Hardness of Untimed Language Universality

Romain Brenguier   Ocan Sankur
Thanks to Stefan Göller

LSV, CNRS & ENS Cachan, France
{brenguier,sankur}@lsv.ens-cachan.fr

YR-CONCUR
September 10, 2011

Timed Systems

- Context: Formal verification of systems running in real-time;

Timed Systems

- Context: Formal verification of systems running in real-time;
- Examples: mobile phone, plane, car, train, robot,. . .

Timed Systems

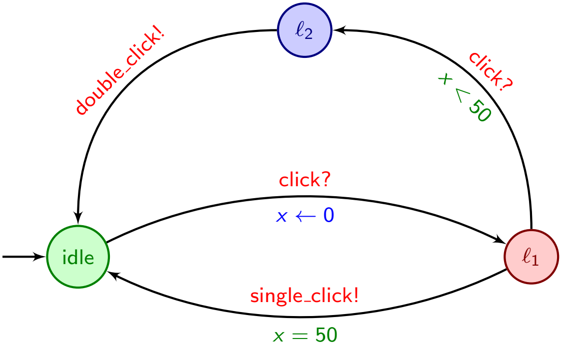
- Context: Formal verification of systems running in real-time;
- Examples: mobile phone, plane, car, train, robot,. . .

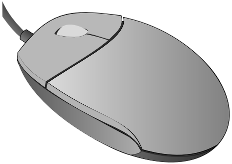
    

- Modeled by Timed Automata

Timed Automata (TA) [Alur and Dill 1994]

Finite automata + Analog clocks 

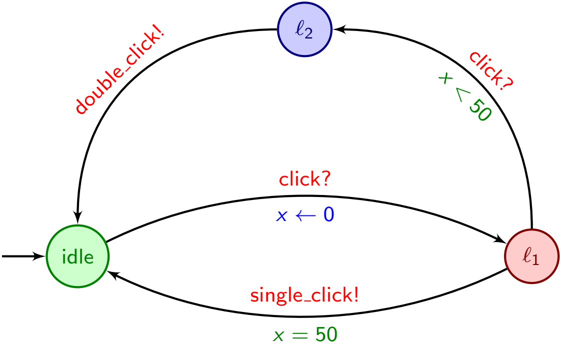


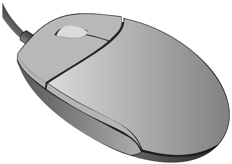


- Clocks cannot be stopped, all grow at the same rate.
- An edge is activated when its **clock constraint** holds.
- A clock can be **reset** by a transition.

Timed Automata (TA) [Alur and Dill 1994]

Finite automata + Analog clocks 



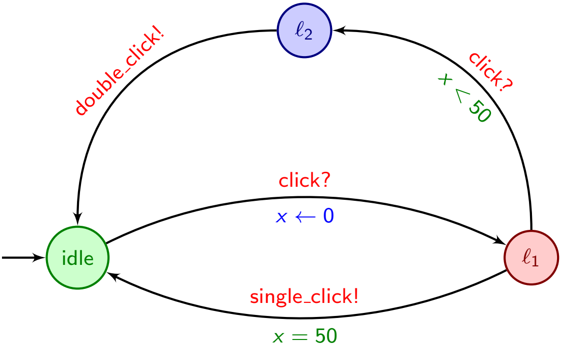


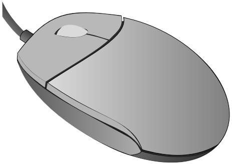
Runs of a timed automaton

$(\text{idle}, x = 0) \xrightarrow{23.7} (\text{idle}, x = 23.7) \xrightarrow{\textit{click?}} (\ell_1, x = 0) \xrightarrow{10} (\ell_1, x = 10)$
$\xrightarrow{\textit{click?}} (\ell_2, x = 10) \xrightarrow{\textit{double\_click!}} (\text{idle}, x = 10) \cdots$

Timed Automata (TA) [Alur and Dill 1994]

Finite automata + Analog clocks 





The **untimed language** of a timed automaton

Sequences of edge labels along which there is a run.

For instance $(\text{click?} \cdot \text{single\_click!})^* \subseteq L(\mathcal{A})$.

