UNIVERSITÉ PARIS 13

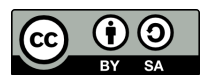
ESWEEK Tutorial

Sunday, 30th of September

Schedulability Analysis under Uncertainty using Formal Methods (part 2)

Étienne André and Giusppe Lipari

LIPN, Université Paris 13, CNRS, France

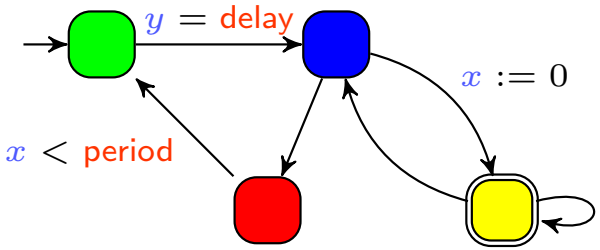


Outline

Model checking timed concurrent systems

- Use formal methods

[Baier and Katoen, 2008]



A model of the system

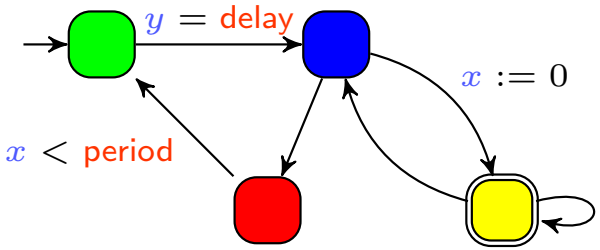
 is unreachable

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

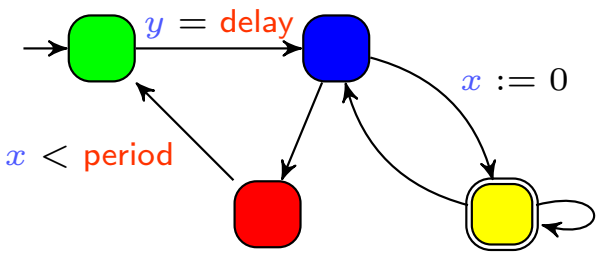
A property to be satisfied

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

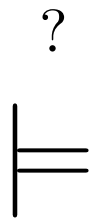
Model checking timed concurrent systems

- Use formal methods

[Baier and Katoen, 2008]



A model of the system



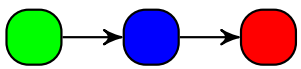

is unreachable

A property to be satisfied

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

Yes

No



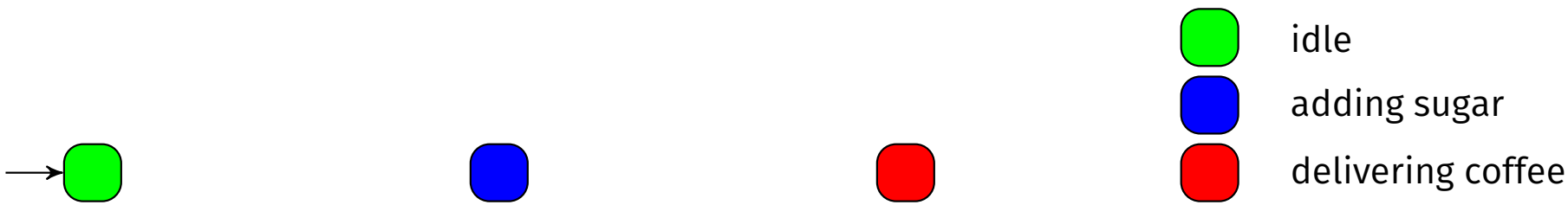
Counterexample

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

Outline

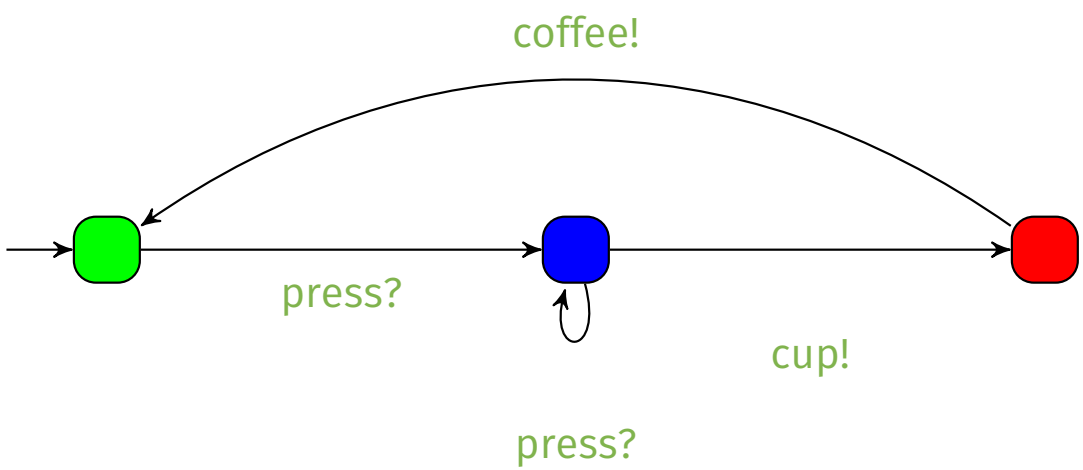
Timed automaton (TA)

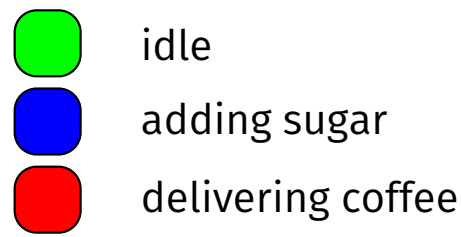
- Finite state automaton (sets of locations)



Timed automaton (TA)

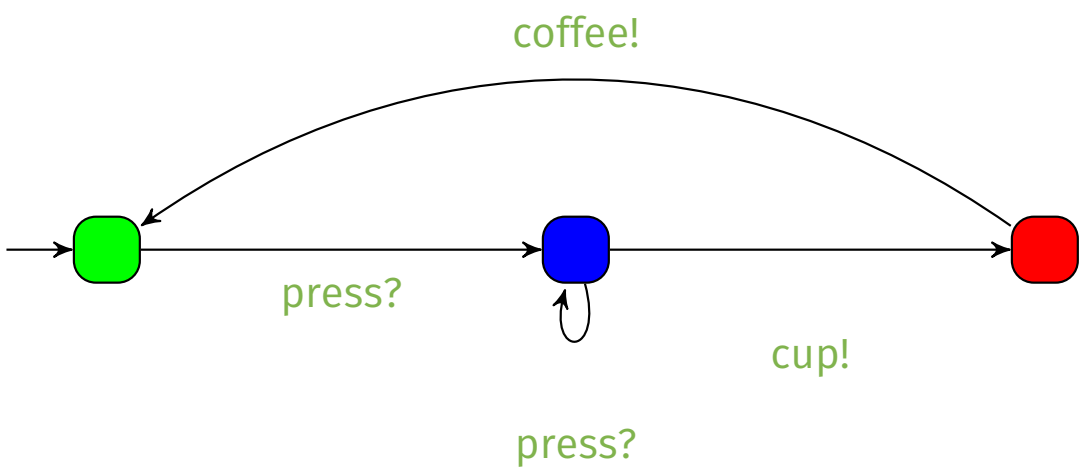
- Finite state automaton (sets of locations and actions)

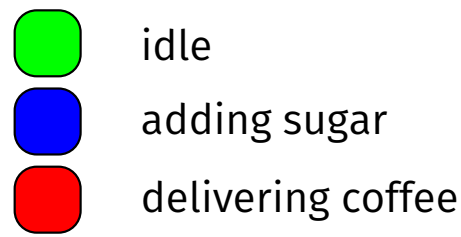




Timed automaton (TA)

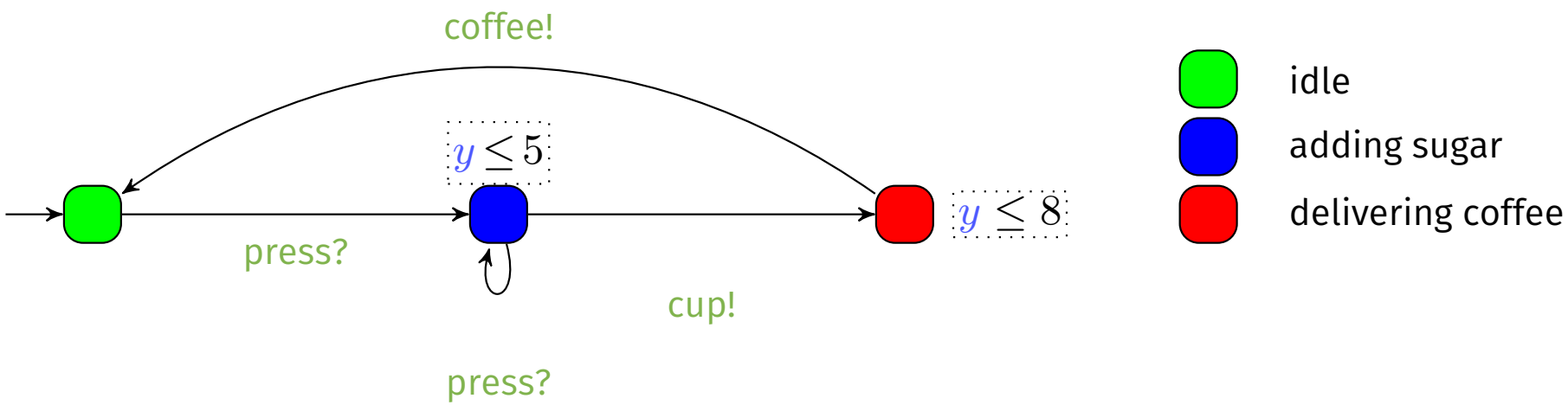
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate





Timed automaton (TA)

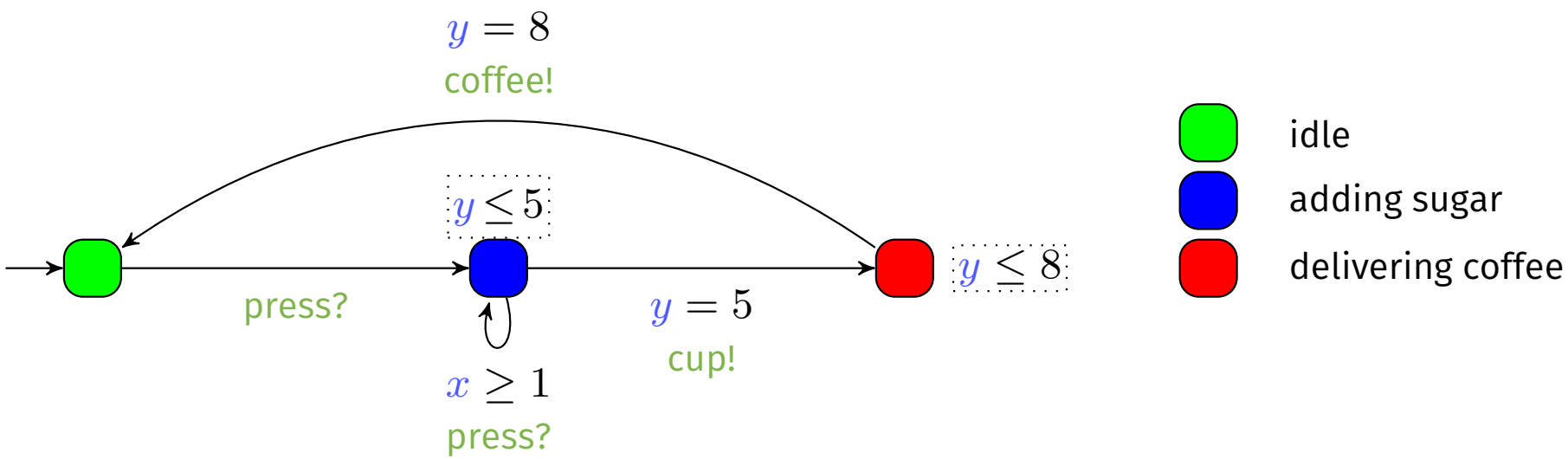
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants
- Features
  - Location invariant: property to be verified to stay at a location



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

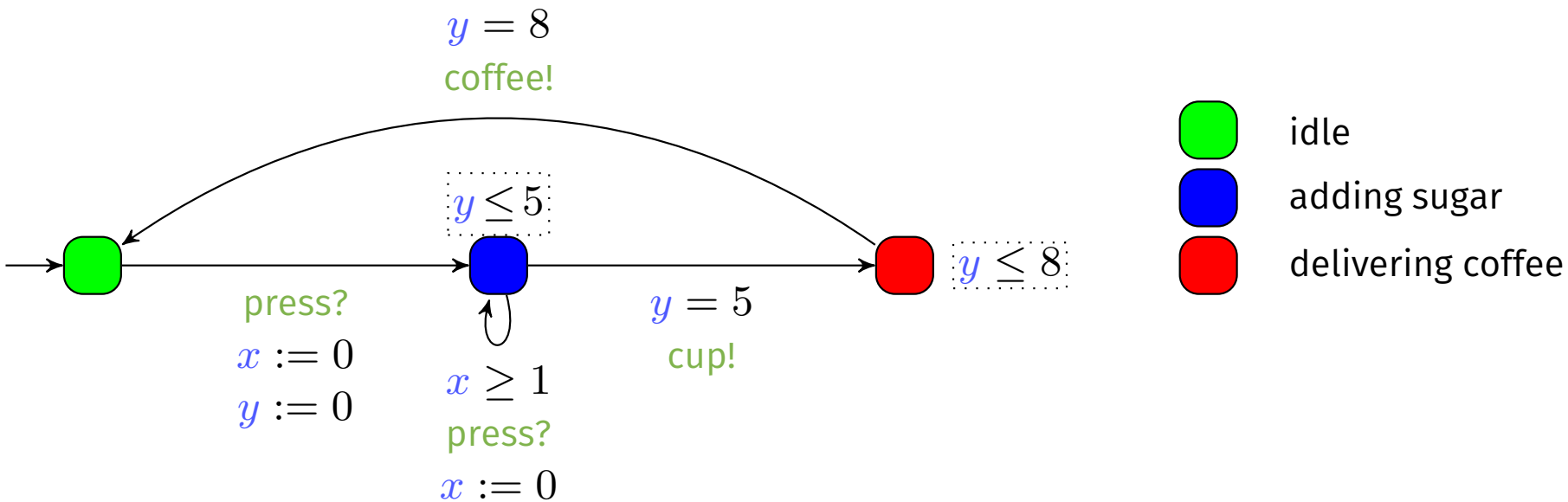
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition



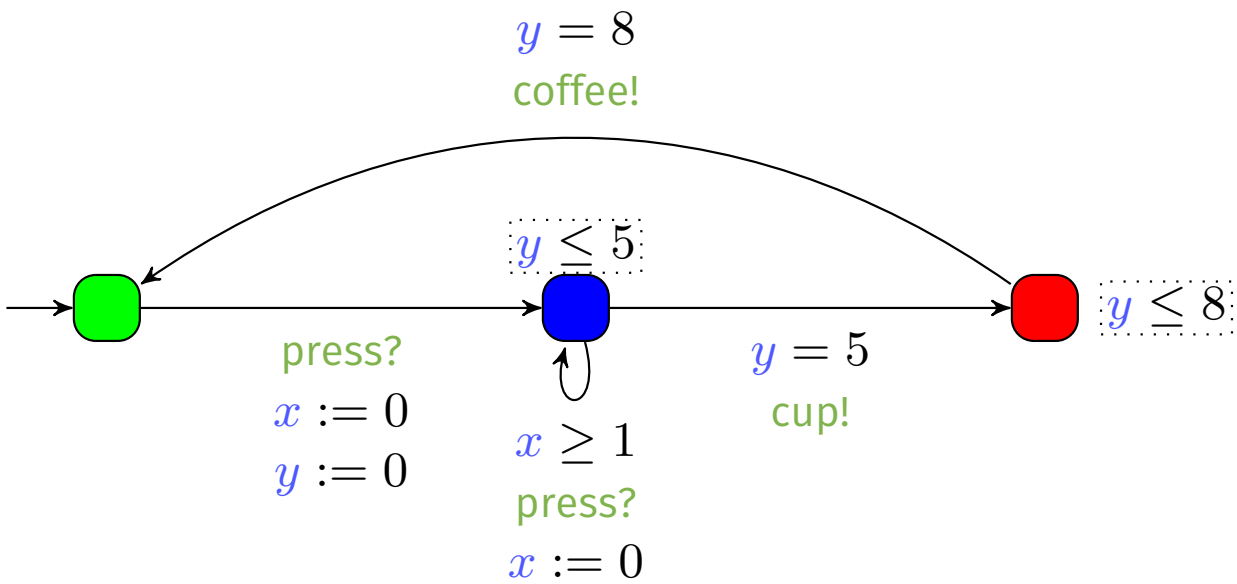
Timed automaton (TA)

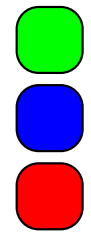
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

