TASE 2019

2019年7月30日
中國，桂林

# Formalizing Time4sys using parametric timed automata

Étienne André

[1] LIPN, Université Paris 13, CNRS, France
[2] National Institute of Informatics, Japan
[3] JFLI, UMI CNRS, Tokyo, Japan

Supported by the ASTREI project funded by the Paris Île-de-France Region, by the ANR national research program PACS (ANR-14-CE28-0002), and by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603).

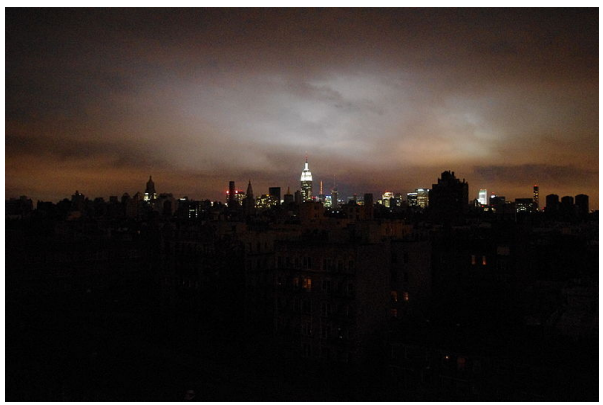
# Context: Verifying critical real-time systems

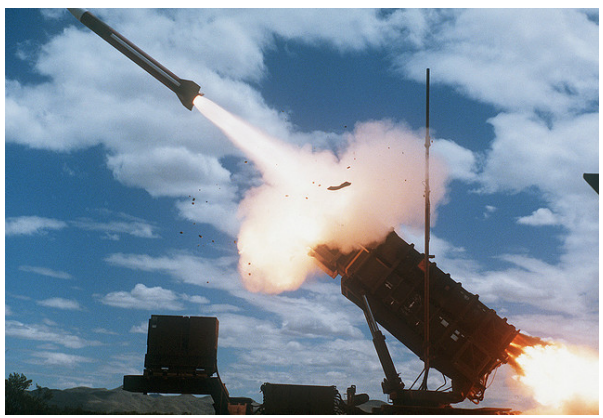
real-time systems:

- Systems for which not only the correctness but also the timely answer is important

# Context: Verifying critical real-time systems

- Critical real-time systems:
  - Systems for which not only the correctness but also the timely answer is important
  - Failures (in correctness or timing) may result in dramatic consequences





# Distributed real-time system

A distributed real-time system is made of a set of tasks to execute on a set of processors

# Distributed real-time system

A distributed real-time system is made of a set of tasks to execute on a set of processors

A task is characterized by:

- $B$: its best-case execution time
- $W$: its worst-case execution time
- $D$: its relative deadline
- $O$: its offset (or phase)

# Distributed real-time system

A distributed real-time system is made of a set of tasks to execute on a set of processors

A task is characterized by:

- $B$: its best-case execution time
- $W$: its worst-case execution time
- $D$: its relative deadline
- $O$: its offset (or phase)

Tasks have instances that can be activated…

- periodically
- sporadically (usually with a minimum interarrival time)
- or following more complex patterns (e. g., activation following the completion of another task instance)

# Scheduler

Activated instances are queued

When the processor is idle, which instance in the queue should be executed?

$\rightsquigarrow$ decision made by the scheduler

# Scheduler

Activated instances are queued

When the processor is idle, which instance in the queue should be executed?

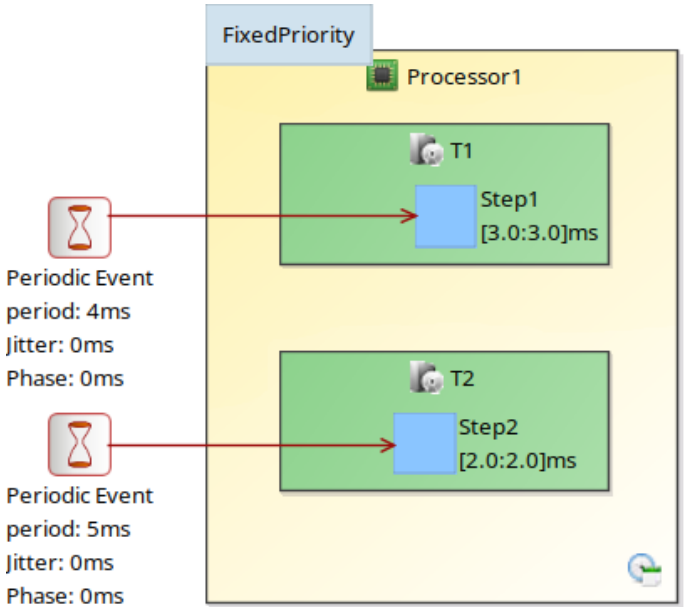
$\rightsquigarrow$ decision made by the scheduler

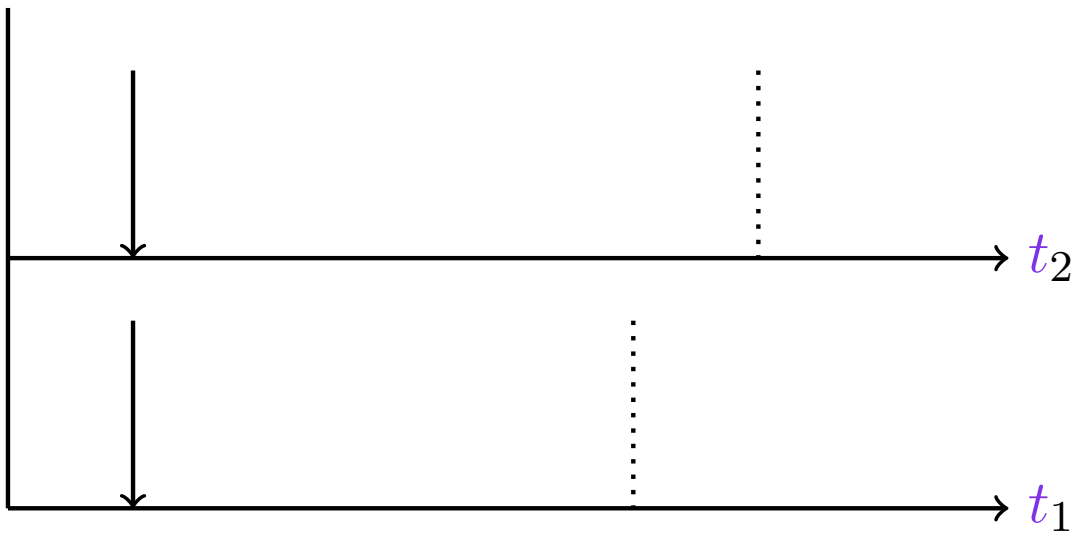
The scheduler can be preemptive

- The execution of a lower priority task can be interrupted when a instance of a task with higher priority is activated
- After completion of the higher priority task, the lower priority task resumes

# Example: earliest deadline first (EDF)

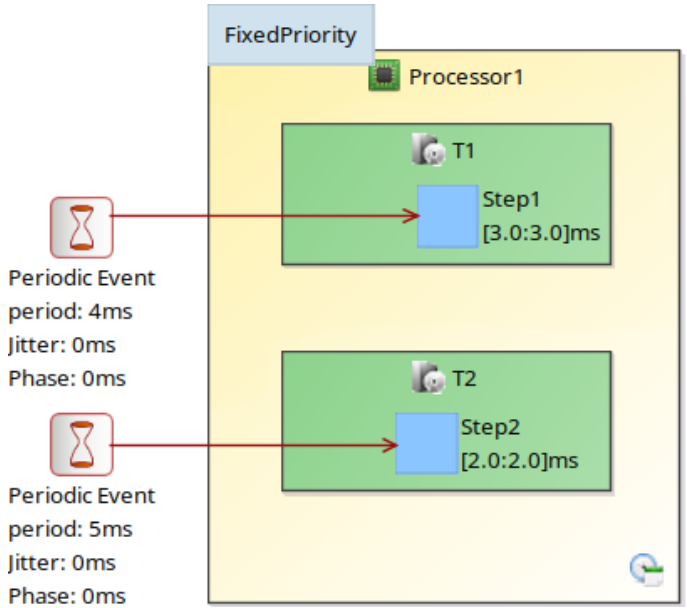
| Task | B | W | D |
|---|---|---|---|
| $t_1$ | 3 | 3 | 4 |
| $t_2$ | 2 | 2 | 5 |

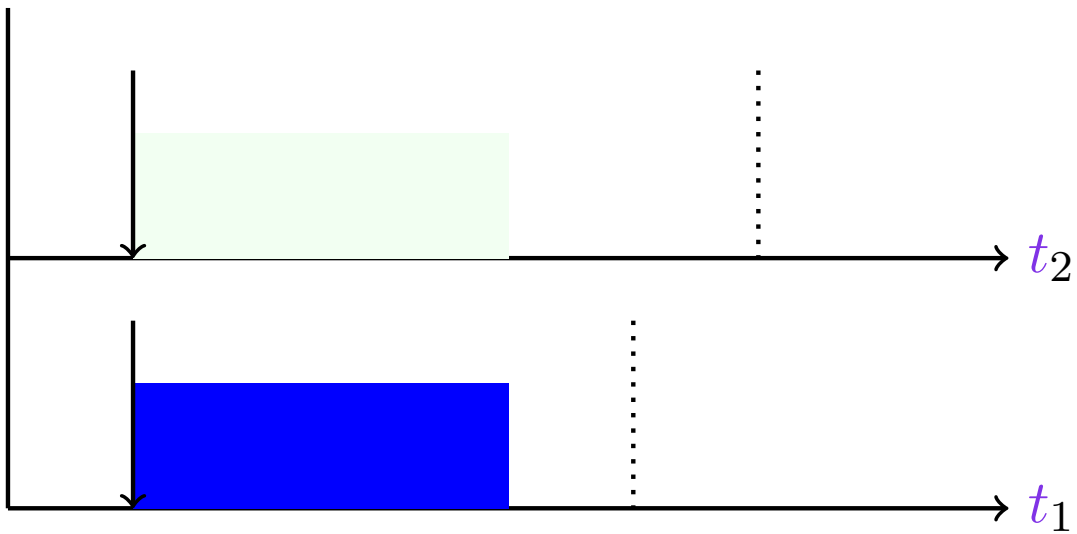


# Example: earliest deadline first (EDF)

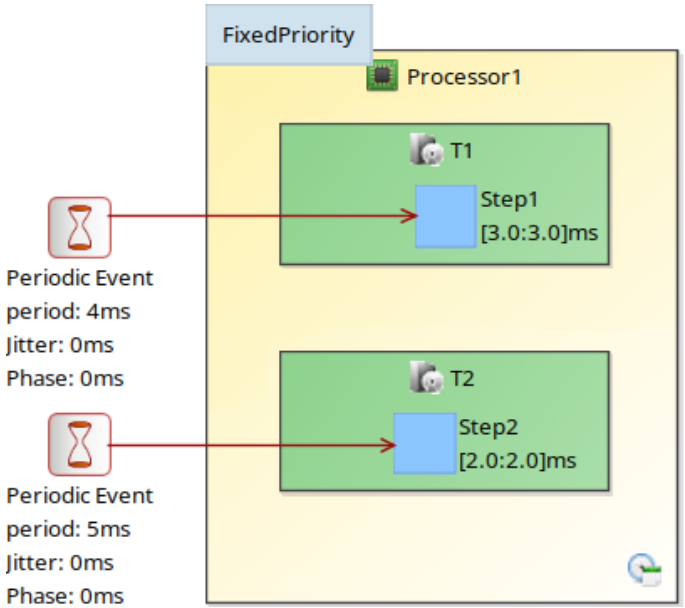
| Task | B | W | D |
|---|---|---|---|
| $t_1$ | 3 | 3 | 4 |
| $t_2$ | 2 | 2 | 5 |

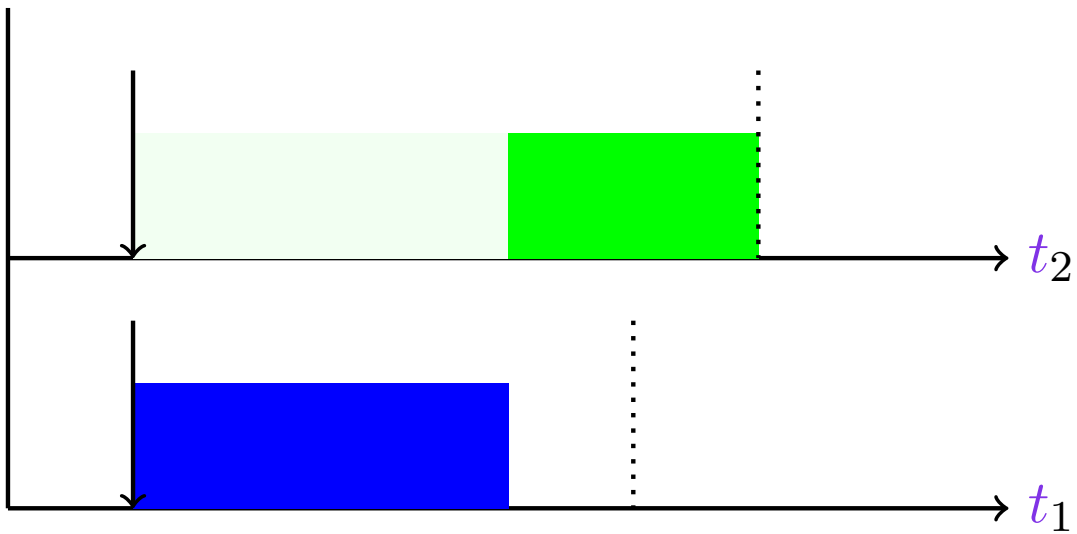


# Example: earliest deadline first (EDF)

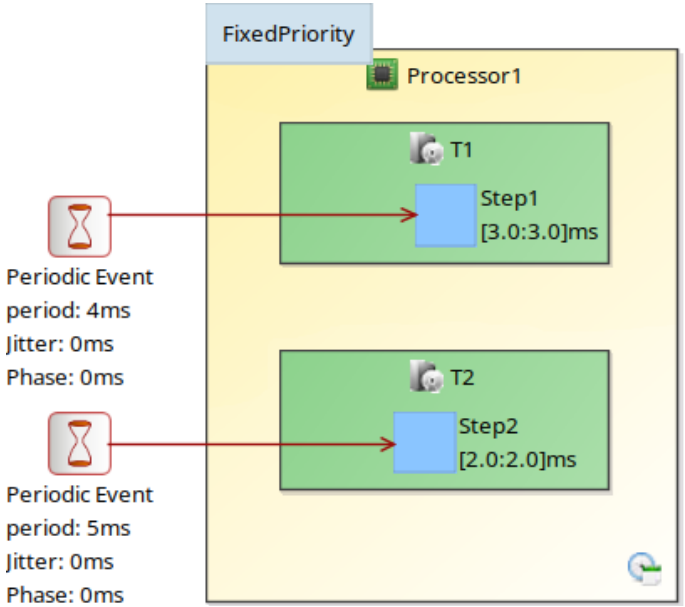
| Task | B | W | D |
|---|---|---|---|
| $t_1$ | 3 | 3 | 4 |
| $t_2$ | 2 | 2 | 5 |

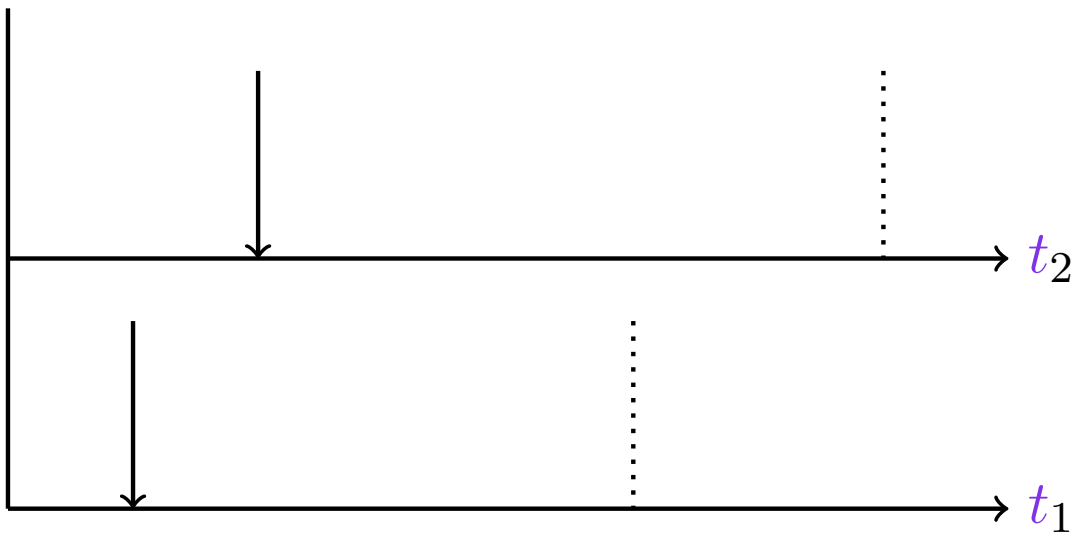


# Example: preemptive fixed priority scheduler (FPS)

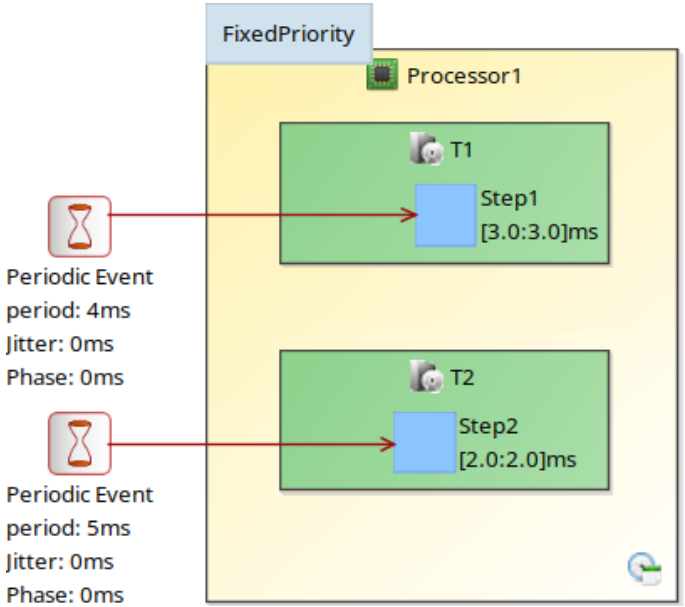
| Task | B | W | D | priority |
|---|---|---|---|---|
| $t_1$ | 3 | 3 | 4 | low |
| $t_2$ | 2 | 2 | 5 | high |