Known Results about TA

- Emptiness is PSPACE-complete; $L(\mathcal{A}) = \varnothing$? $L^t(\mathcal{A}) = \varnothing$?

- Timed language universality is undecidable;
  $L^t(\mathcal{A}) = \Sigma^*$? $L^t(\mathcal{A}) = L^t(\mathcal{B})$? $L^t(\mathcal{A}) \subseteq L^t(\mathcal{B})$?

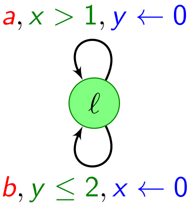
Known Results about TA

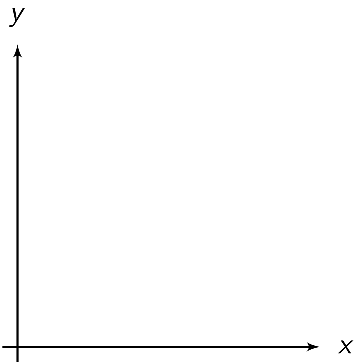
- Emptiness is PSPACE-complete; $L(\mathcal{A}) = \varnothing$? $L^t(\mathcal{A}) = \varnothing$?
- Timed language universality is undecidable; $L^t(\mathcal{A}) = \Sigma^*$? $L^t(\mathcal{A}) = L^t(\mathcal{B})$? $L^t(\mathcal{A}) \subseteq L^t(\mathcal{B})$?
- Untimed language universality and inclusion are PSPACE-hard and in EXPSPACE. $L(\mathcal{A}) = \Sigma^*$? $L(\mathcal{A}) = L(\mathcal{B})$? $L(\mathcal{A}) \subseteq L(\mathcal{B})$?

Known Results about TA

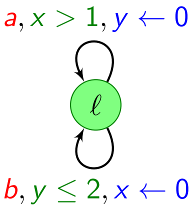
- Emptiness is PSPACE-complete; $L(\mathcal{A}) = \varnothing$? $L^t(\mathcal{A}) = \varnothing$?
- Timed language universality is undecidable;
  $L^t(\mathcal{A}) = \Sigma^*$? $L^t(\mathcal{A}) = L^t(\mathcal{B})$? $L^t(\mathcal{A}) \subseteq L^t(\mathcal{B})$?
- Untimed language universality and inclusion are PSPACE-hard and in EXPSPACE.
  $L(\mathcal{A}) = \Sigma^*$? $L(\mathcal{A}) = L(\mathcal{B})$? $L(\mathcal{A}) \subseteq L(\mathcal{B})$?
  - What is the exact complexity of these problems?

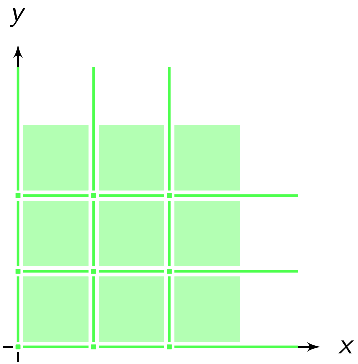
The Region Abstraction





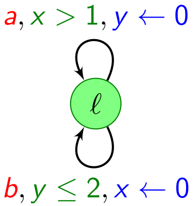
The Region Abstraction

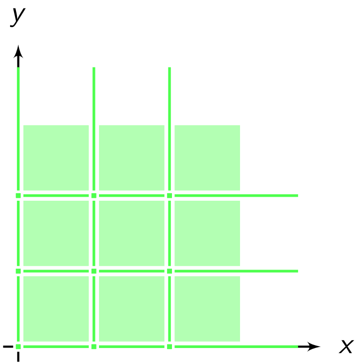




- "Compatibility" between regions and **constraints**;

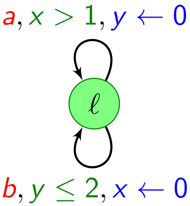
The Region Abstraction

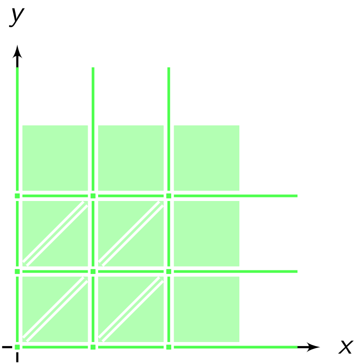




- "Compatibility" between regions and **constraints**;
- "Compatibility" between regions and **resets**;