- 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



The most critical system: The coffee machine



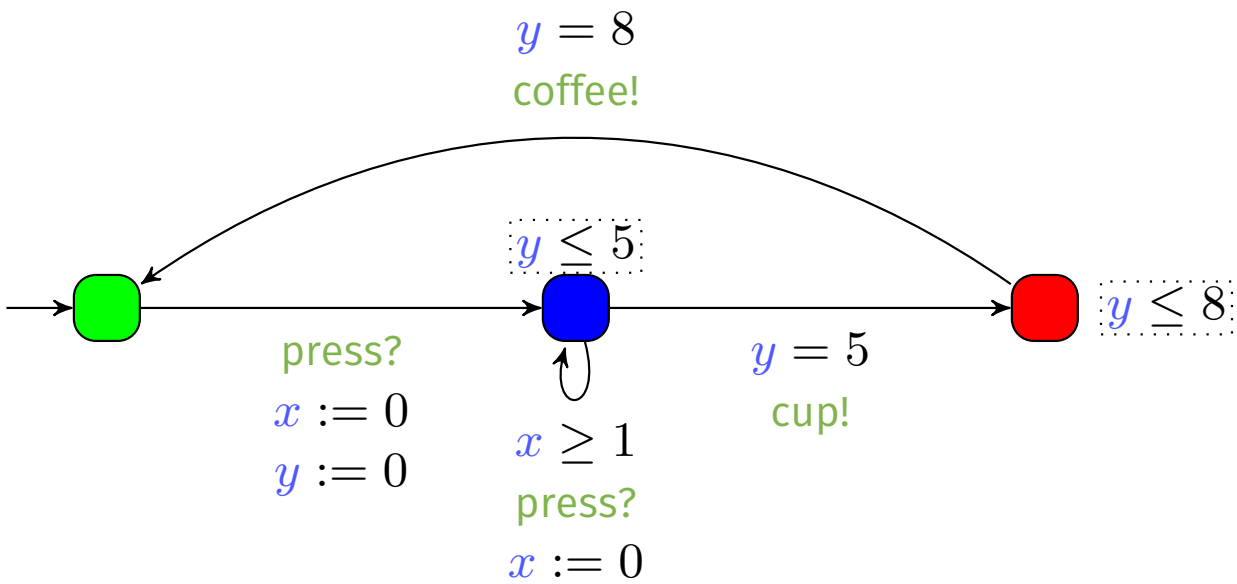


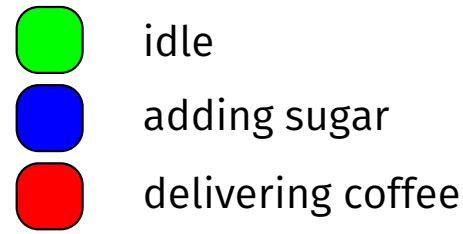
idle

adding sugar

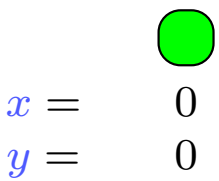
delivering coffee

The most critical system: The coffee machine

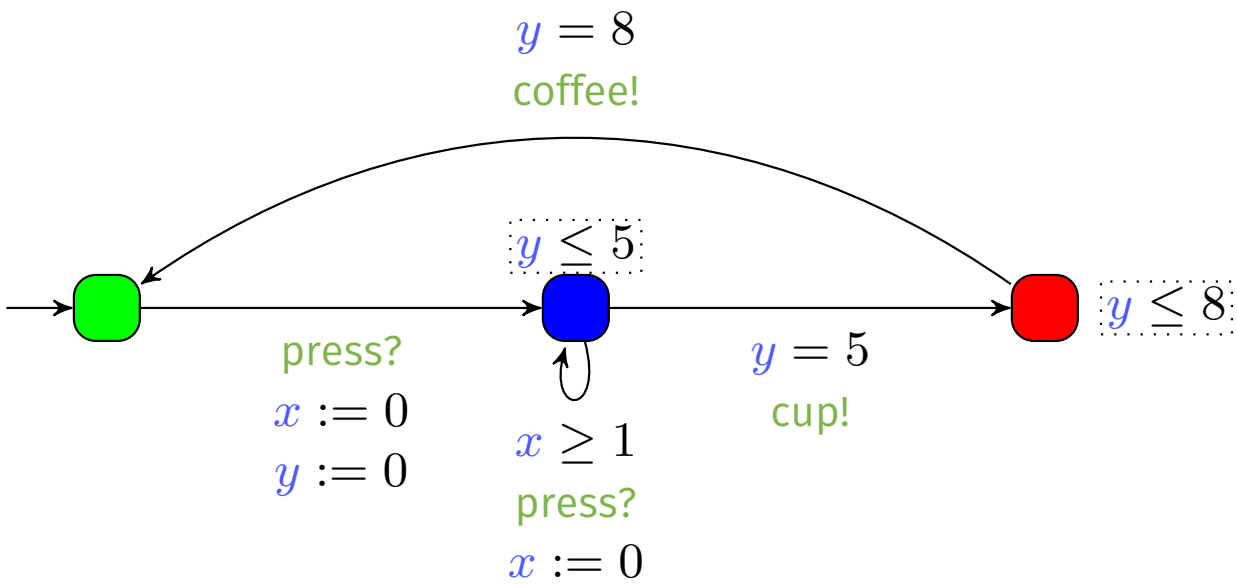


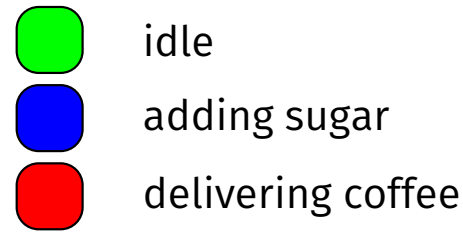


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

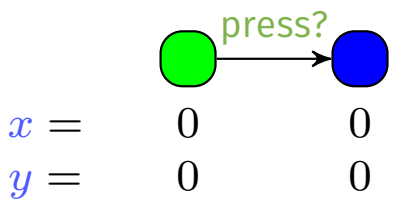


The most critical system: The coffee machine

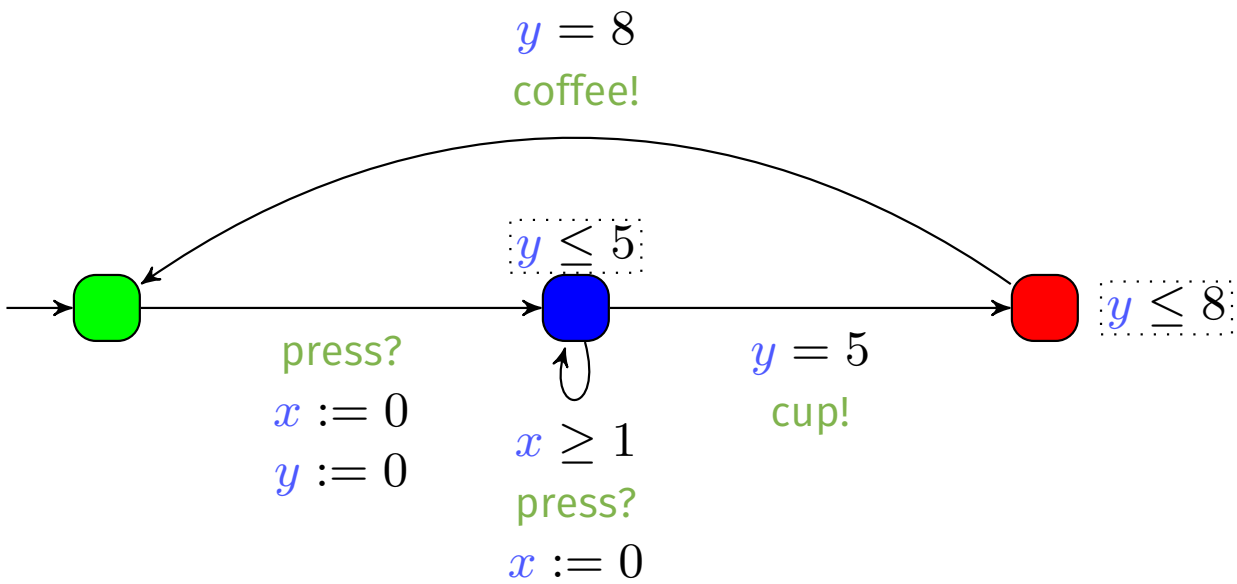


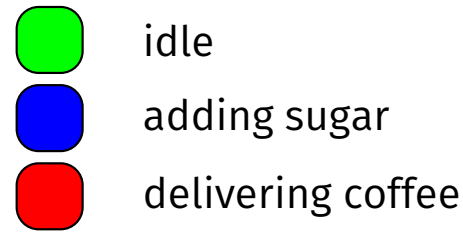


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

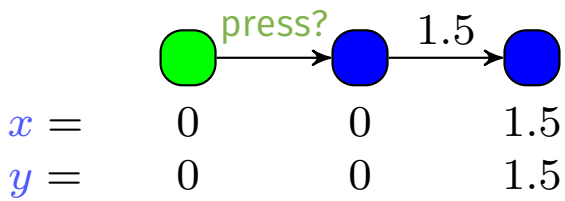


The most critical system: The coffee machine

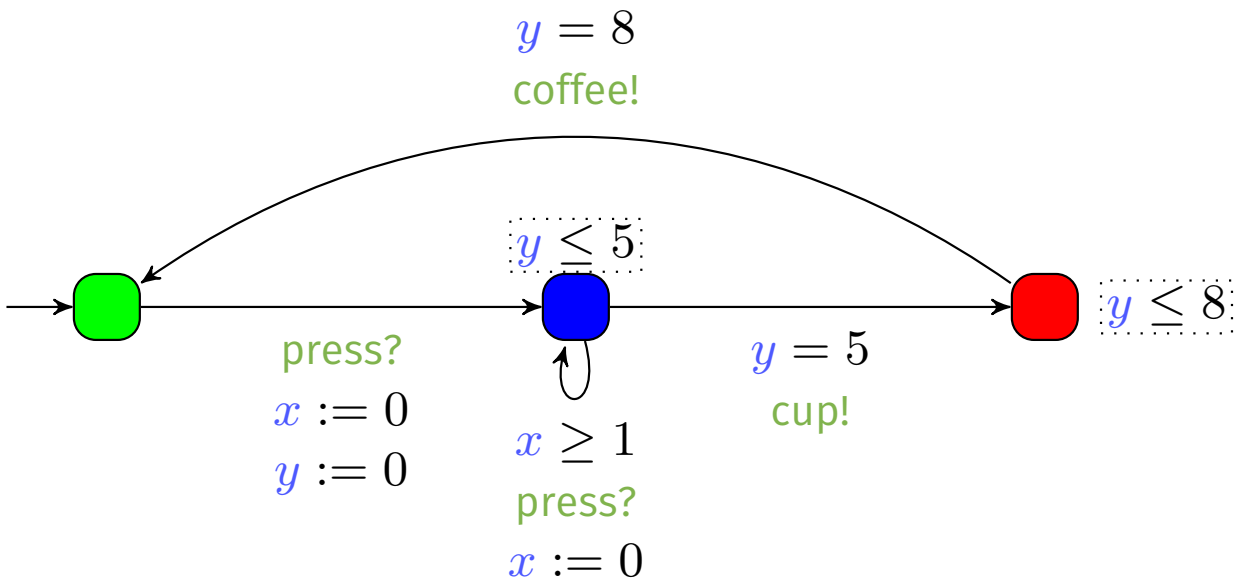


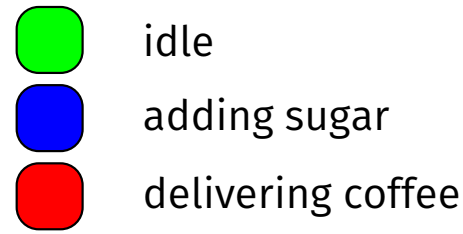


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

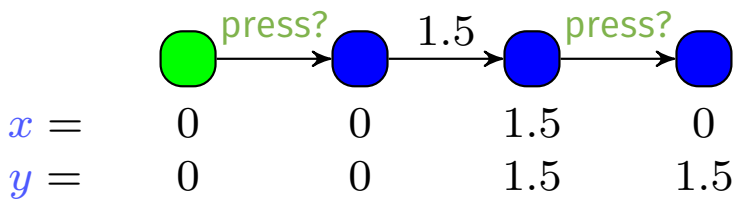


The most critical system: The coffee machine

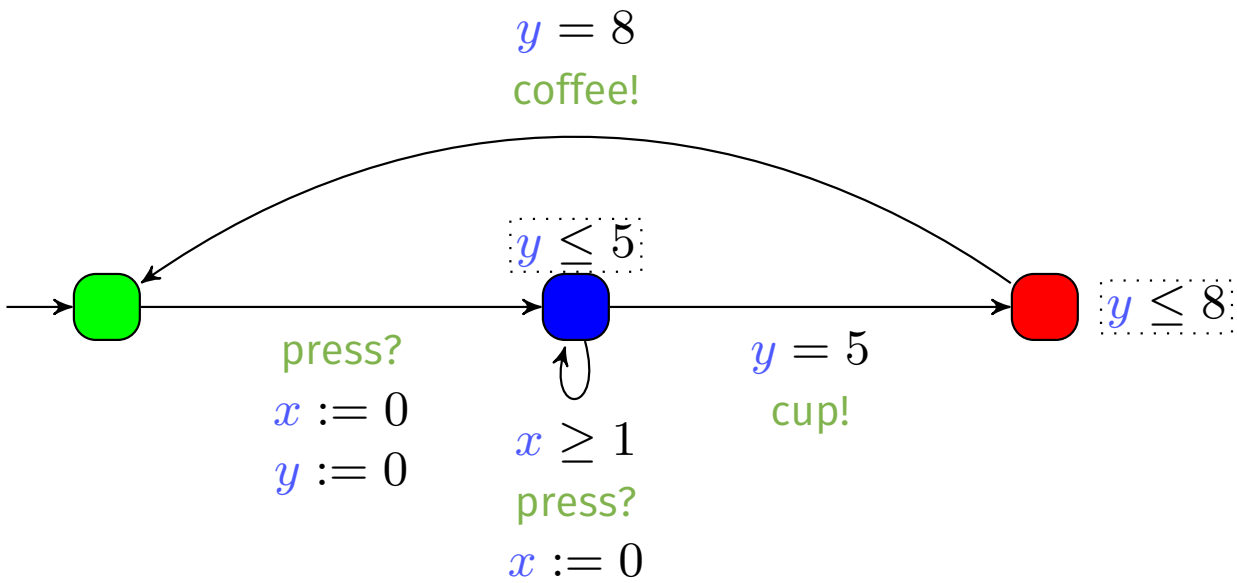


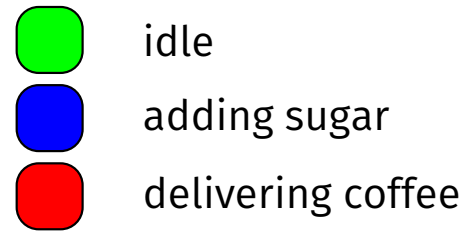


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

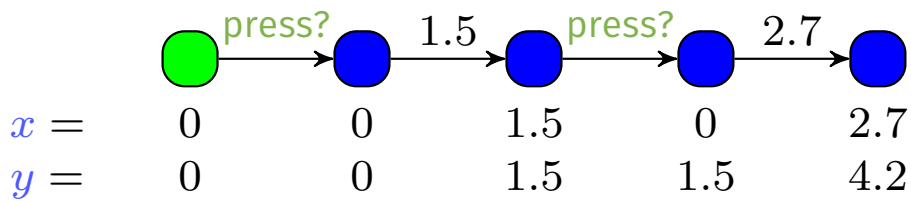


The most critical system: The coffee machine

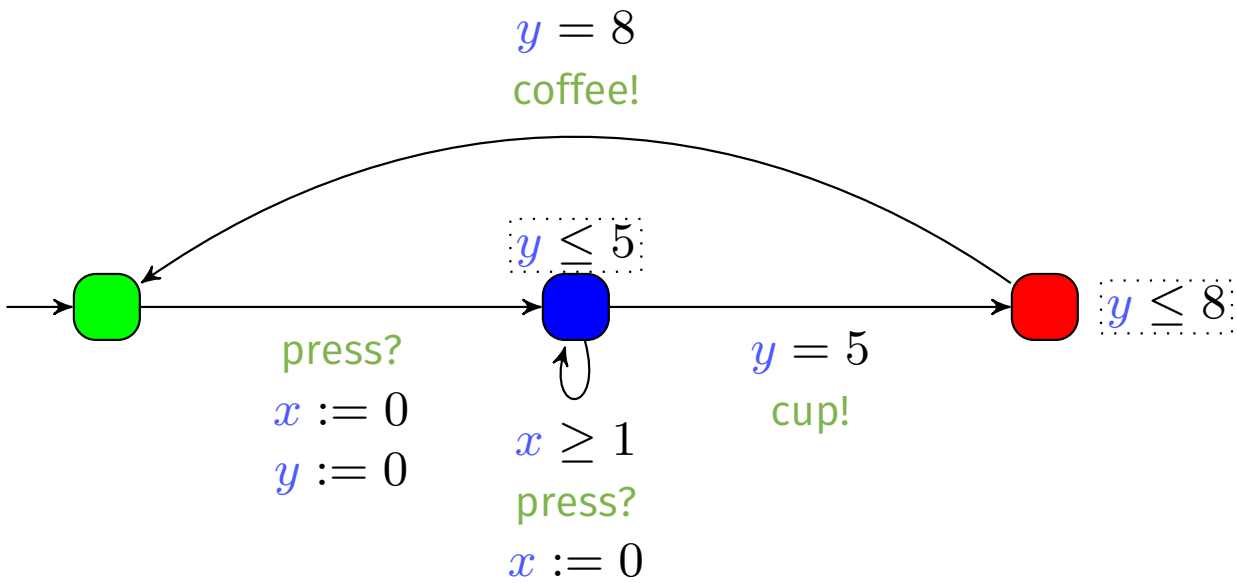


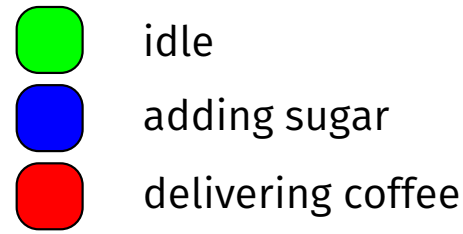


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

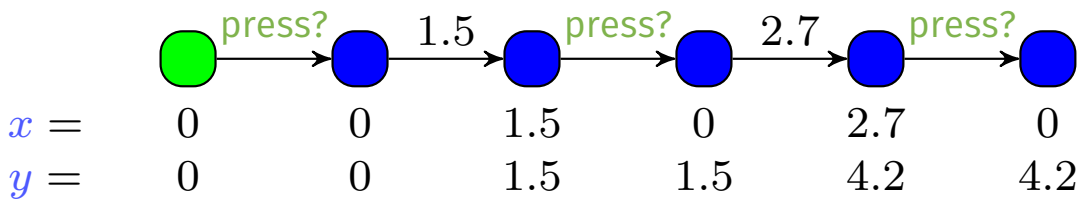


The most critical system: The coffee machine

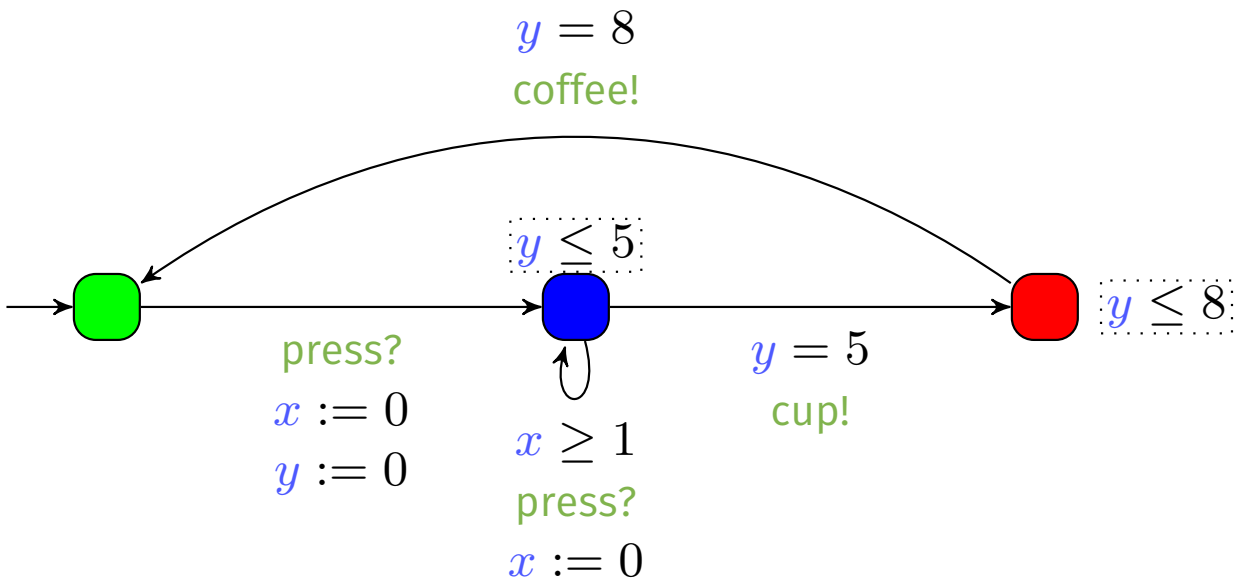


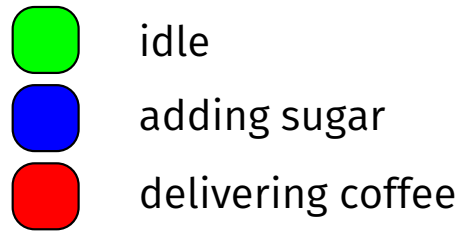


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

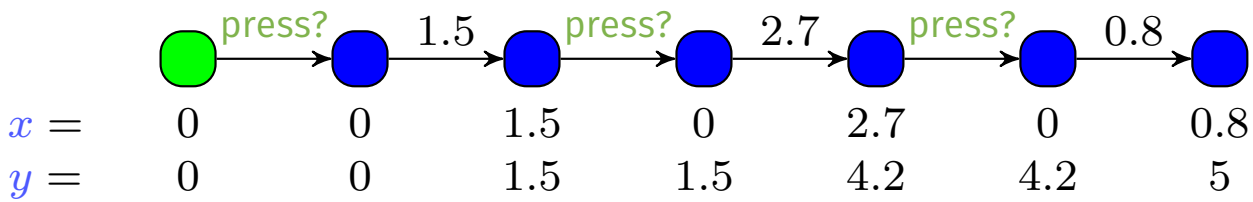


The most critical system: The coffee machine

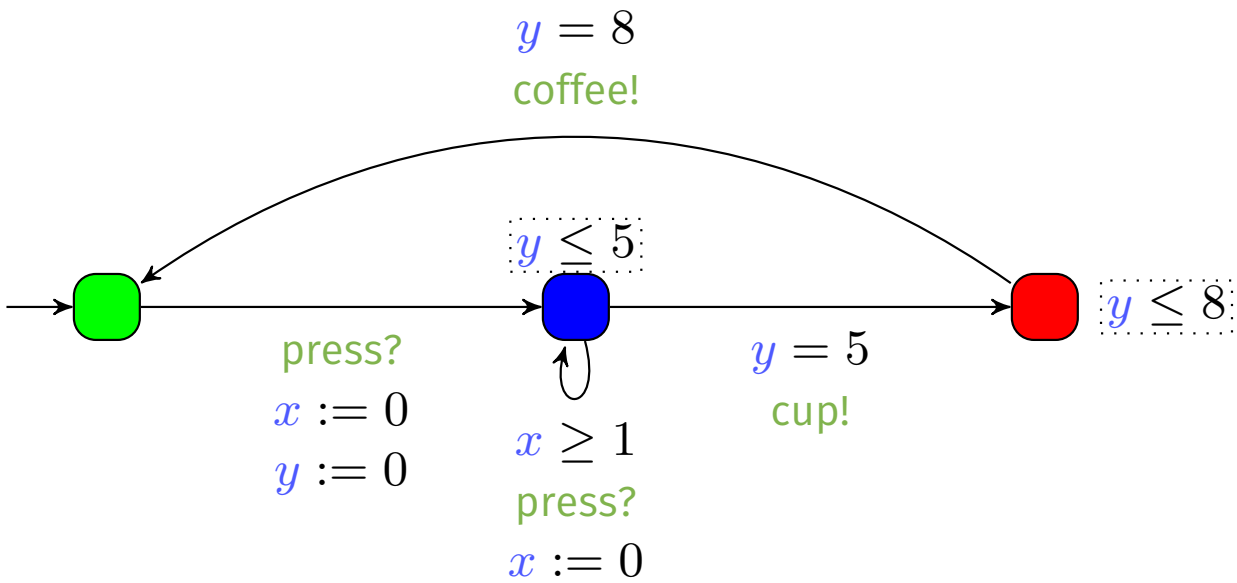


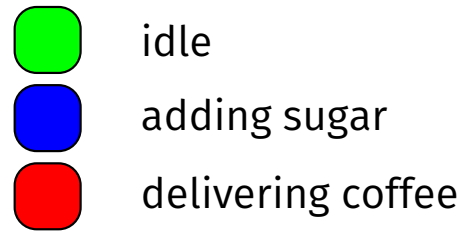


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