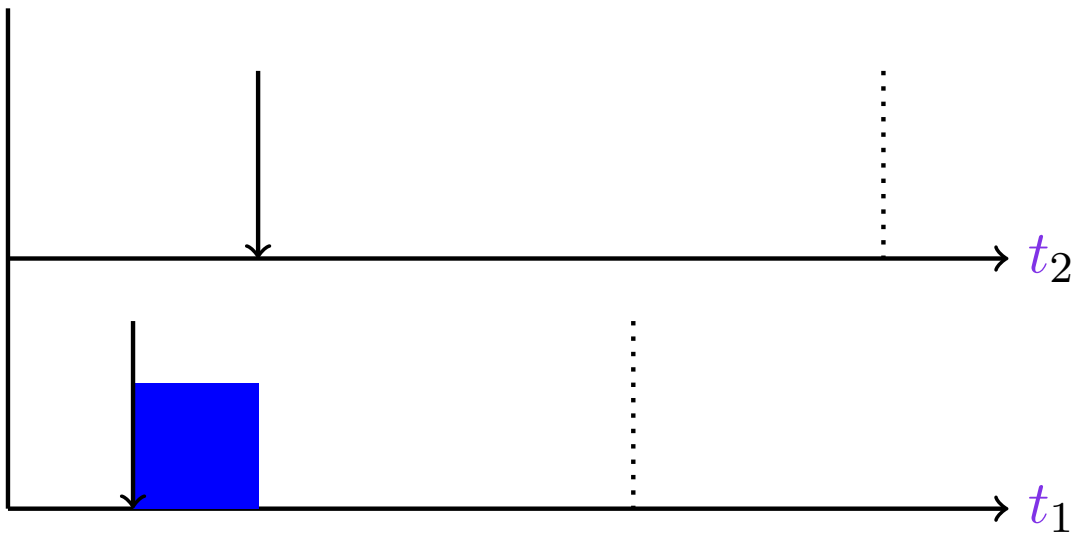


# Example: preemptive fixed priority scheduler (FPS)

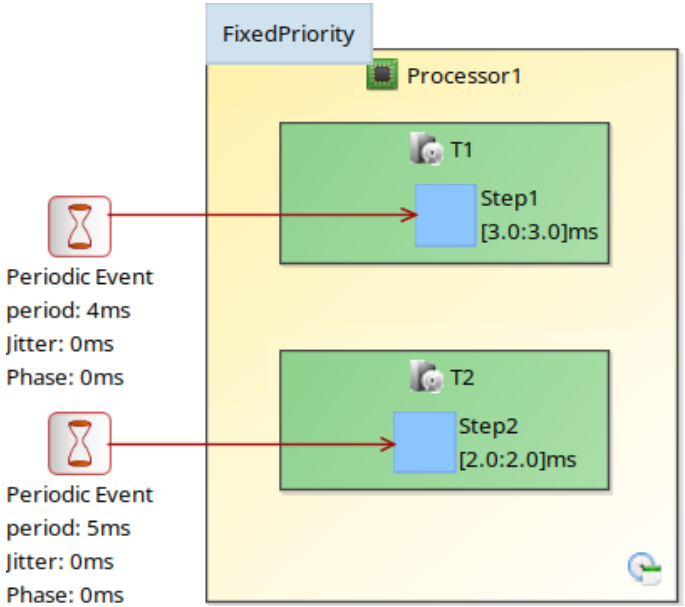
| Task | B | W | D | priority |
|---|---|---|---|---|
| $t_1$ | 3 | 3 | 4 | low |
| $t_2$ | 2 | 2 | 5 | high |

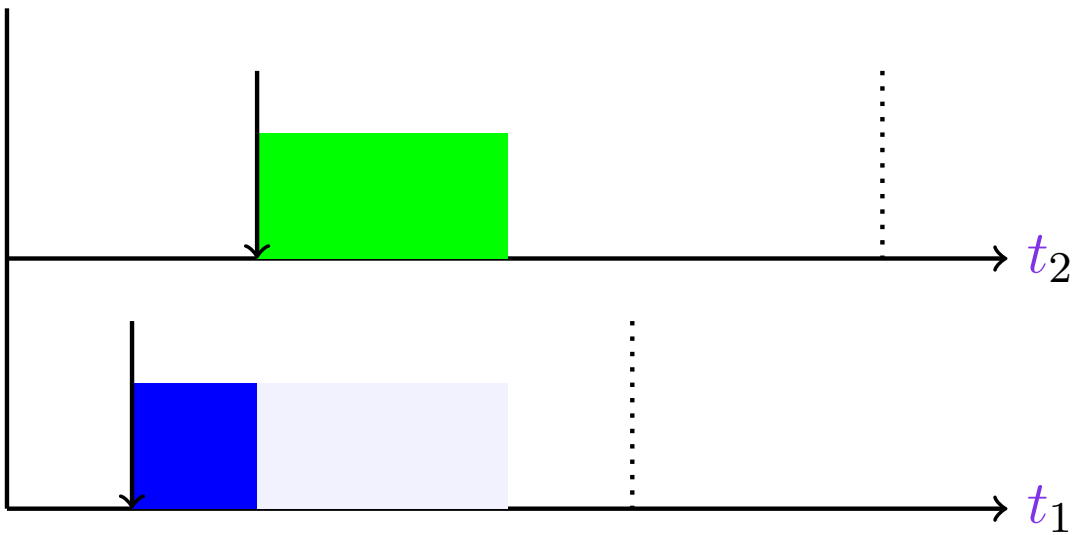


# Example: preemptive fixed priority scheduler (FPS)

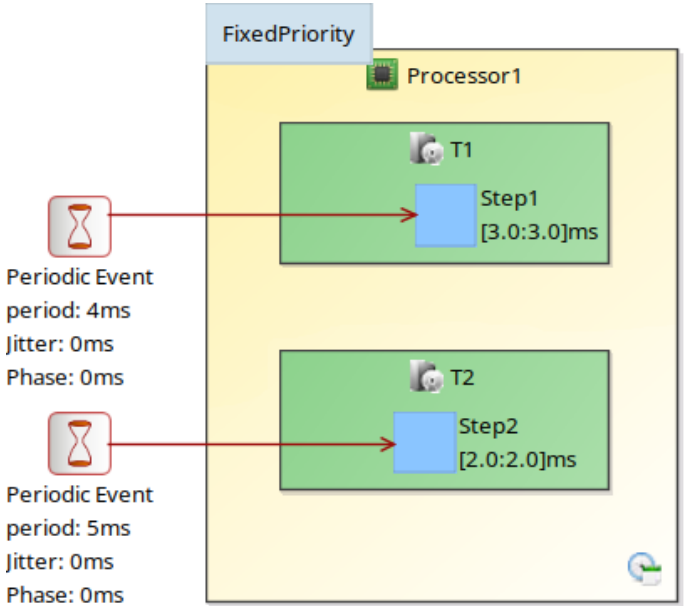
| Task | B | W | D | priority |
|---|---|---|---|---|
| $t_1$ | 3 | 3 | 4 | low |
| $t_2$ | 2 | 2 | 5 | high |

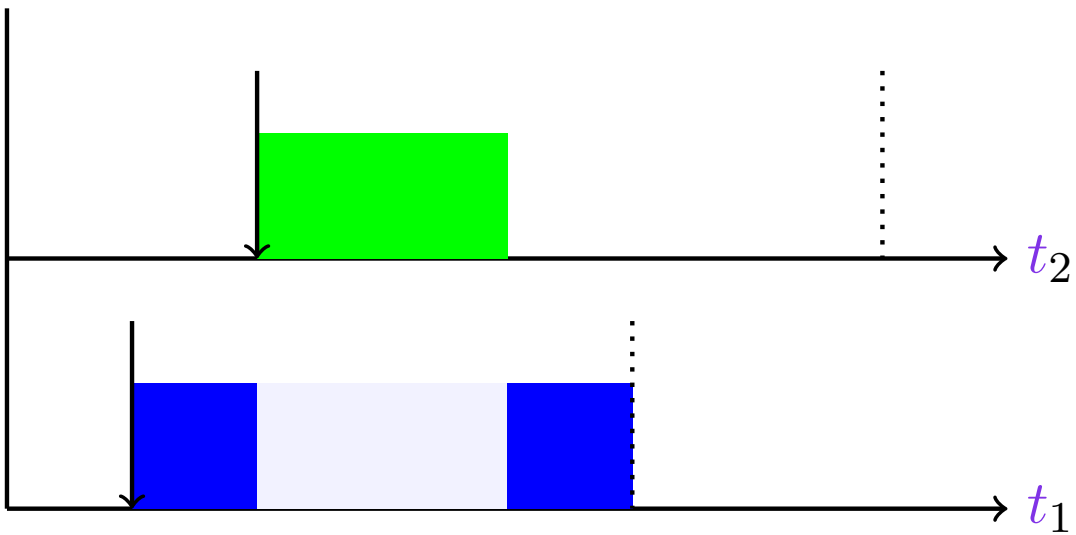


# Example: preemptive fixed priority scheduler (FPS)

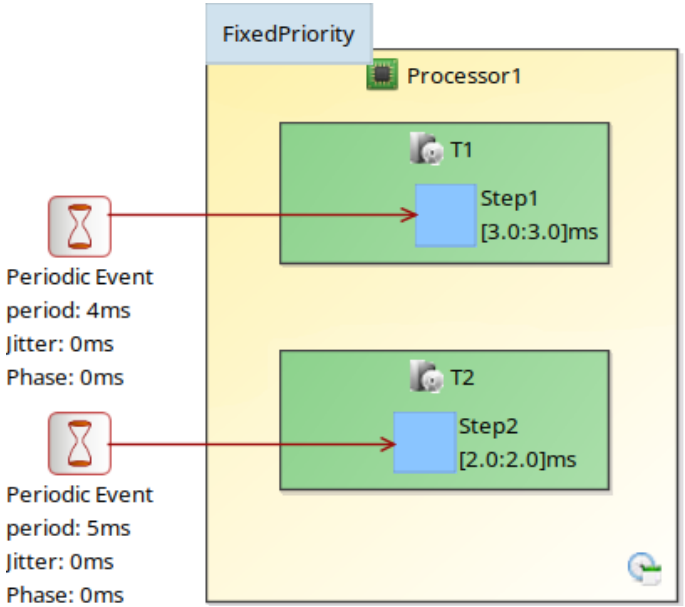
| Task | B | W | D | priority |
|---|---|---|---|---|
| $t_1$ | 3 | 3 | 4 | low |
| $t_2$ | 2 | 2 | 5 | high |

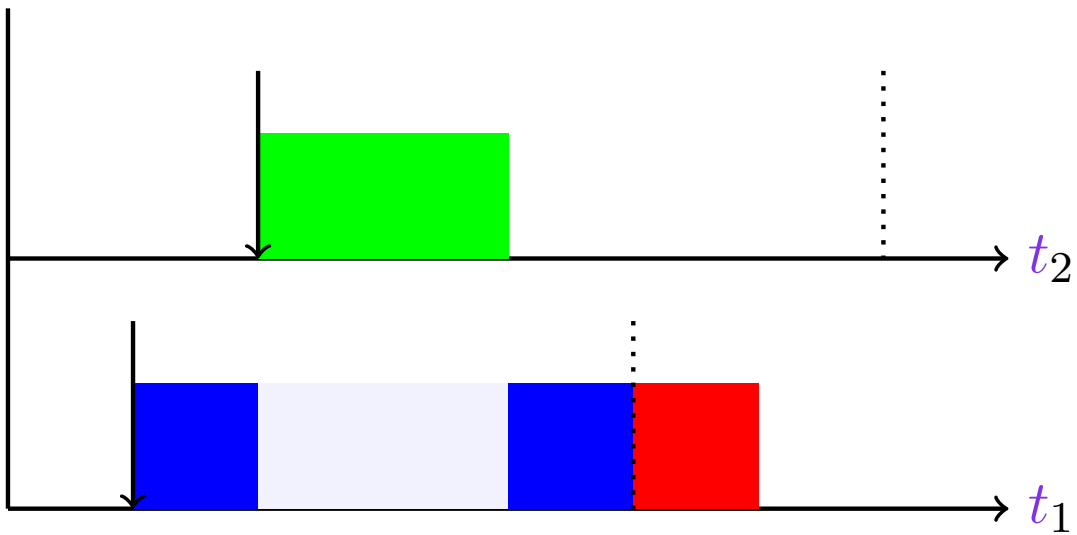


# Example: preemptive fixed priority scheduler (FPS)

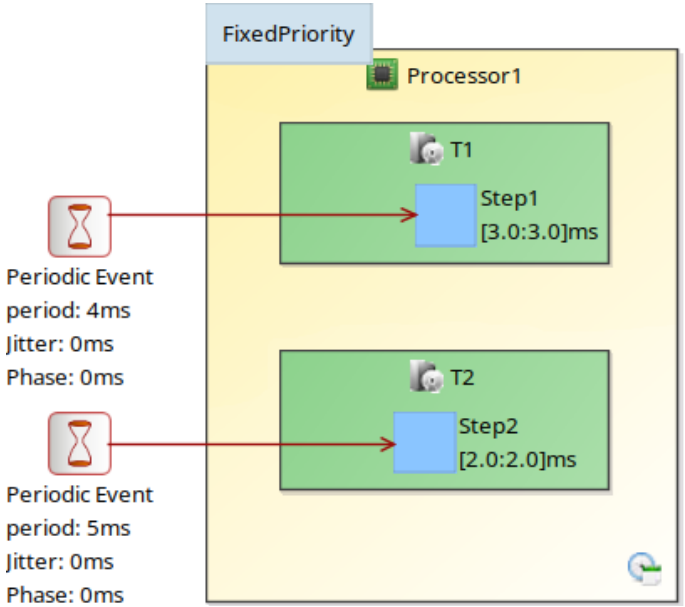
| Task | B | W | D | priority |
|---|---|---|---|---|
| $t_1$ | 3 | 3 | 4 | low |
| $t_2$ | 2 | 2 | 5 | high |

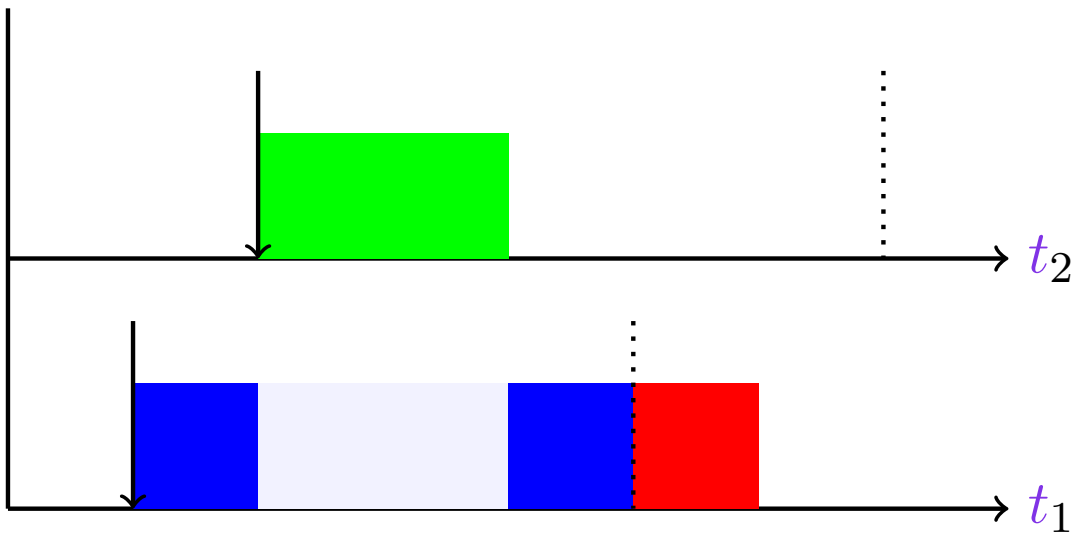


# Example: preemptive fixed priority scheduler (FPS)

| Task | B | W | D | priority |
|---|---|---|---|---|
| $t_1$ | 3 | 3 | 4 | low |
| $t_2$ | 2 | 2 | 5 | high |




Task $t_1$ misses its deadline

# Schedulability analysis

## Definition (schedulability analysis)

Given a real-time system and a scheduling policy for each processor, the schedulability analysis checks whether the system is schedulable (i. e., all tasks meet their deadline) *for all possible behaviors*.

# Schedulability analysis

**Definition (schedulability analysis)**

Given a real-time system and a scheduling policy for each processor, the schedulability analysis checks whether the system is schedulable (i. e., all tasks meet their deadline) *for all possible behaviors*.

All possible behaviors:

- Depends on the periods, interarrival rates, dependencies between tasks…

# Schedulability analysis

**Definition (schedulability analysis)**

Given a real-time system and a scheduling policy for each processor, the schedulability analysis checks whether the system is schedulable (i. e., all tasks meet their deadline) *for all possible behaviors*.

