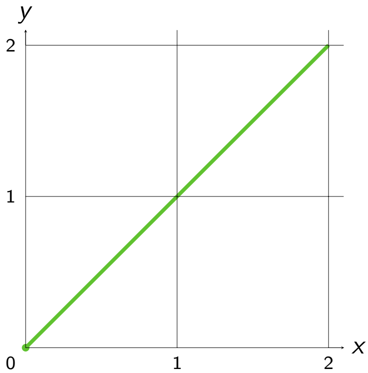
It is equivalent to checking that $\left(\bigcap_{\Delta>0} \mathtt{Reach}(\mathcal{A}^\Delta)\right) \cap \mathsf{Bad} = \emptyset$.

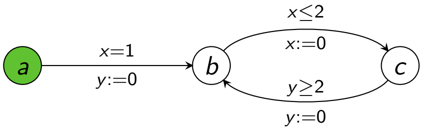
An example: standard semantics



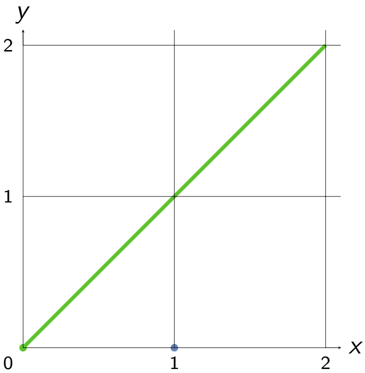


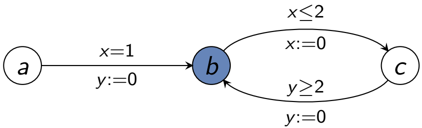
An example: standard semantics



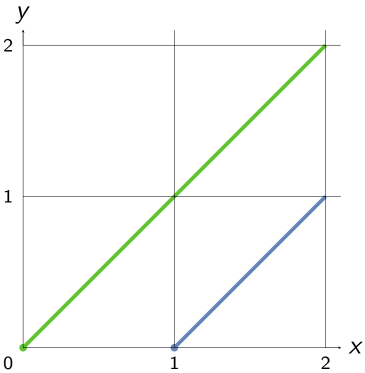


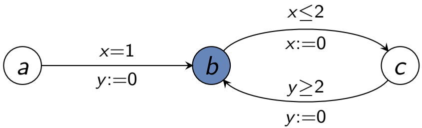
An example: standard semantics



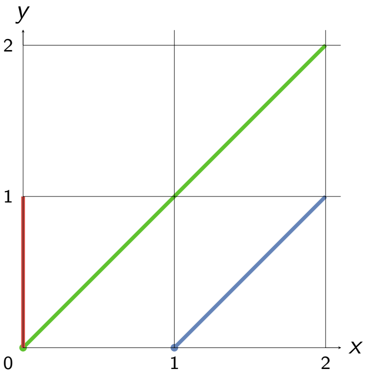


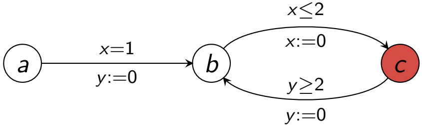
An example: standard semantics



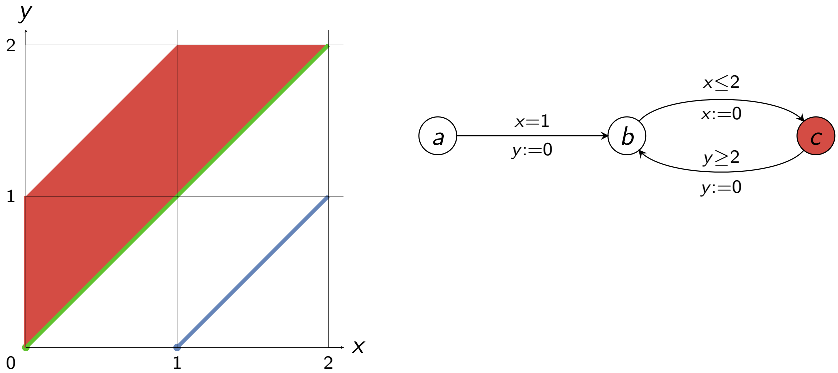


An example: standard semantics

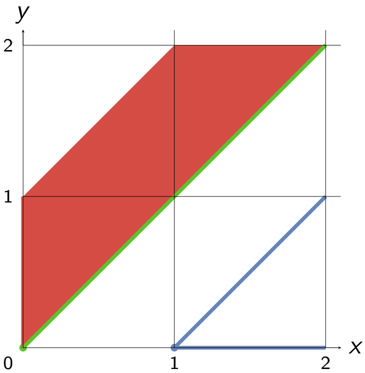


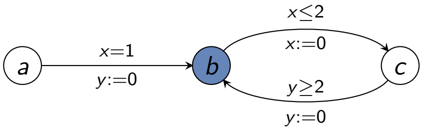


An example: standard semantics

