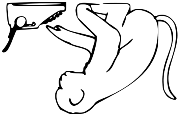
École Temps-Réel 2015
Rennes

IMITATOR Tutorial
Parametric Timed Systems



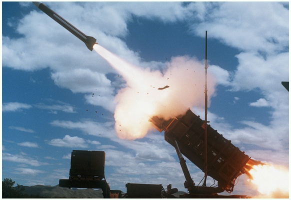
Étienne ANDRÉ

Etienne.Andre@univ-paris13.fr

Version: August 26, 2015 (slideshow version)

Context: Verifying critical real-time systems

- Need for early bug detection
  - Bugs discovered when final testing: expensive
  - ⤳ Need for a thorough specification and verification phase



The Therac-25 radiation therapy machine (1/2)

- Radiation therapy machine used in the 1980s
- Involved in accidents between 1985 and 1987, in which patients were given massive overdoses of radiation
    - Approximately 100 times the intended dose!
    - Numerous causes, including race condition

The Therac-25 radiation therapy machine (1/2)

- Radiation therapy machine used in the 1980s
- Involved in accidents between 1985 and 1987, in which patients were given massive overdoses of radiation
    - Approximately 100 times the intended dose!
    - Numerous causes, including race condition

> *"The failure only occurred when a particular nonstandard sequence of keystrokes was entered on the VT-100 terminal which controlled the PDP-11 computer: an* `X` *to (erroneously) select 25MV photon mode followed by* ↑, `E` *to (correctly) select 25 MeV Electron mode, then* `Enter`*, all within eight seconds."*

The Therac-25 radiation therapy machine (2/2)

The testing engineers could obviously not detect this strange (and quick!) sequence leading to the failure.

The Therac-25 radiation therapy machine (2/2)

The testing engineers could obviously not detect this strange (and quick!) sequence leading to the failure.

Limits of testing

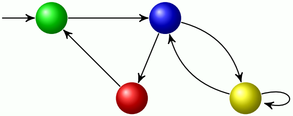
This case illustrates the difficulty of bug detection without formal methods.

Plan: Timed Automata

1. Timed Automata
2. Parametric Timed Automata
3. IMITATOR
4. Perspectives

Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]



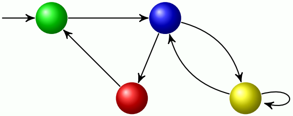
A model of the system

🔴 is unreachable

A property to be satisfied

Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]



A model of the system

$\overset{?}{\models}$

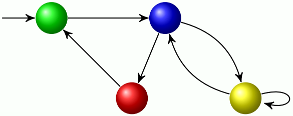
● is unreachable

A property to be satisfied

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

Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]



$\overset{?}{\models}$

● is unreachable

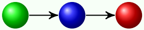
A model of the system

A property to be satisfied

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

Yes

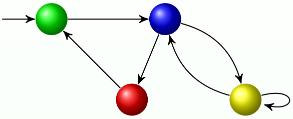
No



Counterexample

Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]



A model of the system

$\overset{?}{\models}$

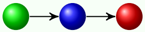
● is unreachable

A property to be satisfied

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

Yes

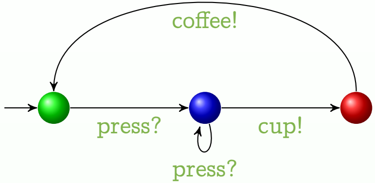
No



Counterexample

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

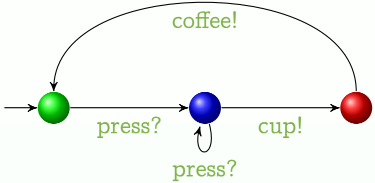
A coffee vending machine $\mathcal{A}_C$



- 🟢 Idle
- 🔵 Adding sugar
- 🔴 Delivering coffee

- Example of runs
  - Coffee with no sugar

A coffee vending machine $\mathcal{A}_C$

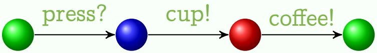


🟢 Idle

🔵 Adding sugar

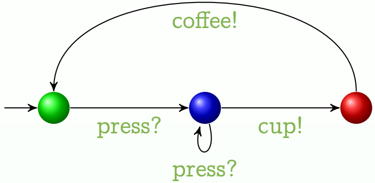
🔴 Delivering coffee

- Example of runs
  - Coffee with no sugar



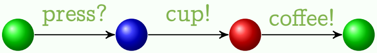
  - Coffee with 2 doses of sugar

A coffee vending machine $\mathcal{A}_C$

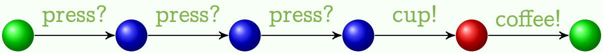


- 🟢 Idle
- 🔵 Adding sugar
- 🔴 Delivering coffee

- Example of runs
  - Coffee with no sugar



  - Coffee with 2 doses of sugar



  - And so on

Verification of properties

Decide whether the following properties are satisfied for the coffee vending machine

- “It is possible to get a coffee with 3 doses of sugar.”