All possible behaviors:

- Depends on the periods, interarrival rates, dependencies between tasks…

Difficulties:

- distributed (several processors)
- tasks dependencies (potentially between different processors)
- uncertainty

# Outline

# Thales

Thales: A multinational company with 80,000 employees in 68 countries

- Digital identity and security
- Ground transportation
- Defense and security
- Space and aerospace

A key focus on R&D

- 1 G € R&D in 2018
- Total sales in 2018: 19 G €

# Time4sys: A Pivot Model

## Objective of Time4sys

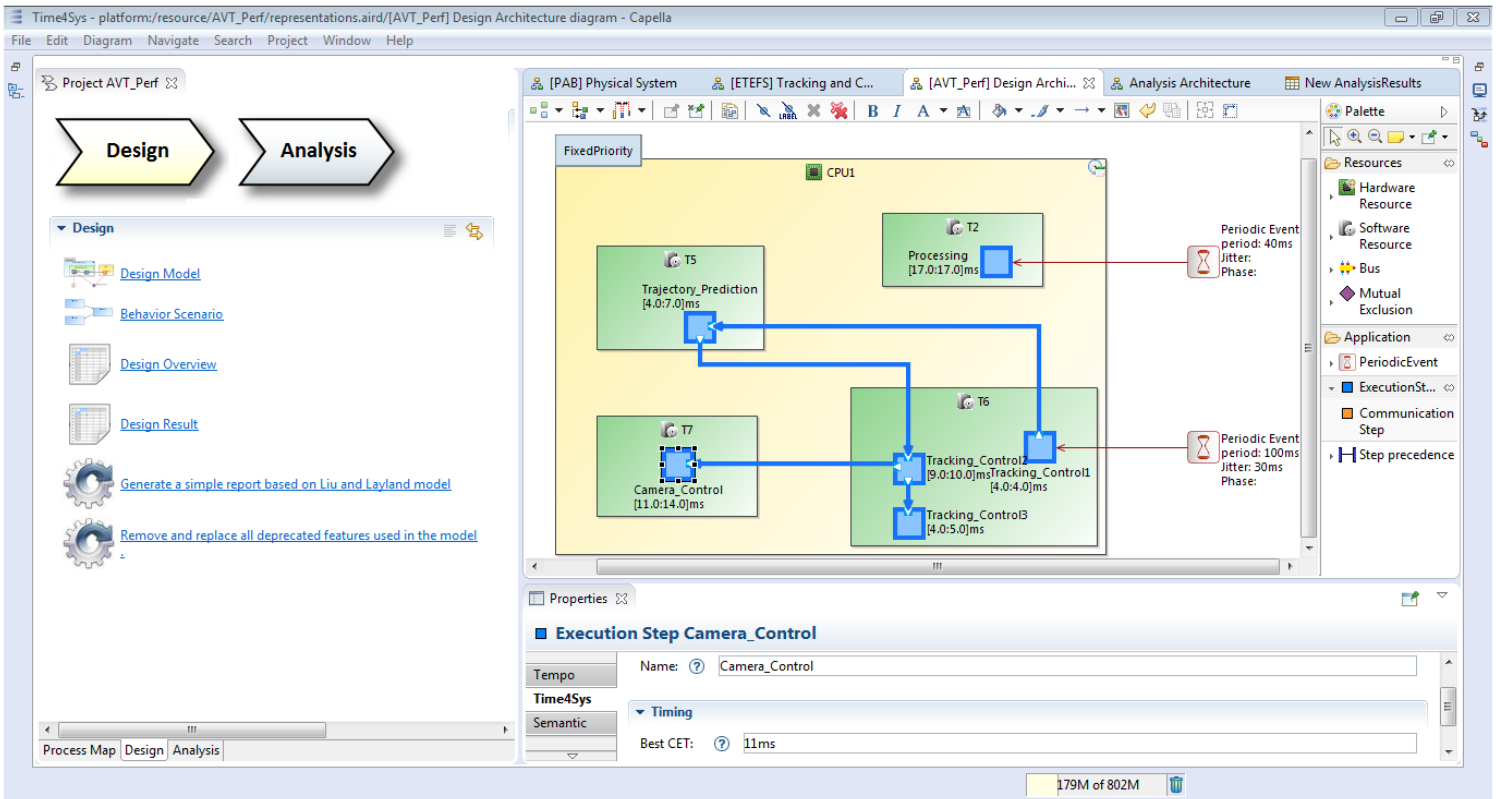
Fill the gap between the capture of timing aspects in the design phase of a real-time system and the ability of specific/dedicated tools to verify the consistency and performances of a given scheduling

- Developed by Thales
- Entirely open-source
- Time4Sys Design model uses a subset of the MARTE OMG standard

# Time4sys: A graphical user interface

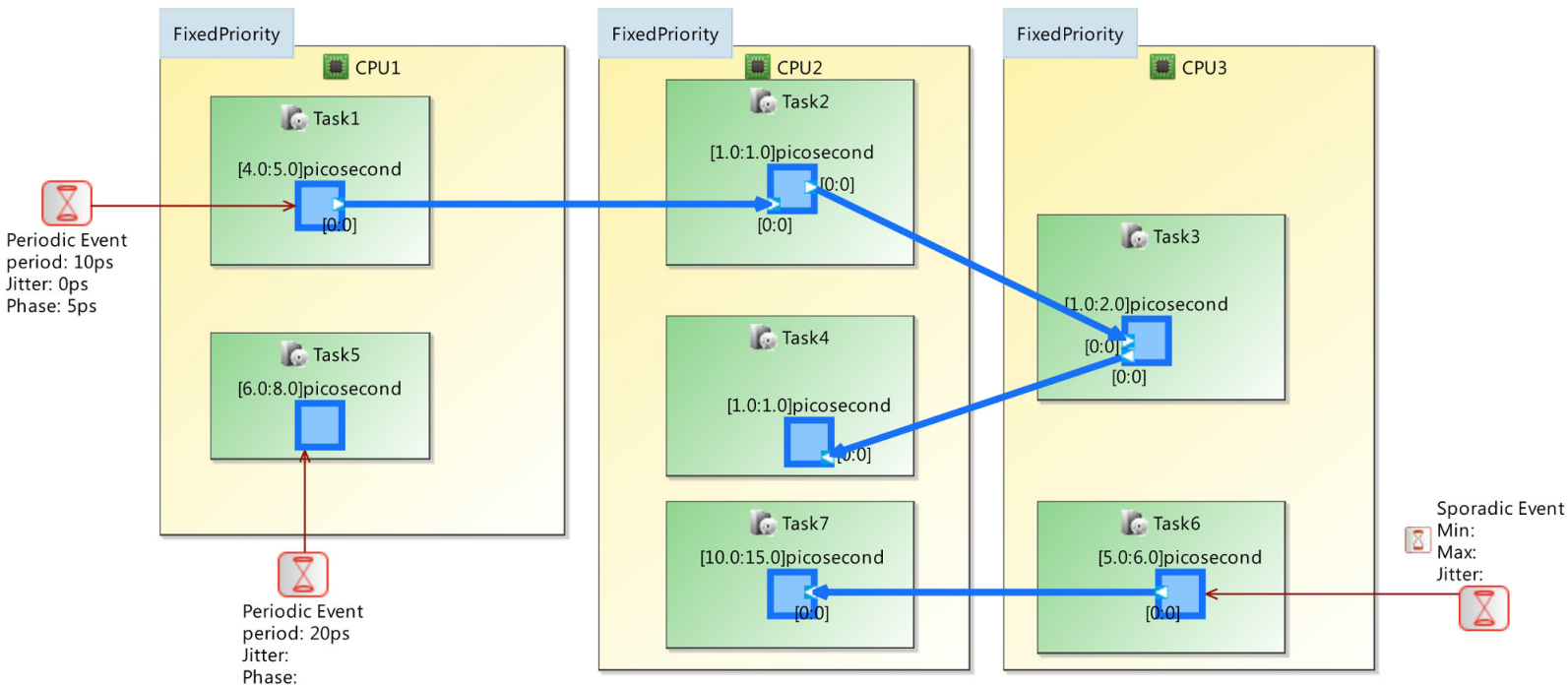
Comes in the form of an Eclipse plugin

- Java module based on Sirius

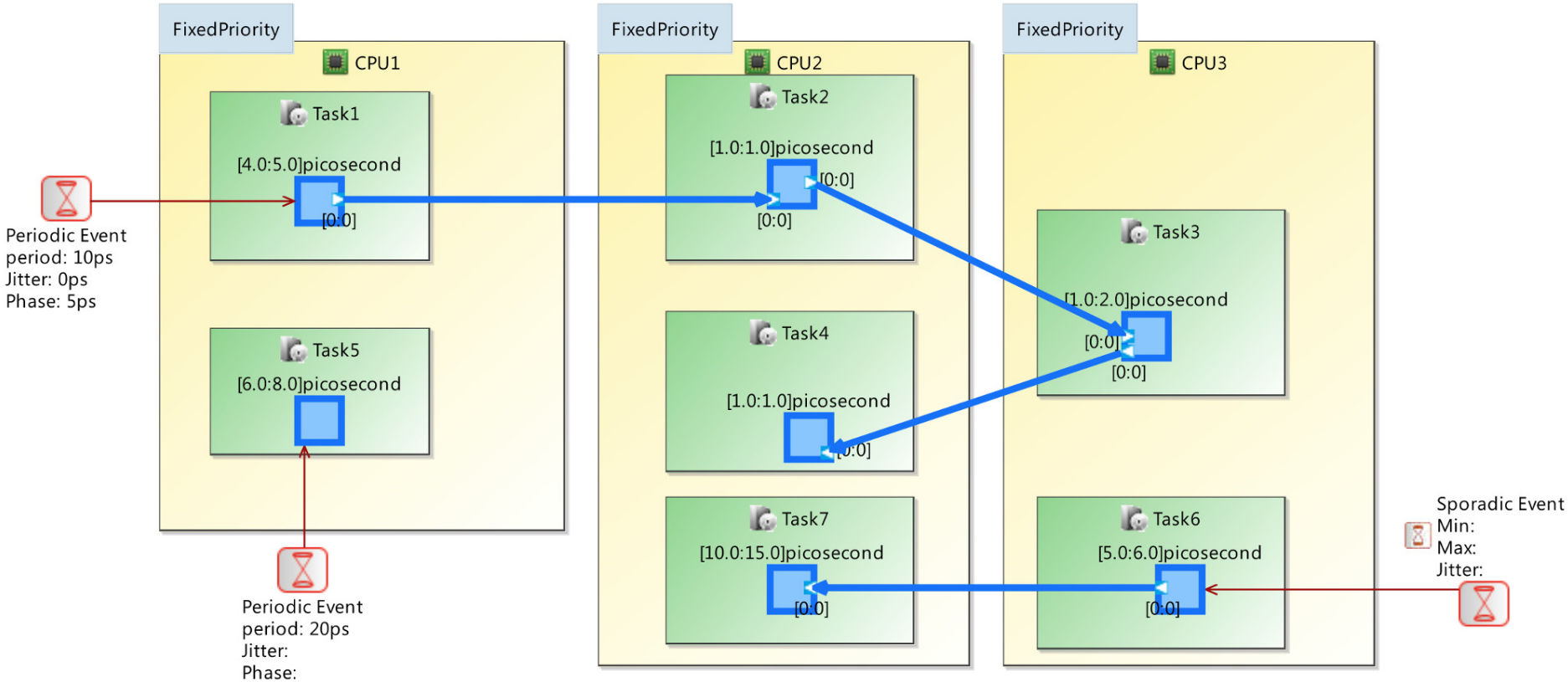


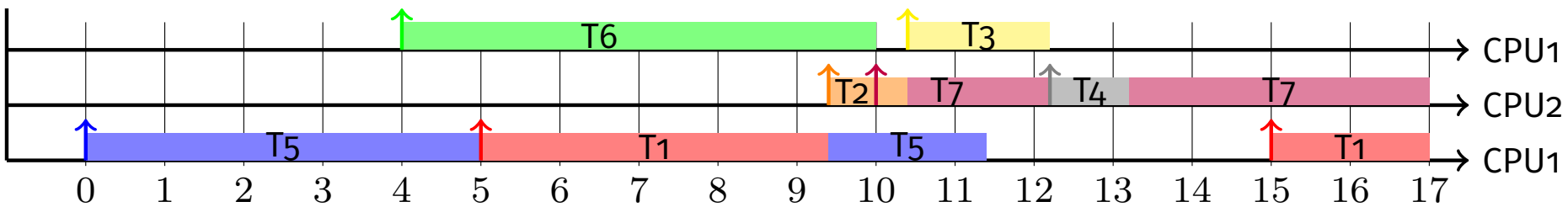
# Time4sys: features

- Uniprocessor or multiprocessor
- Different scheduling policies
  - EDF, FPS, SJF, …
- Rich task dependency mechanisms
  - Task chains: activation of a task upon completion of a previous task

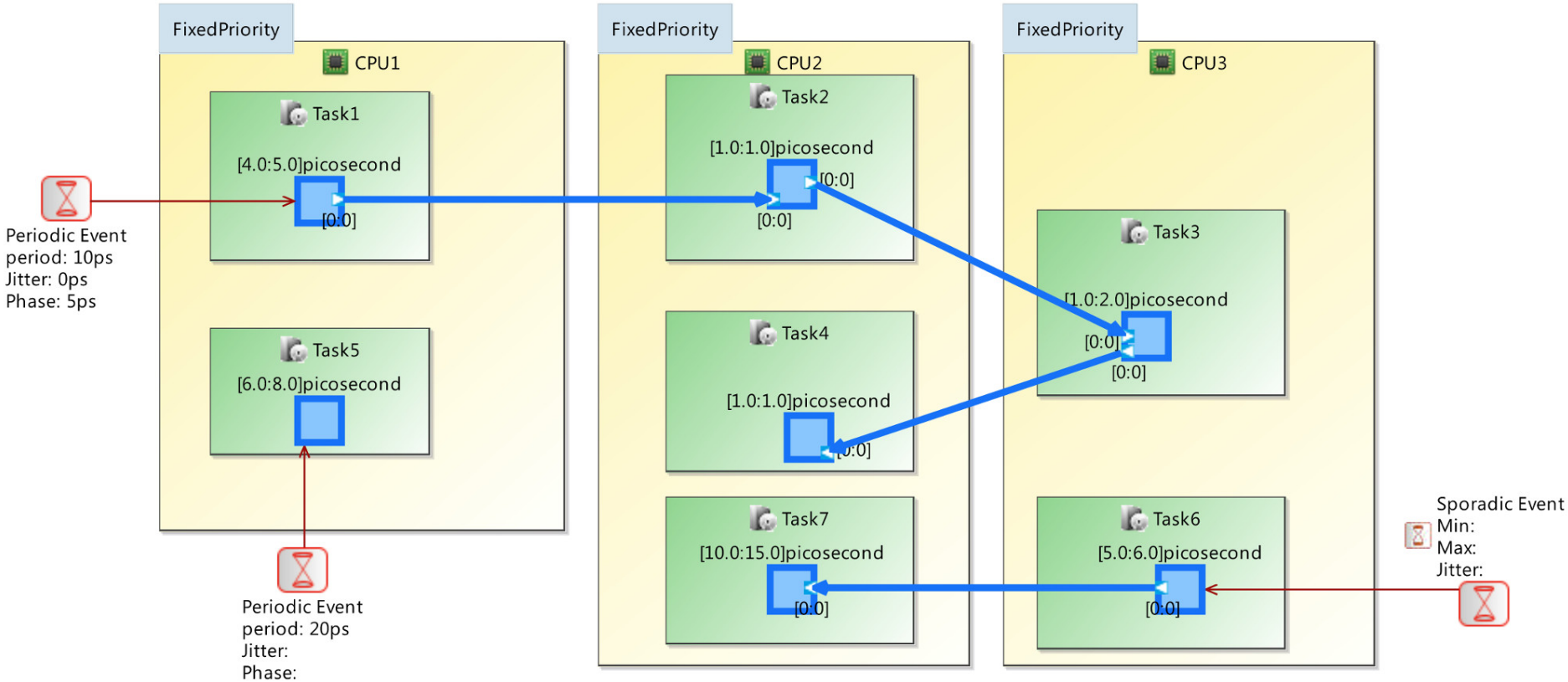


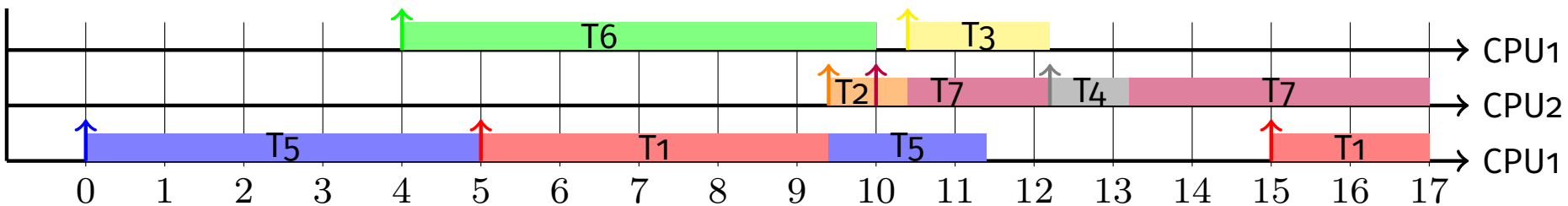
# Time4sys: Complex systems



# Time4sys: Complex systems




Is this system schedulable?

# Outline

# Time4sys: no semantics

**Problem**

Time4sys features no formal semantics

Can be used to model real-time systems, but not to execute, test or formally verify them

# Time4sys: no semantics

## Problem

Time4sys features no formal semantics

Can be used to model real-time systems, but not to execute, test or formally verify them

## Preliminary objective

First challenge: formalize Time4sys

Existing translations to tools such as Cheddar

# Problem: schedulability analysis under uncertainty

Problem: what if some timing constants (deadlines, execution times, periods, interarrival times…) are unknown or known with a limited precision?