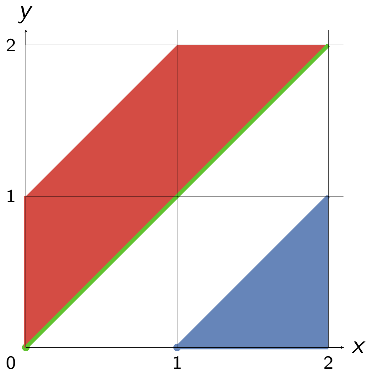


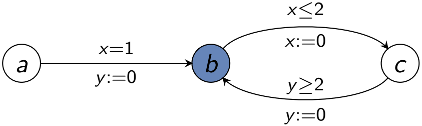
An example: standard semantics



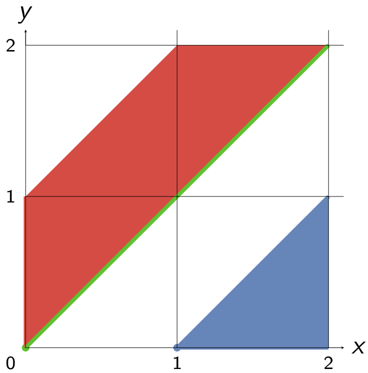


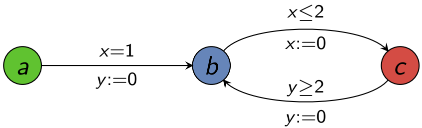
An example: standard semantics



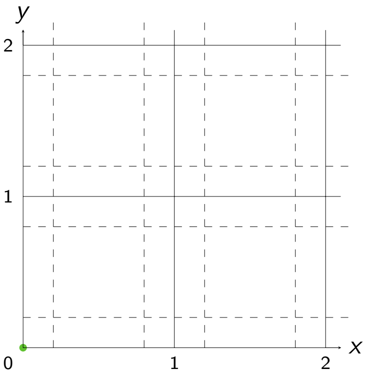


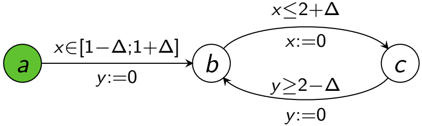
An example: standard semantics



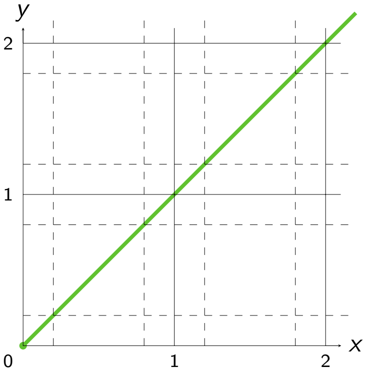


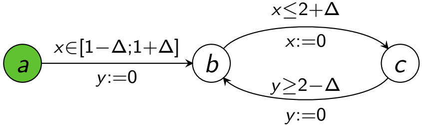
An example of enlarged semantics with $\Delta > 0$



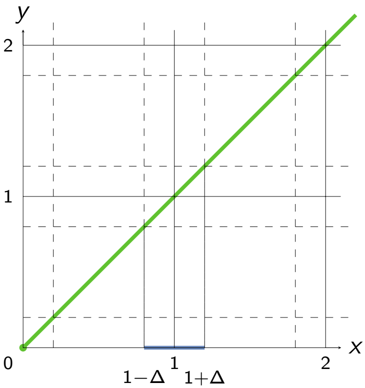


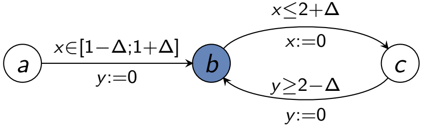
An example of enlarged semantics with $\Delta > 0$



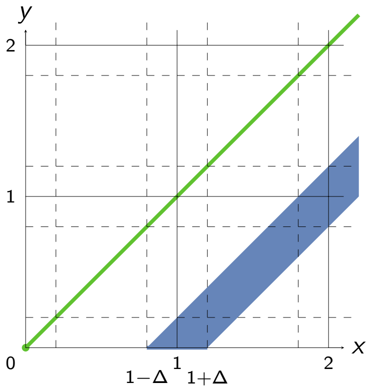


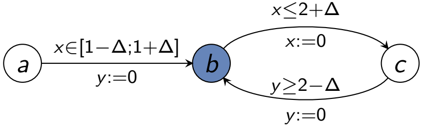
An example of enlarged semantics with $\Delta > 0$