Verification of properties

Decide whether the following properties are satisfied for the coffee vending machine

- "It is possible to get a coffee with 3 doses of sugar."
  ($\surd$)

Verification of properties

Decide whether the following properties are satisfied for the coffee vending machine

- "It is possible to get a coffee with 3 doses of sugar." $(\surd)$
- "After the button is pressed, a coffee is always eventually delivered."

Verification of properties

Decide whether the following properties are satisfied for the coffee vending machine

- “It is possible to get a coffee with 3 doses of sugar.”
  ($\surd$)
- “After the button is pressed, a coffee is always eventually delivered.”
  ($\times$)

Verification of properties

Decide whether the following properties are satisfied for the coffee vending machine

- "It is possible to get a coffee with 3 doses of sugar."
  ($\surd$)
- "After the button is pressed, a coffee is always eventually delivered."
  ($\times$)
- "After the button is pressed, there exists an execution such that a coffee is eventually delivered."

Verification of properties

Decide whether the following properties are satisfied for the coffee vending machine

- "It is possible to get a coffee with 3 doses of sugar."
  ($\surd$)
- "After the button is pressed, a coffee is always eventually delivered."
  ($\times$)
- "After the button is pressed, there exists an execution such that a coffee is eventually delivered."
  ($\surd$)

Verification of properties

Decide whether the following properties are satisfied for the coffee vending machine

- "It is possible to get a coffee with 3 doses of sugar."
  ($\surd$)
- "After the button is pressed, a coffee is always eventually delivered."
  ($\times$)
- "After the button is pressed, there exists an execution such that a coffee is eventually delivered."
  ($\surd$)
- "Once the cup is delivered, coffee will necessarily come next."

Verification of properties

Decide whether the following properties are satisfied for the coffee vending machine

- "It is possible to get a coffee with 3 doses of sugar."
  ($\surd$)
- "After the button is pressed, a coffee is always eventually delivered."
  ($\times$)
- "After the button is pressed, there exists an execution such that a coffee is eventually delivered."
  ($\surd$)
- "Once the cup is delivered, coffee will necessarily come next."
  ($\surd$)

Beyond finite state automata

Finite State Automata: powerful formalism to model qualitative aspects of systems

Beyond finite state automata

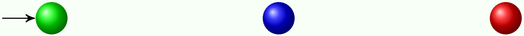
Finite State Automata: powerful formalism to model qualitative aspects of systems

But what about quantitative aspects:

- Time ("the airbag always eventually inflates, but maybe 10 seconds after the crash")
- Temperature ("the alarm always eventually ring, but maybe when the temperature is above 75 degrees")

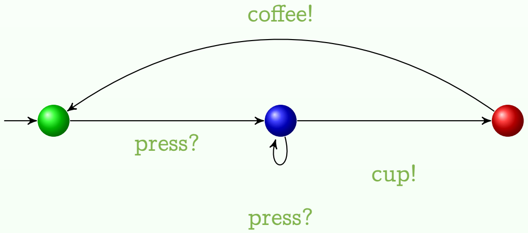
Timed automaton (TA)

- Finite state automaton (sets of locations)



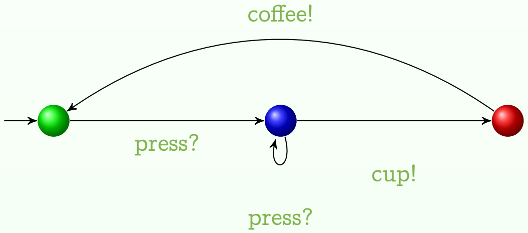
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)



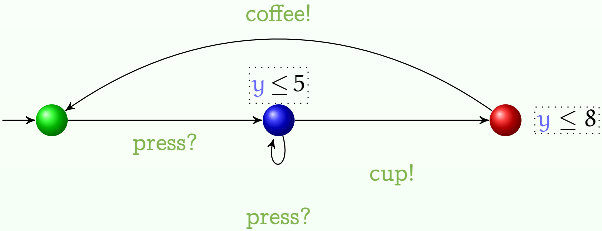
Timed automaton (TA)

- 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



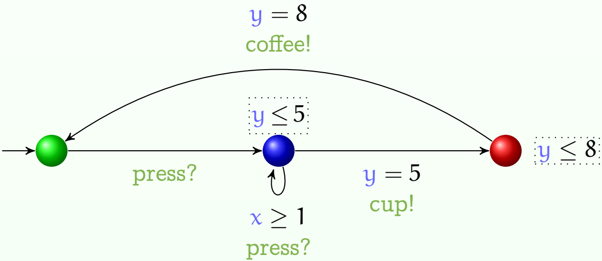
Timed automaton (TA)

- 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
- Features
  - Location invariant: property to be verified to stay at a location



Timed automaton (TA)