# Problem: schedulability analysis under uncertainty

Problem: what if some timing constants (deadlines, execution times, periods, interarrival times…) are unknown or known with a limited precision?

## Objective

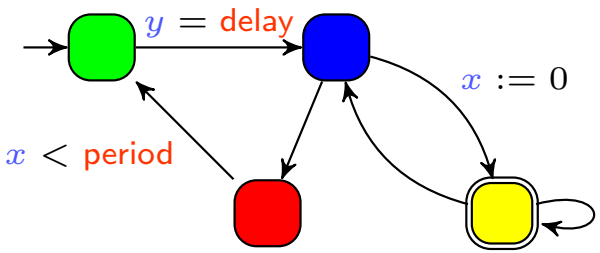
Formalize Time4sys so as to allow for schedulability analysis of real-time systems under uncertainty

# Outline

# Model checking timed concurrent systems

- Use formal methods

[Baier and Katoen, 2008]




A model of the system

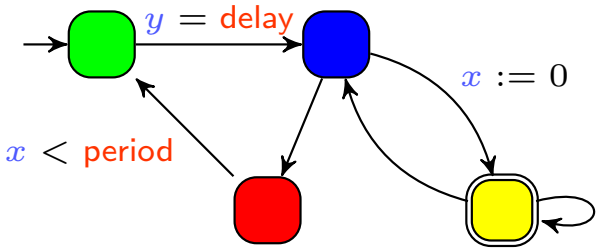
is unreachable

A property to be satisfied

# Model checking timed concurrent systems

- Use formal methods

[Baier and Katoen, 2008]




A model of the system

?

$\models$

is unreachable

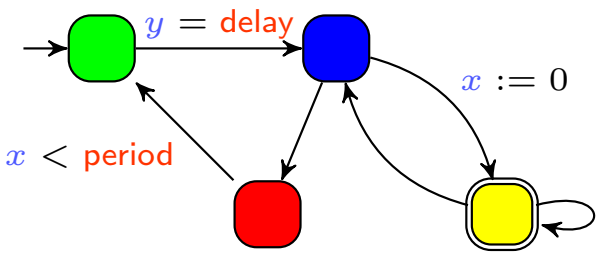
A property to be satisfied

- Question: does the model of the system satisfy the property?

# Model checking timed concurrent systems

- Use formal methods

[Baier and Katoen, 2008]




A model of the system

$\stackrel{?}{\models}$

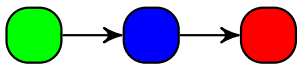

is unreachable

A property to be satisfied

- Question: does the model of the system satisfy the property?

**Yes**

**No**

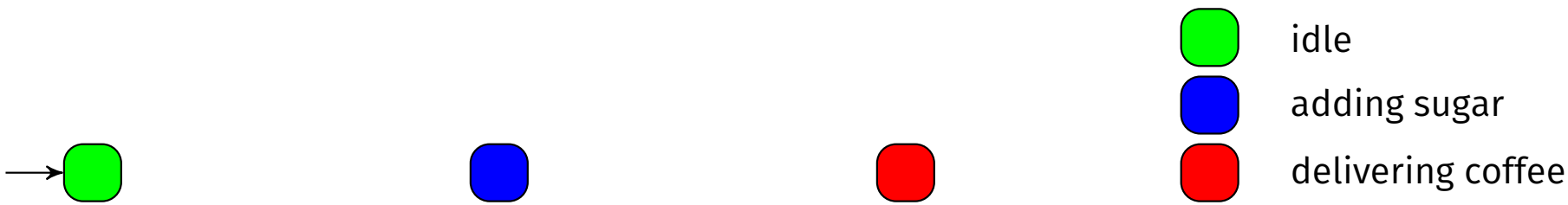


Counterexample

Turing award (2007) to Edmund M. Clarke, Allen Emerson and Joseph Sifakis

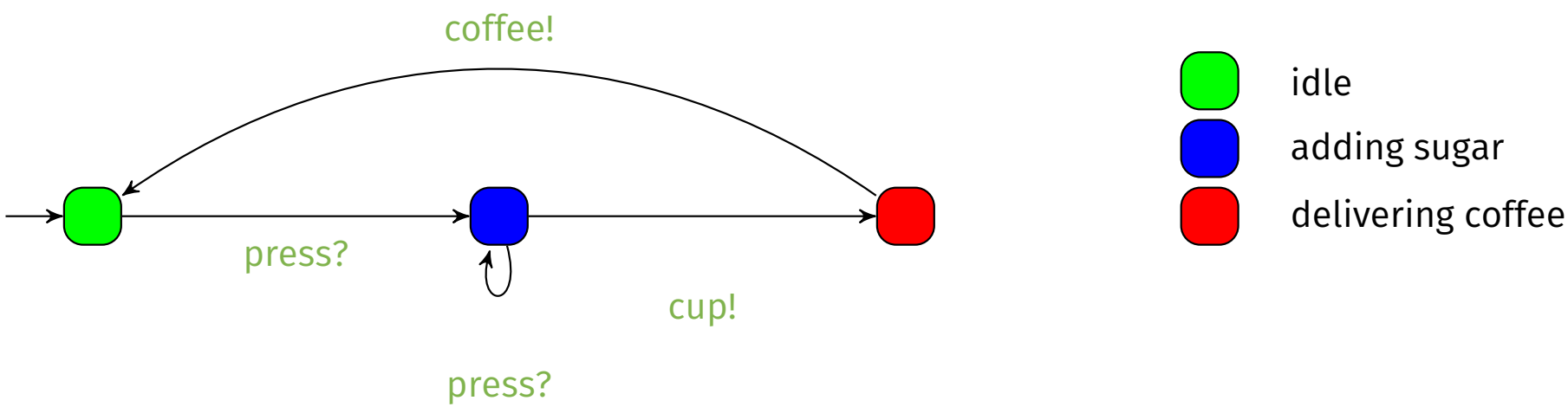
# Timed automaton (TA)

- Finite state automaton (sets of locations)



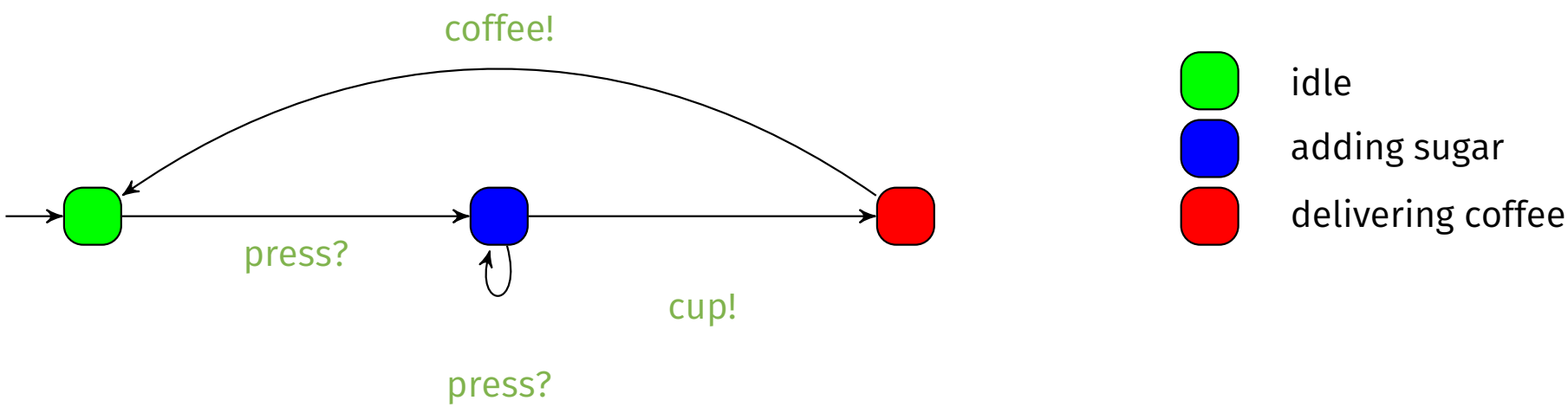
# Timed automaton (TA)

- Finite state automaton (sets of locations and actions)



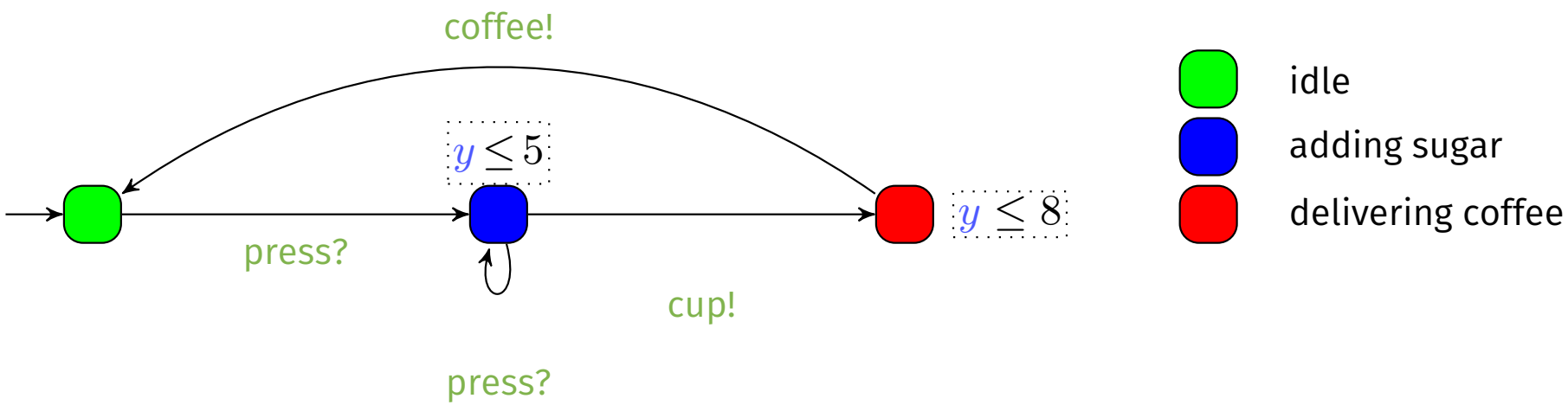
# Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate



# Timed automaton (TA)

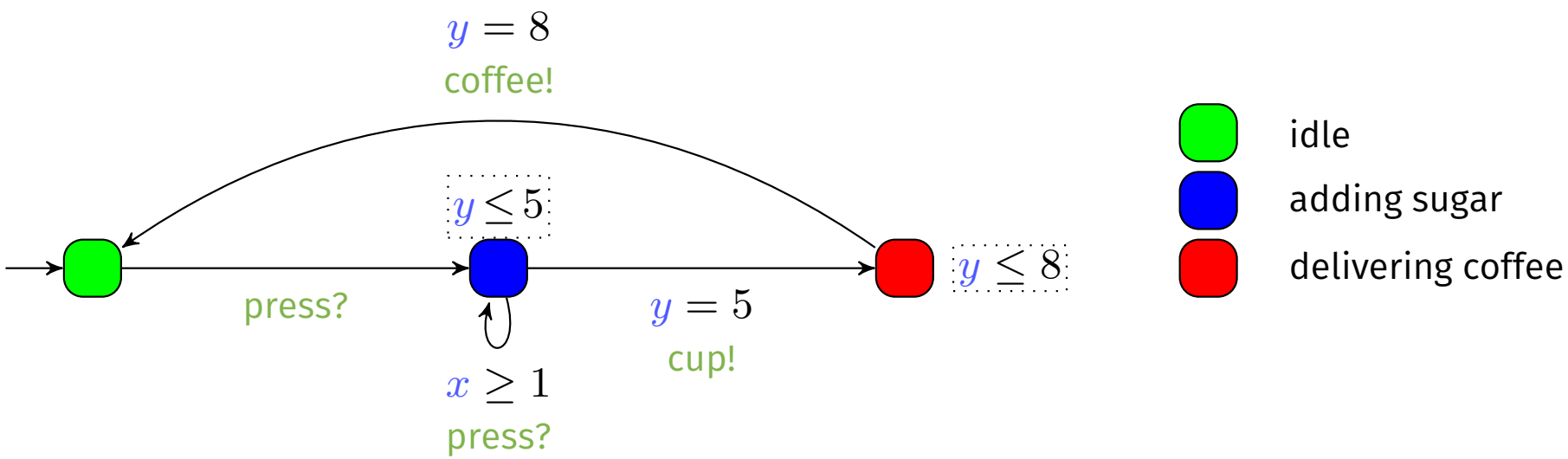
- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants
- Features
  - Location invariant: property to be verified to stay at a location



# Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

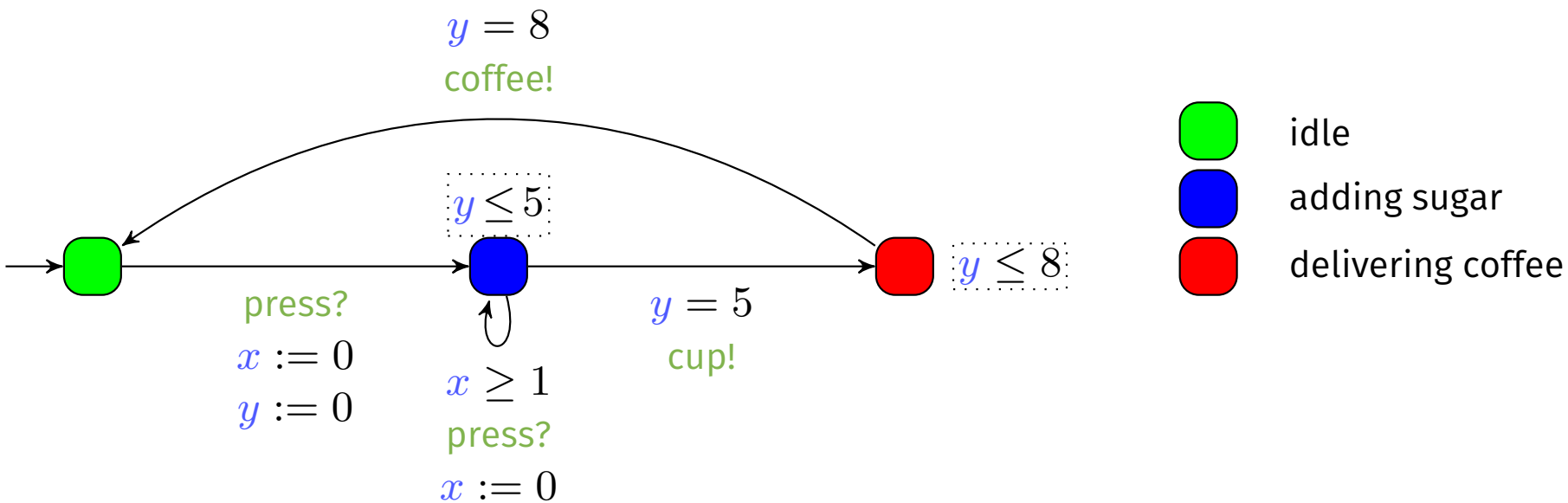
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition



# Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to $0$ along transitions



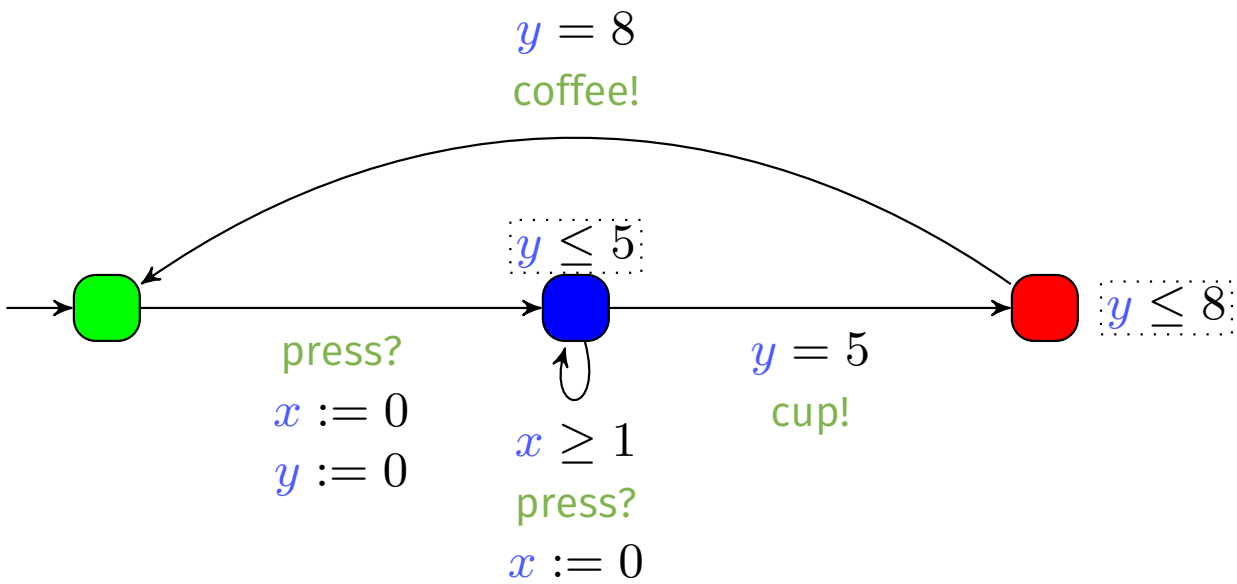
# Concrete semantics of timed automata

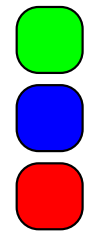
- Concrete state of a TA: pair $(l, w)$, where
  - $l$ is a location,
  - $w$ is a valuation of each clock

  Example: $\left(\text{  }, \begin{pmatrix} x=1.2 \\ y=3.7 \end{pmatrix}\right)$

- Concrete run: alternating sequence of concrete states and actions or time elapse