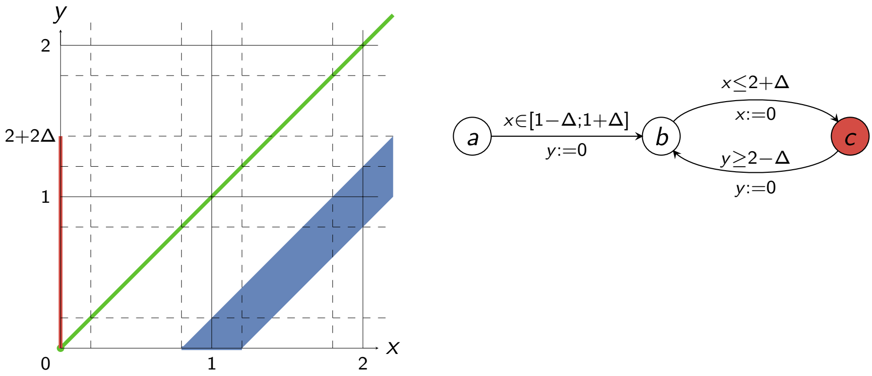


An example of enlarged semantics with $\Delta > 0$

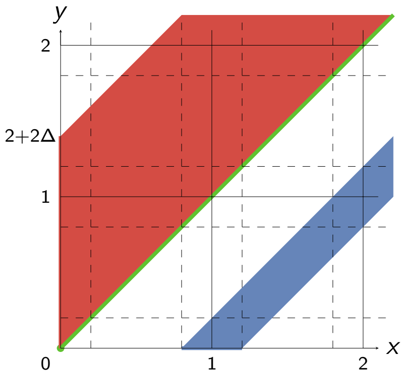


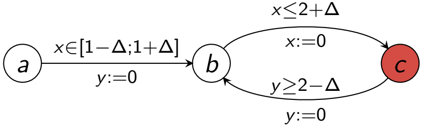


An example of enlarged semantics with $\Delta > 0$

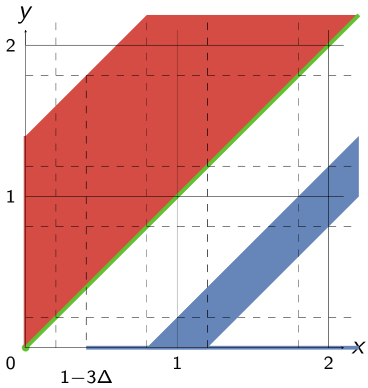


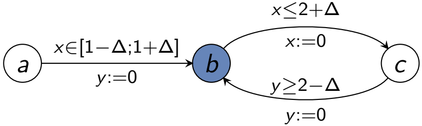
An example of enlarged semantics with $\Delta > 0$



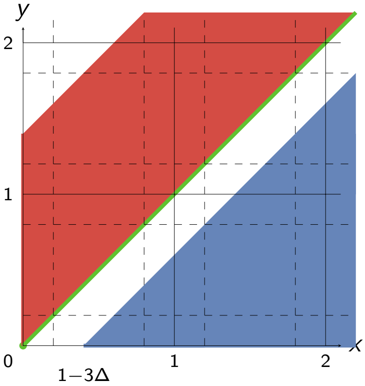


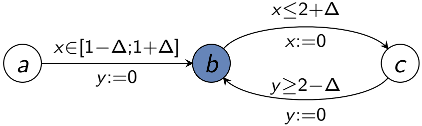
An example of enlarged semantics with $\Delta > 0$



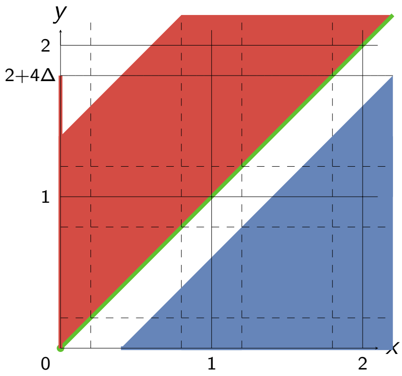


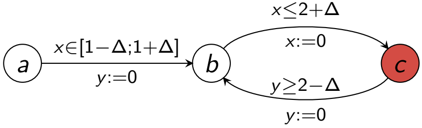
An example of enlarged semantics with $\Delta > 0$



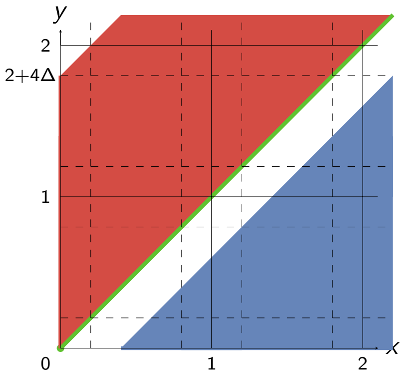


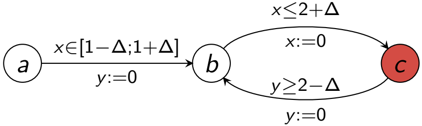
An example of enlarged semantics with $\Delta > 0$



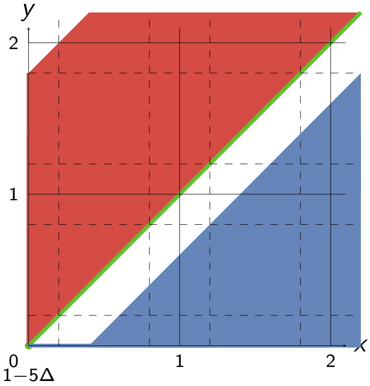


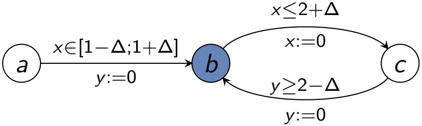
An example of enlarged semantics with $\Delta > 0$