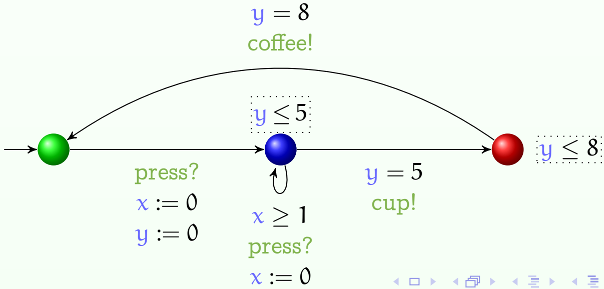
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to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

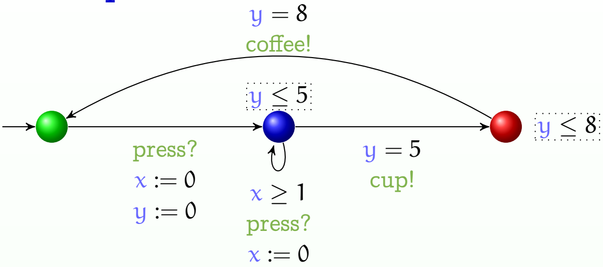
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 at each transition



Concrete semantics of timed automata

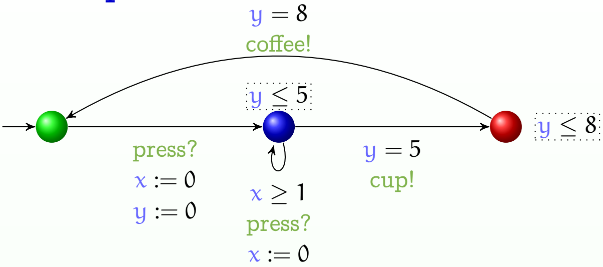
- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or elapsing of time

Examples of concrete runs



- Possible concrete runs for the coffee machine

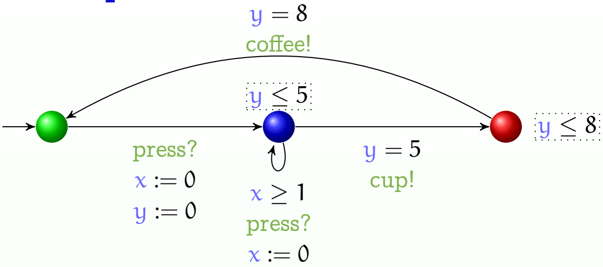
Examples of concrete runs



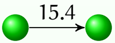
- Possible concrete runs for the coffee machine
  - Coffee with no sugar