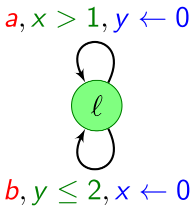
The Region Abstraction

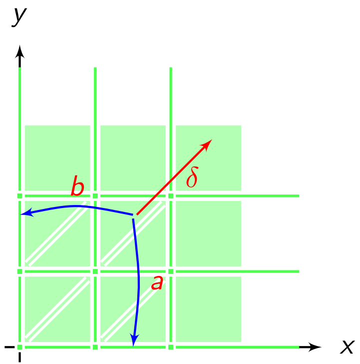




- "Compatibility" between regions and **constraints**;
- "Compatibility" between regions and **resets**;
- "Compatibility" between regions and **time elapsing**;

The Region Abstraction

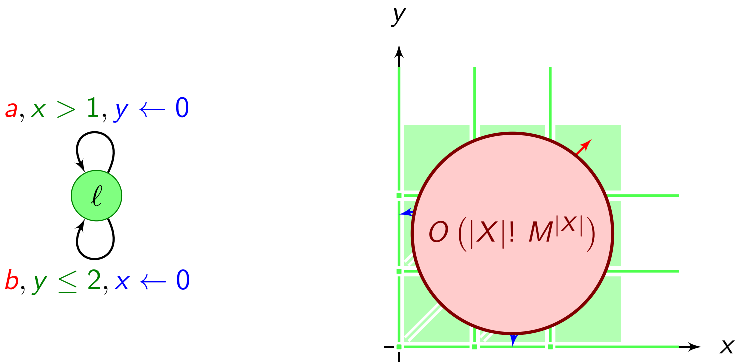




- "Compatibility" between regions and **constraints**;
- "Compatibility" between regions and **resets**;
- "Compatibility" between regions and **time elapsing**;

$\Rightarrow$ Bisimulation property.

The Region Abstraction



- "Compatibility" between regions and **constraints**;
- "Compatibility" between regions and **resets**;
- "Compatibility" between regions and **time elapsing**;

$\Rightarrow$ Bisimulation property.

Main Result

Theorem

Untimed language inclusion and universality problems for timed automata are EXPSPACE-complete.

Main Result

Theorem

Untimed language inclusion and universality problems for timed automata are EXPSPACE-complete.

Algorithm

Given timed automata $\mathcal{A}$ and $\mathcal{B}$

- construct the corresponding region automata, $R(\mathcal{A})$, $R(\mathcal{B})$;
- use a PSPACE algorithm to check language inclusion on the region automata, $R(\mathcal{A}) \subseteq R(\mathcal{B})$.

Encoding an Exponential Space Turing Machine

Given an exponential space Turing machine $\mathcal{T}$ and an input x,

Encoding an Exponential Space Turing Machine

Given an exponential space Turing machine $\mathcal{T}$ and an input x,

- Executions are encoded by some words;

Encoding an Exponential Space Turing Machine

Given an exponential space Turing machine $\mathcal{T}$ and an input x,

- Executions are encoded by some words;
- We construct an automaton that recognizes timed words that are:
  - either **not correct** executions of $\mathcal{T}$,

Encoding an Exponential Space Turing Machine

Given an exponential space Turing machine $\mathcal{T}$ and an input x,

- Executions are encoded by some words;
- We construct an automaton that recognizes timed words that are:
  - either **not correct** executions of $\mathcal{T}$,
  - or executions that do **not start** with the input x,

Encoding an Exponential Space Turing Machine

Given an exponential space Turing machine $\mathcal{T}$ and an input x,

- Executions are encoded by some words;
- We construct an automaton that recognizes timed words that are:
  - either **not correct** executions of $\mathcal{T}$,
  - or executions that do **not start** with the input x,
  - or executions that are **not accepting**;

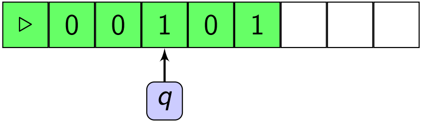
Encoding an Exponential Space Turing Machine

Given an exponential space Turing machine $\mathcal{T}$ and an input x,

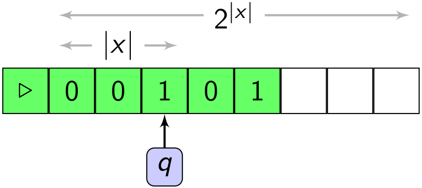
- Executions are encoded by some words;
- We construct an automaton that recognizes timed words that are:
  - either **not correct** executions of $\mathcal{T}$,
  - or executions that do **not start** with the input x,
  - or executions that are **not accepting**;

$\Rightarrow$ The automaton is **universal** if and only if the input x is **not accepted** by the Turing machine.