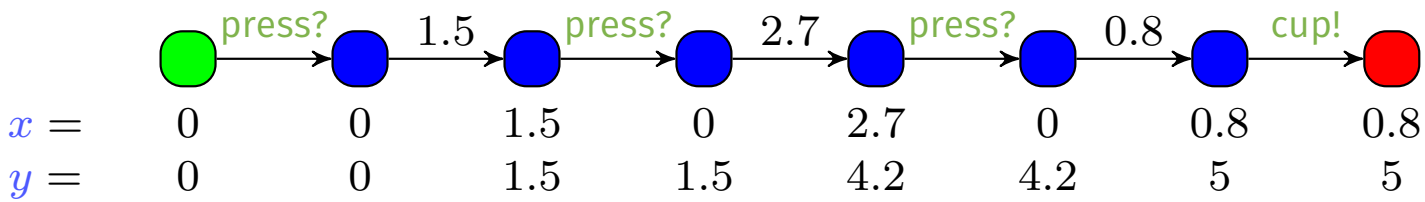


The most critical system: The coffee machine

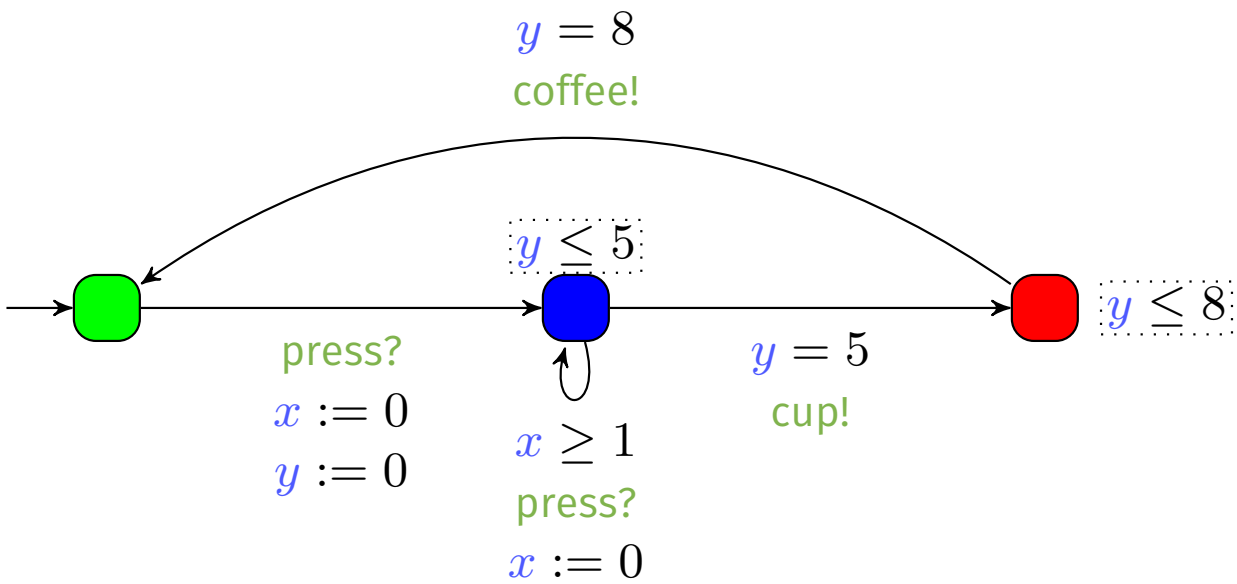


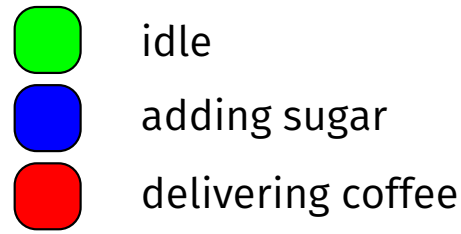


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

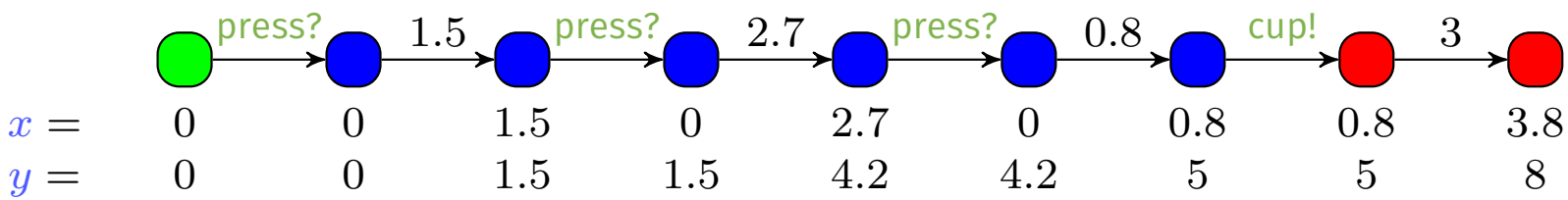


The most critical system: The coffee machine

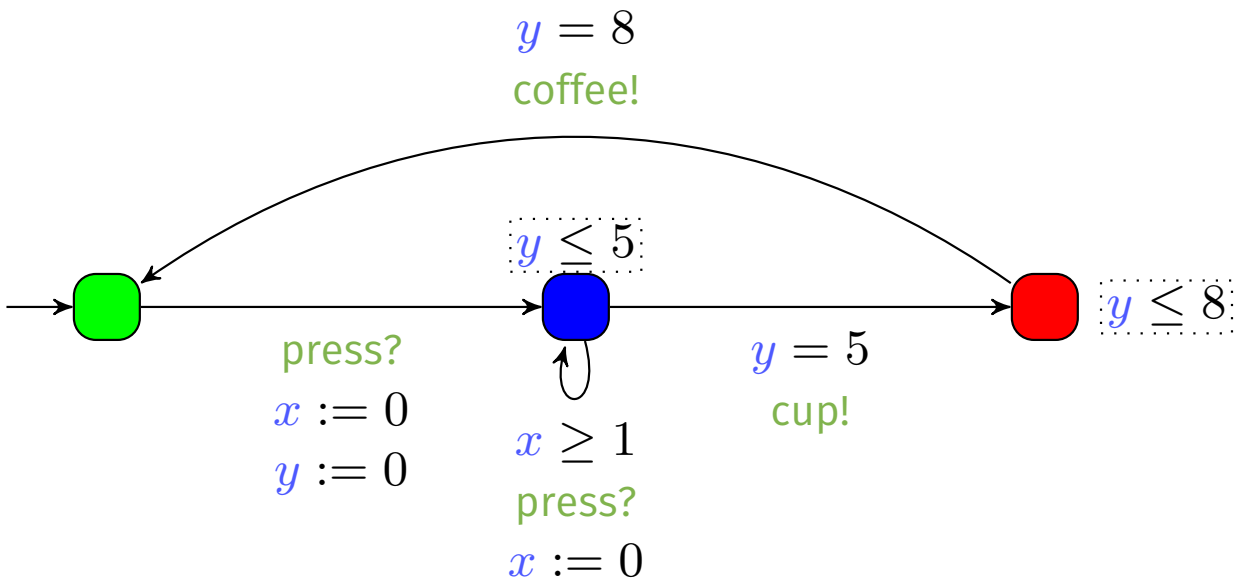


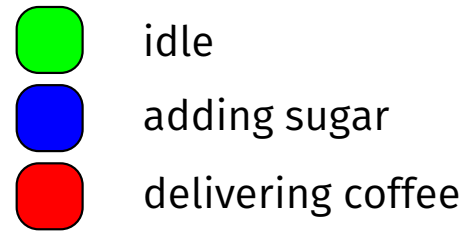


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

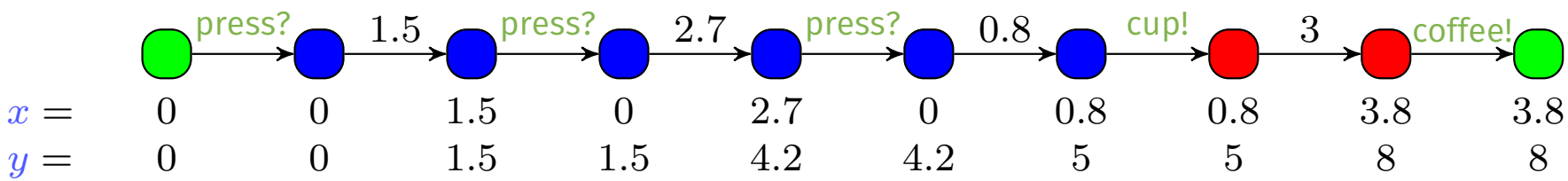


The most critical system: The coffee machine





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



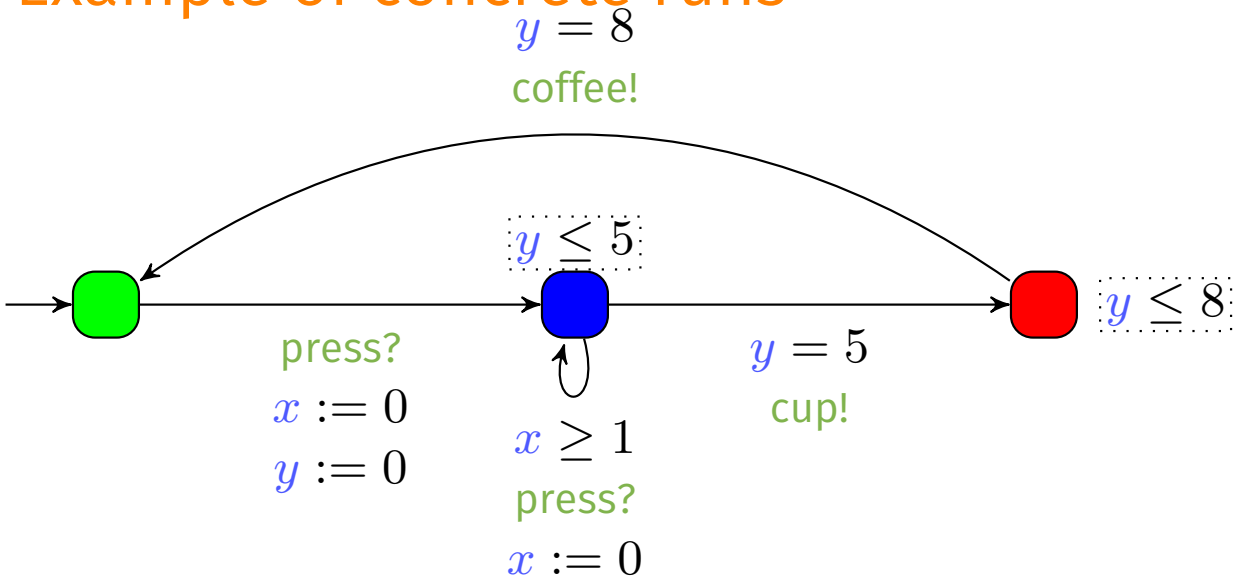
Concrete semantics of timed automata

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

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

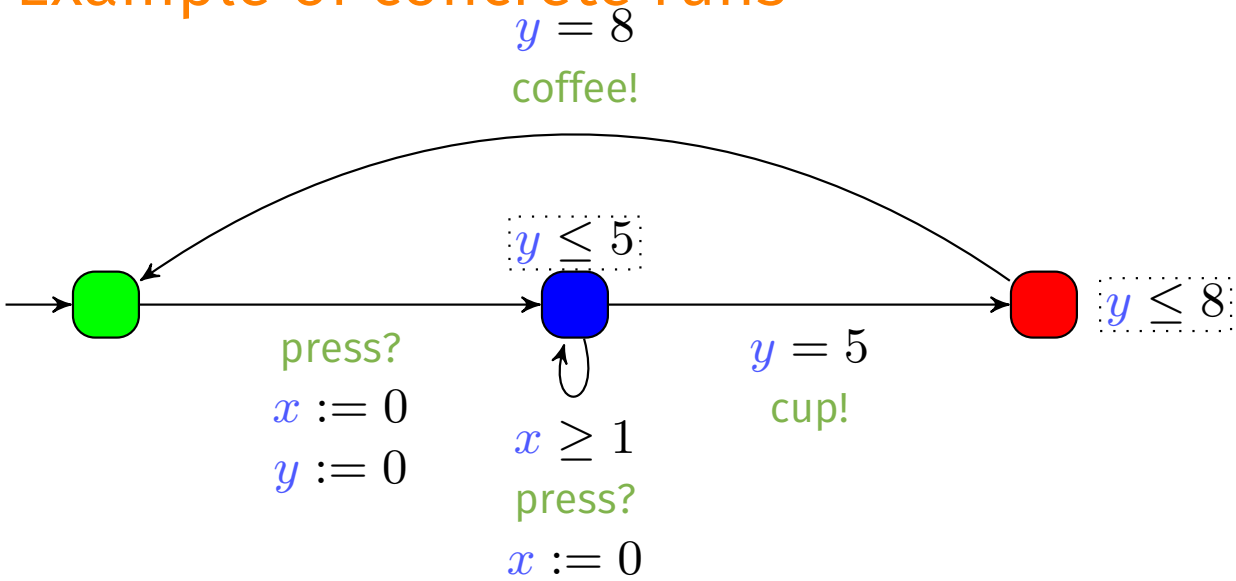
- Concrete run: alternating sequence of concrete states and actions or time elapse

Example of concrete runs

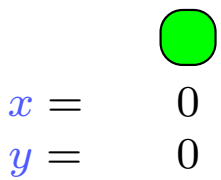


- Possible concrete runs for the coffee machine

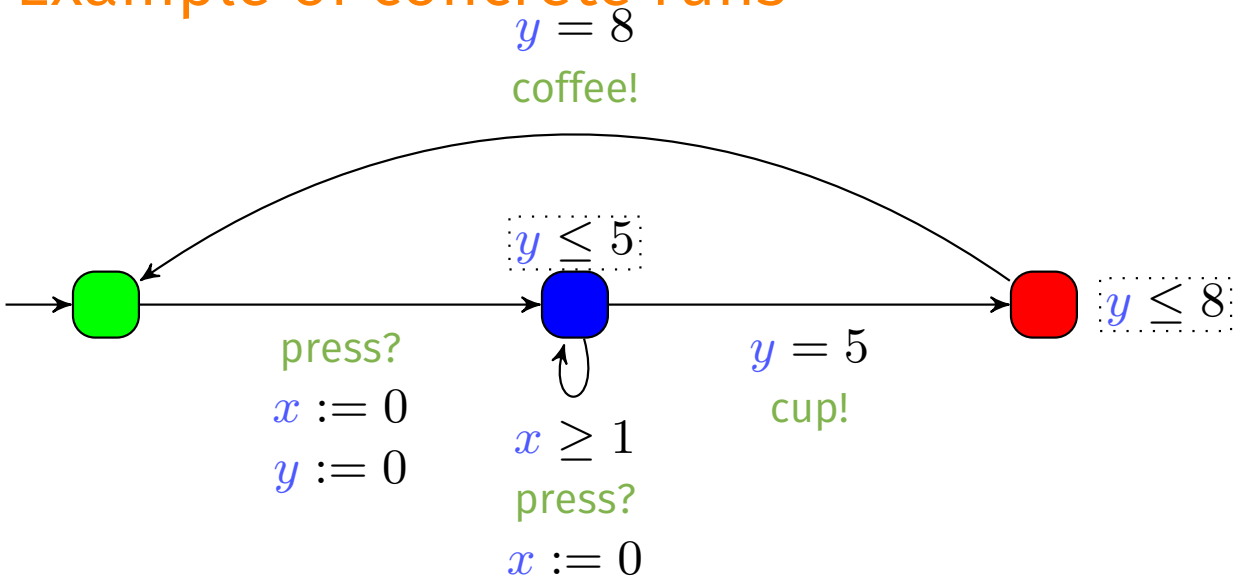
Example of concrete runs



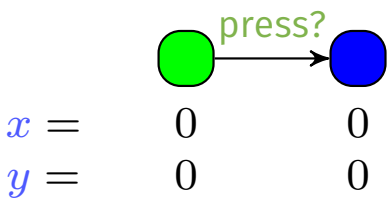
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



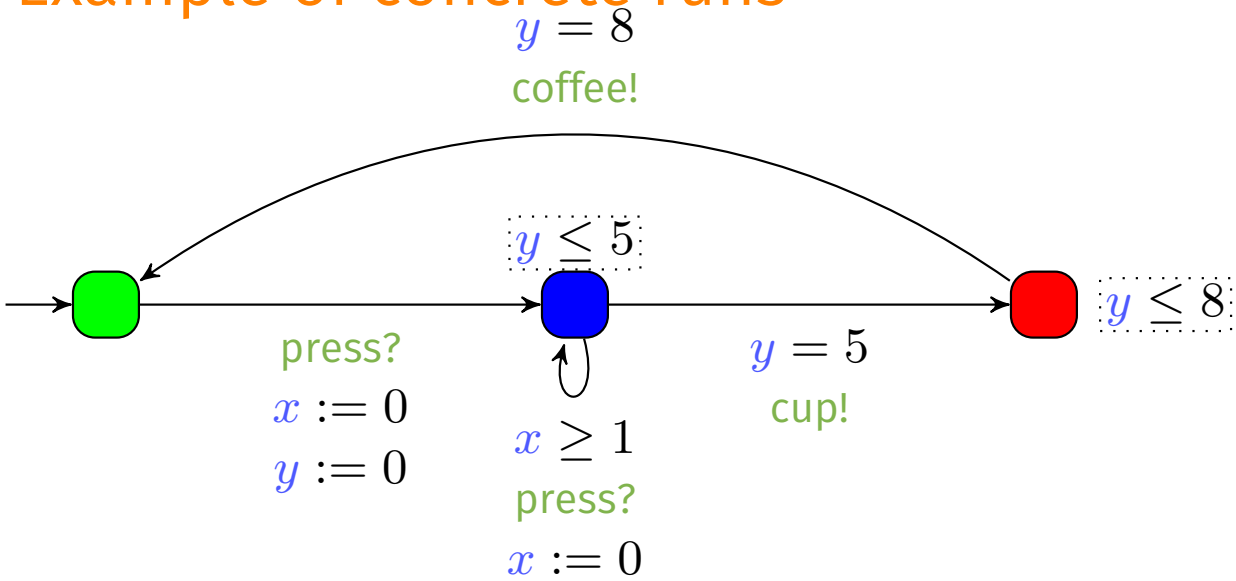
Example of concrete runs



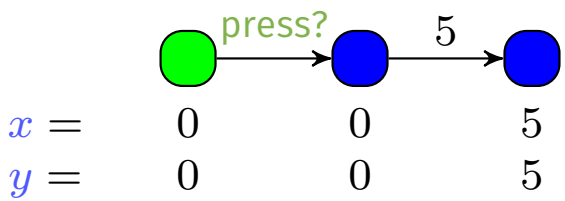
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



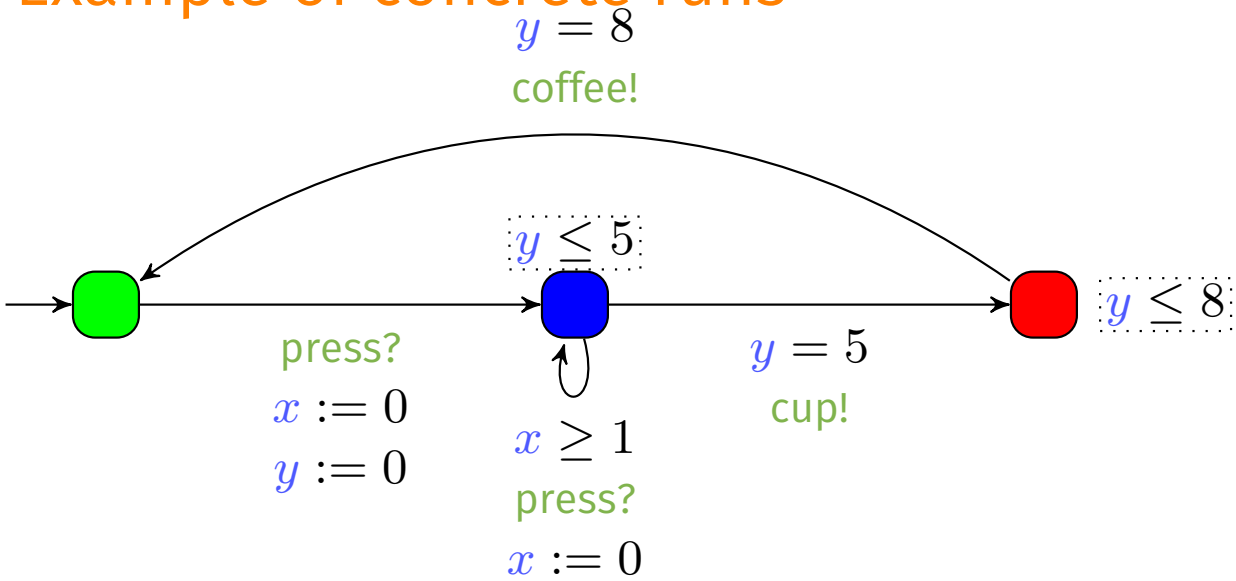
Example of concrete runs



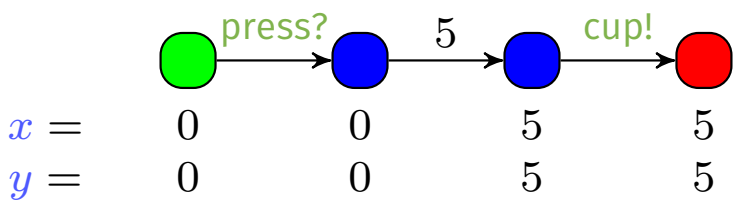
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



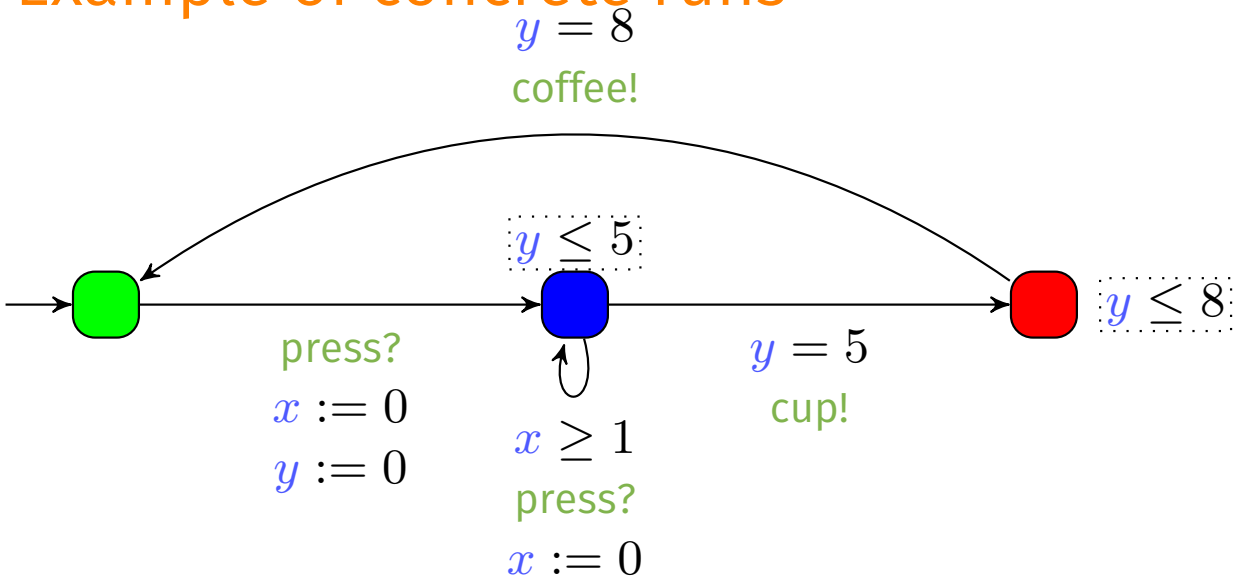
Example of concrete runs



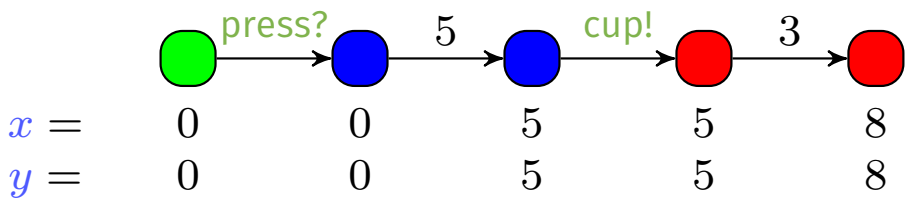
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



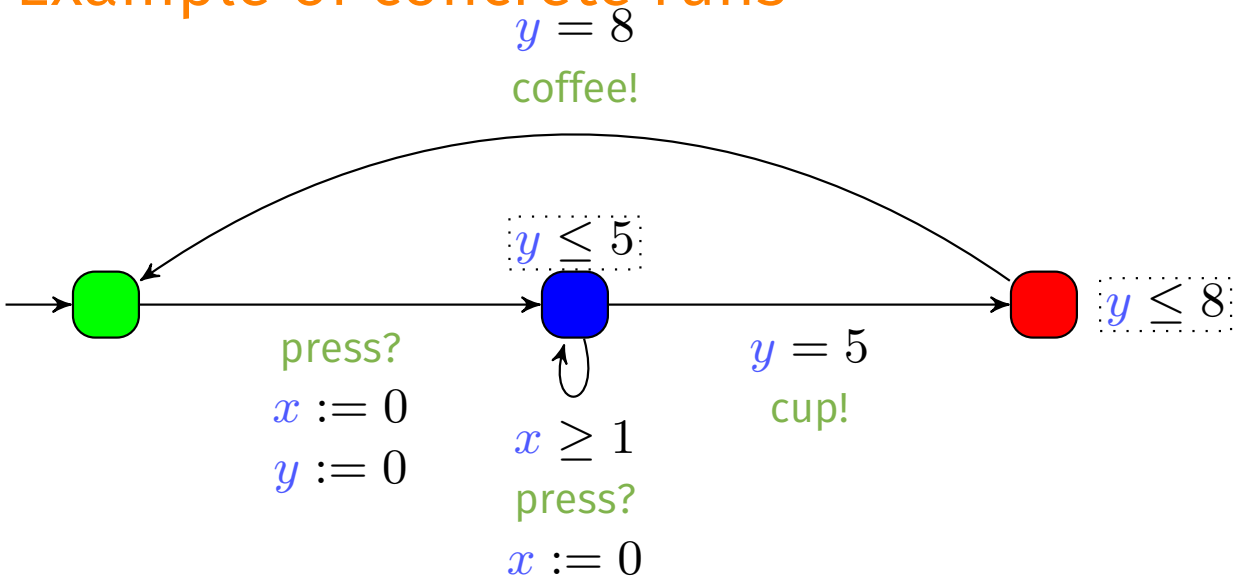
Example of concrete runs



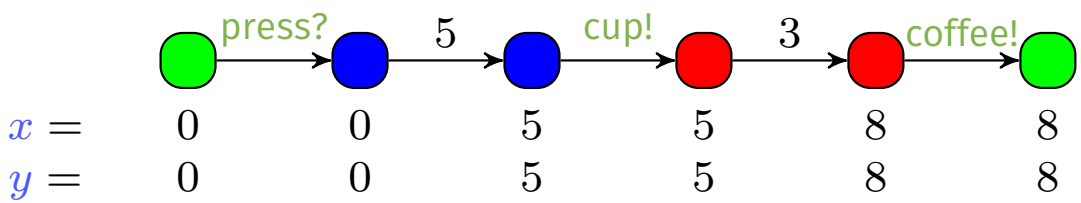
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