x	0
y	0

Examples of concrete runs

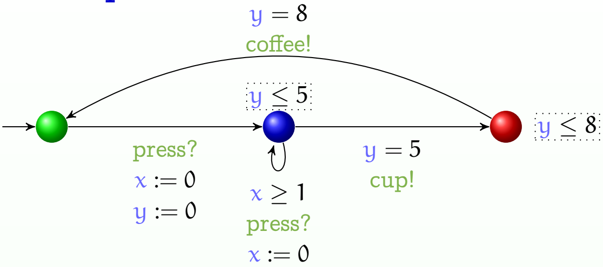


- Possible concrete runs for the coffee machine
    - Coffee with no sugar

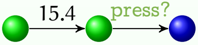


x	0	15.4
y	0	15.4

Examples of concrete runs

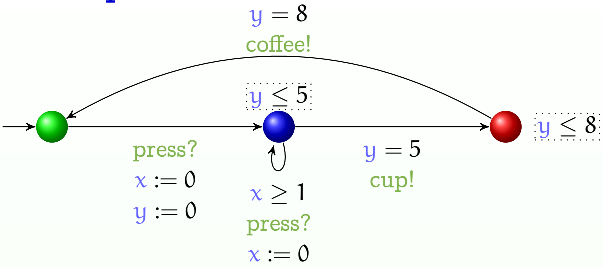


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

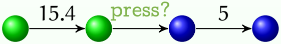


x	0	15.4	0
y	0	15.4	0

Examples of concrete runs

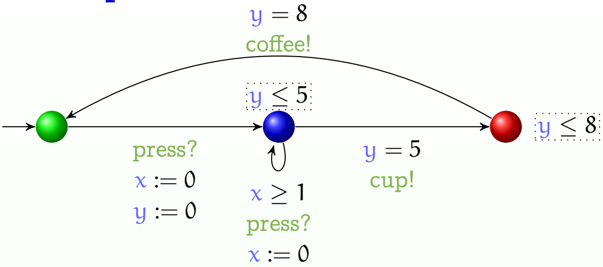


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

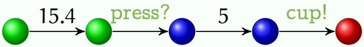


x	0	15.4	0	5
y	0	15.4	0	5

Examples of concrete runs

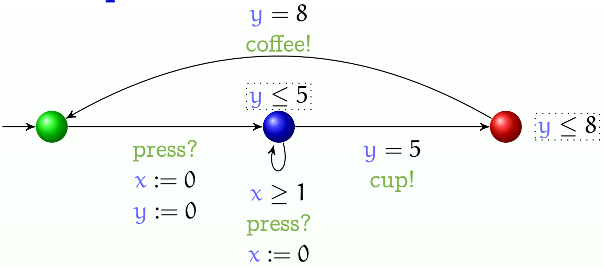


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

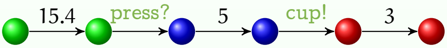


x	0	15.4	0	5	5
y	0	15.4	0	5	5

Examples of concrete runs

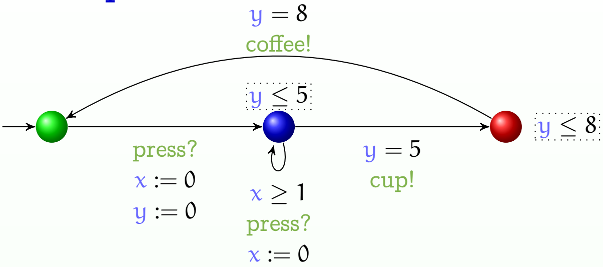


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

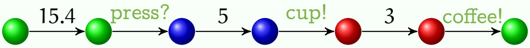


x	0	15.4	0	5	5	8
y	0	15.4	0	5	5	8

Examples of concrete runs

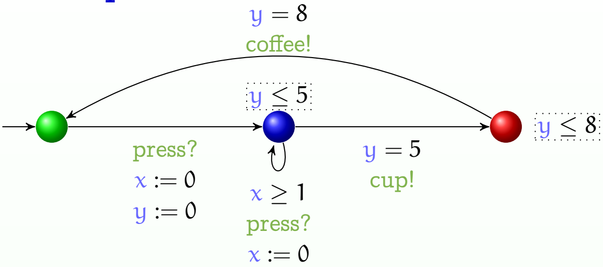


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

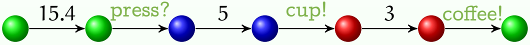


x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

Examples of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar

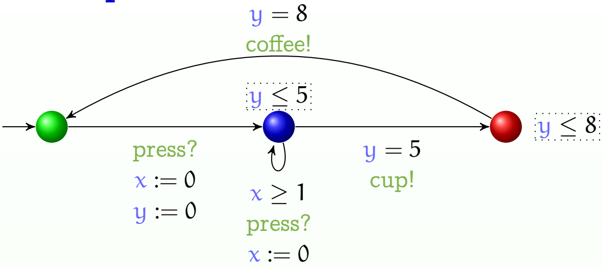


x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

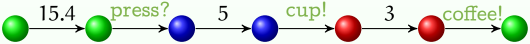
  - Coffee with 2 doses of sugar

x	0
y	0

Examples of concrete runs

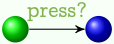


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



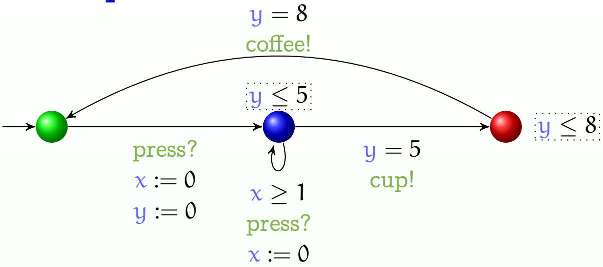
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

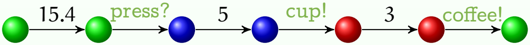


x	0	0
y	0	0

Examples of concrete runs

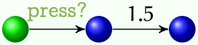


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



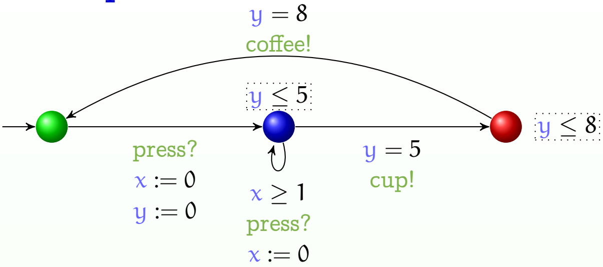
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

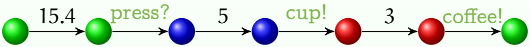


x	0	0	1.5
y	0	0	1.5

Examples of concrete runs

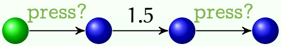


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



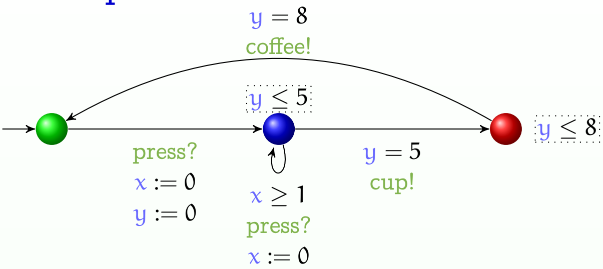
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