# The most critical system: The coffee machine



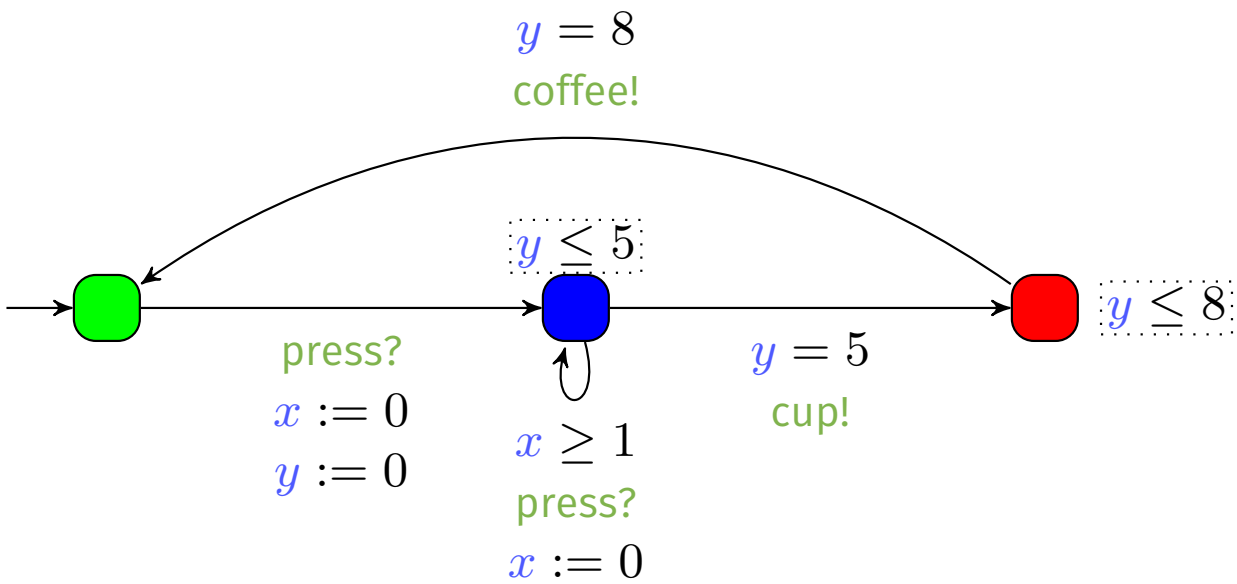



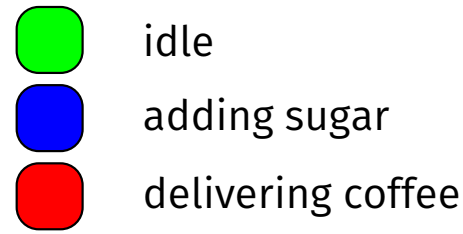
idle

adding sugar

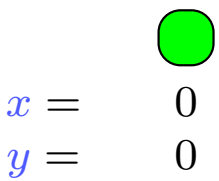
delivering coffee

# The most critical system: The coffee machine

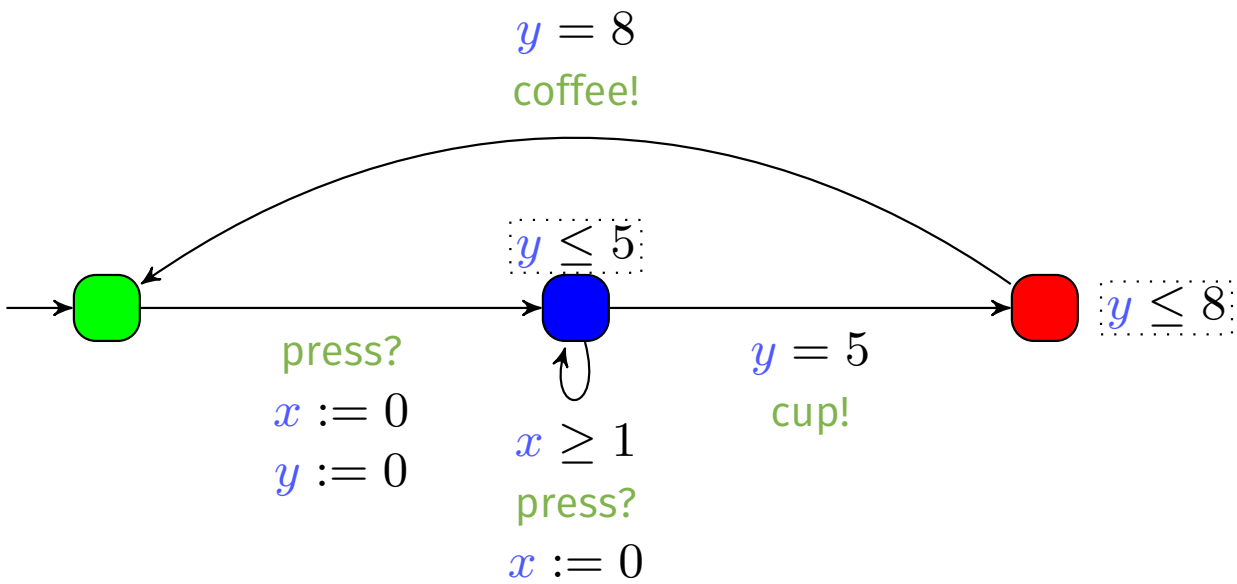


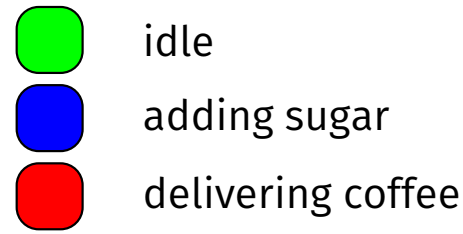




- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

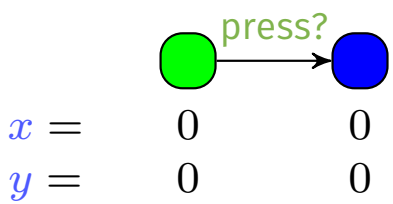


# The most critical system: The coffee machine

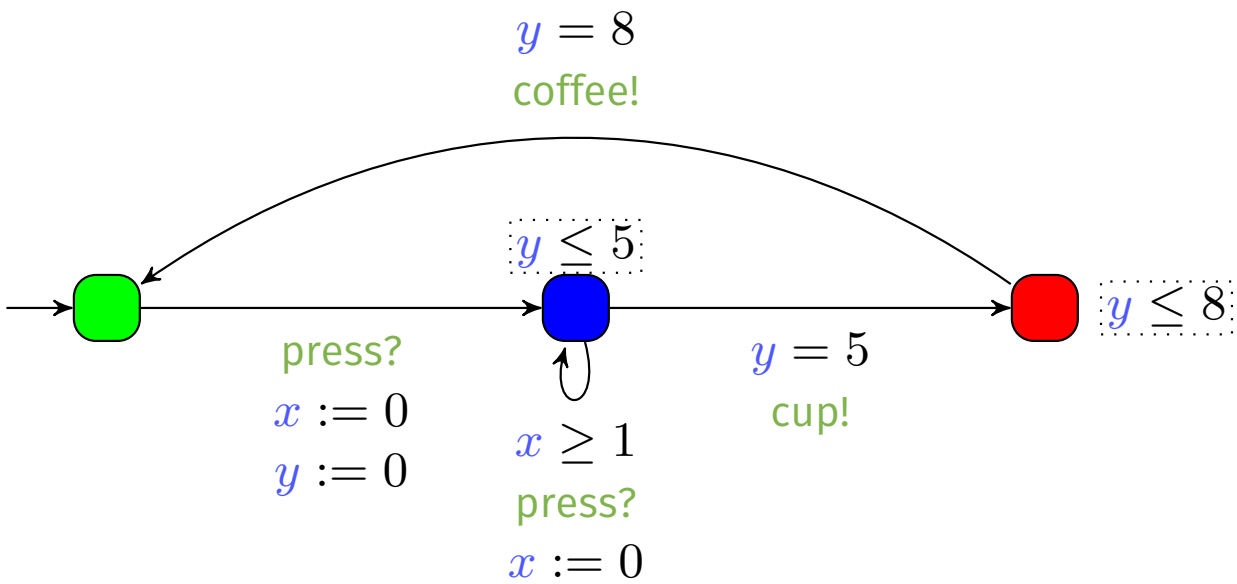


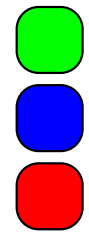




- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



# The most critical system: The coffee machine



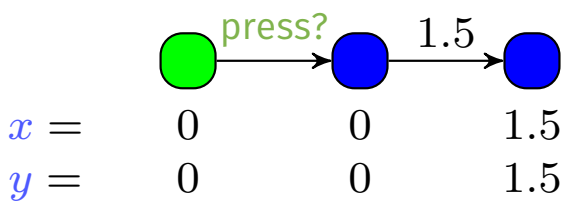



idle

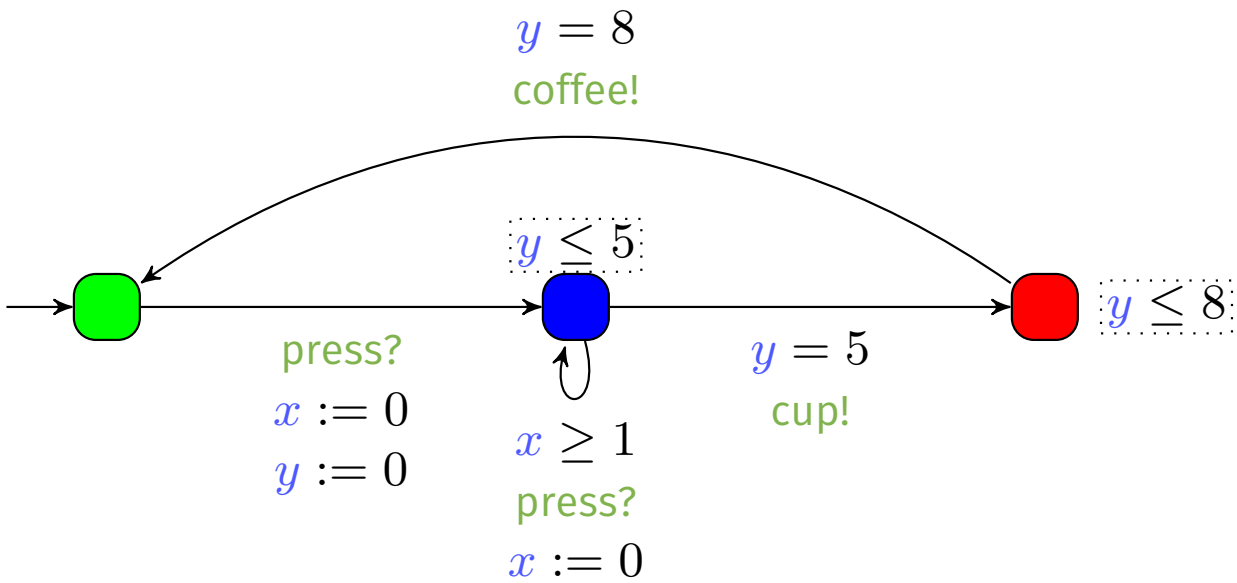
adding sugar

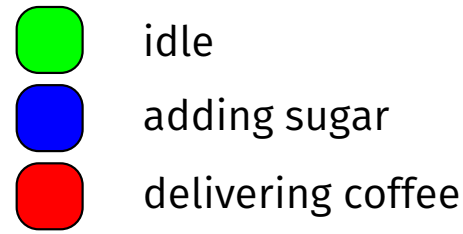
delivering coffee

- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

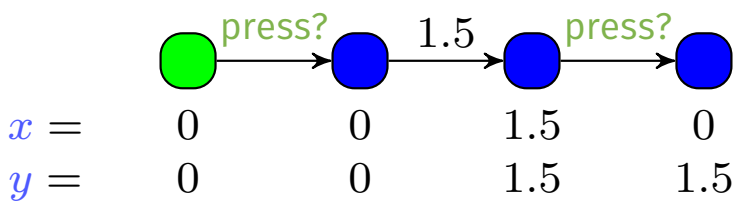


# The most critical system: The coffee machine

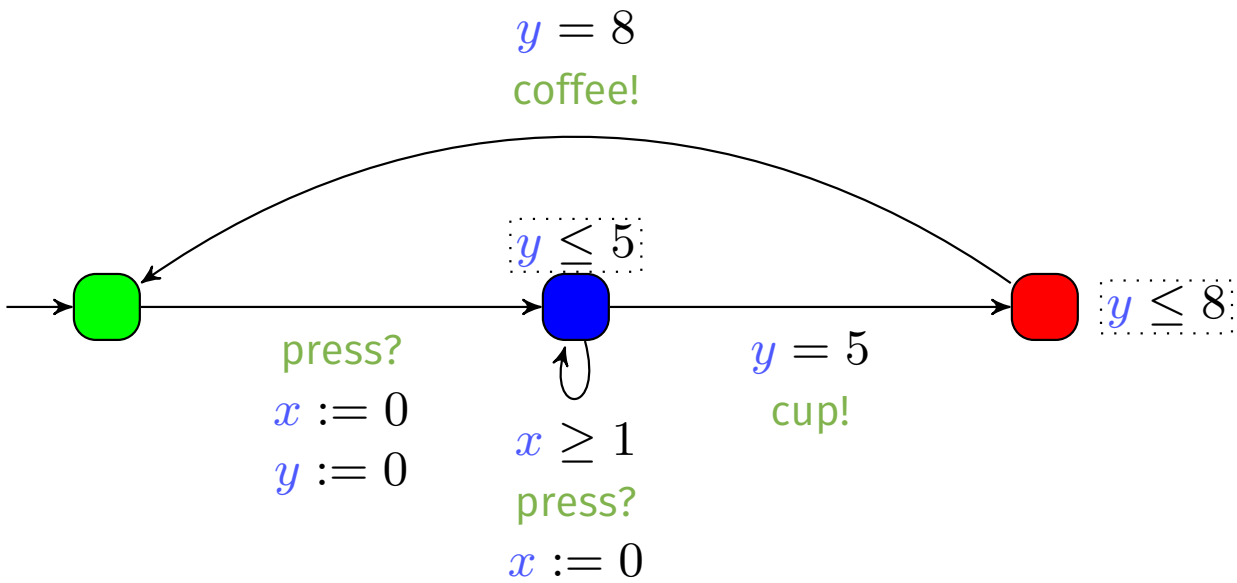


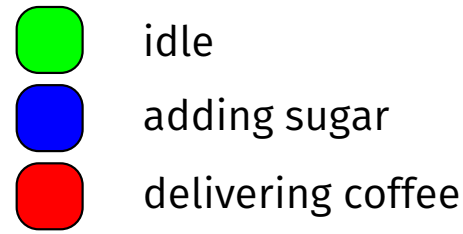




- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

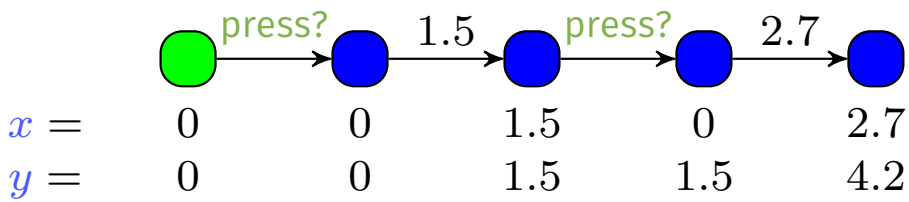


# The most critical system: The coffee machine

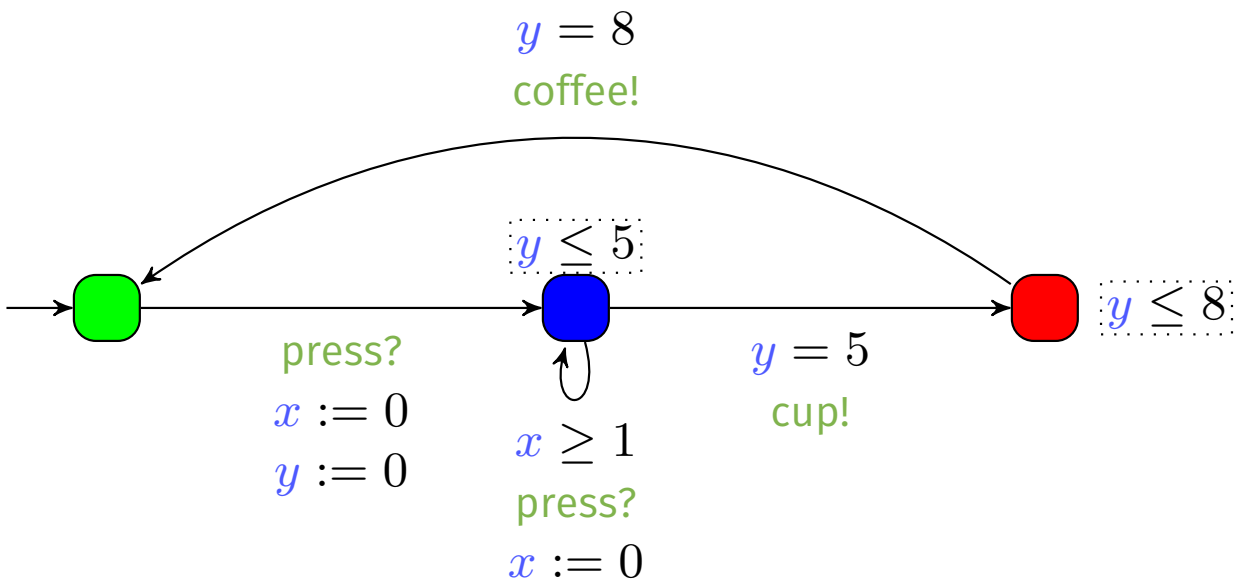


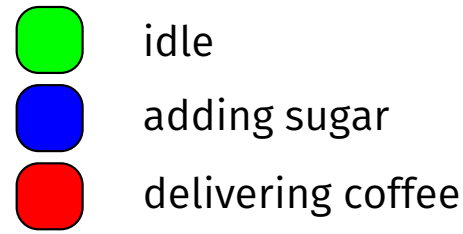




- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