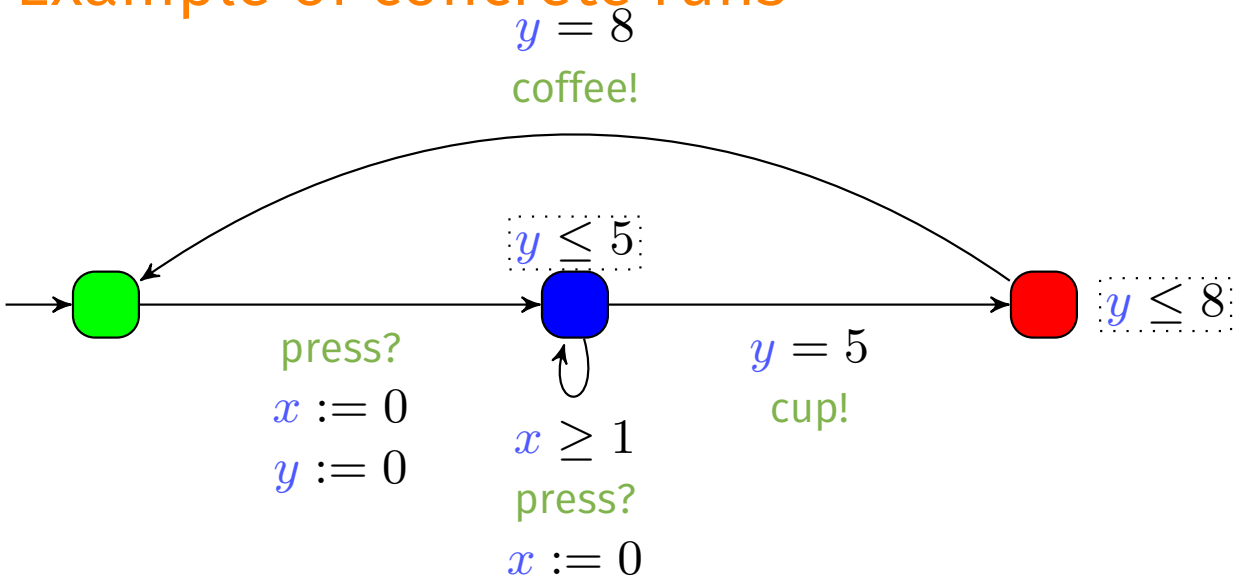
Example of concrete runs



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



Example of concrete runs



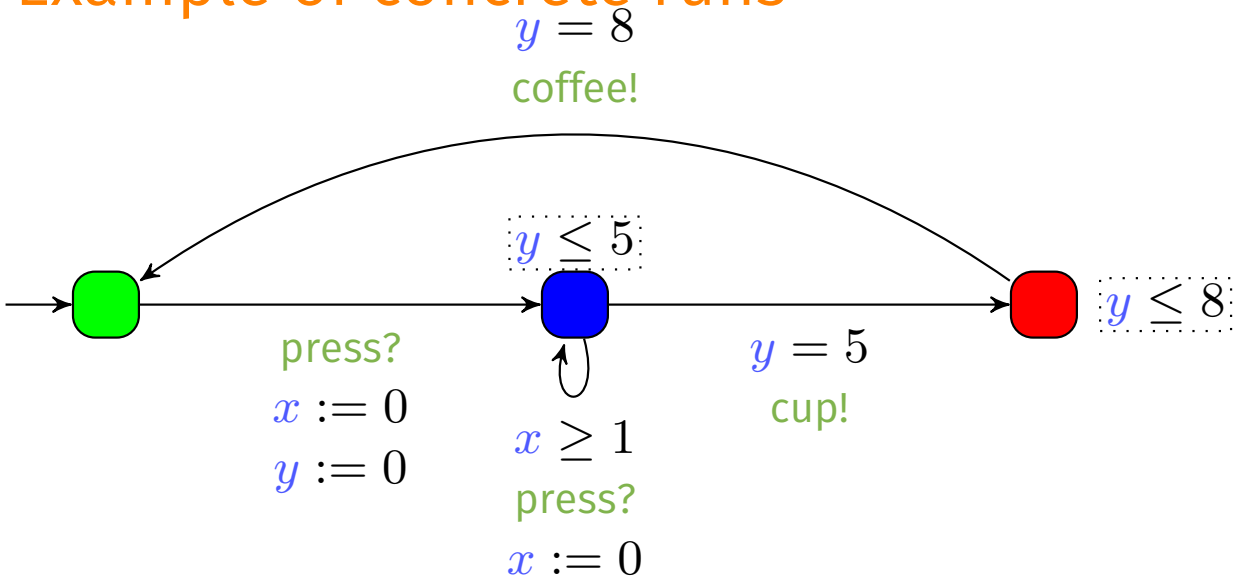
- Possible concrete runs for the coffee machine
  - Coffee with no sugar

$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

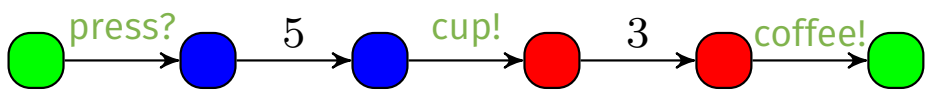
  - Coffee with 2 doses of sugar

$x =$	0
$y =$	0

Example of concrete runs



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

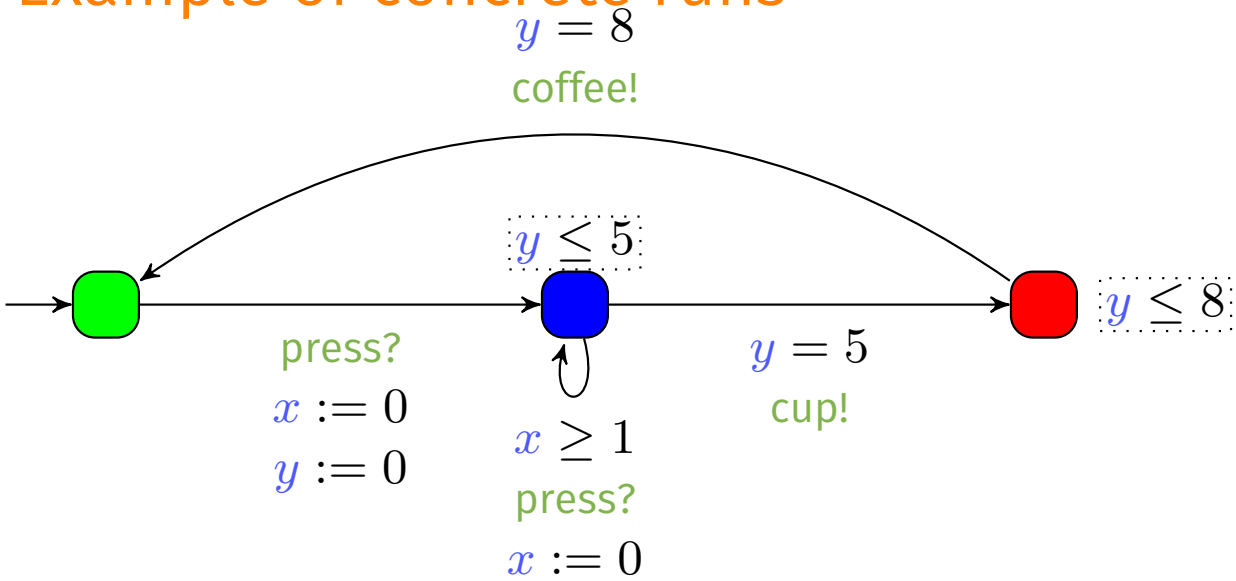


$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

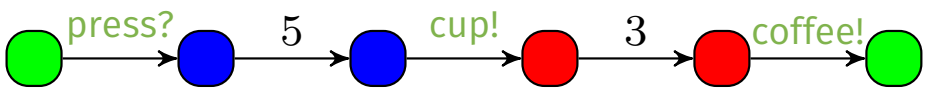
  - Coffee with 2 doses of sugar

$x =$	0	0
$y =$	0	0

Example of concrete runs

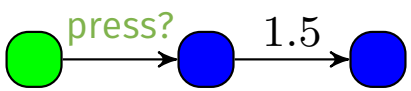


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



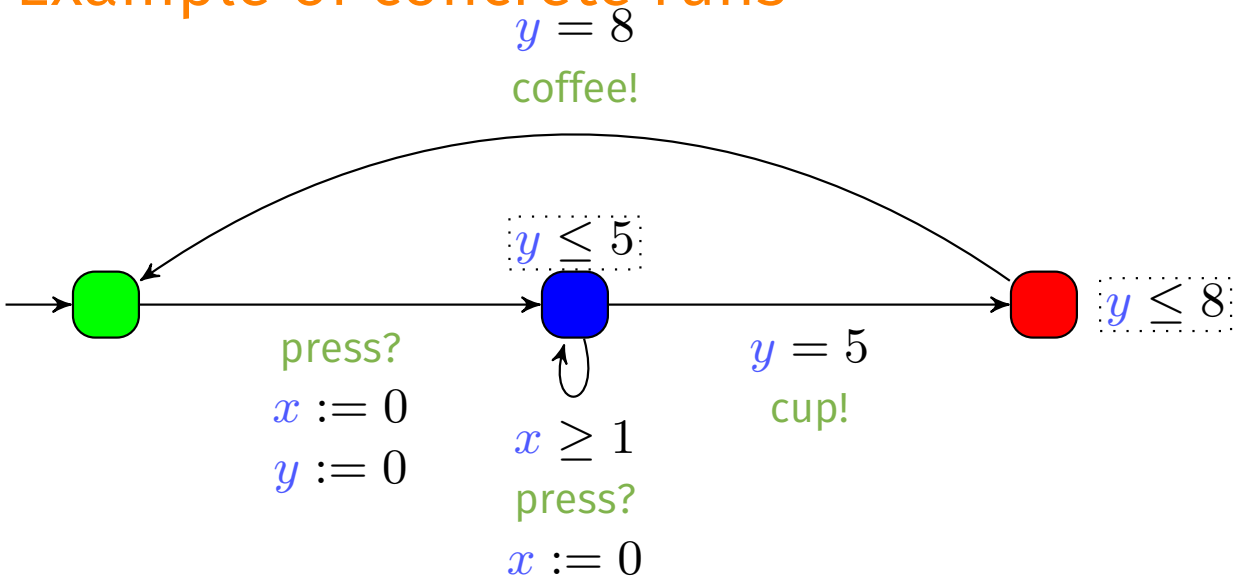
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar

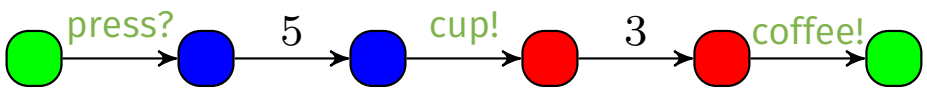


$x =$	0	0	1.5
$y =$	0	0	1.5

Example of concrete runs

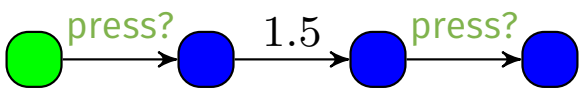


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



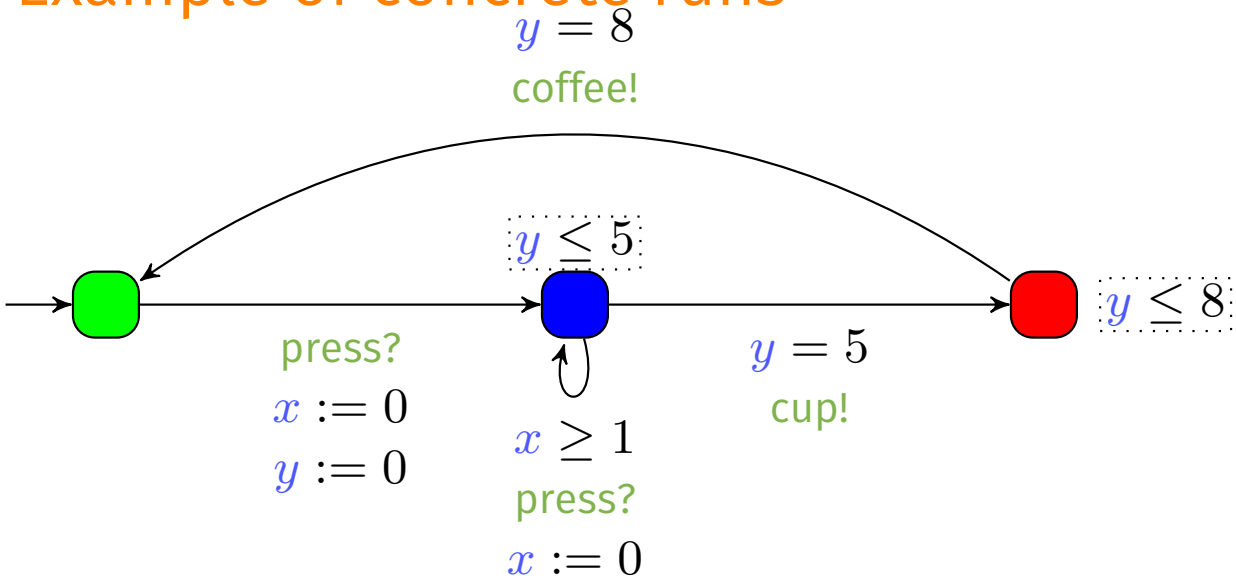
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar

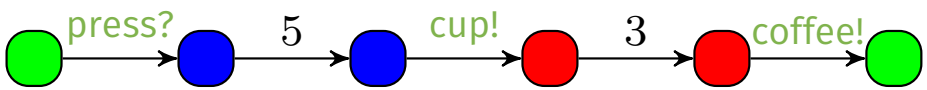


$x =$	0	0	1.5	0
$y =$	0	0	1.5	1.5

Example of concrete runs

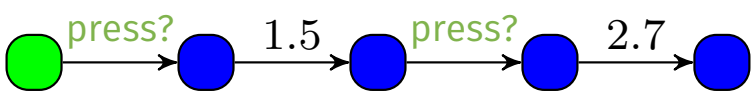


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



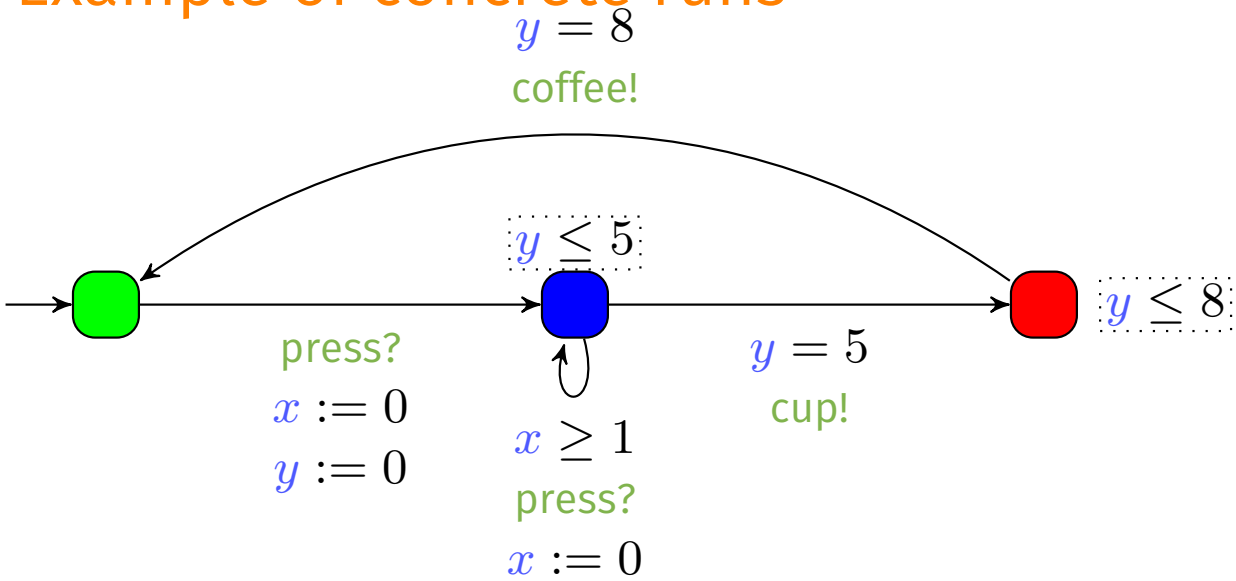
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar

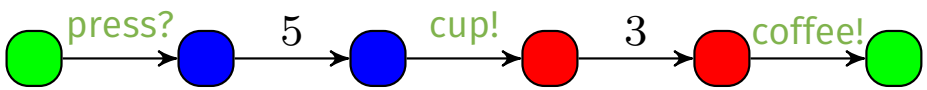


$x =$	0	0	1.5	0	2.7
$y =$	0	0	1.5	1.5	4.2

Example of concrete runs



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

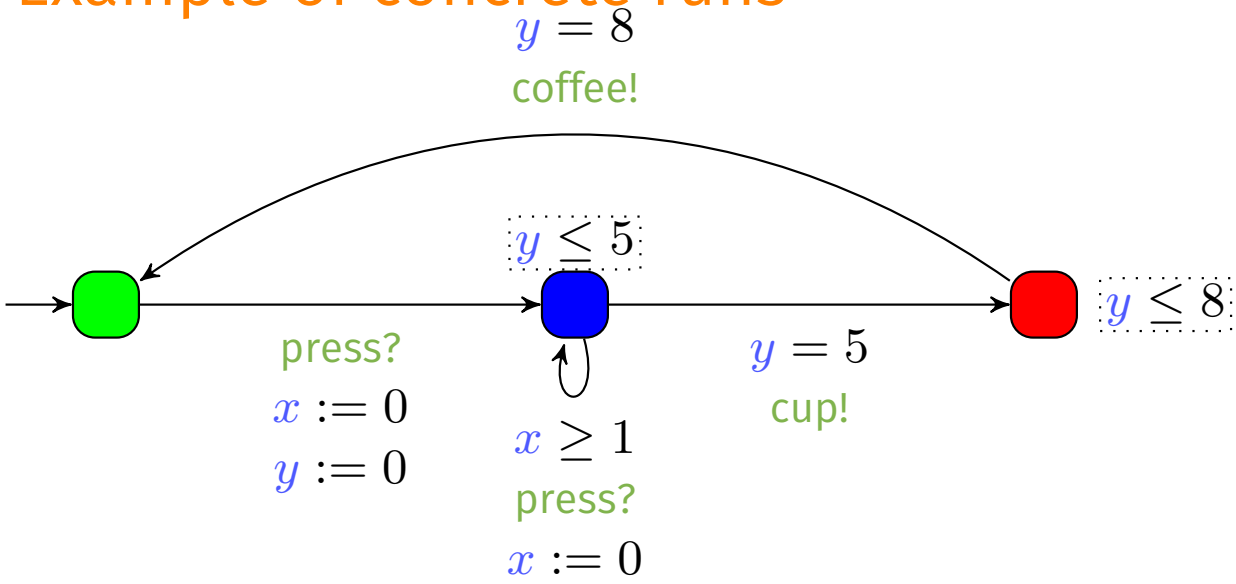


$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

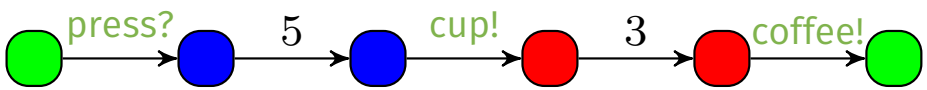
  - Coffee with 2 doses of sugar

$x =$	0	0	1.5	0	2.7	0
$y =$	0	0	1.5	1.5	4.2	4.2

Example of concrete runs

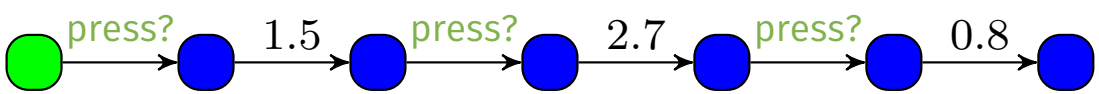


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



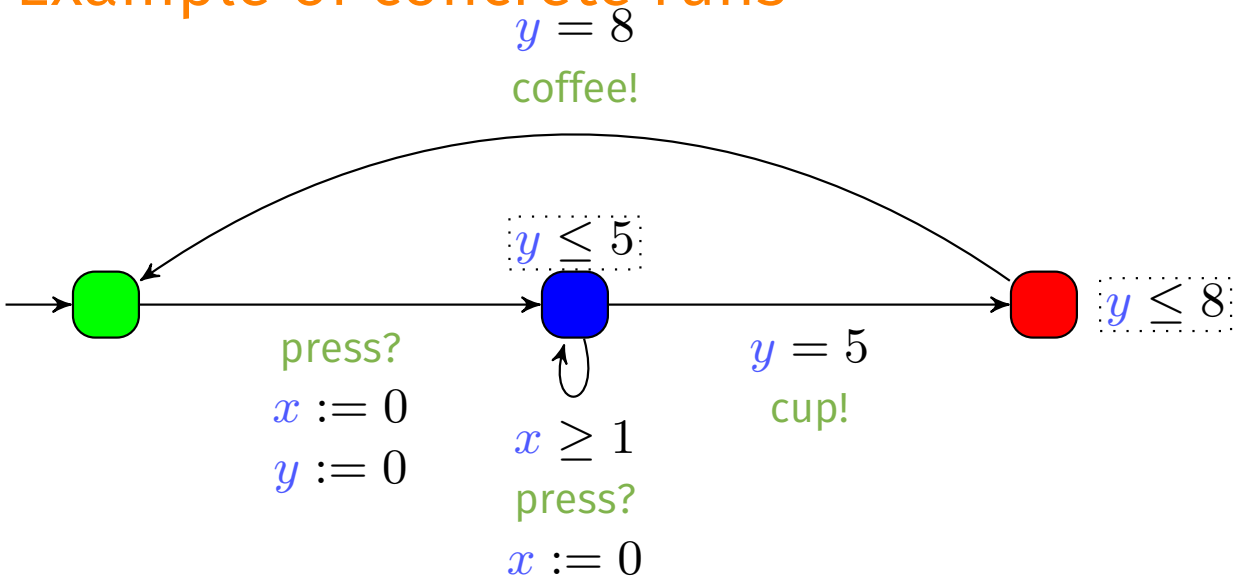
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar

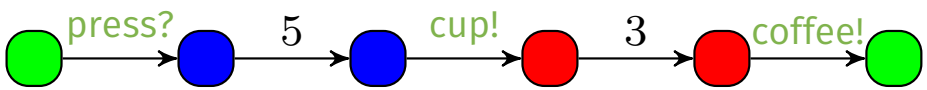


$x =$	0	0	1.5	0	2.7	0	0.8
$y =$	0	0	1.5	1.5	4.2	4.2	5

Example of concrete runs

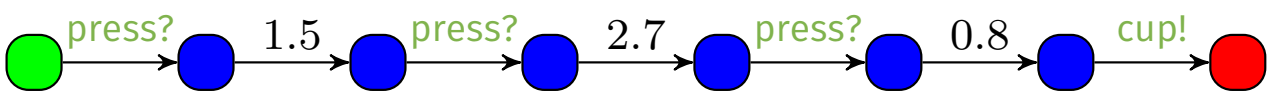


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



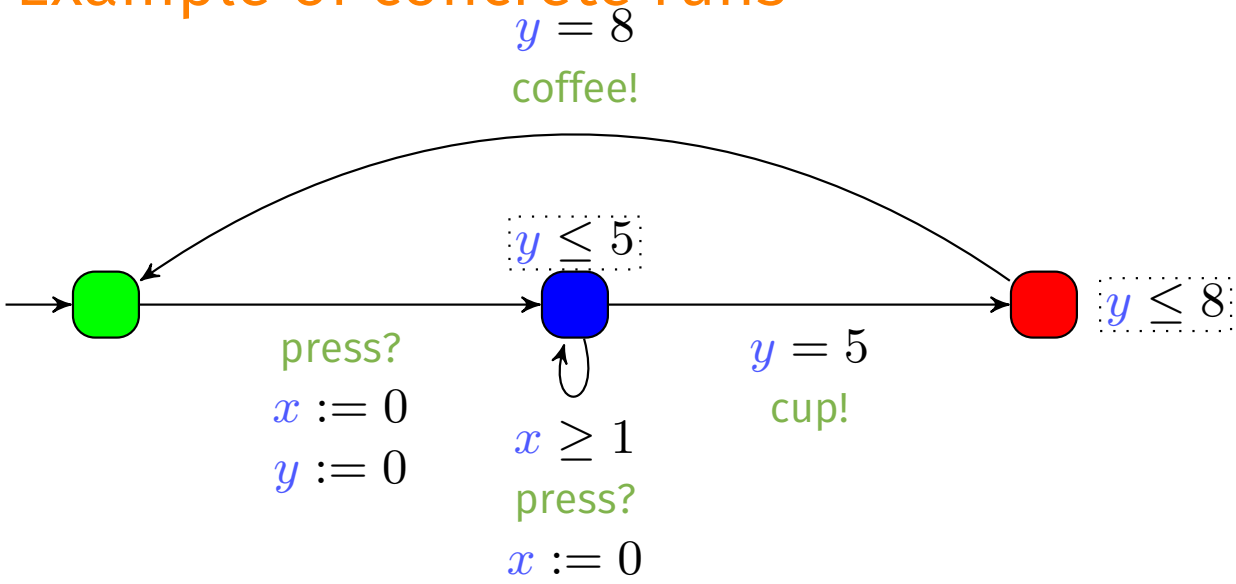
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar