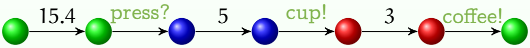


x	0	0	1.5	0
y	0	0	1.5	1.5

Examples of concrete runs

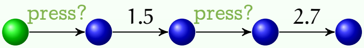


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



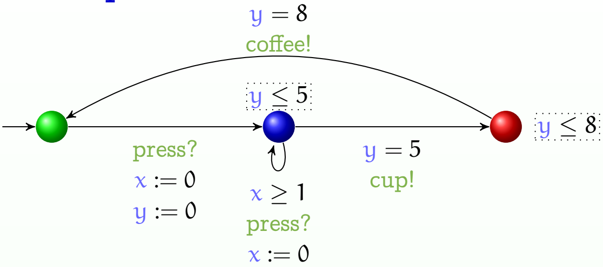
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

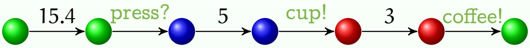


x	0	0	1.5	0	2.7
y	0	0	1.5	1.5	4.2

Examples of concrete runs

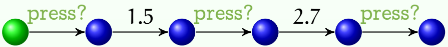


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



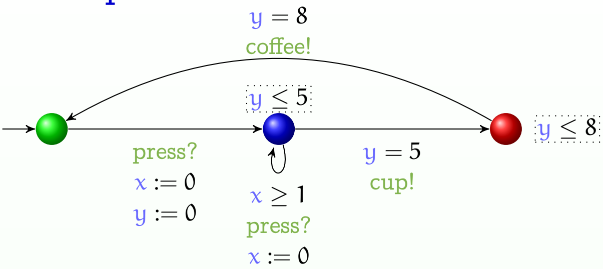
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

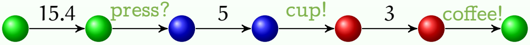


x	0	0	1.5	0	2.7	0
y	0	0	1.5	1.5	4.2	4.2

Examples of concrete runs

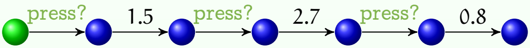


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



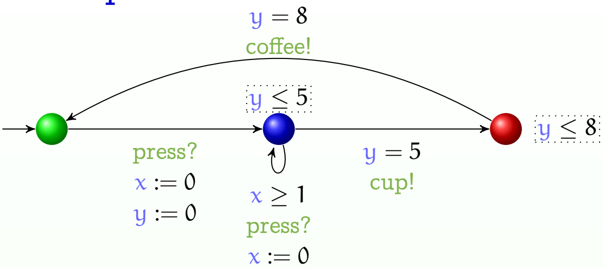
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

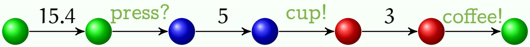


x	0	0	1.5	0	2.7	0	0.8
y	0	0	1.5	1.5	4.2	4.2	5

Examples of concrete runs

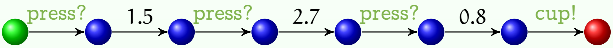


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



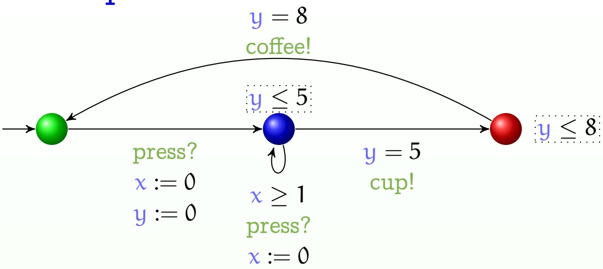
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

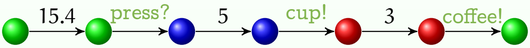


x	0	0	1.5	0	2.7	0	0.8	0.8
y	0	0	1.5	1.5	4.2	4.2	5	5

Examples of concrete runs

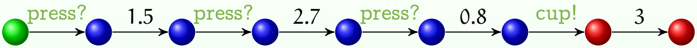


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



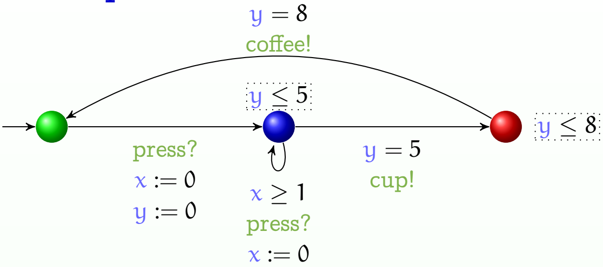
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

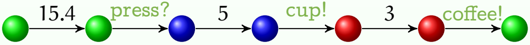


x	0	0	1.5	0	2.7	0	0.8	0.8	3.8
y	0	0	1.5	1.5	4.2	4.2	5	5	8

Examples of concrete runs

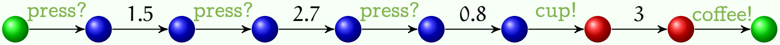


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar



x	0	0	1.5	0	2.7	0	0.8	0.8	3.8	3.8
y	0	0	1.5	1.5	4.2	4.2	5	5	8	8

Verification of (timed) properties

Decide whether the following properties are satisfied for the timed coffee vending machine

- "It is possible to get a coffee with 5 doses of sugar."