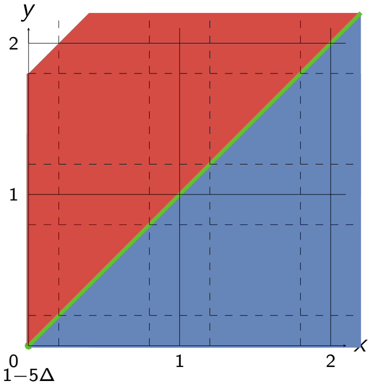


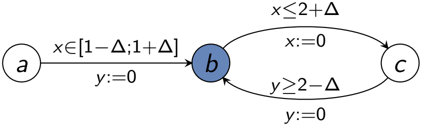
An example of enlarged semantics with $\Delta > 0$



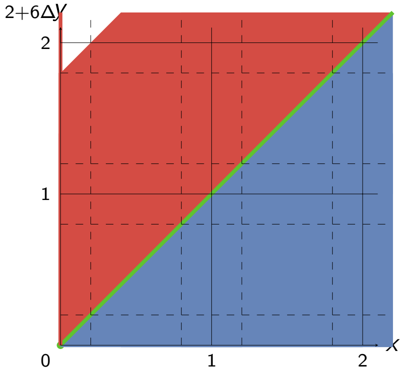


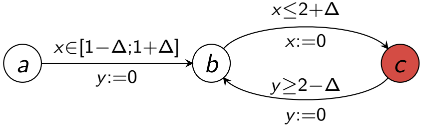
An example of enlarged semantics with $\Delta > 0$



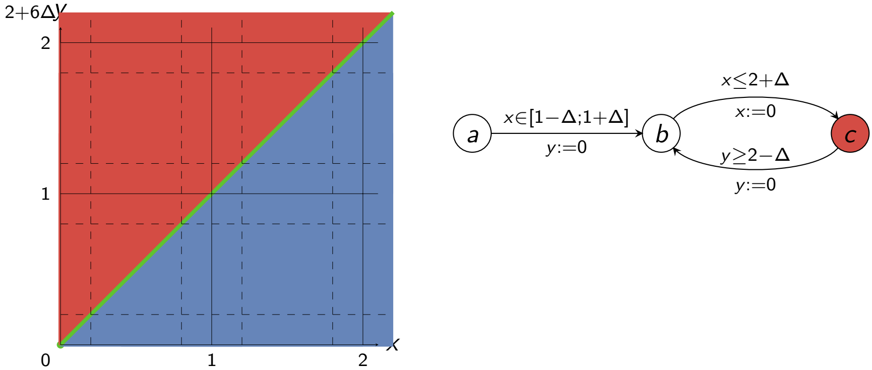


An example of enlarged semantics with $\Delta > 0$

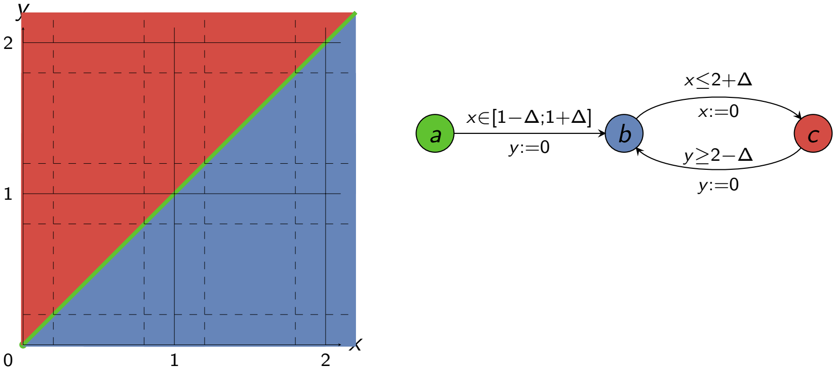




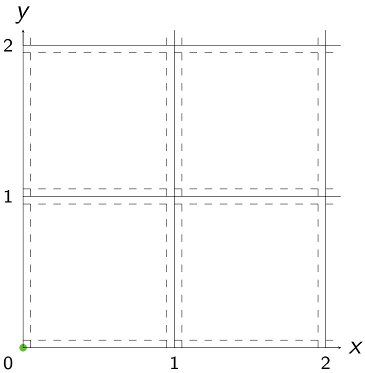
An example of enlarged semantics with $\Delta > 0$