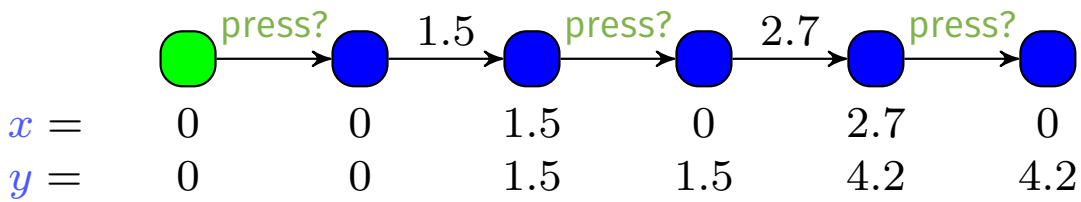


# The most critical system: The coffee machine

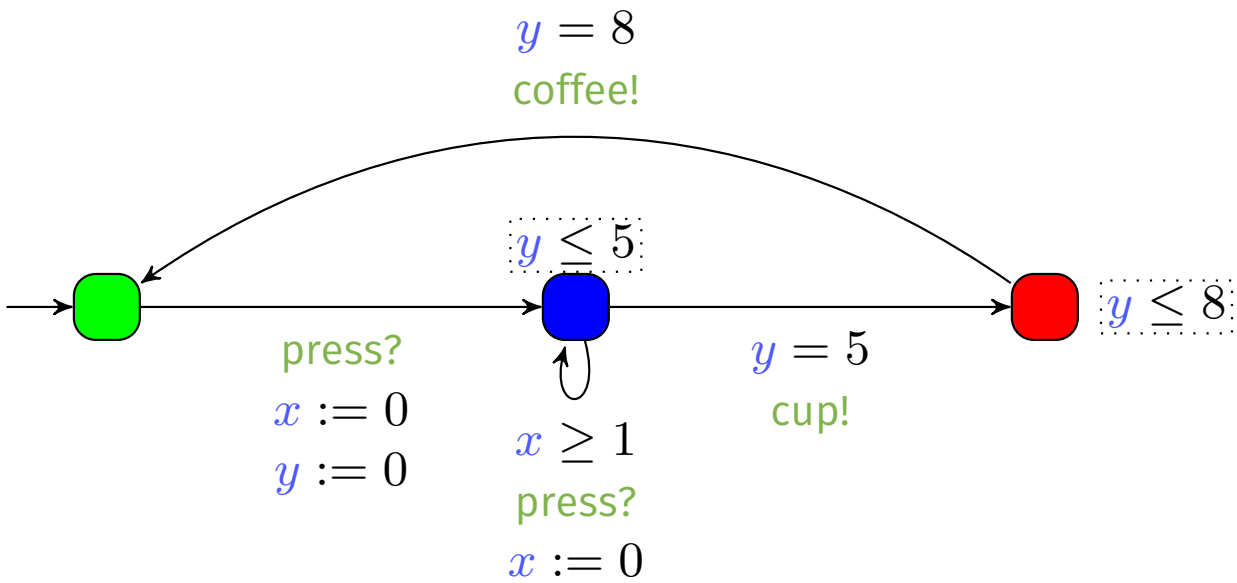


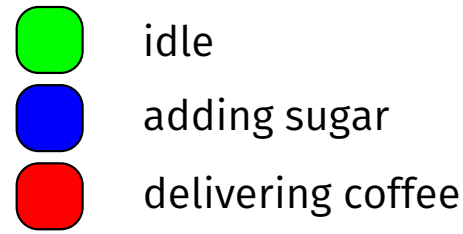




- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

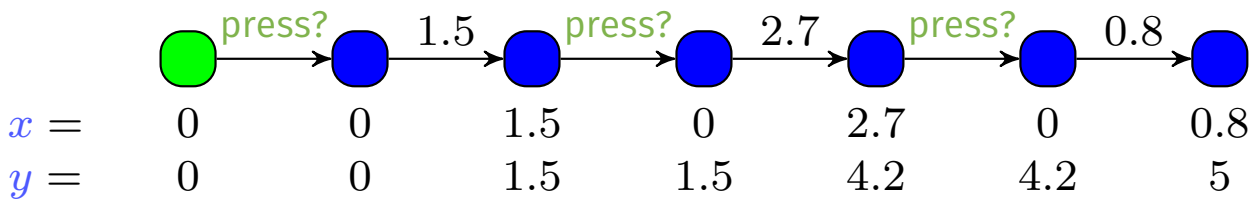


# The most critical system: The coffee machine

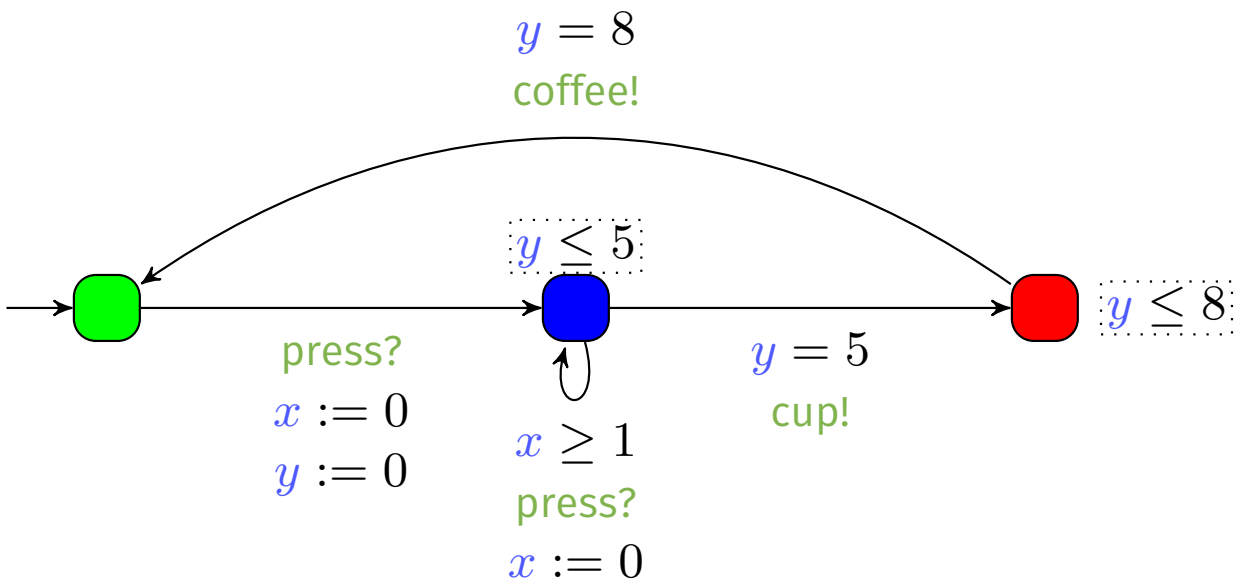


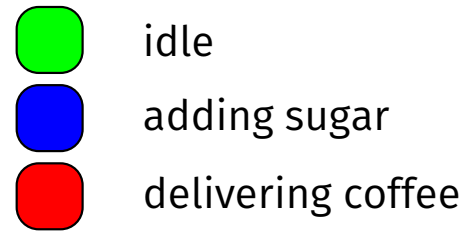




- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

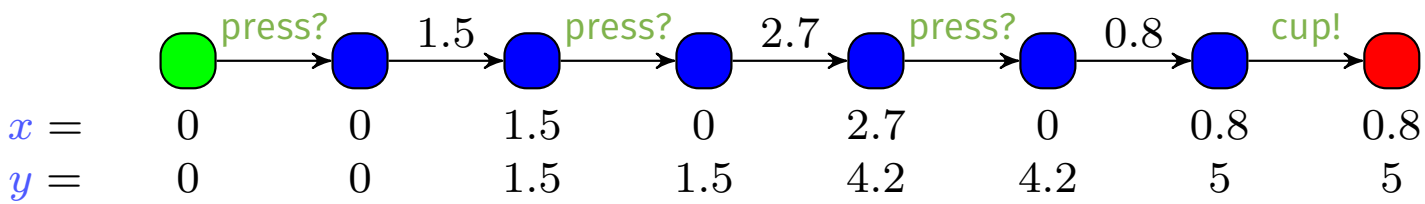


# The most critical system: The coffee machine

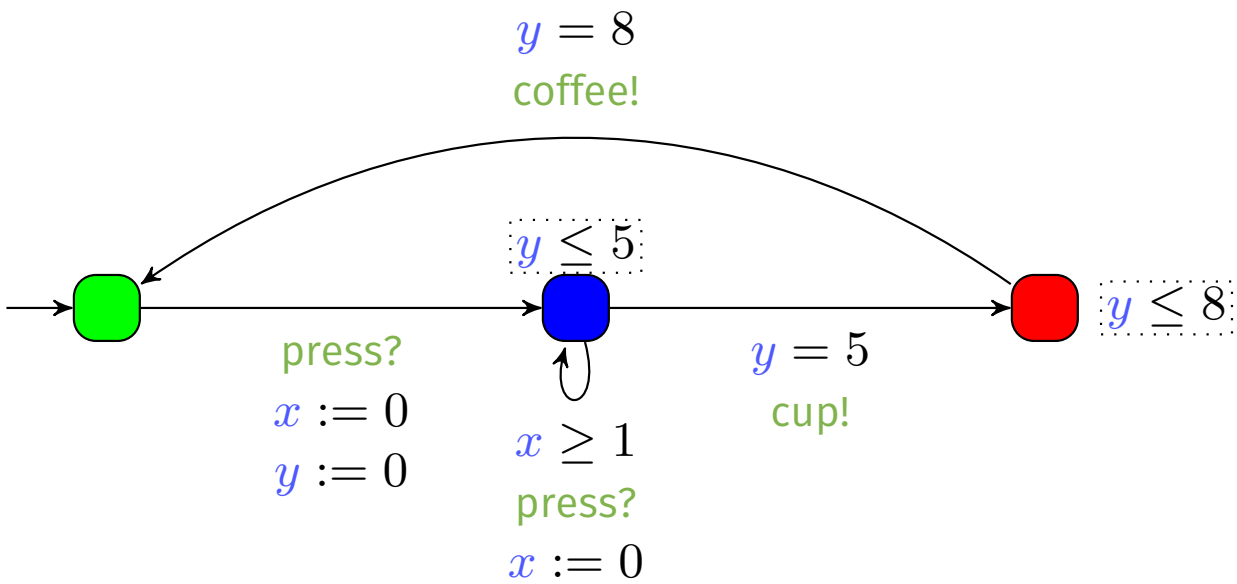


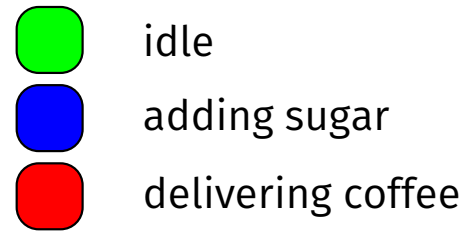




- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

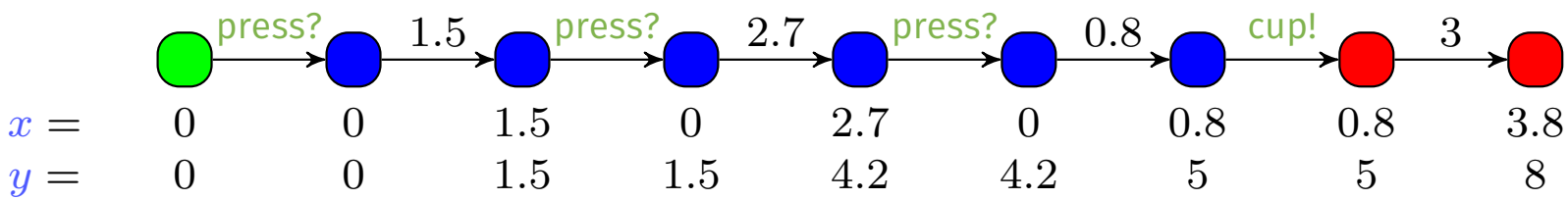


# The most critical system: The coffee machine

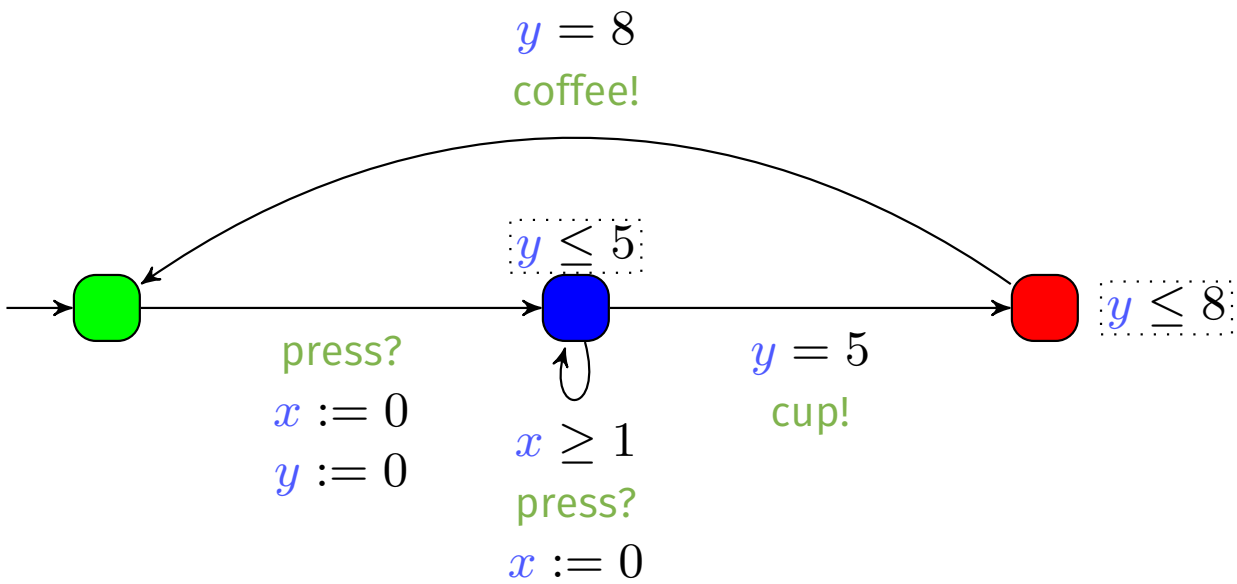


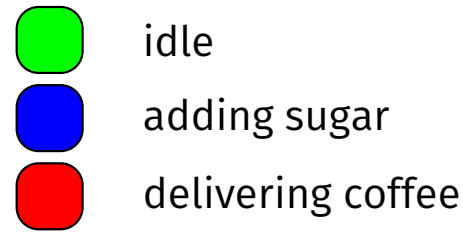




- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

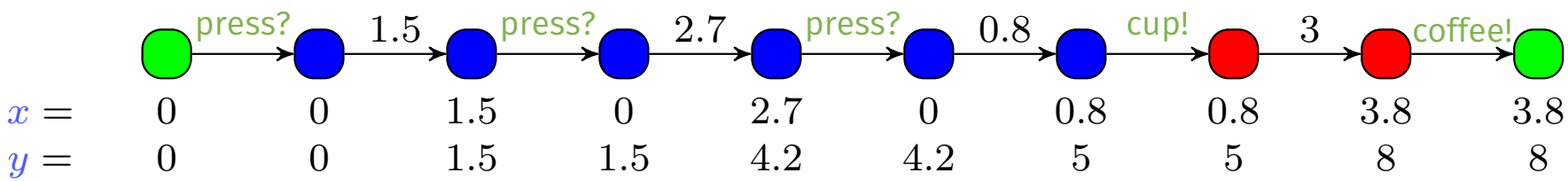


# The most critical system: The coffee machine

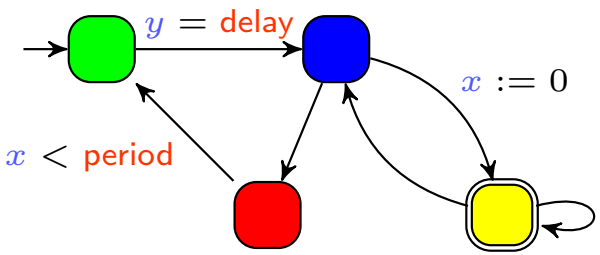



- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



# timed model checking




A model of the system

$?$

$\models$

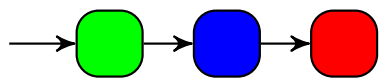
is unreachable

A property to be satisfied

- Question: does the model of the system satisfy the property?