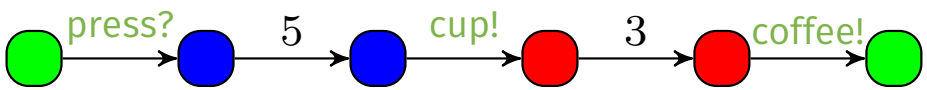


$x =$	0	0	1.5	0	2.7	0	0.8	0.8
$y =$	0	0	1.5	1.5	4.2	4.2	5	5

Example of concrete runs

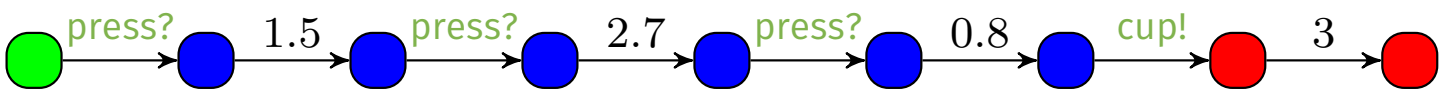


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



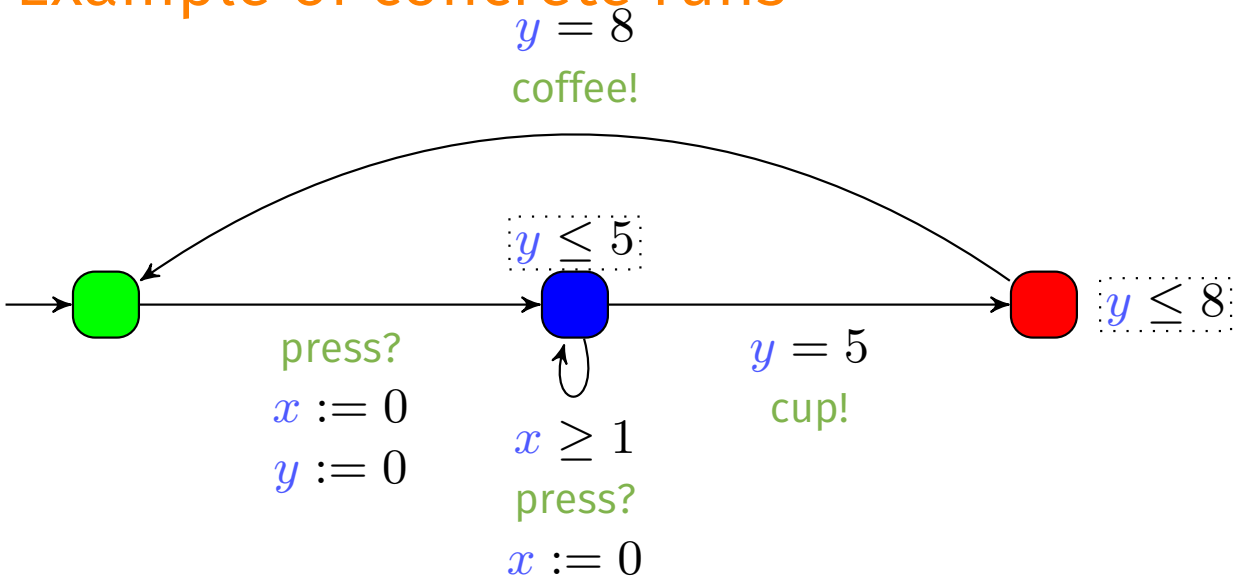
$x =$	0	0	5	5	8	8
$y =$	0	0	5	5	8	8

  - Coffee with 2 doses of sugar

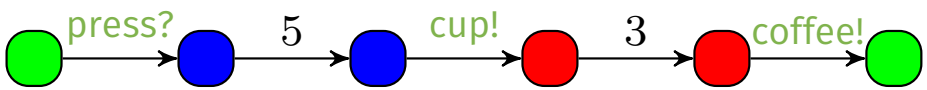


$x =$	0	0	1.5	0	2.7	0	0.8	0.8	3.8
$y =$	0	0	1.5	1.5	4.2	4.2	5	5	8

Example of concrete runs



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



$x =$	0	0	5	5	8	8
$y =$	0	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

Timed automata: A success story

- An expressive formalism
  - Dense time
  - Concurrency
- A tractable verification in theory
  - Reachability is PSPACE-complete [Alur and Dill, 1994]
- A very efficient verification in practice
  - Symbolic verification: relatively insensitive to constants
  - Several model checkers, notably UPPAAL [Larsen et al., 1997]
  - Long list of successful case studies

Outline

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 [Markey, 2011]: 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?

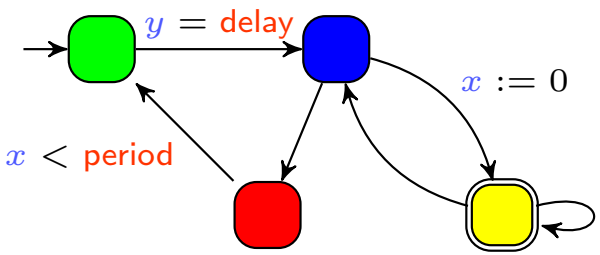
Beyond timed model checking: parameter synthesis

- 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 [Markey, 2011]: 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)

timed model checking



A model of the system

$?$

$\models$

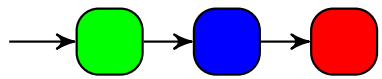
is unreachable

A property to be satisfied

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

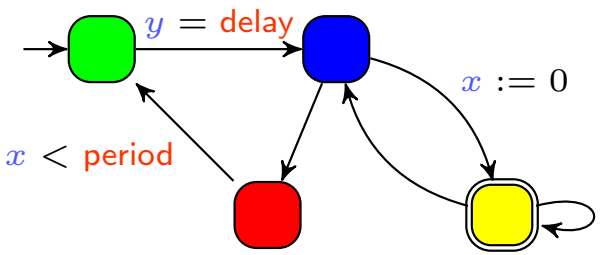
Yes

No



Counterexample

Parametric timed model checking



A model of the system

$?$

$\models$

A property to be satisfied

is unreachable

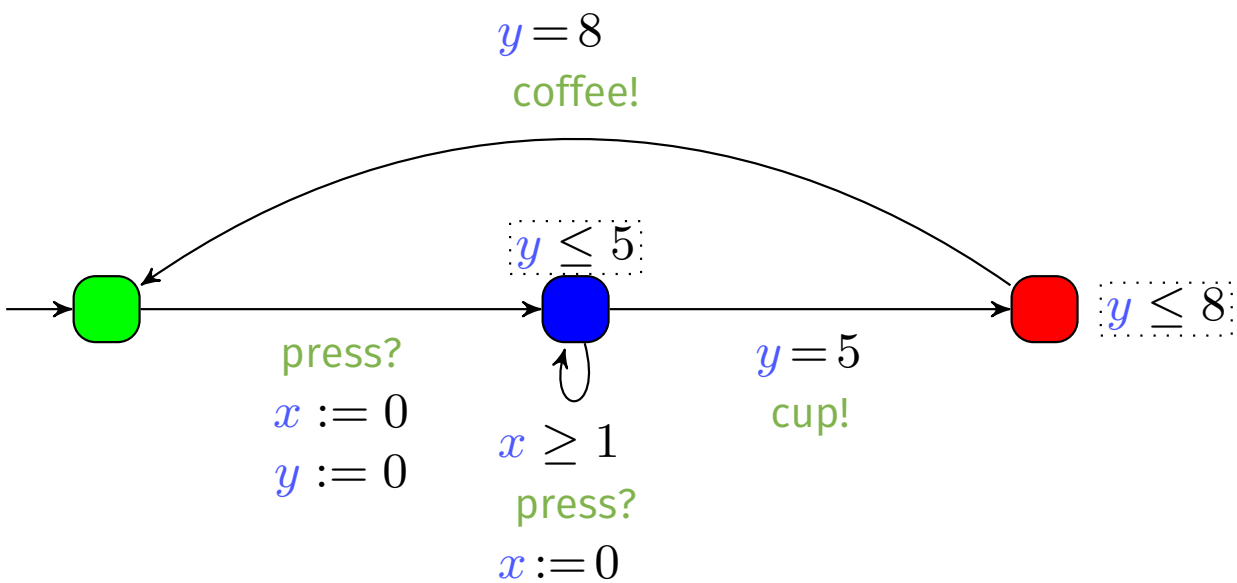
- Question: for what values of the parameters does the model of the system satisfy the property?

Yes if...

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

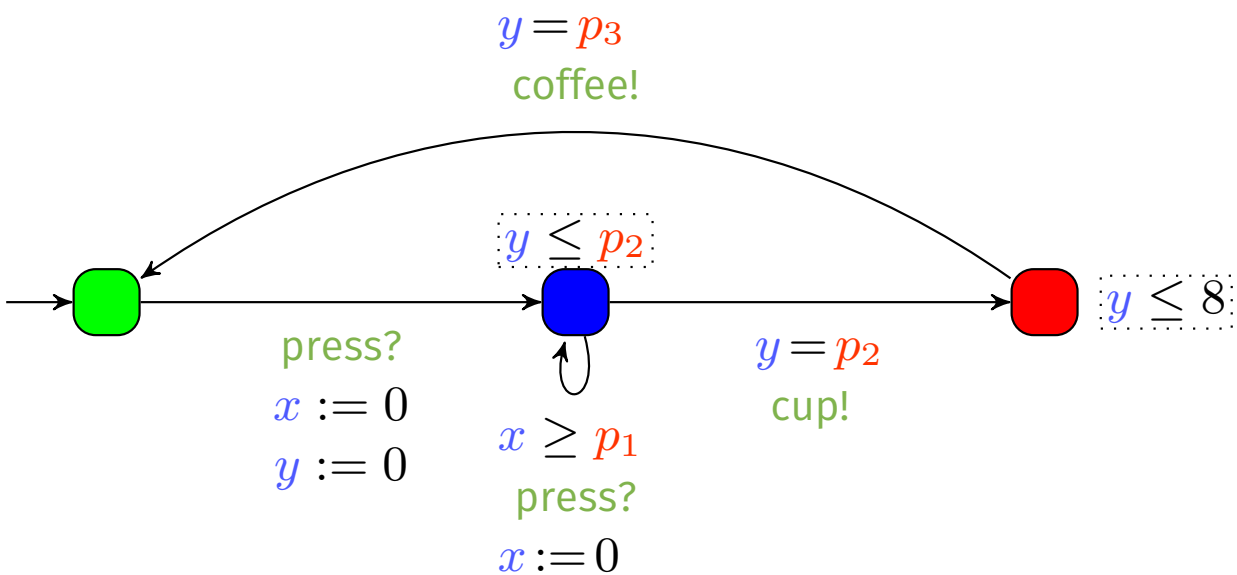
Parametric Timed Automaton (PTA)

- 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



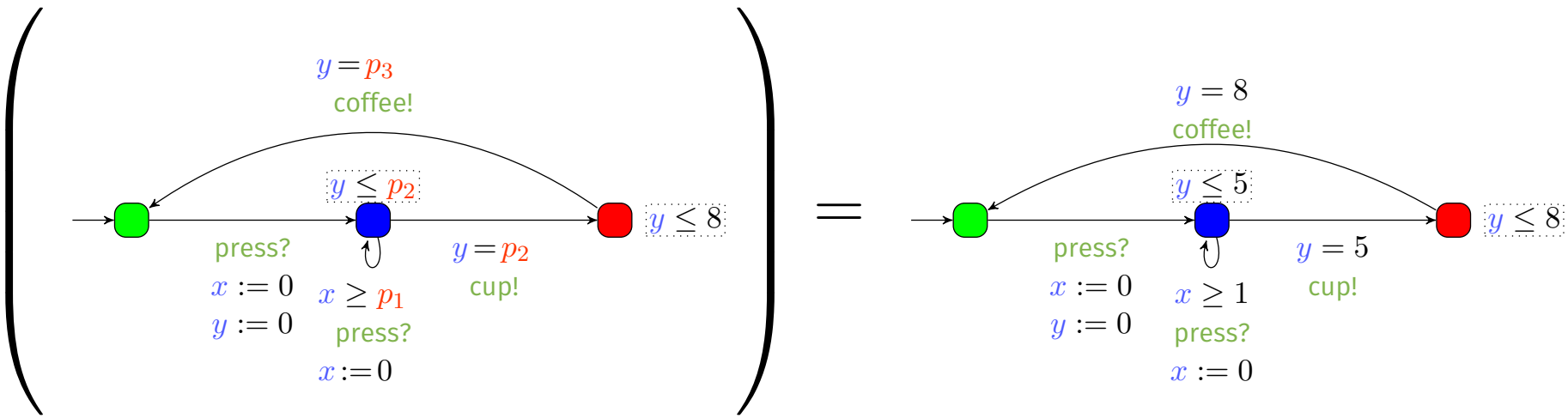
Notation: Valuation of a PTA

- Given a PTA $\mathcal{A}$ and a parameter valuation v, we denote by $v(\mathcal{A})$ the (non-parametric) timed automaton where each parameter p is valuated by $v(p)$

Notation: Valuation of a PTA

- Given a PTA $\mathcal{A}$ and a parameter valuation v, we denote by $v(\mathcal{A})$ the (non-parametric) timed automaton where each parameter p is valuated by $v(p)$

v



with v : $\begin{cases} p_1 & \rightarrow & 1 \\ p_2 & \rightarrow & 5 \\ p_3 & \rightarrow & 8 \end{cases}$

Symbolic semantics of parametric timed automata

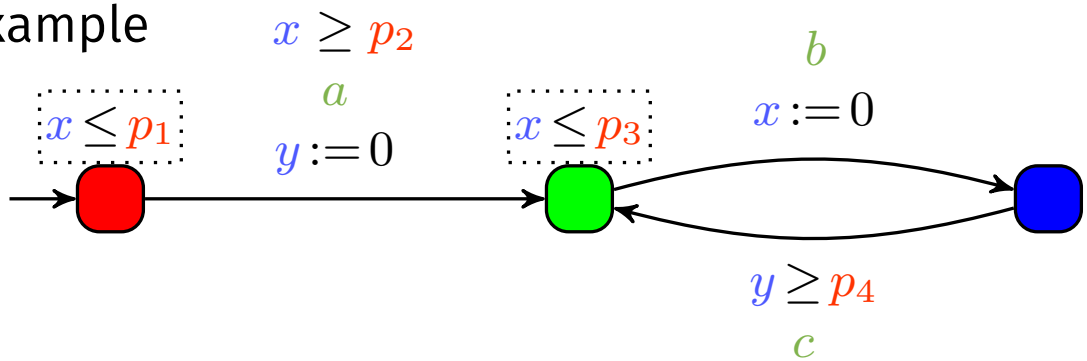
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]

Symbolic semantics of parametric timed automata

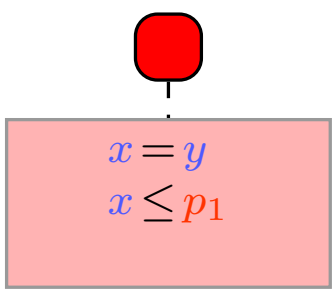
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]
- Symbolic run: alternating sequence of symbolic states and actions

Symbolic semantics of parametric timed automata

- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]
- Symbolic run: alternating sequence of symbolic states and actions
- Example

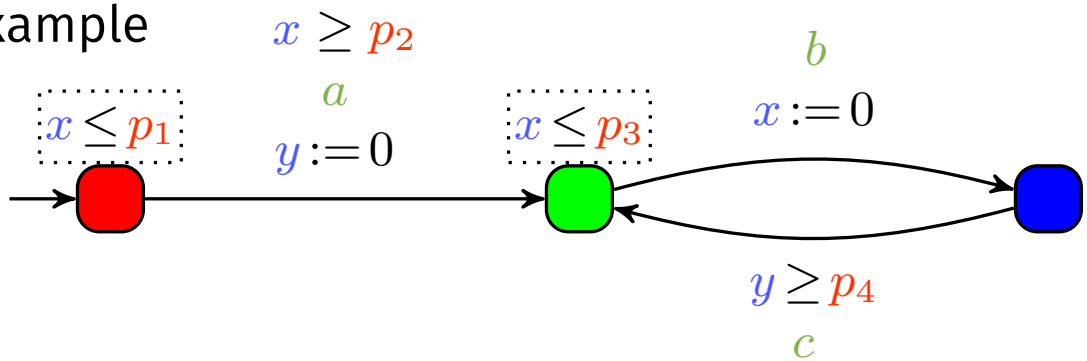


  - Possible symbolic run for this PTA

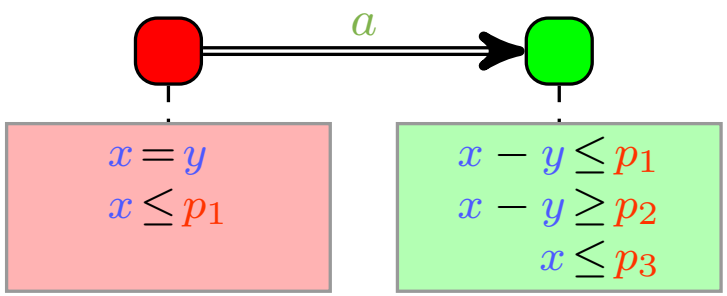


Symbolic semantics of parametric timed automata

- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]
- Symbolic run: alternating sequence of symbolic states and actions
- Example

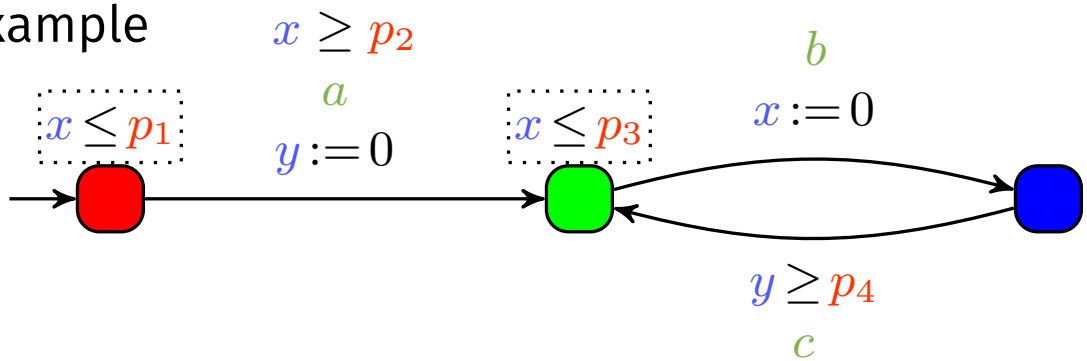


  - Possible symbolic run for this PTA

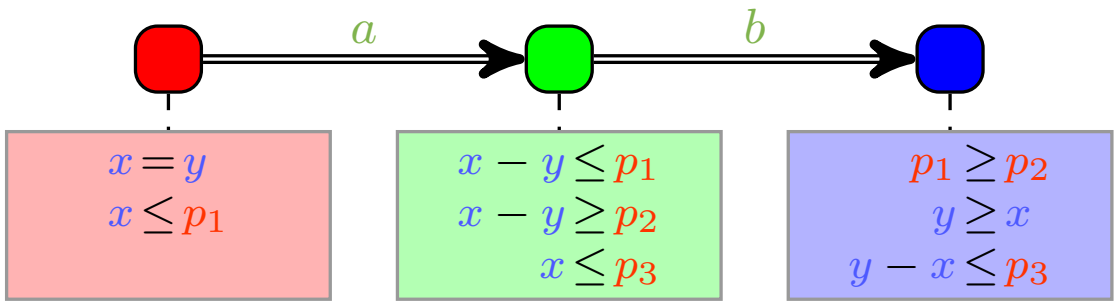


Symbolic semantics of parametric timed automata

- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]
- Symbolic run: alternating sequence of symbolic states and actions
- Example

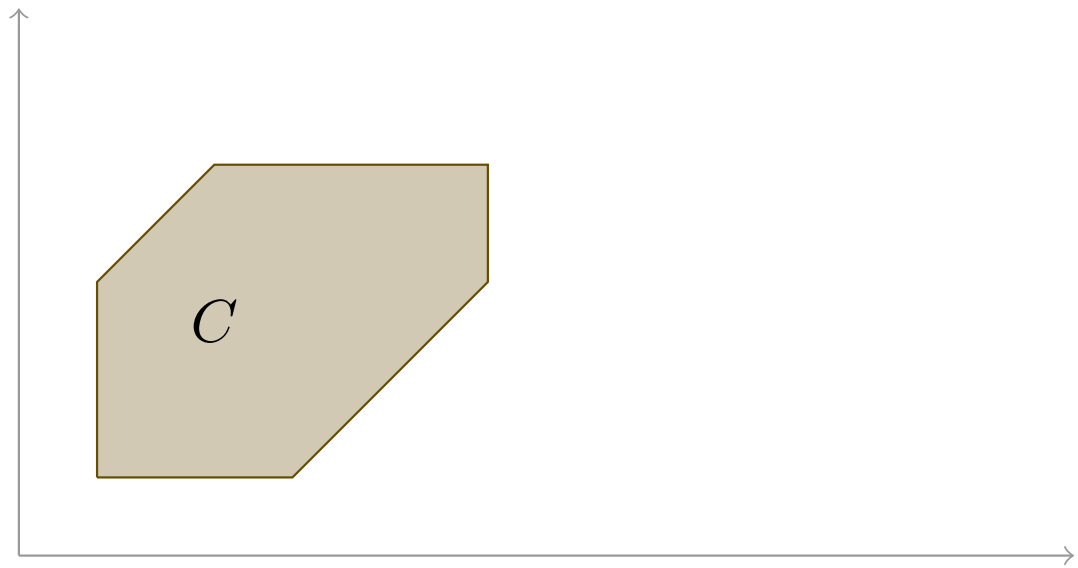


  - Possible symbolic run for this PTA



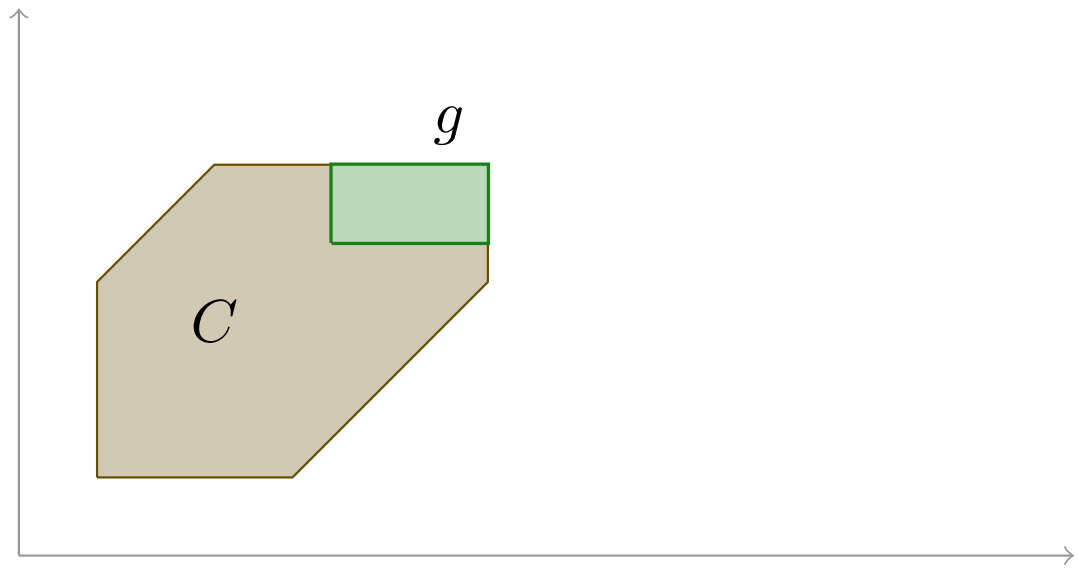
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$