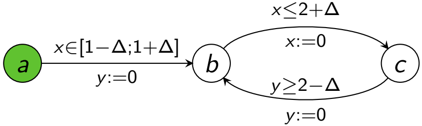


An example of enlarged semantics with $\Delta > 0$

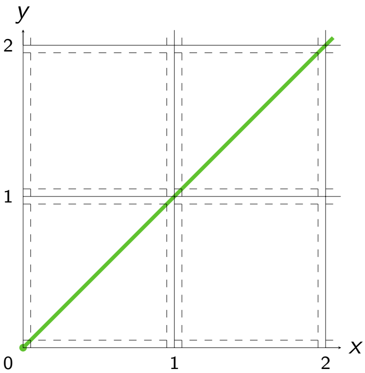


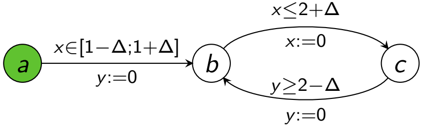
An example with Δ very small



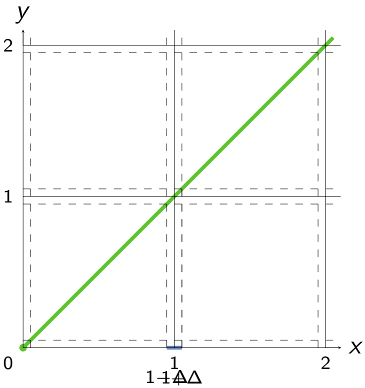


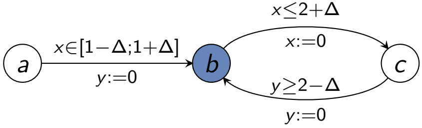
An example with Δ very small



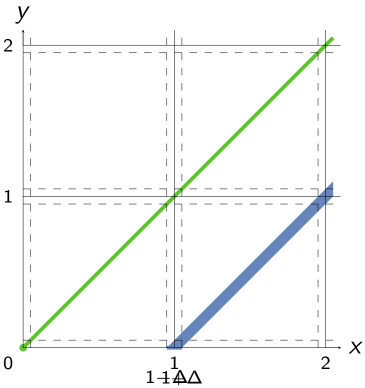


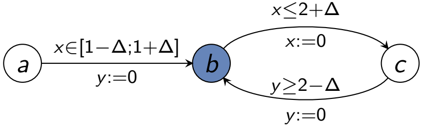
An example with Δ very small



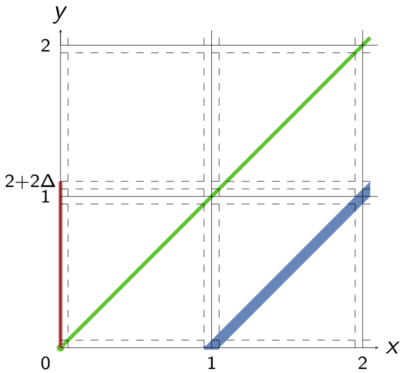


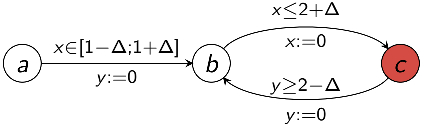
An example with Δ very small



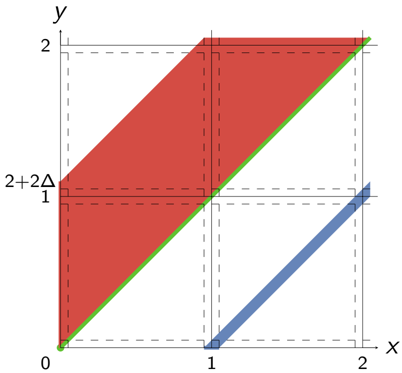


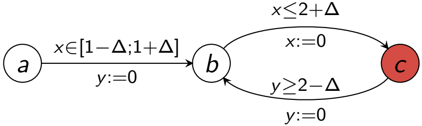
An example with Δ very small



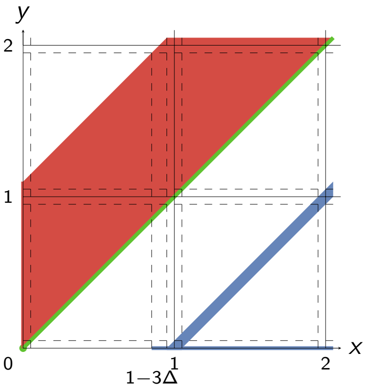


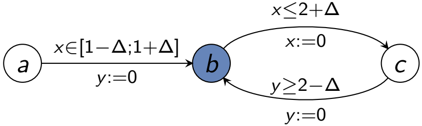
An example with Δ very small



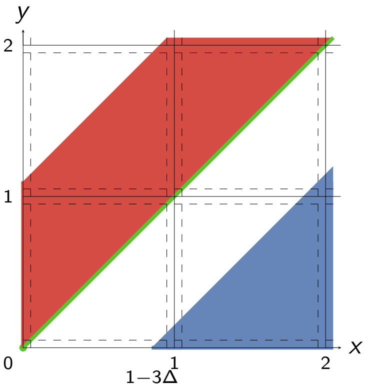


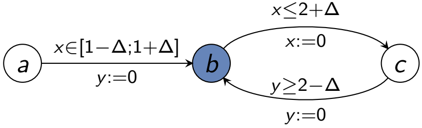
An example with Δ very small



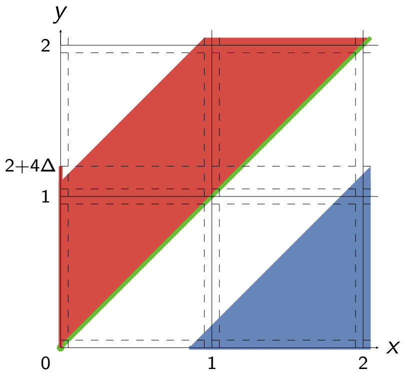


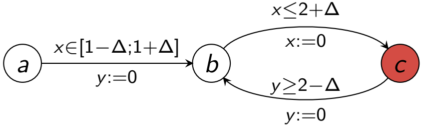
An example with Δ very small



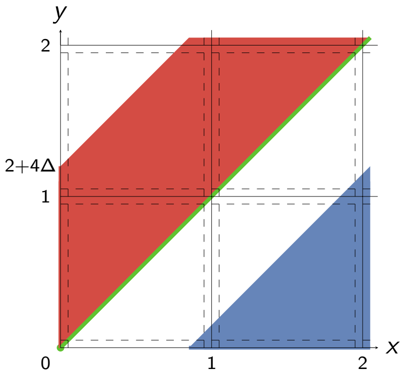


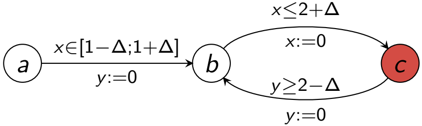
An example with Δ very small



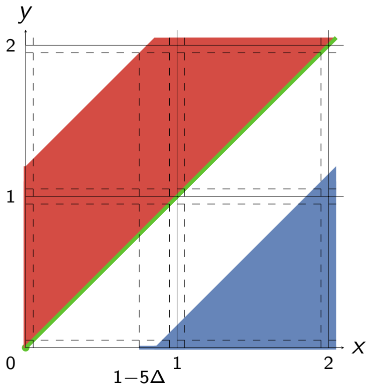


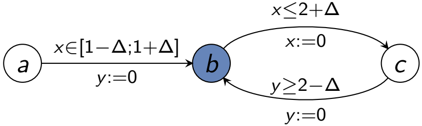
An example with Δ very small



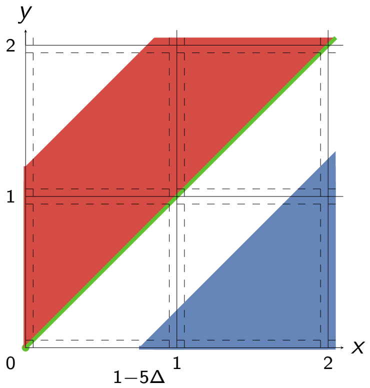