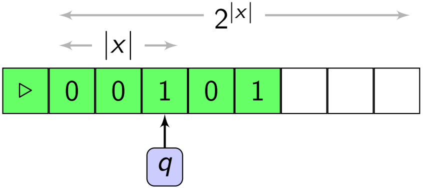
A configuration



A configuration

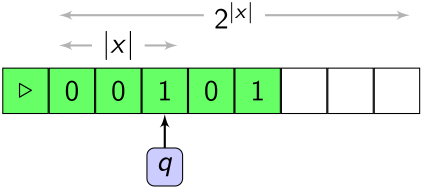


A configuration



is encoded by the word $w = \triangleright\, 0\, 0\, 1\ q\ 0\, 1\, \sqcup\ \sqcup\ \sqcup$

A configuration

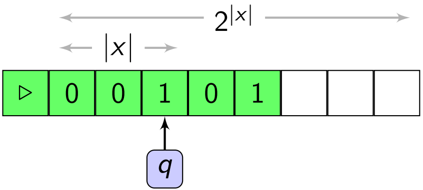


is encoded by the word $w = \triangleright\, 0\, 0\, 1\, q\, 0\, 1\, \sqcup\, \sqcup\, \sqcup$

An execution is encoded by

$$w_1 \; \$ \; w_2 \; \$ \; w_3 \; \$ \; w_4 \; \cdots$$

A configuration



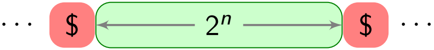
is encoded by the word $w = \triangleright\, 0\, 0\, 1\; q\; 0\, 1 \sqcup \sqcup \sqcup$