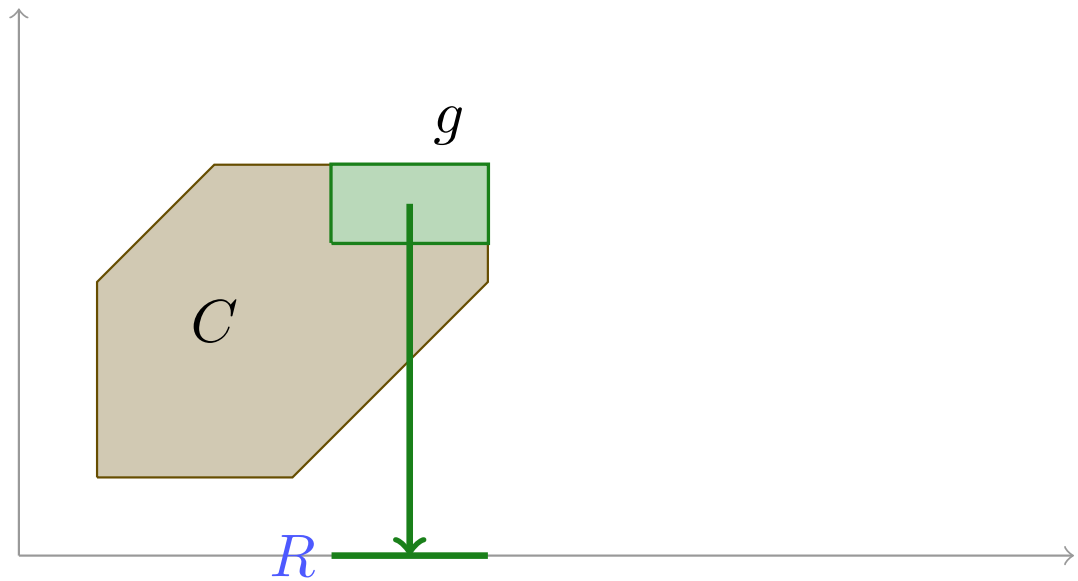
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$



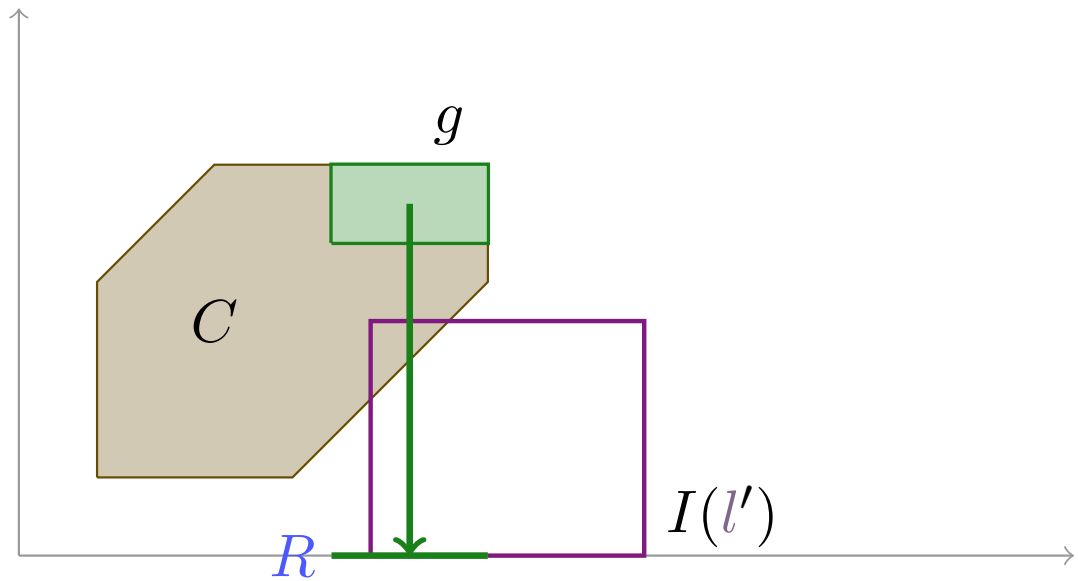
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$



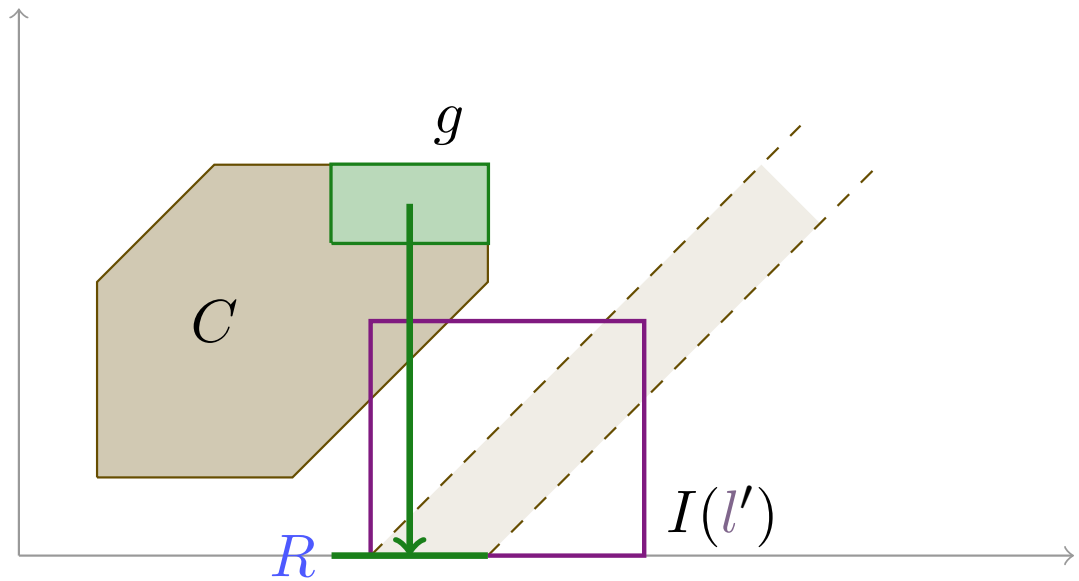
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$



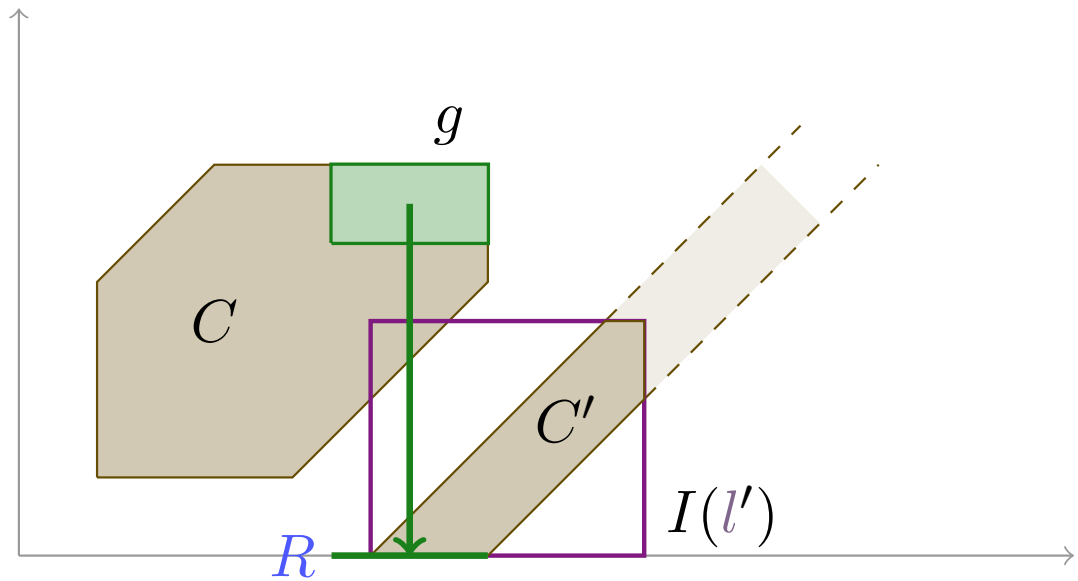
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$