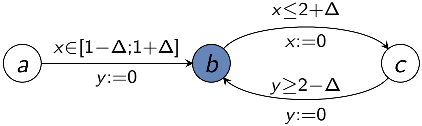
An example with Δ very small



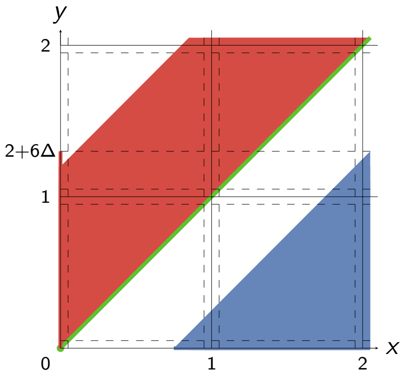


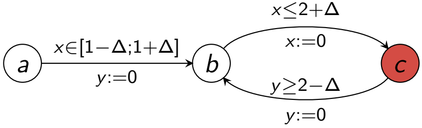
An example with Δ very small



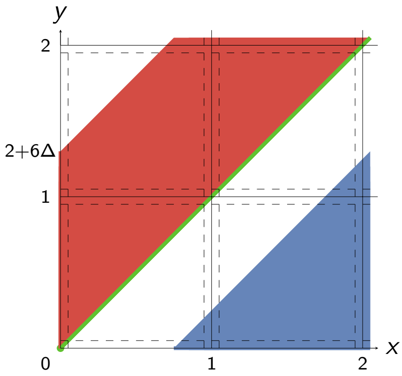


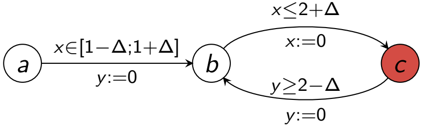
An example with Δ very small



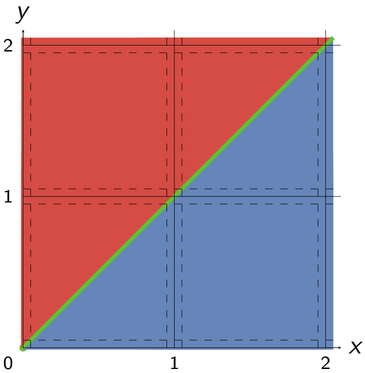


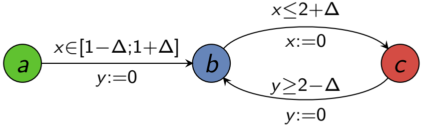
An example with Δ very small



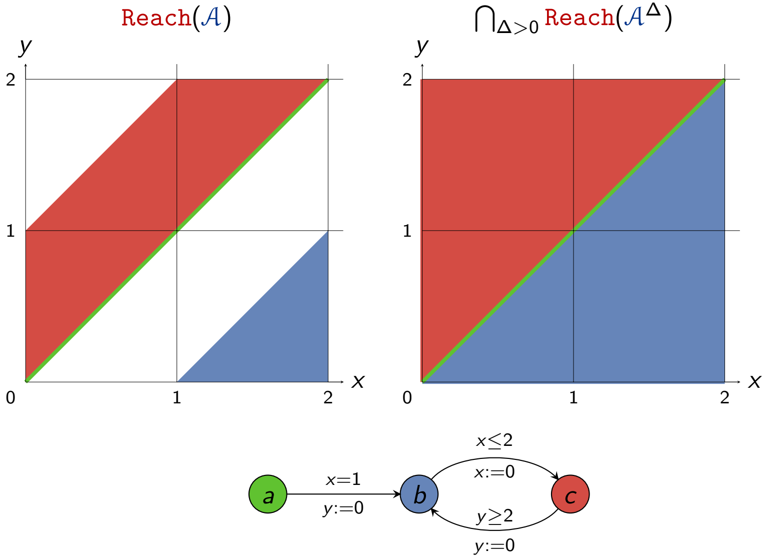


An example with Δ very small





Difference between $\mathcal{A}$ and $\mathcal{A}^{\Delta}$



Case of pure-safety properties

[Puri – FTRTFT'98]
[De Wulf, Doyen, Markey, Raskin – FORMATS'04]

Theorem

Given a timed automaton $\mathcal{A}$ and a set of bad states Bad, we can decide whether there exists $\Delta > 0$ s.t. $\mathtt{Reach}(\mathcal{A}^{\Delta}) \cap \mathsf{Bad} = \emptyset$.

It is equivalent to checking that $\left(\bigcap_{\Delta>0} \mathtt{Reach}(\mathcal{A}^{\Delta})\right) \cap \mathsf{Bad} = \emptyset$.

Case of pure-safety properties

[Puri – FTRTFT'98]
[De Wulf, Doyen, Markey, Raskin – FORMATS'04]

Theorem

Given a timed automaton $\mathcal{A}$ and a set of bad states Bad, we can decide whether there exists $\Delta > 0$ s.t. $\texttt{Reach}(\mathcal{A}^{\Delta}) \cap \text{Bad} = \emptyset$.