Verification of (timed) properties

Decide whether the following properties are satisfied for the timed coffee vending machine

- "It is possible to get a coffee with 5 doses of sugar." ($\surd$)

Verification of (timed) properties

Decide whether the following properties are satisfied for the timed coffee vending machine

- "It is possible to get a coffee with 5 doses of sugar."
  ($\surd$)
- "Once the cup is delivered, coffee will come next within 2 seconds."

Verification of (timed) properties

Decide whether the following properties are satisfied for the timed coffee vending machine

- "It is possible to get a coffee with 5 doses of sugar."
  ($\surd$)
- "Once the cup is delivered, coffee will come next within 2 seconds."
  ($\times$)

Verification of (timed) properties

Decide whether the following properties are satisfied for the timed coffee vending machine

- "It is possible to get a coffee with 5 doses of sugar." ($\surd$)
- "Once the cup is delivered, coffee will come next within 2 seconds." ($\times$)
- "After the button is pressed, a coffee is always eventually delivered."

Verification of (timed) properties

Decide whether the following properties are satisfied for the timed coffee vending machine

- "It is possible to get a coffee with 5 doses of sugar."
  (✓)
- "Once the cup is delivered, coffee will come next within 2 seconds."
  (×)
- "After the button is pressed, a coffee is always eventually delivered."
  (✓)

Verification of (timed) properties

Decide whether the following properties are satisfied for the timed coffee vending machine

- "It is possible to get a coffee with 5 doses of sugar."
  ($\surd$)
- "Once the cup is delivered, coffee will come next within 2 seconds."
  ($\times$)
- "After the button is pressed, a coffee is always eventually delivered."
  ($\surd$)
- "It is impossible to press the button twice within 1 second."

Verification of (timed) properties

Decide whether the following properties are satisfied for the timed coffee vending machine

- "It is possible to get a coffee with 5 doses of sugar."
  ($\surd$)
- "Once the cup is delivered, coffee will come next within 2 seconds."
  ($\times$)
- "After the button is pressed, a coffee is always eventually delivered."
  ($\surd$)
- "It is impossible to press the button twice within 1 second."
  ($\times$)

Plan: Parametric Timed Automata

1. Timed Automata
2. Parametric Timed Automata
3. IMITATOR
4. Perspectives

Motivation

- Challenge 1: systems incompletely specified
  - Some delays may not be known yet, or may change

Motivation

- Challenge 1: systems incompletely specified
  - Some delays may not be known yet, or may change
- Challenge 2: Robustness [Markey, 2011]
  - What happens if 8 is implemented with 7.99?
  - Can I really get a coffee with 5 doses of sugar?

Motivation

- Challenge 1: systems incompletely specified
  - Some delays may not be known yet, or may change
- Challenge 2: Robustness [Markey, 2011]
  - What happens if 8 is implemented with 7.99?
  - Can I really get a coffee with 5 doses of sugar?
- Challenge 3: Optimization of timing constants
  - Up to which value of the delay between two actions press? can I still order a coffee with 3 doses of sugar?

Motivation

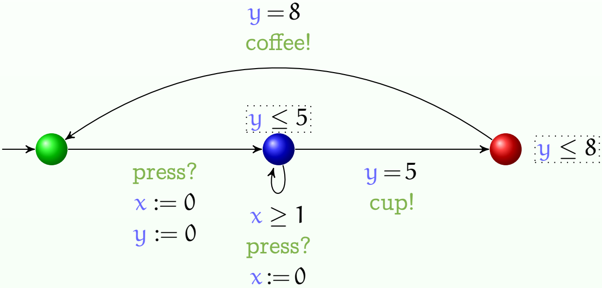
- Challenge 1: systems incompletely specified
  - Some delays may not be known yet, or may change
- Challenge 2: Robustness [Markey, 2011]
  - What happens if *8* is implemented with *7.99*?
  - Can I really get a coffee with 5 doses of sugar?
- Challenge 3: Optimization of timing constants
  - Up to which value of the delay between two actions press? can I still order a coffee with 3 doses of sugar?
- Challenge 4: Avoidance of numerous verifications
  - If one of the timing delays of the model changes, should I model check again the whole system?

Motivation

- Challenge 1: systems incompletely specified
  - Some delays may not be known yet, or may change
- Challenge 2: Robustness [Markey, 2011]
  - What happens if 8 is implemented with 7.99?
  - Can I really get a coffee with 5 doses of sugar?
- Challenge 3: Optimization of timing constants
  - Up to which value of the delay between two actions press? can I still order a coffee with 3 doses of sugar?
- Challenge 4: Avoidance of numerous verifications
  - If one of the timing delays of the model changes, should I model check again the whole system?
- A solution: Parametric analysis
  - Consider that timing constants are unknown (parameters)
  - Find good values for the parameters s.t. the system behaves well

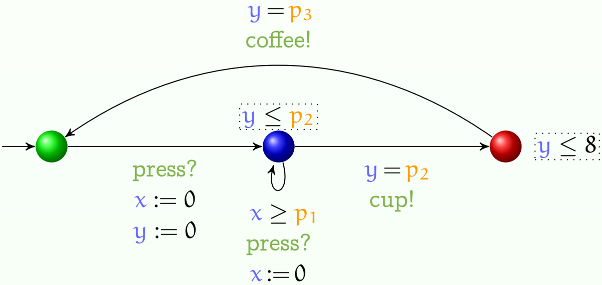
Parametric Timed Automaton (PTA)

- 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 used in guards and invariants



Decision and computation problems for PTA

- Emptiness “Does there at least one parameter valuation for which I can get a coffee with 2 sugars?”