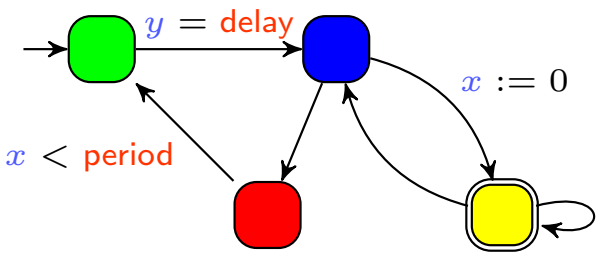
**Yes**

**No**



Counterexample

# Parametric timed model checking




A model of the system

$?$

$\models$

is unreachable

A property to be satisfied

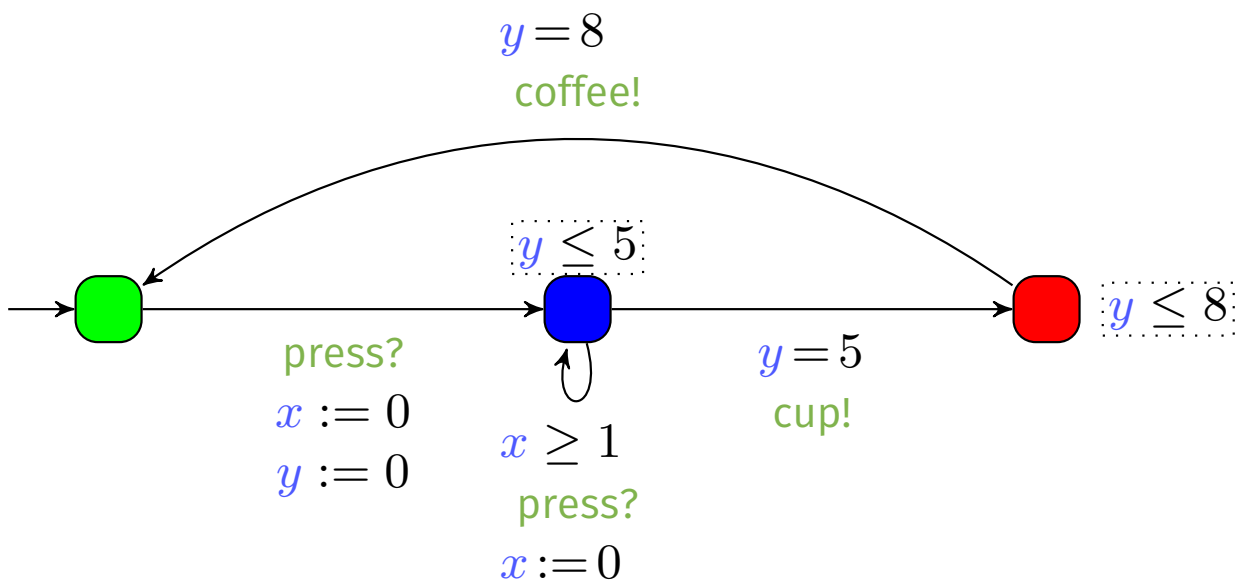
- Question: for what values of the parameters does the model of the system satisfy the property?

**Yes if...**

$$2\text{delay} > \text{period}$$
$$\wedge\ \text{period} < 20.46$$

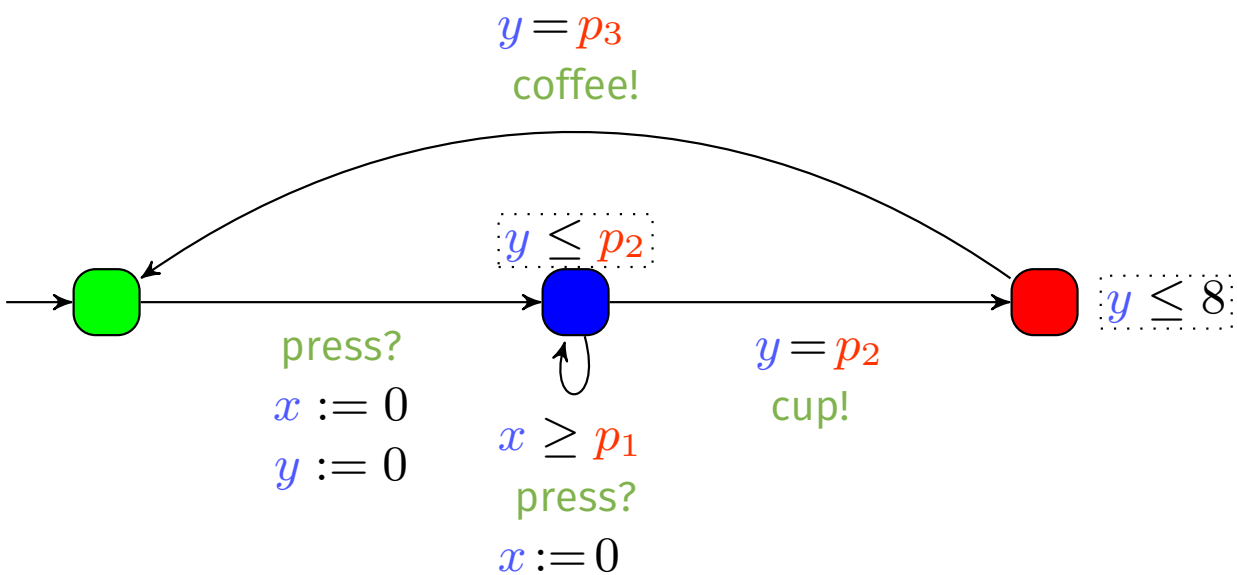
# Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks)



# Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks) augmented with a set $P$ of parameters [Alur et al., 1993]
  - Unknown constants compared to a clock in guards and invariants



# Outline

# Objective and methodology

Objective: translate Time4sys into parametric timed automata

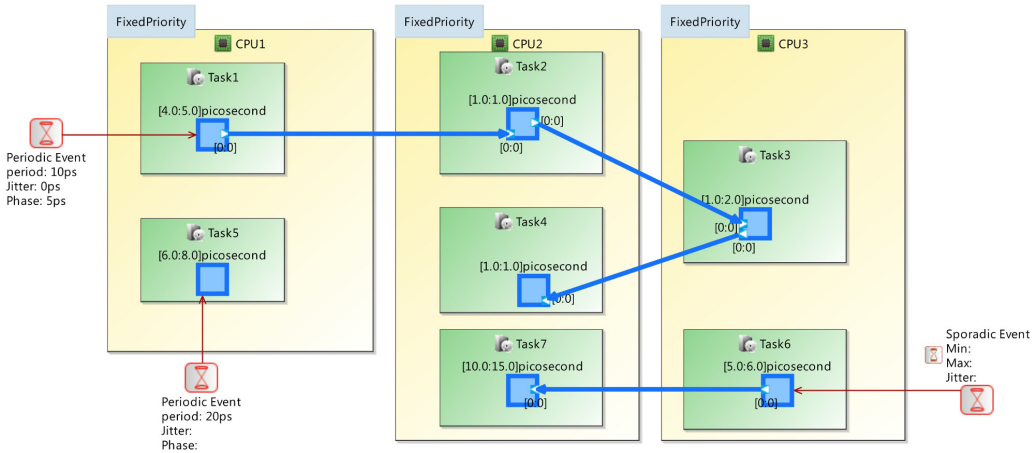
Schedulability analysis reduces to reachability synthesis

## General scheme

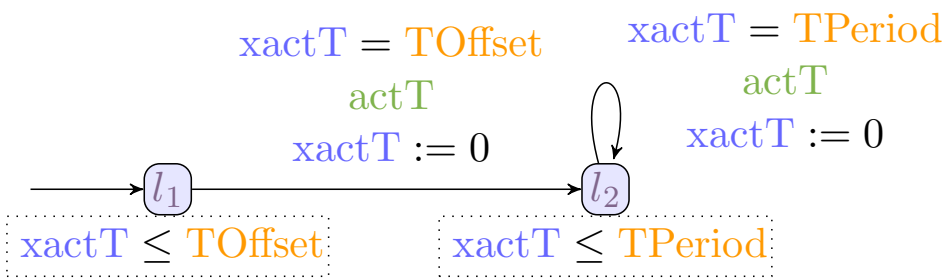
- Translate each task activation pattern (sporadic, periodic) into a PTA
- Translate each precedence constraint (task chain) into a (set of) PTA
- Translate the scheduling policy of each processor into a PTA
- Synchronization between these PTAs by synchronization on actions

# Outline

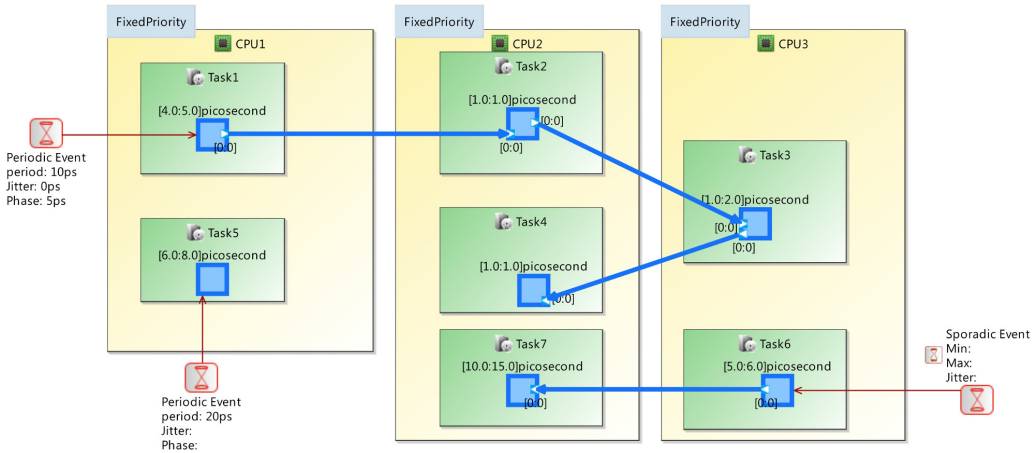
# Translating activation patterns



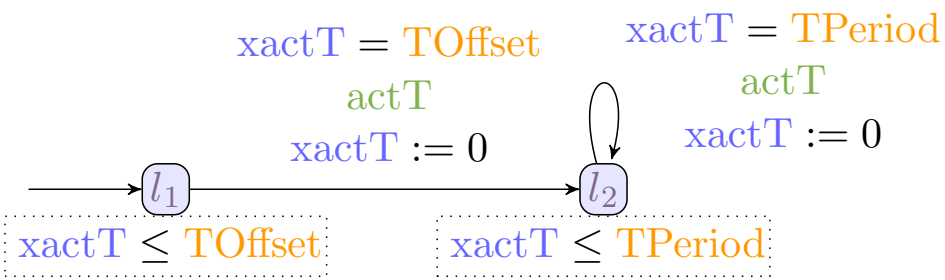

Translating a periodic task:



# Translating activation patterns




Translating a periodic task:




Sporadic task: identical, without invariant in $l_2$ and $\mathrm{xactT} = \mathrm{TPeriod}$ becomes $\mathrm{xactT} \geq \mathrm{TIAT}$

(see paper)

# Translating precedence constraints

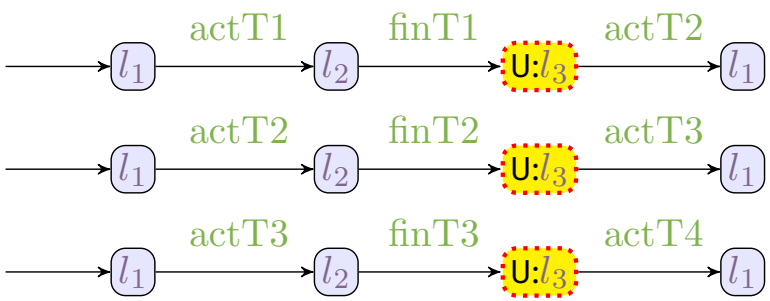
Objectives:

- ensure that, upon completion of a task, the instance of the following task is immediately created
- do not constrain the relative order between the tasks creations and completions

Method:

- Decompose the task chain, and generate one PTA per dependency
- Add urgent locations to enforce immediate activation

Translating task chain $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$:

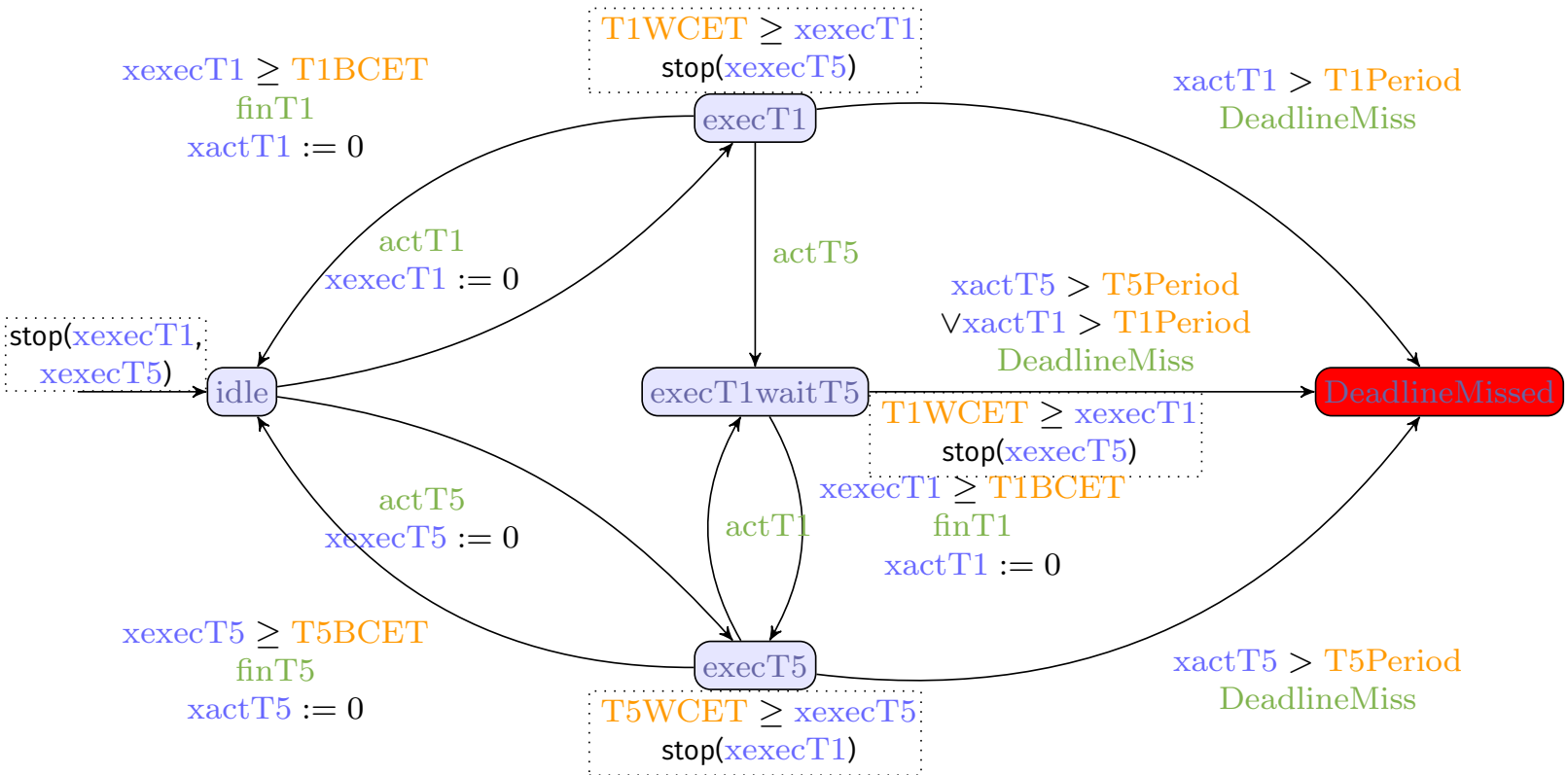


# Translating scheduling policies

Not intrinsically difficult:

- some attempts in the literature [Fersman et al., 2007, Sun et al., 2013]
- manually error prone, but can be automated (reasonably) easily

Example: Translating the scheduler CPU1 with a preemptive FPS scheduling policy



# Reduction to reachability synthesis

During the translation, we define a set of “bad locations”:

- Corresponding to deadline misses on the various CPUs

# Reduction to reachability synthesis

During the translation, we define a set of “bad locations”:

- Corresponding to deadline misses on the various CPUs

**Fact**

The values for which the real-time system is schedulable are exactly the values of the timing parameters of the translated PTA for which this set of bad locations cannot be reached (reachability synthesis).