It is equivalent to checking that $\left(\bigcap_{\Delta>0} \texttt{Reach}(\mathcal{A}^{\Delta})\right) \cap \text{Bad} = \emptyset$.

Algorithm for computing $\left(\bigcap_{\Delta>0} \texttt{Reach}(\mathcal{A}^{\Delta})\right)$

1. build the region automaton G of $\mathcal{A}$;
2. compute $\texttt{SCC}(G)$, the set of SCCs of G;
3. $J := \texttt{Reach}(G, [q_0])$;
4. while $\exists S \in \texttt{SCC}(G).\ S \not\subseteq J$ and $S \cap J \neq \varnothing$,
   $J := J \cup S$;
   $J := \texttt{Reach}(G, J)$;
5. return(J);

The logic LTL

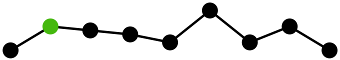
- The linear-time temporal logic LTL **[Pnueli – FOCS'77]**

$$\mathsf{LTL} \ni \psi, \varphi \ ::= \ p \ | \ \varphi \wedge \psi \ | \ \varphi \vee \psi \ | \ \neg \varphi \ | \ \mathbf{X}\, \varphi \ | \ \varphi \,\mathbf{U}\, \psi$$

The logic LTL

- The linear-time temporal logic LTL **[Pnueli – FOCS'77]**

$$\text{LTL} \ni \psi, \varphi ::= p \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\psi$$

$\mathbf{X}\,\varphi$  "In the next state" φ holds

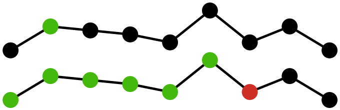
The logic LTL

- The linear-time temporal logic LTL **[Pnueli – FOCS'77]**

$$\text{LTL} \ni \psi, \varphi \ ::= \ p \ \mid \ \varphi \wedge \psi \ \mid \ \varphi \vee \psi \ \mid \ \neg \varphi \ \mid \ \mathbf{X}\, \varphi \ \mid \ \varphi\, \mathbf{U}\, \psi$$