Decision and computation problems for PTA

- Emptiness "Does there at least one parameter valuation for which I can get a coffee with 2 sugars?"
  $\surd$, e.g., $p_1 = 1, p_2 = 5, p_3 = 8$

Decision and computation problems for PTA

- Emptiness "Does there at least one parameter valuation for which I can get a coffee with 2 sugars?"
  $\surd$, e.g., $p_1 = 1, p_2 = 5, p_3 = 8$
- Universality "Are all parameter valuations such that I may eventually get a coffee?"

Decision and computation problems for PTA

- Emptiness "Does there at least one parameter valuation for which I can get a coffee with 2 sugars?"
  $\surd$, e.g., $p_1 = 1, p_2 = 5, p_3 = 8$
- Universality "Are all parameter valuations such that I may eventually get a coffee?"
  $\times$, e.g., $p_1 = 1, p_2 = 5, p_3 = 2$

Decision and computation problems for PTA

- Emptiness "Does there at least one parameter valuation for which I can get a coffee with 2 sugars?"
  $\surd$, e.g., $p_1 = 1, p_2 = 5, p_3 = 8$

- Universality "Are all parameter valuations such that I may eventually get a coffee?"
  $\times$, e.g., $p_1 = 1, p_2 = 5, p_3 = 2$

- Preservation of the untimed language "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviors?"

Decision and computation problems for PTA

- Emptiness "Does there at least one parameter valuation for which I can get a coffee with 2 sugars?"
  $\surd$, e.g., $p_1 = 1, p_2 = 5, p_3 = 8$

- Universality "Are all parameter valuations such that I may eventually get a coffee?"
  $\times$, e.g., $p_1 = 1, p_2 = 5, p_3 = 2$

- Preservation of the untimed language "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviors?"
  $\surd$

Decision and computation problems for PTA

- Emptiness "Does there at least one parameter valuation for which I can get a coffee with 2 sugars?"
  $\surd$, e.g., $p_1 = 1, p_2 = 5, p_3 = 8$

- Universality "Are all parameter valuations such that I may eventually get a coffee?"
  $\times$, e.g., $p_1 = 1, p_2 = 5, p_3 = 2$

- Preservation of the untimed language "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviors?"
  $\surd$

- Synthesis "What are all parameter valuations such that one can always eventually get a coffee?"

Decision and computation problems for PTA

- Emptiness "Does there at least one parameter valuation for which I can get a coffee with 2 sugars?"
  $\surd$, e.g., $p_1 = 1, p_2 = 5, p_3 = 8$
- Universality "Are all parameter valuations such that I may eventually get a coffee?"
  $\times$, e.g., $p_1 = 1, p_2 = 5, p_3 = 2$
- Preservation of the untimed language "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviors?"
  $\surd$
- Synthesis "What are all parameter valuations such that one can always eventually get a coffee?"
  $0 \leq p_2 \leq p_3 \leq 8$

Software supporting parametric timed automata

Specification and verification of parametric models using parametric timed automata are supported by several software

- HyTech (also hybrid automata) [Henzinger et al., 1997]
- PHAVer (also hybrid systems) [Frehse, 2005]
- Roméo (also parametric time Petri nets) [Lime et al., 2009]
- Imitator [A. et al, 2012]

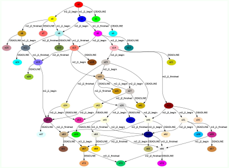
Plan: IMITATOR

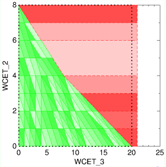
1. Timed Automata
2. Parametric Timed Automata
3. IMITATOR
4. Perspectives

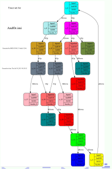
IMITATOR

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

- Verification
  - Computation of the symbolic state space
  - Parametric model checking (using a subset of TCTL)
  - Language and trace preservation, and robustness analysis
  - Behavioral cartography