# Outline

# Implementing the translation

Automated translation

- Input: a Time4sys real-time system
- Output: a network of PTAs described in the IMITATOR input language
- Translation implemented by Jawher Jerray and Sahar Mhiri
  - tool Time4sys2imi
  - 5.5 kLoC in Java

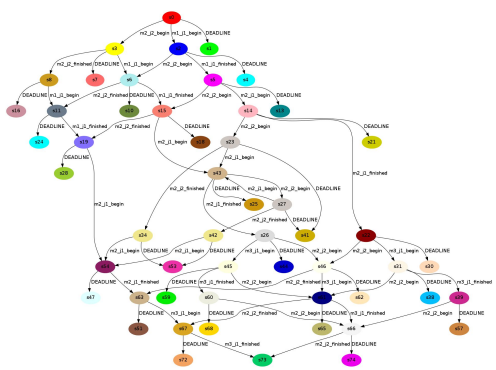
[ÉA, Jerray, Mhiri @ ICTAC 2019]

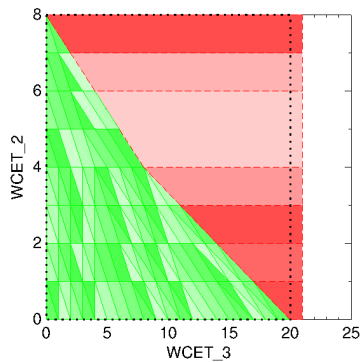
# Outline

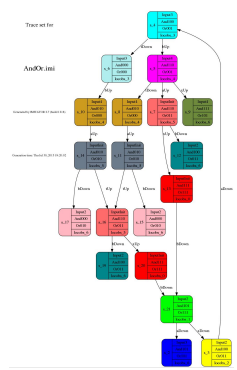
# IMITATOR

- A tool for modeling and verifying timed concurrent systems with unknown constants modeled with parametric timed automata
  - Communication through (strong) broadcast synchronization
  - Rational-valued shared discrete variables
  - Stopwatches, to model schedulability problems with preemption

- Synthesis algorithms
  - (non-Zeno) parametric model checking (using a subset of TCTL)
  - Language and trace preservation, and robustness analysis
  - Parametric deadlock-freeness checking





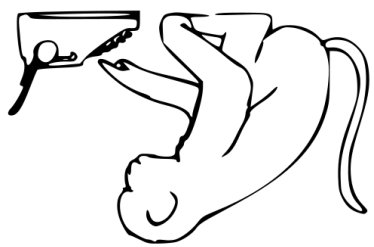
# IMITATOR

Under continuous development since 2008 [André et al., FM'12]

A library of benchmarks [André, FTSCS'18]

- Communication protocols
- Schedulability problems
- Asynchronous circuits
- …and more

Free and open source software: Available under the GNU-GPL license



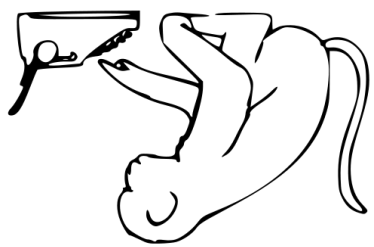
# IMITATOR

Under continuous development since 2008 [André et al., FM'12]

A library of benchmarks [André, FTSCS'18]

- Communication protocols
- Schedulability problems
- Asynchronous circuits
- …and more

Free and open source software: Available under the GNU-GPL license

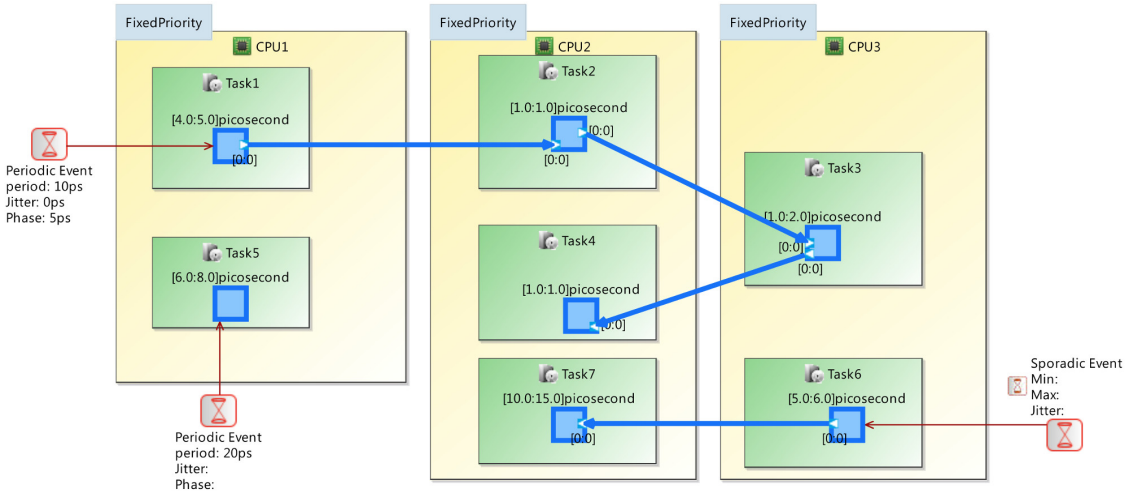


Try it!

www.imitator.fr

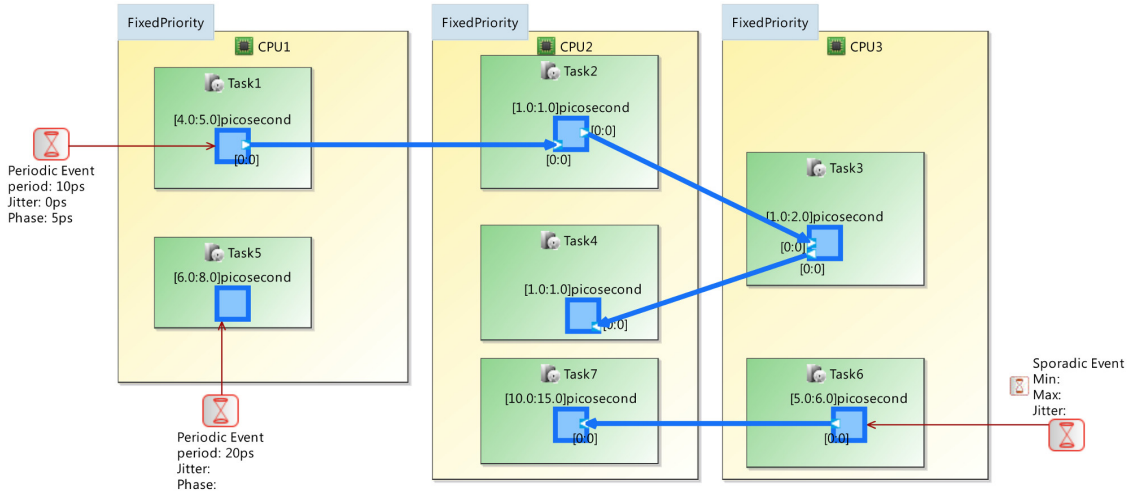
# Outline

# Proof of concept



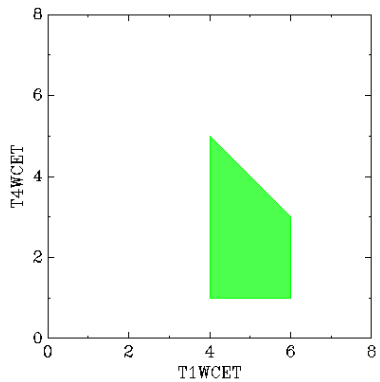

1. Synthesize $T1WCET$ and $T4WCET$ for which the system is schedulable:

# Proof of concept

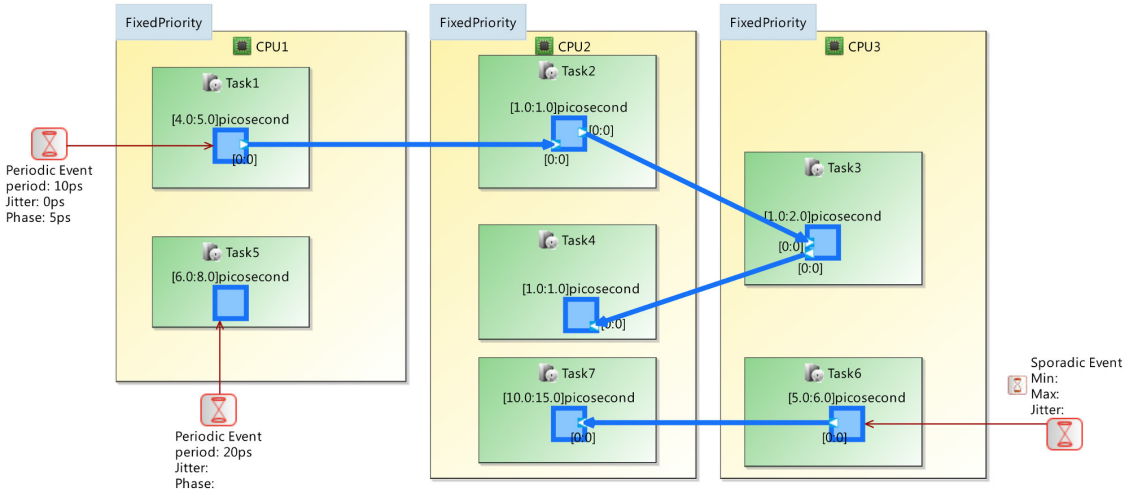



1. Synthesize $\mathrm{T1WCET}$ and $\mathrm{T4WCET}$ for which the system is schedulable:

$$4 \leq \mathrm{T1WCET} \leq 6 \wedge \mathrm{T4WCET} \geq 1 \wedge \mathrm{T1WCET} + \mathrm{T4WCET} < 9$$

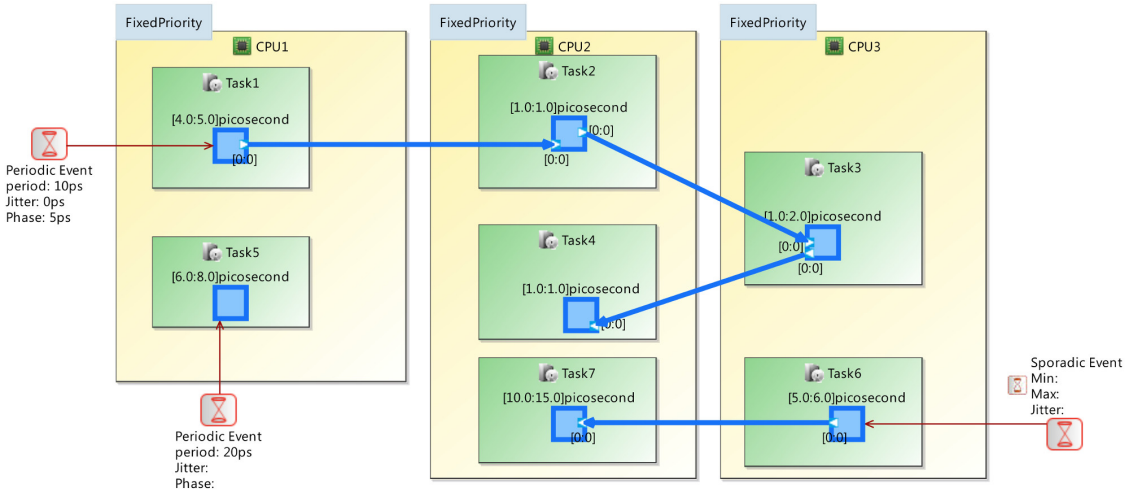


# Proof of concept (cont.)



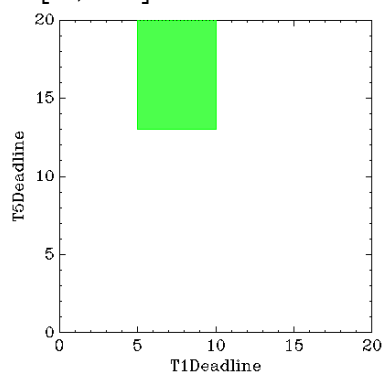

2. Synthesize deadlines of tasks 1 and 5 ensuring schedulability:

# Proof of concept (cont.)




2 Synthesize deadlines of tasks 1 and 5 ensuring schedulability:

$$\text{T1Deadline} \in [5, 13] \wedge \text{T5Deadline} \in [10, 20]$$




Computation time using IMITATOR 2.10.4 “Butter Jellyfish”: few seconds

Additional experiments in up to 4 dimensions (see paper)

# Outline

# Conclusion

Formalization of an industrial formalism for real-time system

- Uniprocessor or multiprocessor
- Periodic, sporadic tasks
- Tasks dependencies
- Various scheduling policies
- Notably supports uncertainty

Verification and synthesis using IMITATOR using an automated translation

# Perspectives

## More expressiveness

- Remove some (mild?) assumptions
  - "Task period = task deadline"
- Support more scheduling policies
- Mixed-criticality scheduling

## More efficiency

- Optimizations (in our scheme, and in IMITATOR) dedicated to schedulability

## Certification of translation

- Issue: no formal semantics!
- Opportunity: [Halchin et al., 2019]

# Bibliography

# References I



Alur, R. and Dill, D. L. (1994).
A theory of timed automata.
*Theoretical Computer Science*, 126(2):183–235.


Alur, R., Henzinger, T. A., and Vardi, M. Y. (1993).
Parametric real-time reasoning.
In Kosaraju, S. R., Johnson, D. S., and Aggarwal, A., editors, *STOC*, pages 592–601, New York, NY, USA. ACM.


André, É. (2019a).
A benchmark library for parametric timed model checking.
In Artho, C. and Ölveczky, P. C., editors, *FTSCS*, volume 1008 of *CCIS*, pages 75–83. Springer.


André, É. (2019b).
Formalizing Time4sys using parametric timed automata.
In Méry, D. and Qin, S., editors, *TASE*, pages 176–183. IEEE.


André, É., Fribourg, L., Kühne, U., and Soulat, R. (2012).
IMITATOR 2.5: A tool for analyzing robustness in scheduling problems.
In Giannakopoulou, D. and Méry, D., editors, *FM*, volume 7436 of *LNCS*, pages 33–36. Springer.


André, É., Hasuo, I., and Waga, M. (2018).
Offline timed pattern matching under uncertainty.
In Lin, A. W. and Sun, J., editors, *ICECCS*, pages 10–20. IEEE CPS.

# References II


André, É., Jerray, J., and Mhiri, S. (2019).
Time4sys2imi: A tool to formalize real-time system models under uncertainty.
In Jmaiel, M. and Gaaloul, W., editors, *ICTAC*. Springer.
To appear.

Baier, C. and Katoen, J.-P. (2008).
*Principles of Model Checking*.
MIT Press.

Bouyer, P., Markey, N., and Sankur, O. (2013).
Robustness in timed automata.
In Abdulla, P. A. and Potapov, I., editors, *RP*, volume 8169 of *LNCS*, pages 1–18. Springer.
Invited paper.

Fanchon, L. and Jacquemard, F. (2013).
Formal timing analysis of mixed music scores.
In *ICMC*. Michigan Publishing.

Fersman, E., Krcál, P., Pettersson, P., and Yi, W. (2007).
Task automata: Schedulability, decidability and undecidability.
*Information and Computation*, 205(8):1149–1172.

Fribourg, L., Lesens, D., Moro, P., and Soulat, R. (2012).
Robustness analysis for scheduling problems using the inverse method.
In Reynolds, M., Terenziani, P., and Moszkowski, B., editors, *TIME*, pages 73–80. IEEE Computer Society Press.

# References III

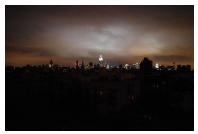

Halchin, A., Ameur, Y. A., Singh, N., Feliachi, A., and Ordioni, J. (2019).
Certified embedding of B models in an integrated verification framework.
In Méry, D. and Qin, S., editors, *TASE*, pages 168–175. IEEE.

Luthmann, L., Gerecht, T., Stephan, A., Bürdek, J., and Lochau, M. (2019).
Minimum/maximum delay testing of product lines with unbounded parametric real-time constraints.
*Journal of Systems and Software*, 149:535–553.

Sun, Y., Soulat, R., Lipari, G., André, É., and Fribourg, L. (2013).
Parametric schedulability analysis of fixed priority real-time distributed systems.
In Artho, C. and Ölveczky, P., editors, *FSTCS*, volume 419 of *CCIS*, pages 212–228. Springer.

# Additional explanation

# Explanation for the 4 pictures in the beginning



Allusion to the Northeast blackout (USA, 2003)
Computer bug
Consequences: 11 fatalities, huge cost
(Picture actually from the Sandy Hurricane, 2012)


Error screen on the earliest versions of Macintosh

Allusion to the sinking of the Sleipner A offshore platform (Norway, 1991)
No fatalities
Computer bug: inaccurate finite element analysis modeling
(Picture actually from the Deepwater Horizon Offshore Drilling Platform)

Allusion to the MIM-104 Patriot Missile Failure (Iraq, 1991)
28 fatalities, hundreds of injured
Computer bug: software error (clock drift)
(Picture of an actual MIM-104 Patriot Missile, though not the one of 1991)

# Beyond timed model checking: parameter synthesis

- Verification for one set of constants does not usually guarantee the correctness for other values

- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase $10$?
  - Robustness [Bouyer et al., 2013]: What happens if $50$ is implemented with $49.99$?
  - System incompletely specified: Can I verify my system even if I don't know the period value with full certainty?

# Beyond timed model checking: parameter synthesis

- Verification for one set of constants does not usually guarantee the correctness for other values

- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase $10$?
  - Robustness [Bouyer et al., 2013]: What happens if $50$ is implemented with $49.99$?
  - System incompletely specified: Can I verify my system even if I don't know the period value with full certainty?

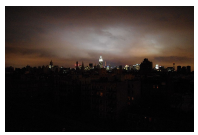
- Parameter synthesis
  - Consider that timing constants are unknown constants (parameters)

# Some success stories of IMITATOR

- Modeled and verified an asynchronous memory circuit by ST-Microelectronics
- Parametric schedulability analysis of a prospective architecture for the flight control system of the next generation of spacecrafts designed at ASTRIUM Space Transportation [Fribourg et al., 2012]
- Verification of software product lines [Luthmann et al., 2019]
- Formal timing analysis of music scores [Fanchon and Jacquemard, 2013]
- Solution to a challenge related to a distributed video processing system by Thales
- Monitoring cyber-physical systems [ÉA, Hasuo, Waga @ ICECCS'18]

# Licensing

# Source of the graphics used I



Title: Hurricane Sandy Blackout New York Skyline
Author: David Shankbone
Source: `https://commons.wikimedia.org/wiki/File:Hurricane_Sandy_Blackout_New_York_Skyline.JPG`
License: CC BY 3.0


Title: Sad mac
Author: Przemub
Source: `https://commons.wikimedia.org/wiki/File:Sad_mac.png`
License: Public domain

Title: Deepwater Horizon Offshore Drilling Platform on Fire
Author: ideum
Source: `https://secure.flickr.com/photos/ideum/4711481781/`
License: CC BY-SA 2.0

Title: DA-SC-88-01663
Author: imcomkorea
Source: `https://secure.flickr.com/photos/imcomkorea/3017886760/`
License: CC BY-NC-ND 2.0

# Source of the graphics used II

Title: Smiley green alien big eyes (aaah)
Author: LadyofHats
Source: `https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg`
License: public domain

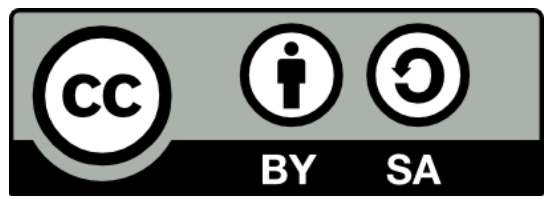
Title: Smiley green alien big eyes (cry)
Author: LadyofHats
Source: `https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg`
License: public domain

# License of this document

This presentation can be published, reused and modified under the terms of the license Creative Commons **Attribution-ShareAlike 4.0 Unported (CC BY-SA 4.0)**

(LaTeX source available on demand)

Author: **Étienne André**




https://creativecommons.org/licenses/by-sa/4.0/