$\mathbf{X}\, \varphi$

$\varphi\, \mathbf{U}\, \psi$



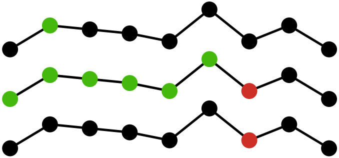
"In the next state" φ holds

φ holds "Until" ψ holds

The logic LTL

- The linear-time temporal logic LTL **[Pnueli – FOCS'77]**

$$\mathsf{LTL} \ni \psi, \varphi ::= p \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\psi$$



$\mathbf{X}\,\varphi$ — "In the next state" φ holds

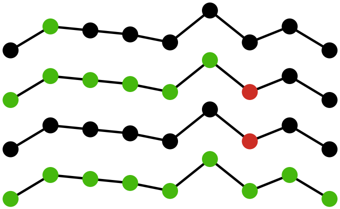
$\varphi\,\mathbf{U}\,\psi$ — φ holds "Until" ψ holds

$\mathbf{F}\,\varphi \equiv \top\,\mathbf{U}\,\varphi$ — φ holds "Eventually"

The logic LTL

- The linear-time temporal logic LTL **[Pnueli – FOCS'77]**

$$\mathsf{LTL} \ni \psi, \varphi \ ::= \ p \ \mid \ \varphi \wedge \psi \ \mid \ \varphi \vee \psi \ \mid \ \neg\varphi \ \mid \ \mathbf{X}\,\varphi \ \mid \ \varphi\,\mathbf{U}\,\psi$$

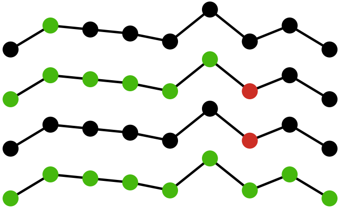


$\mathbf{X}\,\varphi$	"In the next state" φ holds
$\varphi\,\mathbf{U}\,\psi$	φ holds "Until" ψ holds
$\mathbf{F}\,\varphi \equiv \top\,\mathbf{U}\,\varphi$	φ holds "Eventually"
$\mathbf{G}\,\varphi \equiv \neg(\mathbf{F}\,\neg\varphi)$	φ "Always" holds

The logic LTL

- The linear-time temporal logic LTL **[Pnueli – FOCS'77]**

$$\text{LTL} \ni \psi, \varphi ::= p \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\psi$$



$\mathbf{X}\,\varphi$	"In the next state" φ holds
$\varphi\,\mathbf{U}\,\psi$	φ holds "Until" ψ holds
$\mathbf{F}\,\varphi \equiv \top\,\mathbf{U}\,\varphi$	φ holds "Eventually"
$\mathbf{G}\,\varphi \equiv \neg(\mathbf{F}\,\neg\varphi)$	φ "Always" holds

- Examples of formulas:
  - "p occurs infinitely often": $\mathbf{G}\,\mathbf{F}\,p$
  - "a request is eventually granted": $\mathbf{G}\,(\mathit{request} \rightarrow \mathbf{F}\,\mathit{grant})$

Robust model-checking of LTL

Theorem

Robust model-checking of LTL properties is decidable and PSPACE-complete.

Robust model-checking of LTL

Theorem

Robust model-checking of LTL properties is decidable and PSPACE-complete.

- Model-checking of LTL properties can be reduced to model-checking of co-Büchi properties **[Wolper, Vardi, Sistla – FOCS'83]**

Robust model-checking of LTL

Theorem

Robust model-checking of LTL properties is decidable and PSPACE-complete.

- Model-checking of LTL properties can be reduced to model-checking of co-Büchi properties **[Wolper, Vardi, Sistla – FOCS'83]**
- Extend the classical region automaton with γ-transitions when a reachable region is adjacent to an SCC $\rightsquigarrow \mathcal{R}^*$

Robust model-checking of LTL

Theorem

Robust model-checking of LTL properties is decidable and PSPACE-complete.

- Model-checking of LTL properties can be reduced to model-checking of co-Büchi properties **[Wolper, Vardi, Sistla – FOCS'83]**
- Extend the classical region automaton with γ-transitions when a reachable region is adjacent to an SCC $\rightsquigarrow$ $\mathcal{R}^*$
- Checking co-Büchi properties in $\mathcal{A}$ and in $\mathcal{R}^*$ is equivalent

  "Taking a γ-transition in $\mathcal{R}^*$ corresponds to taking a certain number of times the corresponding SCC in $\mathcal{A}$"

Conclusion

- robust model-checking of LTL properties is PSPACE-complete
- robust model-checking of a small fragment of MTL (a real-time extension of LTL) in PSPACE:

$$\mathbf{G}\,(p \rightarrow \mathbf{F}_{\leq 5}\, q)$$

Conclusion

- robust model-checking of LTL properties is PSPACE-complete
- robust model-checking of a small fragment of MTL (a real-time extension of LTL) in PSPACE:

$$\mathbf{G}\,(p \rightarrow \mathbf{F}_{\leq 5}\, q)$$

Further work

- robust model-checking of Safety-MTL? Or even of MTL?
- synthesis of robust controllers?
- what about branching-time logics?