IMITATOR

Under continuous development since 2008

A library of benchmarks

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

Open source: Available under the GNU-GPL license

GPL 3 Free Software

Free as in Freedom

IMITATOR

Under continuous development since 2008

A library of benchmarks

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

Open source: Available under the GNU-GPL license

GPL
Free Software
Free as in Freedom

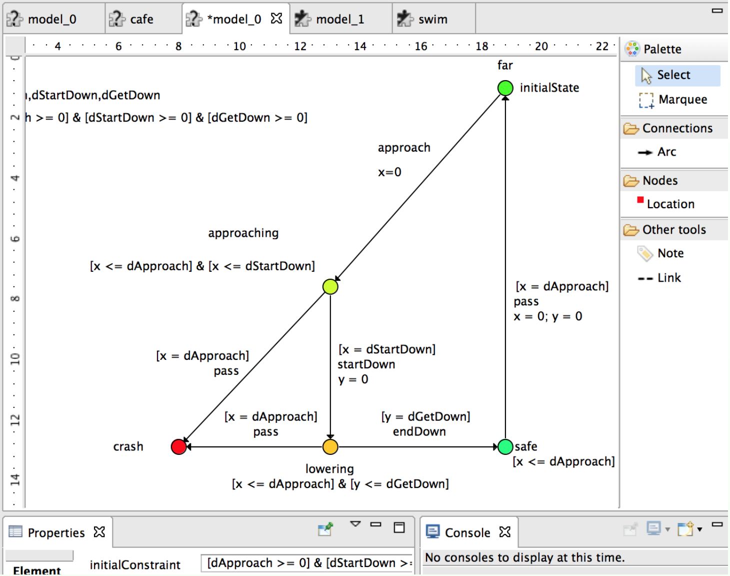
Try it!

www.imitator.fr

Some success stories

- Modeled and verified an asynchronous memory circuit by ST-Microelectronics
  - Project ANR Valmem
- 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]
- Solution to a challenge related to a distributed video processing system by Thales
- Formal timing analysis of music scores [Fanchon and Jacquemard, 2013]

Graphical user interface using *CosyVerif*



initialConstraint [dApproach >= 0] & [dStartDown >:

No consoles to display at this time.

Outline of the practical session

1. Perform parameter synthesis for a railway crossing system
2. Specify and verify the coffee machine
3. . . . and if you are fast: a free bonus exercise!

Plan: Perspectives

1. Timed Automata
2. Parametric Timed Automata
3. IMITATOR
4. Perspectives

Perspectives

- “Small detail”: all known problems for PTA are undecidable
  - No big deal: semi-algorithms, approximations, etc.
  - Challenge: find decidable subclasses
- Other parametric models
  - Number of processes (“discrete” parameters)
  - Challenge: combine different types of parameters (discrete + continuous)

General References

- Systems and Software Verification (Béatrice Bérard, Michel Bidoit, Alain Finkel, François Laroussinie, Antoine Petit, Laure Petrucci, Philippe Schnoebelen), Springer, 2001
- Principles of Model Checking (Christel Baier and Joost-Pieter Katoen), MIT Press, 2008
- The Inverse Method (Étienne André and Romain Soulat), ISTE and Wiley & Sons, 2013

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 *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC'93, pages 592–601, New York, NY, USA. ACM.

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, *Proceedings of the 18th International Symposium on Formal Methods (FM'12)*, volume 7436 of *Lecture Notes in Computer Science*, pages 33–36. Springer.

André, É. and Soulat, R. (2013).
The Inverse Method.
FOCUS Series in Computer Engineering and Information Technology. ISTE Ltd and John Wiley & Sons Inc.
176 pages.

References II

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

Fanchon, L. and Jacquemard, F. (2013).
Formal timing analysis of mixed music scores.
In *ICMC 2013 (International Computer Music Conference)*.

Frehse, G. (2005).
PHAVer: Algorithmic verification of hybrid systems past HyTech.
In Morari, M. and Thiele, L., editors, *HSCC 2005*, volume 3414 of *Lecture Notes in Computer Science*, pages 258–273. Springer.

Fribourg, L., Lesens, D., Moro, P., and Soulat, R. (2012).
Robustness analysis for scheduling problems using the inverse method.
In *TIME'12*, pages 73–80. IEEE Computer Society Press.

Henzinger, T. A., Ho, P.-H., and Wong-Toi, H. (1997).
HyTech: A model checker for hybrid systems.
Software Tools for Technology Transfer, 1:110–122.

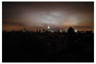
References III

Lime, D., Roux, O. H., Seidner, C., and Traonouez, L.-M. (2009).
Romeo: A parametric model-checker for Petri nets with stopwatches.
In Kowalewski, S. and Philippou, A., editors, *15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2009)*, volume 5505 of *LNCS*, pages 54–57. Springer.

Markey, N. (2011).
Robustness in real-time systems.
In *Proceedings of the 6th IEEE International Symposium on Industrial Embedded Systems (SIES'11)*, pages 28–34, Västerås, Sweden. IEEE Computer Society Press.

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)

```
0000000F
00000003
```

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)

License

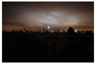
Source of the graphics (1)

Titre: Clock 256
Auteur: Everaldo Coelho
Source: `https://commons.wikimedia.org/wiki/File:Clock_256.png`
Licence: GNU LGPL

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

Source of the graphics (2)



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

License of this document

Author: Étienne André

(LaTeX source available on demand)