An execution is encoded by

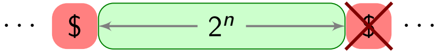
w_1 \$ w_2 \$ w_3 \$ w_4 $\cdots$

The alphabet of the timed automaton is $\Gamma = \Sigma \cup Q \cup \{\$\}$

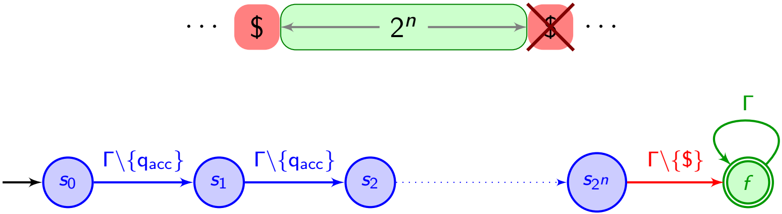
Detecting Errors with an Automaton



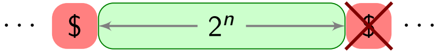
Detecting Errors with an Automaton

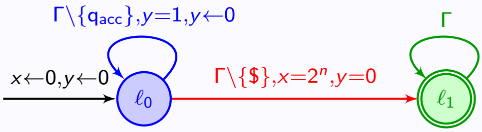


Detecting Errors with an Automaton



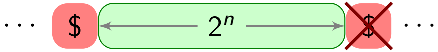
Detecting Errors with a Timed Automaton

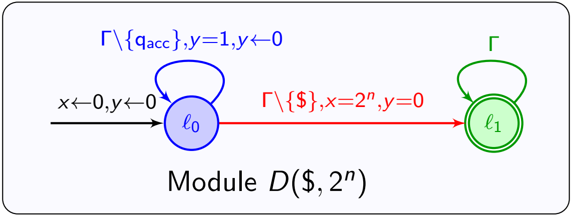




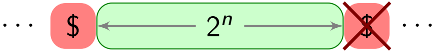
Module $D(\$, 2^n)$

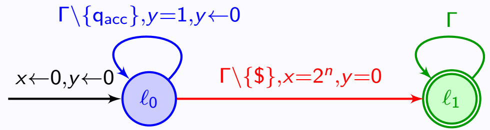
Detecting Errors with a Timed Automaton





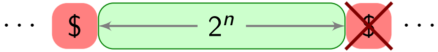
Detecting Errors with a Timed Automaton

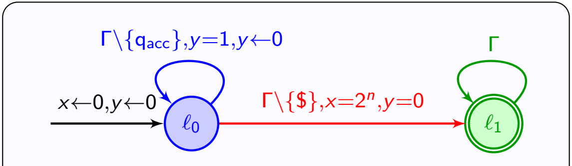




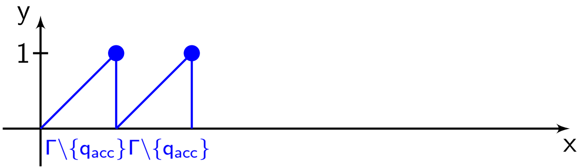
Module $D(\$, 2^n)$

Detecting Errors with a Timed Automaton

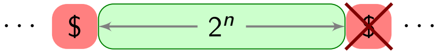


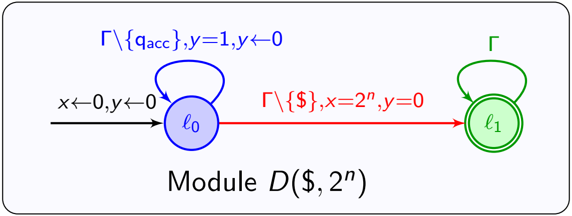


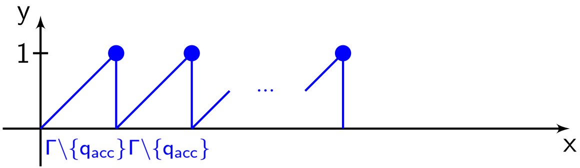
Module $D(\$, 2^n)$



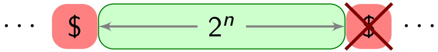
Detecting Errors with a Timed Automaton

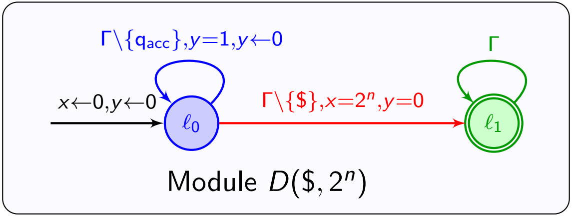


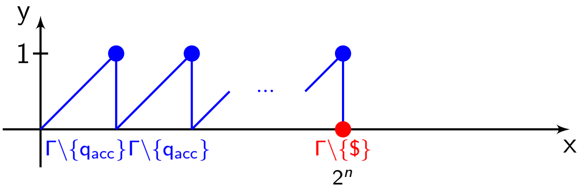




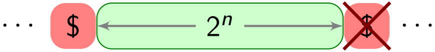
Detecting Errors with a Timed Automaton

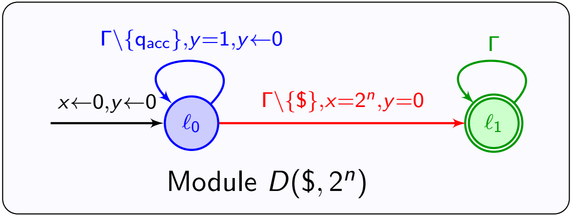


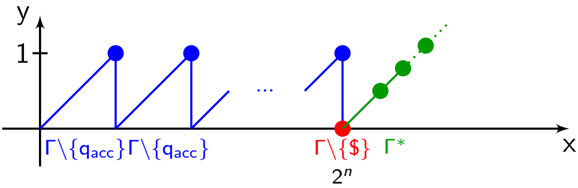




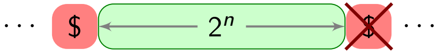
Detecting Errors with a Timed Automaton

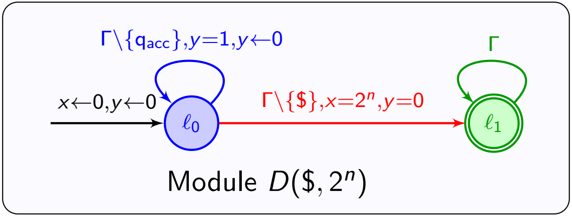


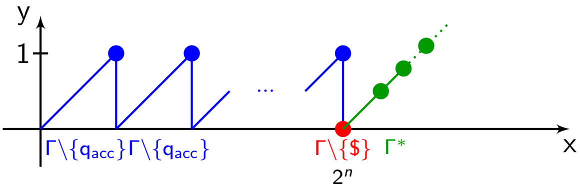




Detecting Errors with a Timed Automaton

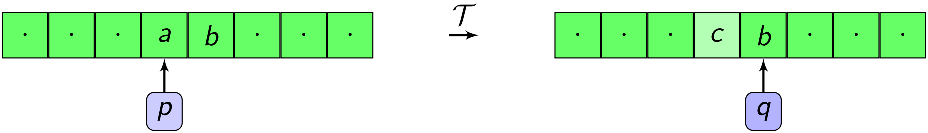




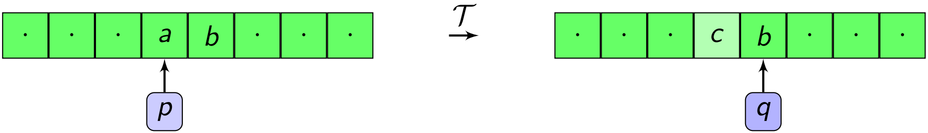


$$L(D(a,m)) = (\Gamma \setminus \{q_{acc}\})^m \cdot (\Gamma \setminus \{a\}) \cdot \Gamma^*$$

Encoding an Instruction



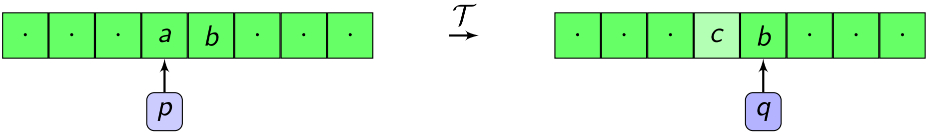
Encoding an Instruction



$$\cdots \quad a \quad p \quad b \quad \cdots \quad \$ \quad \cdots \quad c \quad b \quad q \quad \cdots$$

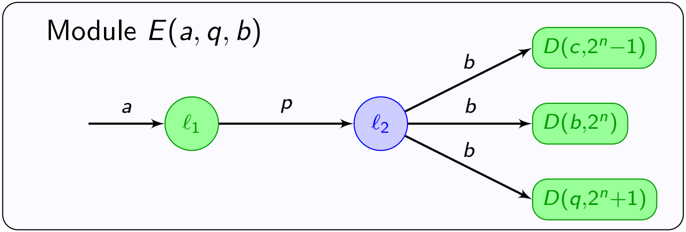
$$\longleftarrow 2^n - 1 \longrightarrow$$

Encoding an Instruction

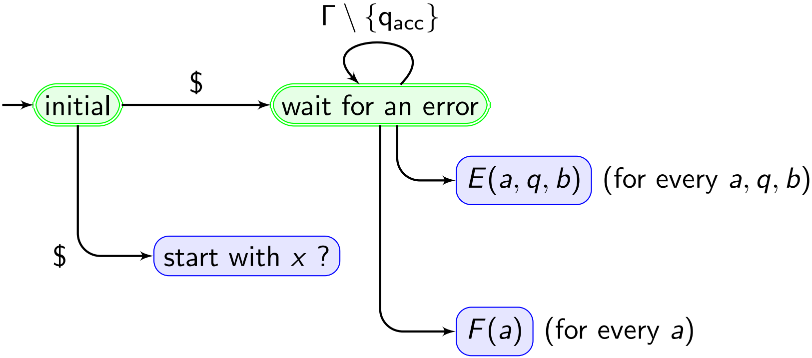


$\cdots \; a \; p \; b \; \cdots \; \$ \; \cdots \; c \; b \; q \; \cdots$

$\longleftarrow 2^n{-}1 \longrightarrow$



The Global Construction



Conclusion

- The algorithm is simple and uses the region abstraction ...

Conclusion

- The algorithm is simple and uses the region abstraction . . .
- But the problem is hard.

Conclusion

- The algorithm is simple and uses the region abstraction ...
- But the problem is hard.

Thank you for your attention