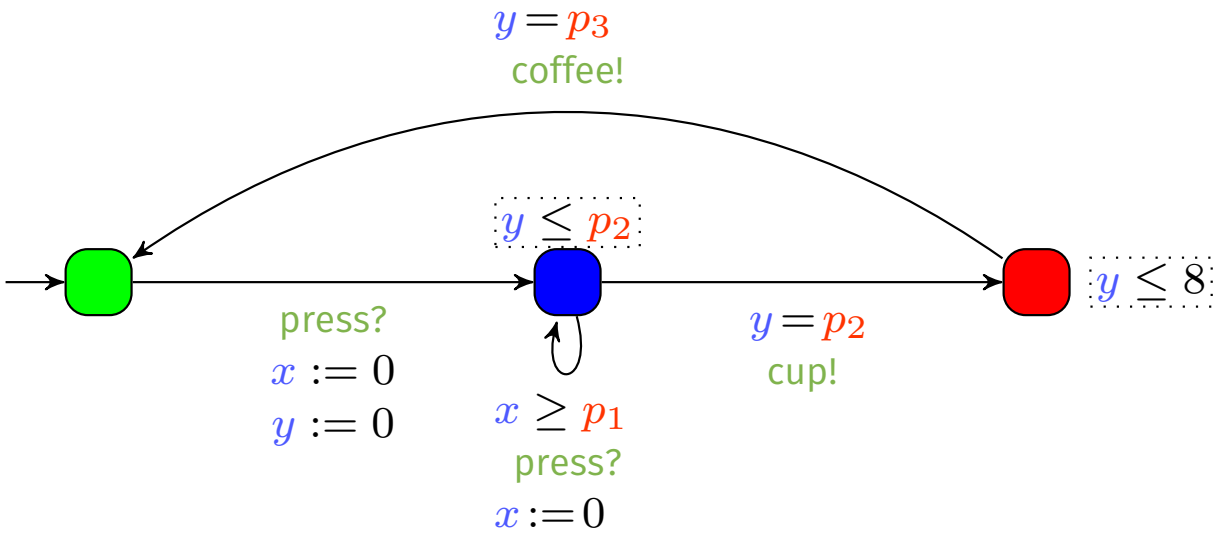


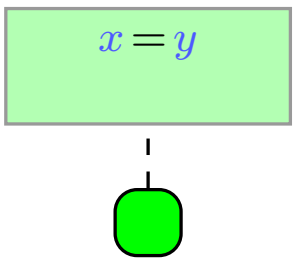
Symbolic semantics of PTA: Illustration

$$C' = [(C \cap g)]_R \cap I(l'))^{\nearrow} \cap I(l'))$$

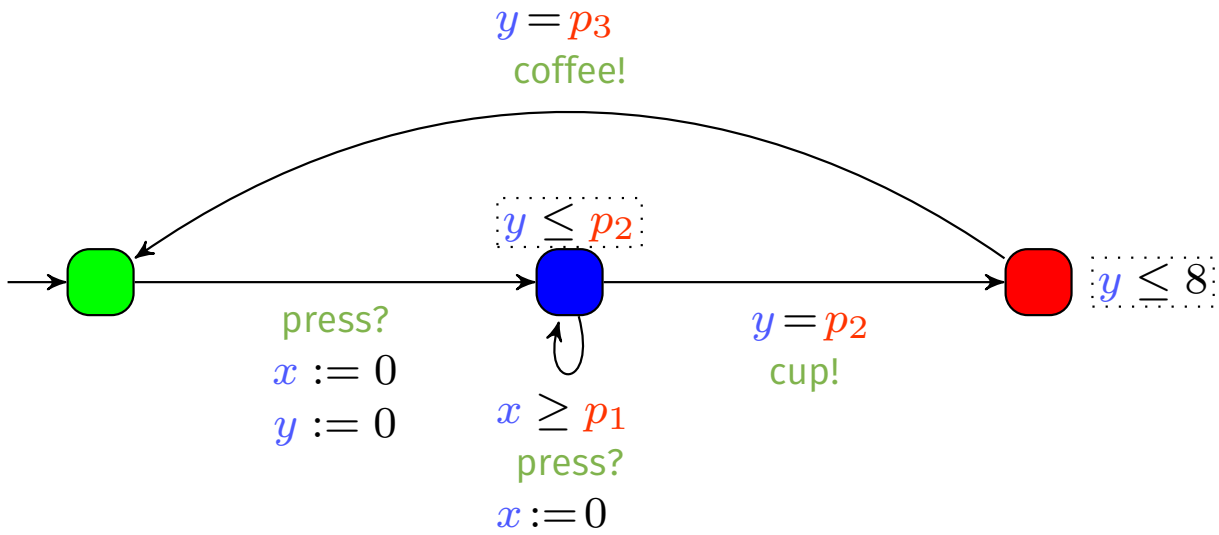


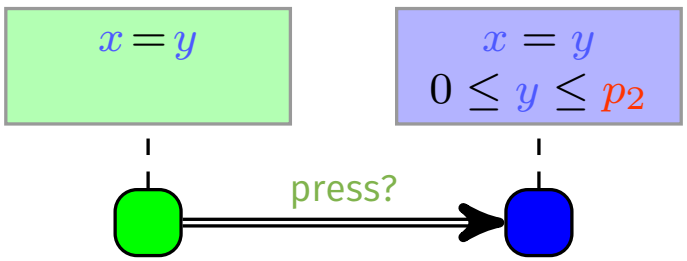
Symbolic exploration: Coffee machine



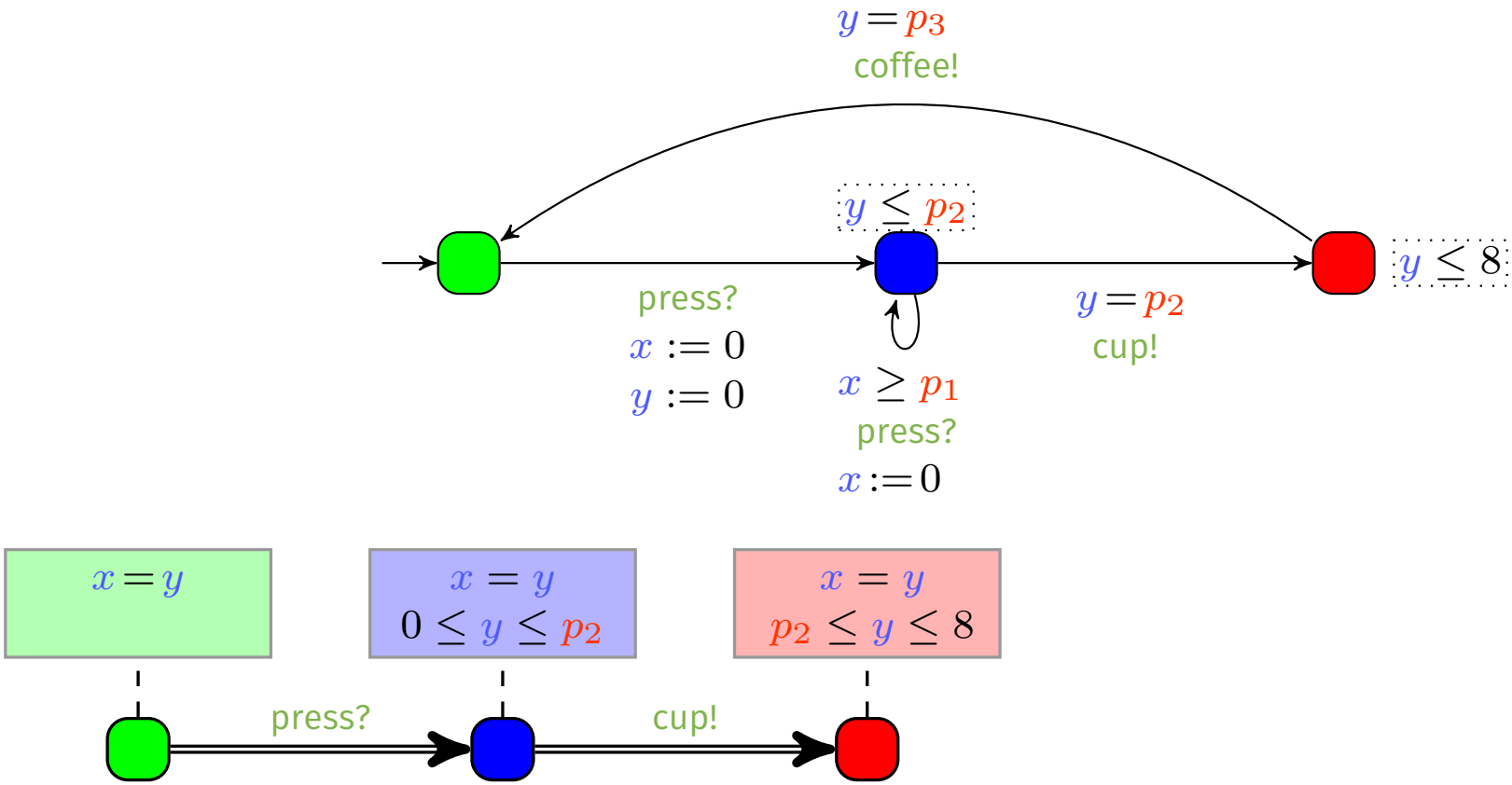


Symbolic exploration: Coffee machine

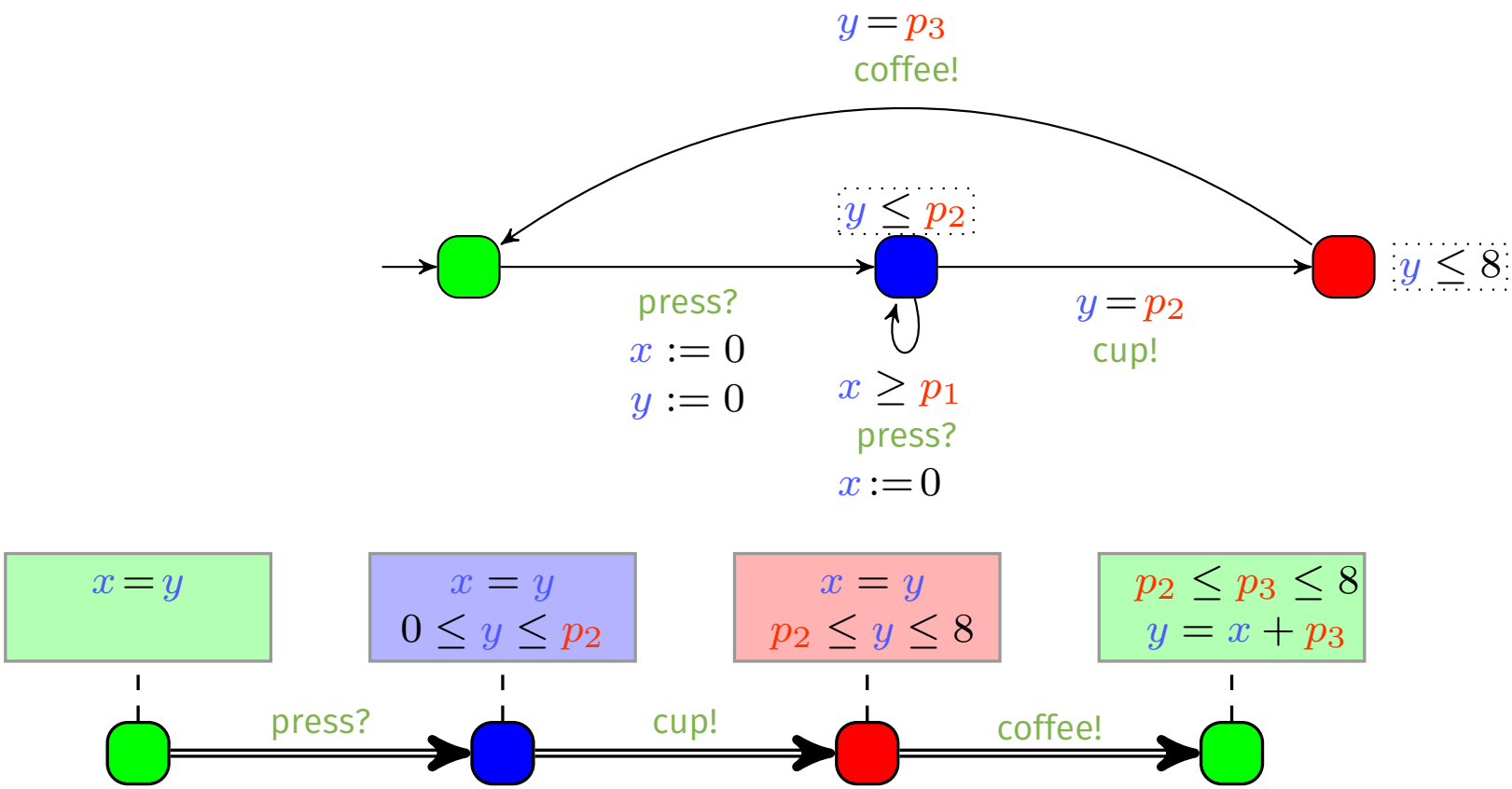




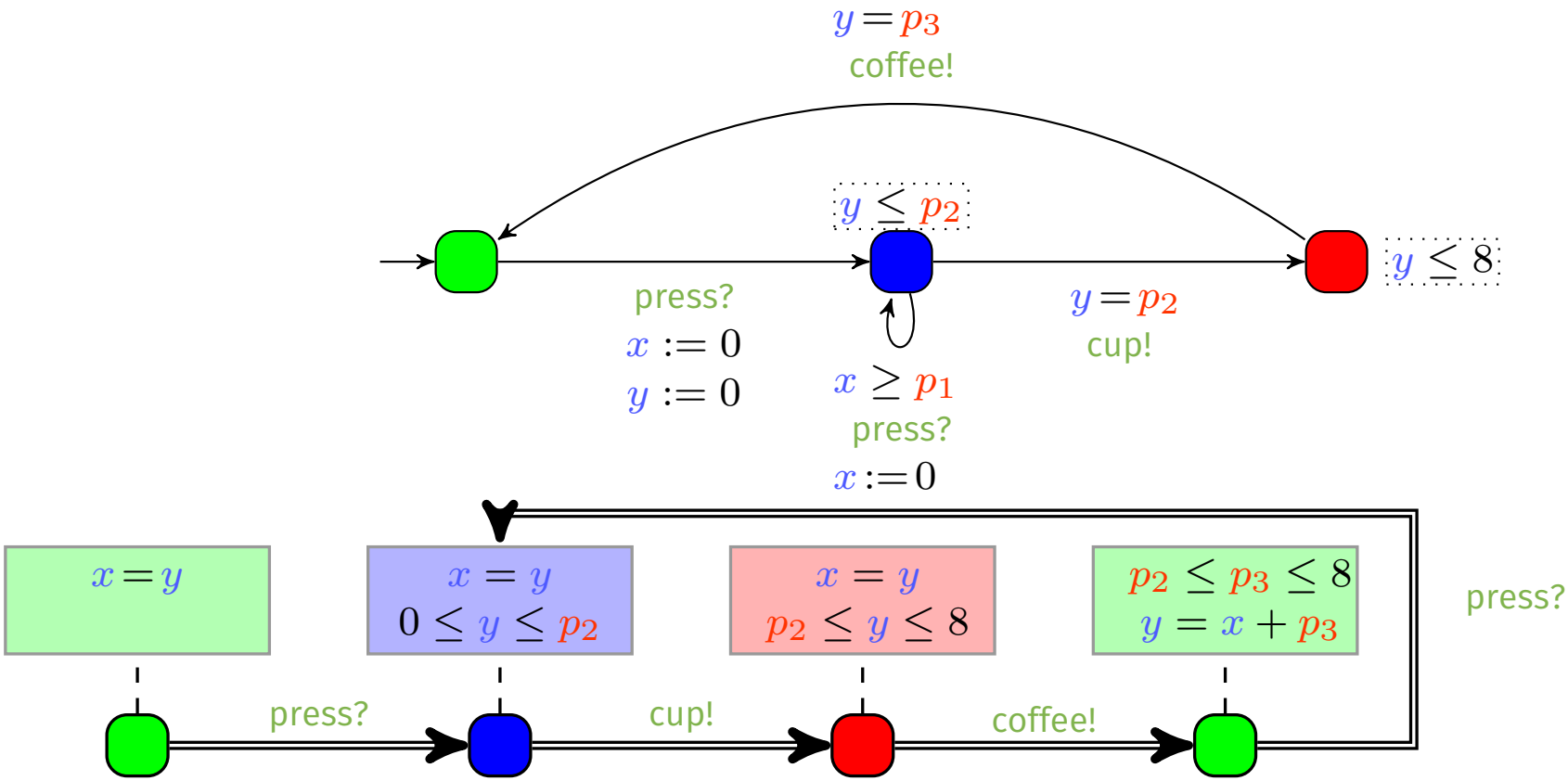
Symbolic exploration: Coffee machine



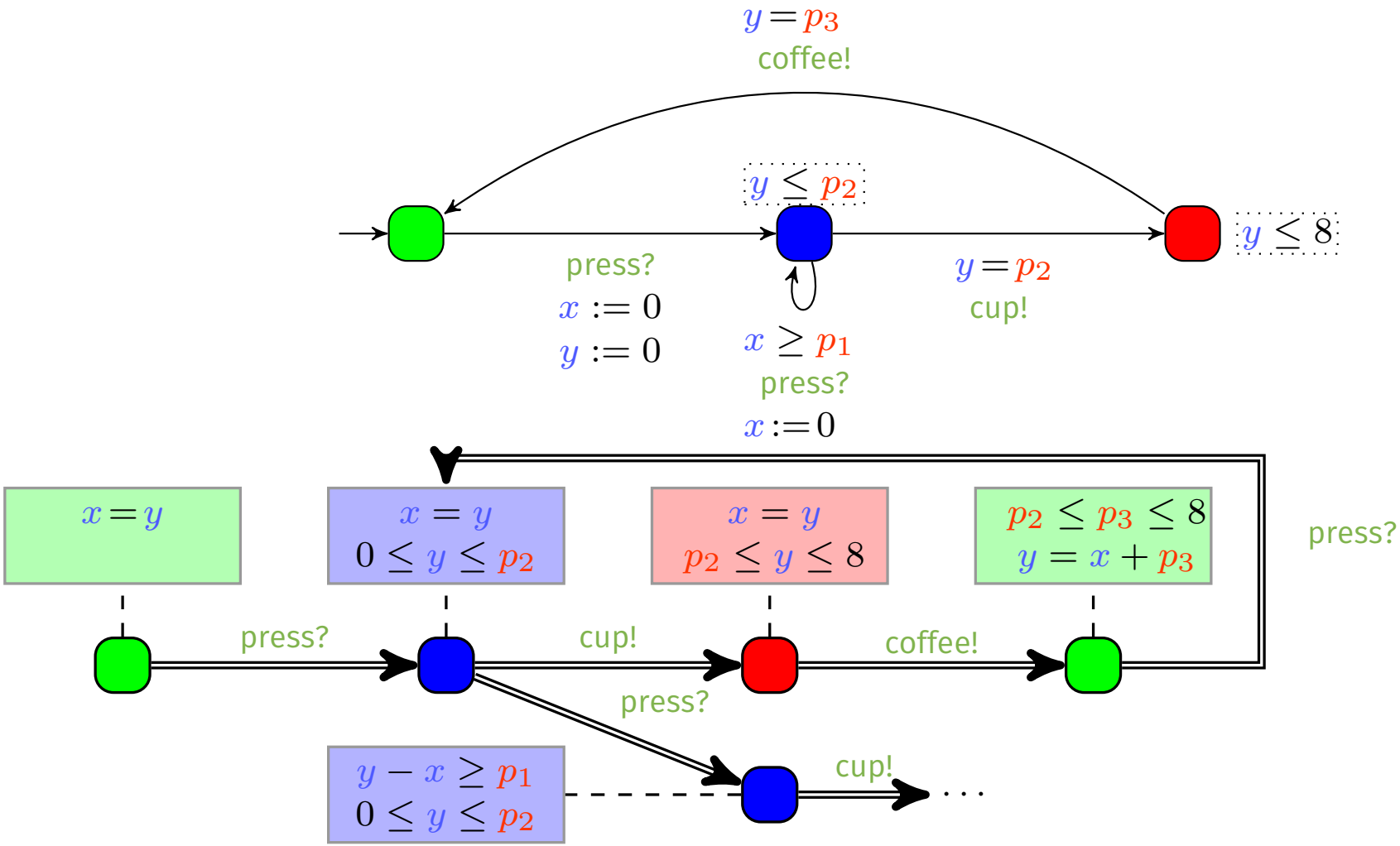
Symbolic exploration: Coffee machine



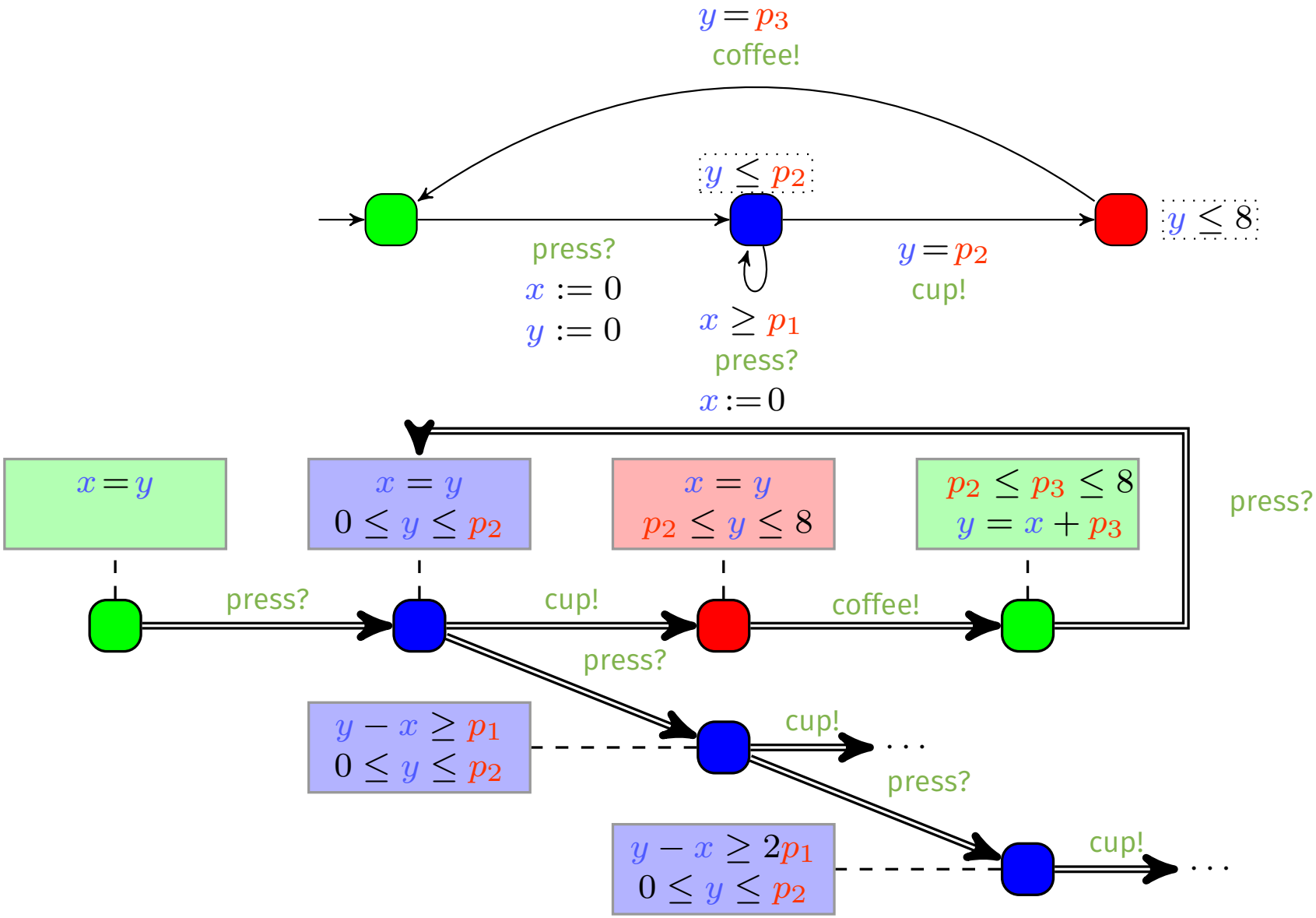
Symbolic exploration: Coffee machine



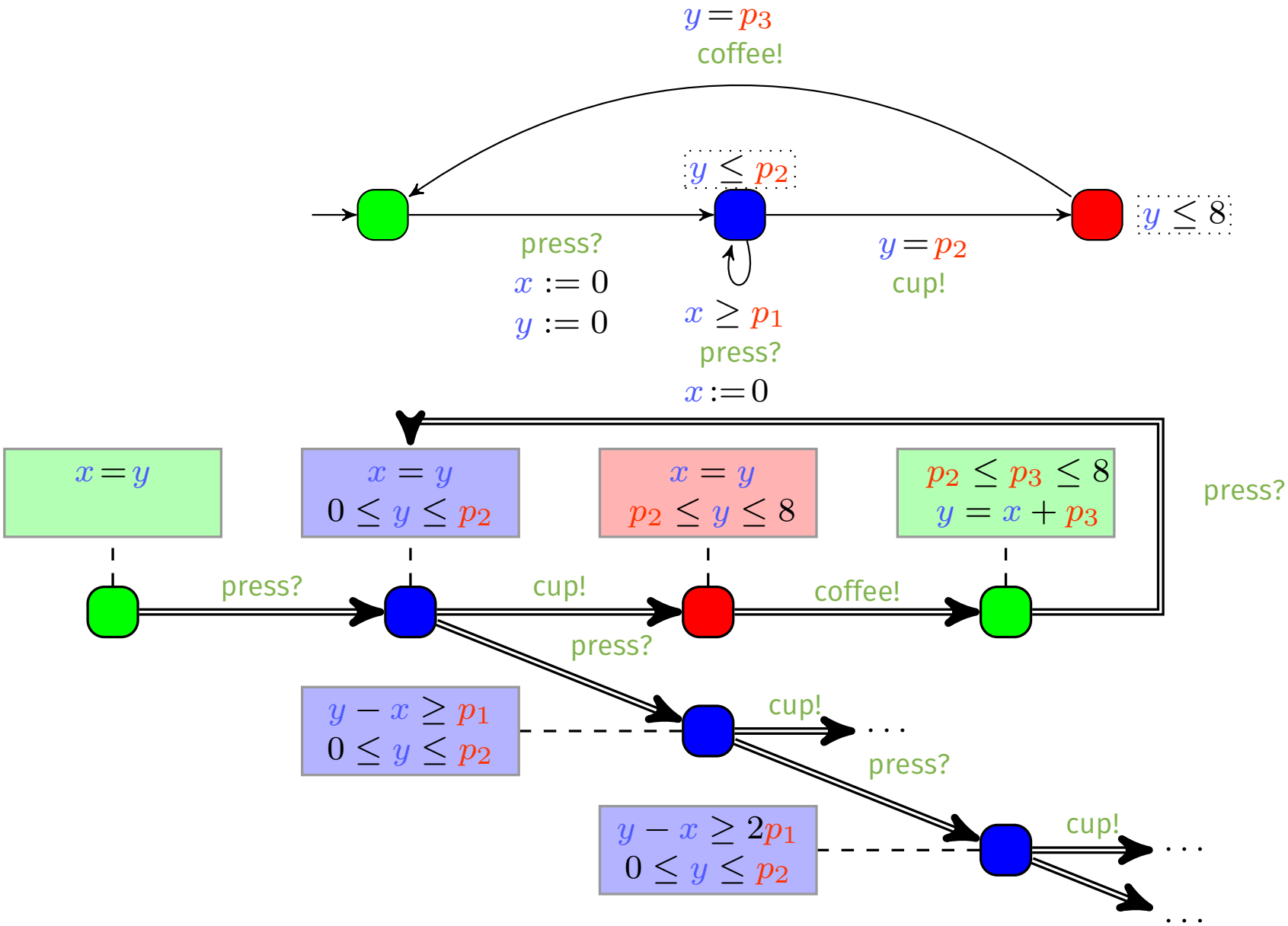
Symbolic exploration: Coffee machine



Symbolic exploration: Coffee machine



Symbolic exploration: Coffee machine



Why studying decidability?

If a decision problem is undecidable, it is hopeless to look for algorithms yielding exact solutions (because that is impossible)

Why studying decidability?

If a decision problem is undecidable, it is hopeless to look for algorithms yielding exact solutions (because that is impossible)

However, one can:

- design semi-algorithms: if the algorithm halts, then its result is correct
- design algorithms yielding over- or under-approximations

Decision and computation problems for PTA

- EF-Emptiness "Is the set of parameter valuations for which a given location l is reachable empty?"

  Example: "Does there exist at least one parameter valuation for which I can get a coffee with 2 sugars?"

- EF-Universality "Do all parameter valuations allow to reach a given location l?"

  Example: "Are all parameter valuations such that I may eventually get a coffee?"

- AF-Emptiness "Is the set of parameter valuations for which a given location l is always eventually reachable empty?"

  Example: "Does there exist at least one parameter valuation for which I can always eventually get a coffee?"

Decision and computation problems for PTA

- EF-Emptiness "Is the set of parameter valuations for which a given location l is reachable empty?"

  Example: "Does there exist at least one parameter valuation for which I can get a coffee with 2 sugars?" $\sqrt{}$, e.g., $p_1 = 1, p_2 = 5, p_3 = 8$

- EF-Universality "Do all parameter valuations allow to reach a given location l?"

  Example: "Are all parameter valuations such that I may eventually get a coffee?"

- AF-Emptiness "Is the set of parameter valuations for which a given location l is always eventually reachable empty?"

  Example: "Does there exist at least one parameter valuation for which I can always eventually get a coffee?"

Decision and computation problems for PTA

- EF-Emptiness "Is the set of parameter valuations for which a given location l is reachable empty?"

  Example: "Does there exist at least one parameter valuation for which I can get a coffee with 2 sugars?" $\sqrt{}$, e.g., $p_1 = 1, p_2 = 5, p_3 = 8$

- EF-Universality "Do all parameter valuations allow to reach a given location l?"

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

- AF-Emptiness "Is the set of parameter valuations for which a given location l is always eventually reachable empty?"

  Example: "Does there exist at least one parameter valuation for which I can always eventually get a coffee?"

Decision and computation problems for PTA

- EF-Emptiness "Is the set of parameter valuations for which a given location l is reachable empty?"

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

- EF-Universality "Do all parameter valuations allow to reach a given location l?"

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

- AF-Emptiness "Is the set of parameter valuations for which a given location l is always eventually reachable empty?"

  Example: "Does there exist at least one parameter valuation for which I can always eventually get a coffee?" $\checkmark$, e.g., $p_1 = 1, p_2 = 5, p_3 = 8$

Undecidability

- The symbolic state space is infinite in general
- No finite abstraction exists (unlike timed automata)

Undecidability

- The symbolic state space is infinite in general
- No finite abstraction exists (unlike timed automata)

Bad news

All interesting problems are undecidable for (general) parametric timed automata.

[ÉA, STTT 2017]

Undecidability in a nutshell

- EF-emptiness problem

  “Is the set of parameter valuations for which a given location l is reachable empty?” [Alur et al., 1993, Miller, 2000, Doyen, 2007, Beneš et al., 2015]

Undecidability in a nutshell

- EF-emptiness problem

  "Is the set of parameter valuations for which a given location l is reachable empty?" [Alur et al., 1993, Miller, 2000, Doyen, 2007, Beneš et al., 2015]

- EF-universality problem

  "Do all parameter valuations allow to reach a given location l?"

  [ÉA, Lime, Roux @ ICFEM'16]

Undecidability in a nutshell

- EF-emptiness problem
  "Is the set of parameter valuations for which a given location l is reachable empty?" [Alur et al., 1993, Miller, 2000, Doyen, 2007, Beneš et al., 2015]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location l?"
  [ÉA, Lime, Roux @ ICFEM'16]

- AF-emptiness and AF-universality problem
  "Is the set of parameter valuations for which all runs eventually reach a given location l empty/universal?" [Jovanović et al., 2015, André et al., 2016]

Undecidability in a nutshell

- EF-emptiness problem
  "Is the set of parameter valuations for which a given location l is reachable empty?" [Alur et al., 1993, Miller, 2000, Doyen, 2007, Beneš et al., 2015]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location l?"
  [ÉA, Lime, Roux @ ICFEM'16]

- AF-emptiness and AF-universality problem
  "Is the set of parameter valuations for which all runs eventually reach a given location l empty/universal?" [Jovanović et al., 2015, André et al., 2016]

- Preservation of the untimed language
  "Given a parameter valuation, does there exist another valuations with the same untimed language?" [André and Markey, 2015]

Decidability in a nutshell

Reducing the number of clocks yields decidability of the EF-emptiness problem:

Decidability in a nutshell

Reducing the number of clocks yields decidability of the EF-emptiness problem:

- ✓ 1 parametric clock and arbitrarily many non-parametric clocks and integer-valued parameters [Beneš et al., 2015]
- ✓ 1 parametric clock and arbitrarily many rational-valued parameters [Miller, 2000]
- ✓ 2 parametric clocks and 1 integer-valued parameter [Bundala and Ouaknine, 2014]

Decidability in a nutshell

Reducing the number of clocks yields decidability of the EF-emptiness problem:

- ✓ 1 parametric clock and arbitrarily many non-parametric clocks and integer-valued parameters [Beneš et al., 2015]
- ✓ 1 parametric clock and arbitrarily many rational-valued parameters [Miller, 2000]
- ✓ 2 parametric clocks and 1 integer-valued parameter [Bundala and Ouaknine, 2014]

Restraining the syntax brings decidability of some problems:

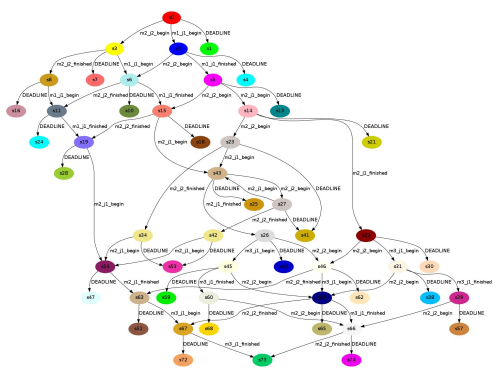
- L/U-PTAs [Hune et al., 2002, Bozzelli and La Torre, 2009, André and Markey, 2015, André and Lime, 2017, André et al., 2018b]
- PTAs with bounded integer-valued parameters [Jovanović et al., 2015]
- reset-PTAs [André et al., 2016, André et al., 2018c]

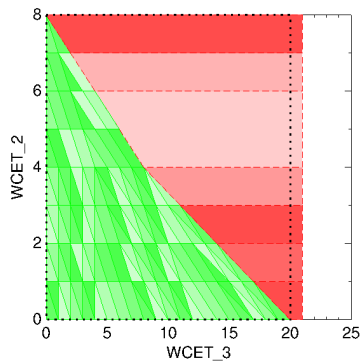
Outline

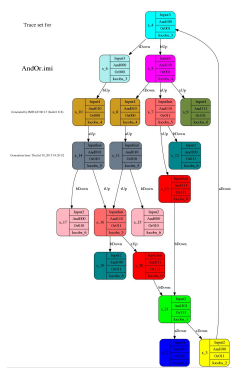
IMITATOR

- 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







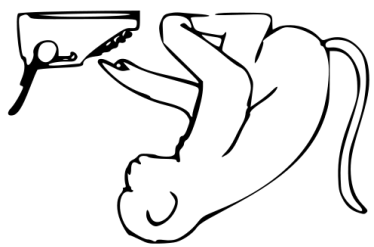
IMITATOR

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

A library of benchmarks

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

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



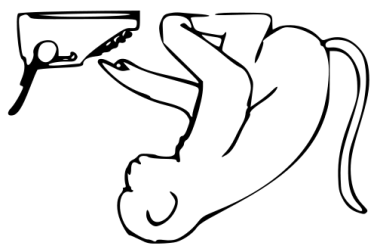
IMITATOR

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

A library of benchmarks

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

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



Try it!

www.imitator.fr

Some success stories

- 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., 2017]
- Offline monitoring [ÉA, Hasuo, Waga @ ICECCS'18]
- Formal timing analysis of music scores [Fanchon and Jacquemard, 2013]
- Solution to a challenge related to a distributed video processing system by Thales

Outline

Modeling real-time systems with timed automata

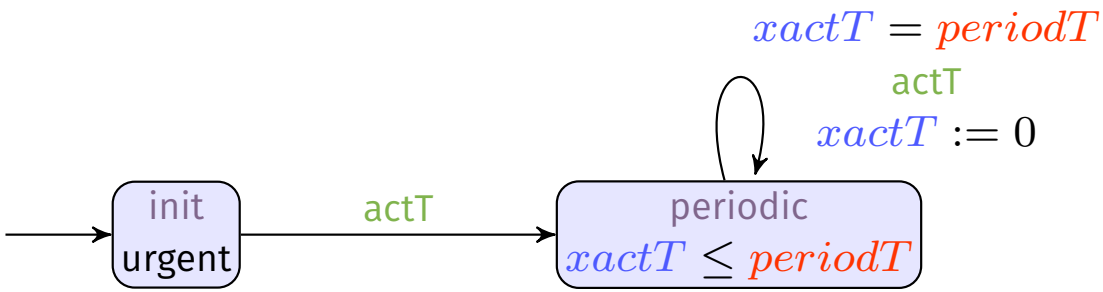
- Using timed automata
  - [Abdeddaïm and Maler, 2001]
- Using stopwatch automata
  - [Adbeddaïm and Maler, 2002]
- Using parametric timed automata
  - [Cimatti et al., 2008]
- Using parametric stopwatch automata
  - [Fribourg et al., 2012, Sun et al., 2013, Lipari et al., 2014]
- Using task automata
  - [Norström et al., 1999, Fersman et al., 2007, André, 2017]

Modeling a periodic task T (exercise)

Periodic task T with period $periodT$:

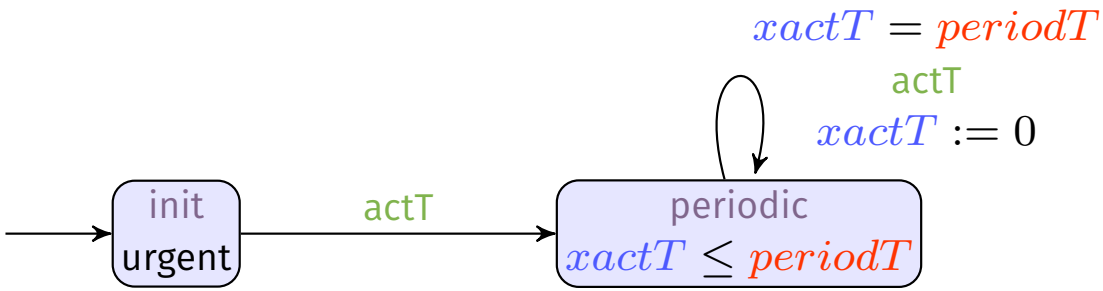
Modeling a periodic task T (exercise)

Periodic task T with period $periodT$:



Modeling a periodic task T (exercise)

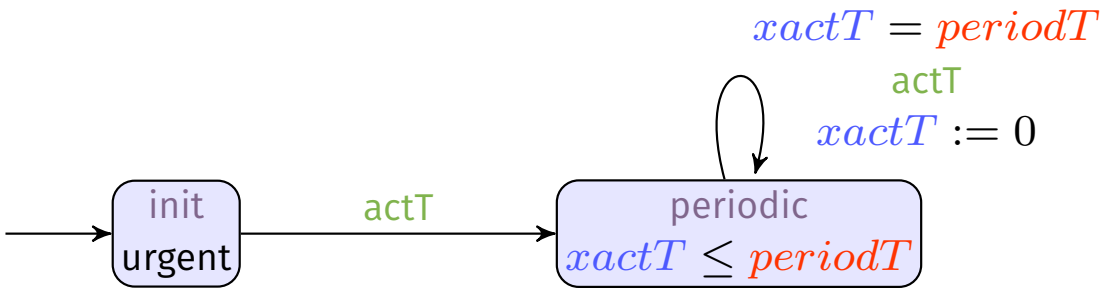
Periodic task T with period $periodT$:



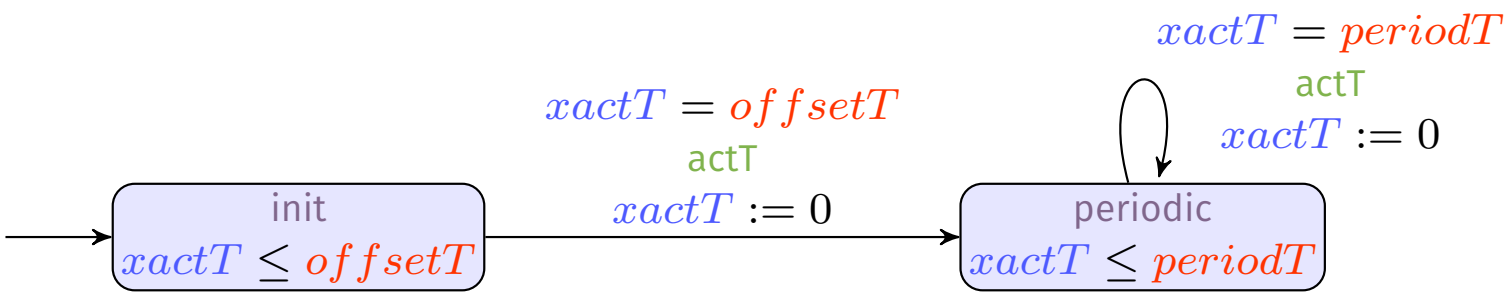
Periodic task T with period $periodT$ and $offsetT$:

Modeling a periodic task T (exercise)

Periodic task T with period $periodT$:



Periodic task T with period $periodT$ and $offsetT$:

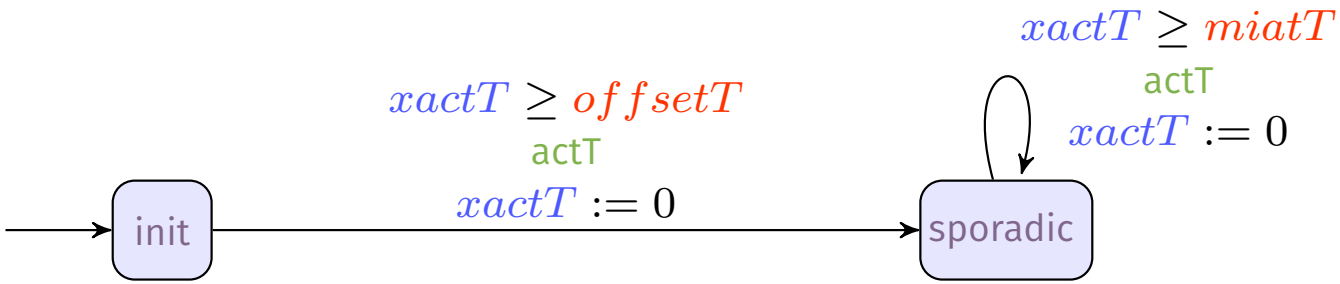


Modeling a sporadic task T (exercise)

Sporadic task T with minimum interarrival time $miatT$ and $offsetT$:

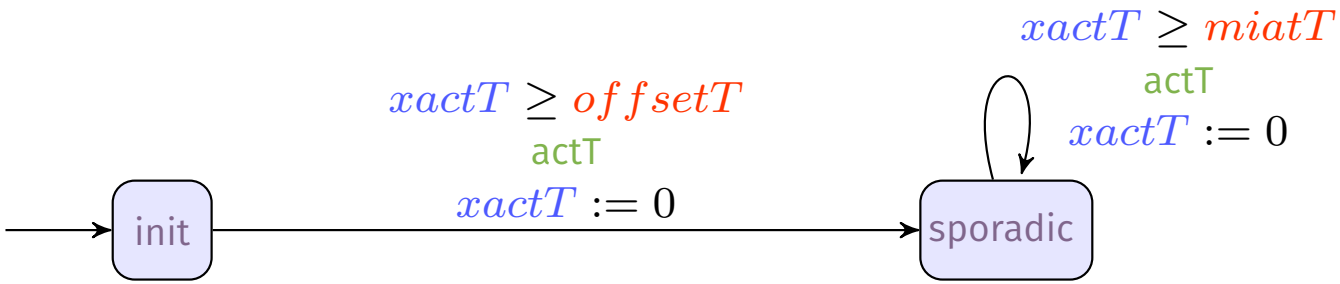
Modeling a sporadic task T (exercise)

Sporadic task T with minimum interarrival time $miatT$ and $offsetT$:



Modeling a sporadic task T (exercise)

Sporadic task T with minimum interarrival time $miatT$ and $offsetT$:

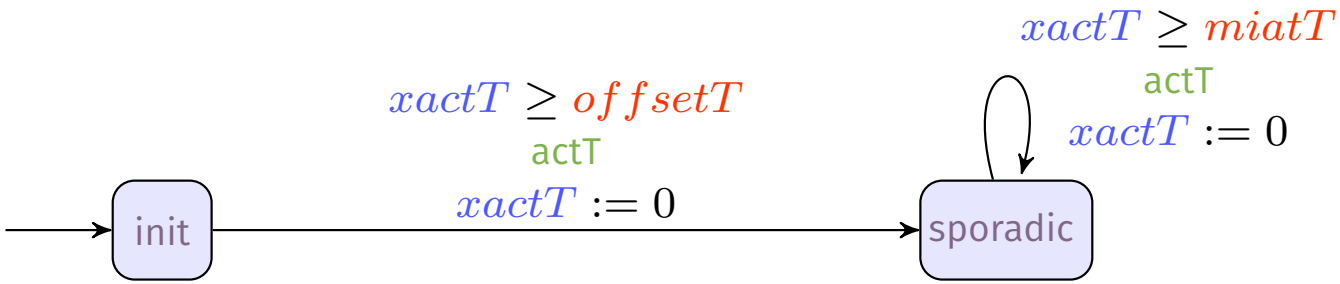


A more efficient modeling to avoid clock divergence in IMITATOR

- and hence optimize the computation

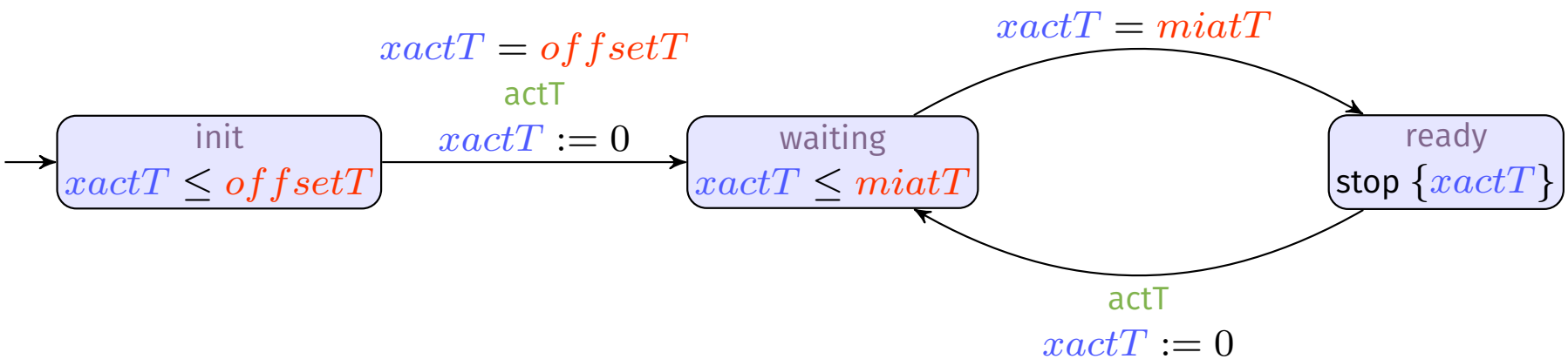
Modeling a sporadic task T (exercise)

Sporadic task T with minimum interarrival time $miatT$ and $offsetT$:



A more efficient modeling to avoid clock divergence in IMITATOR

- and hence optimize the computation



Trick: stop the computation of $xactT$ to avoid diverging

Modeling a task / pipeline

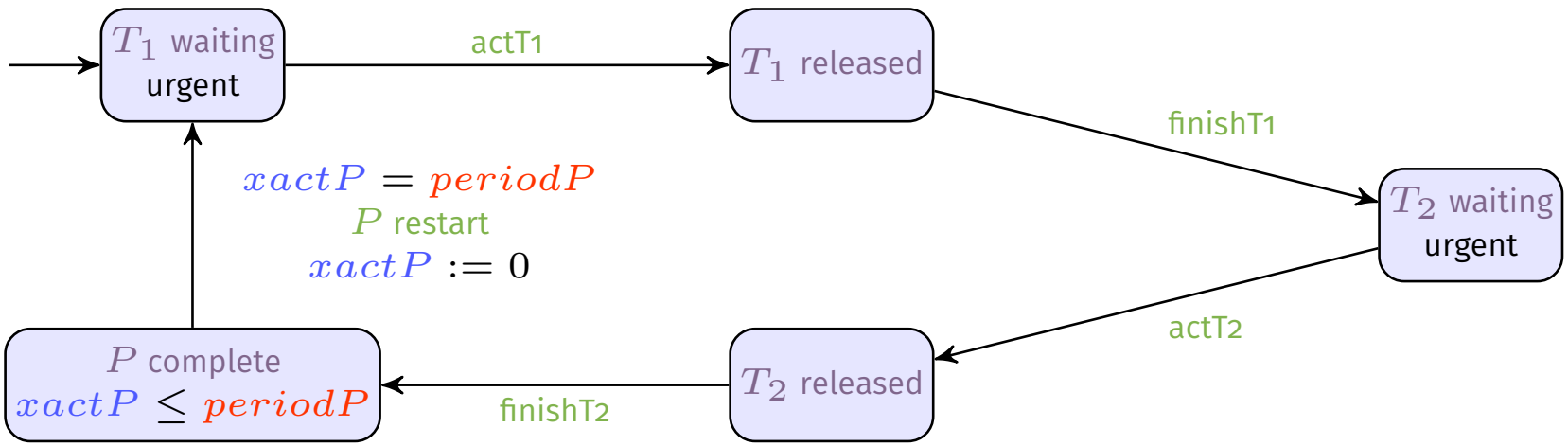
Pipeline P of two tasks T_1 and T_2

The pipeline has a period $periodP$

Modeling a task / pipeline

Pipeline P of two tasks T_1 and T_2

The pipeline has a period $periodP$

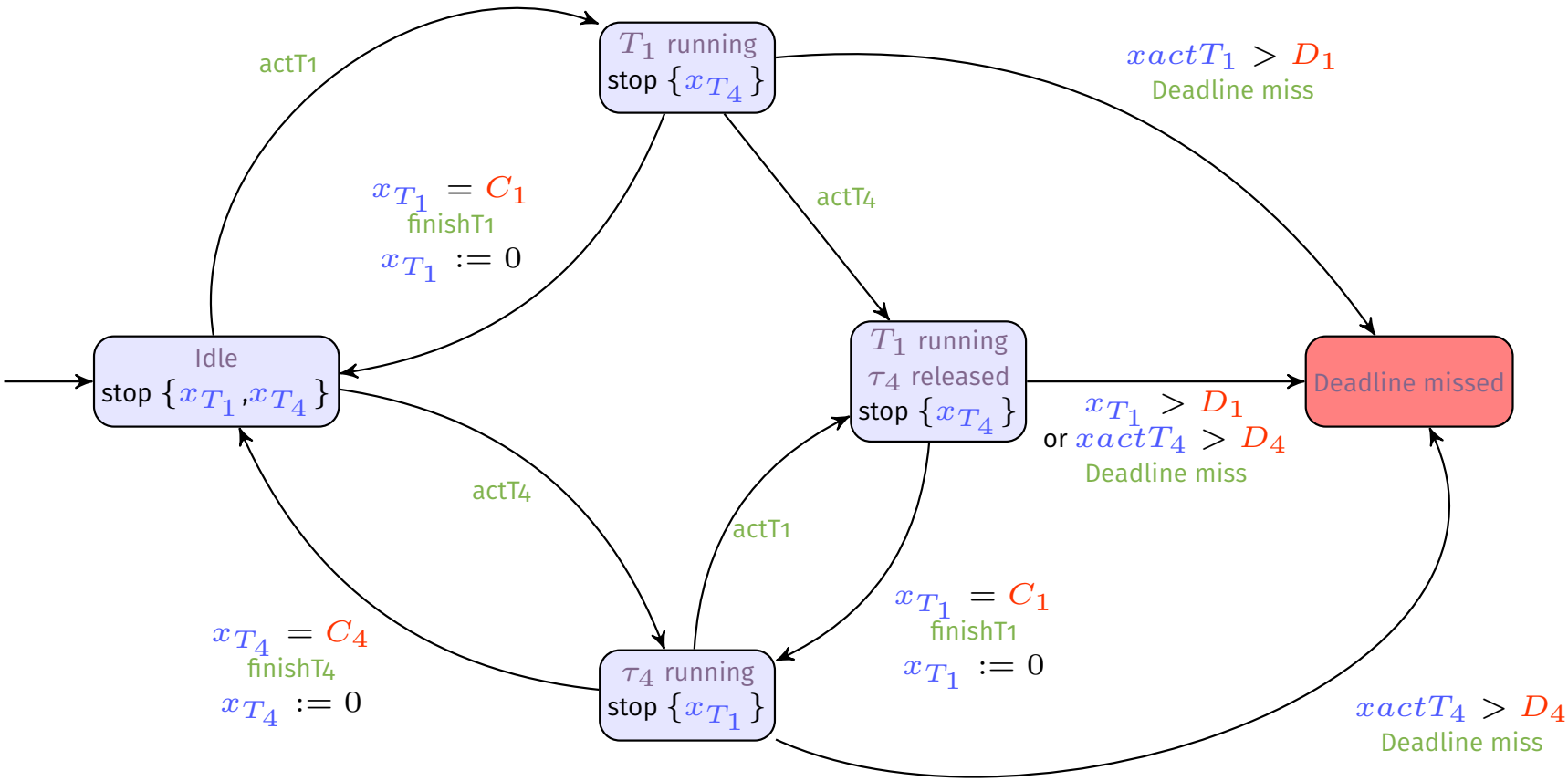


Modeling the preemptive fixed priority scheduler

A fixed-priority preemptive processor with two tasks T_1 and T_4

Timings for T_1: period $periodT1$, execution time period C_1, deadline period D_1

- and similarly for T_4



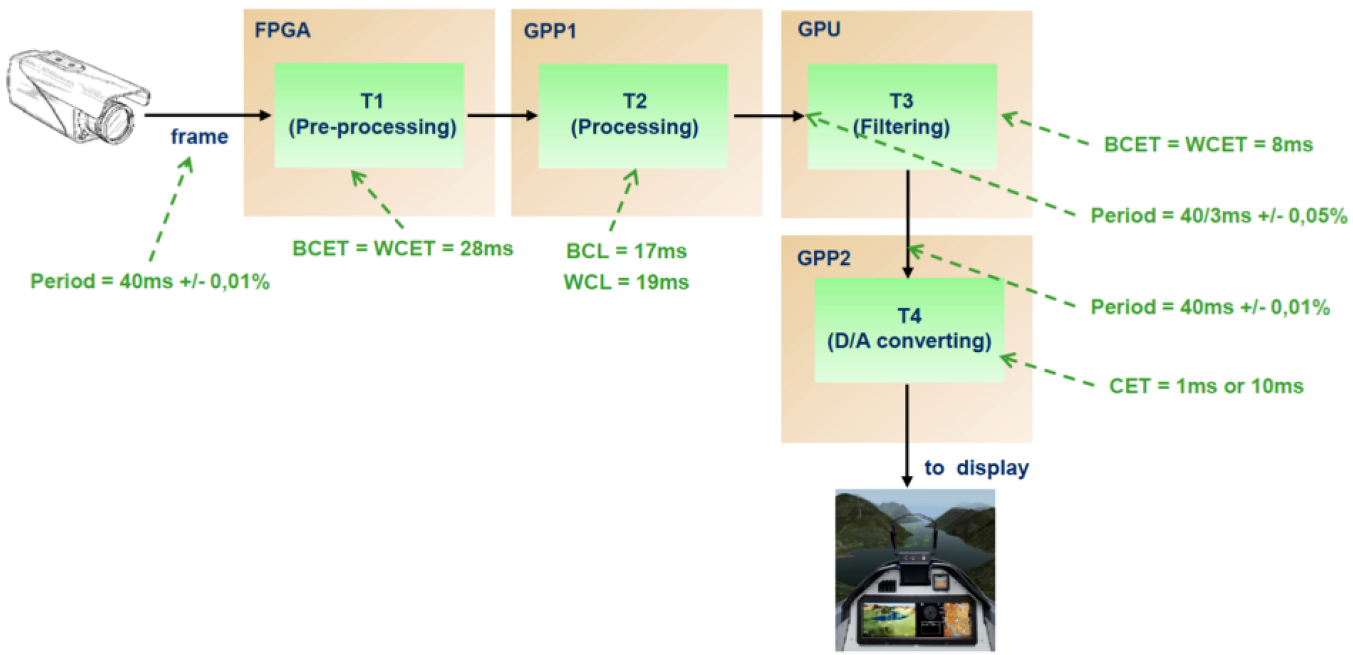
Outline

The FMTV 2015 Challenge (1/2)

Challenge by Thales proposed during the WATERS 2014 workshop
Solutions presented at WATERS 2015

System: an unmanned aerial video system with uncertain periods

- Period constant but with a small uncertainty (typically 0.01 %)
- Not a jitter!



The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of $n = 1$ and $n = 3$

The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of $n = 1$ and $n = 3$

- ☹ Not a typical parameter synthesis problem?
  - No parameters in the specification

The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of $n = 1$ and $n = 3$

☹ Not a typical parameter synthesis problem?

- No parameters in the specification

☺ A typical parameter synthesis problem

- The end-to-end time can be set as a parameter… to be synthesized
- The uncertain period is typically a parameter (with some constraint, e. g., $P1 \in [40 - 0.004, 40 + 0.004]$)

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)
5. Eliminate all parameters but the end-to-end time

Note: not eliminating parameters allows one to know for which values of the periods the best / worst case execution times are obtained.

Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)
5. Eliminate all parameters but the end-to-end time
6. Exhibit the minimum and the maximum

Note: not eliminating parameters allows one to know for which values of the periods the best / worst case execution times are obtained.

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$
- Parameters:
  - P1_uncertain
  - P3_uncertain
  - P4_uncertain

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$
- Parameters:
  - $\mathrm{P1\_uncertain}$
  - $\mathrm{P3\_uncertain}$
  - $\mathrm{P4\_uncertain}$
- The end-to-end latency (another parameter): $\mathrm{E2E}$

To build the PTA model

- Uncertainties in the system:
  - $P1 \in [40 - 0.004, 40 + 0.004]$
  - $P3 \in [\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}]$
  - $P4 \in [40 - 0.004, 40 + 0.004]$
- Parameters:
  - $\mathrm{P1\_uncertain}$
  - $\mathrm{P3\_uncertain}$
  - $\mathrm{P4\_uncertain}$
- The end-to-end latency (another parameter): $\mathrm{E2E}$
- Others:
  - the register between task 2 and task 3: discrete variable $\mathrm{reg}_{2,3}$
  - the buffer between task 3 and task 4: $n = 1$ or $n = 3$

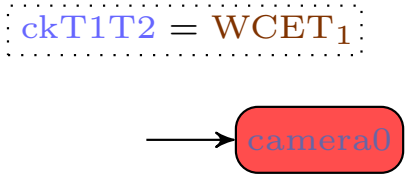
Simplification

- T1 and T2 are synchronised; T1, T3 and T4 are asynchronised
  - (exact modeling of the system behaviour is too heavy)

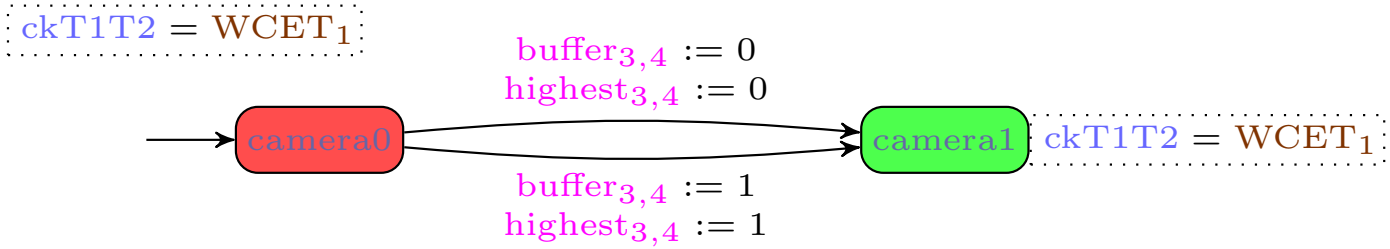
Simplification

- T1 and T2 are synchronised; T1, T3 and T4 are asynchronised
  - (exact modeling of the system behaviour is too heavy)
- We choose a single arbitrary frame, called the target one
- We assume the system is initially in an arbitrary status
  - This is our only uncertain assumption (in other words, can the periods deviate from each other so as to yield any arbitrary deviation?)

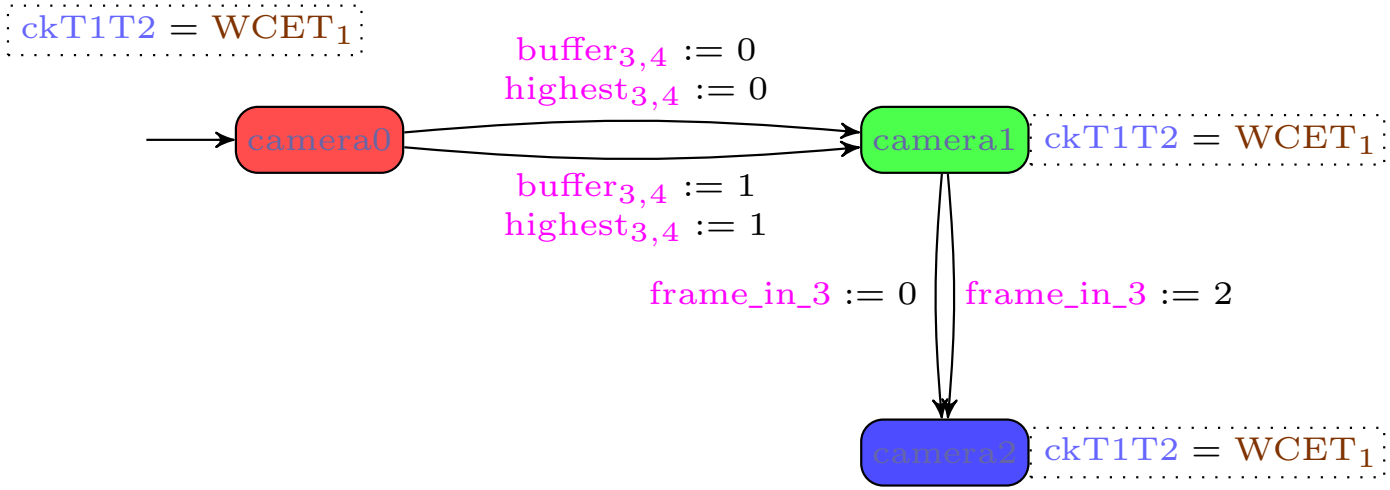
The initialization automaton



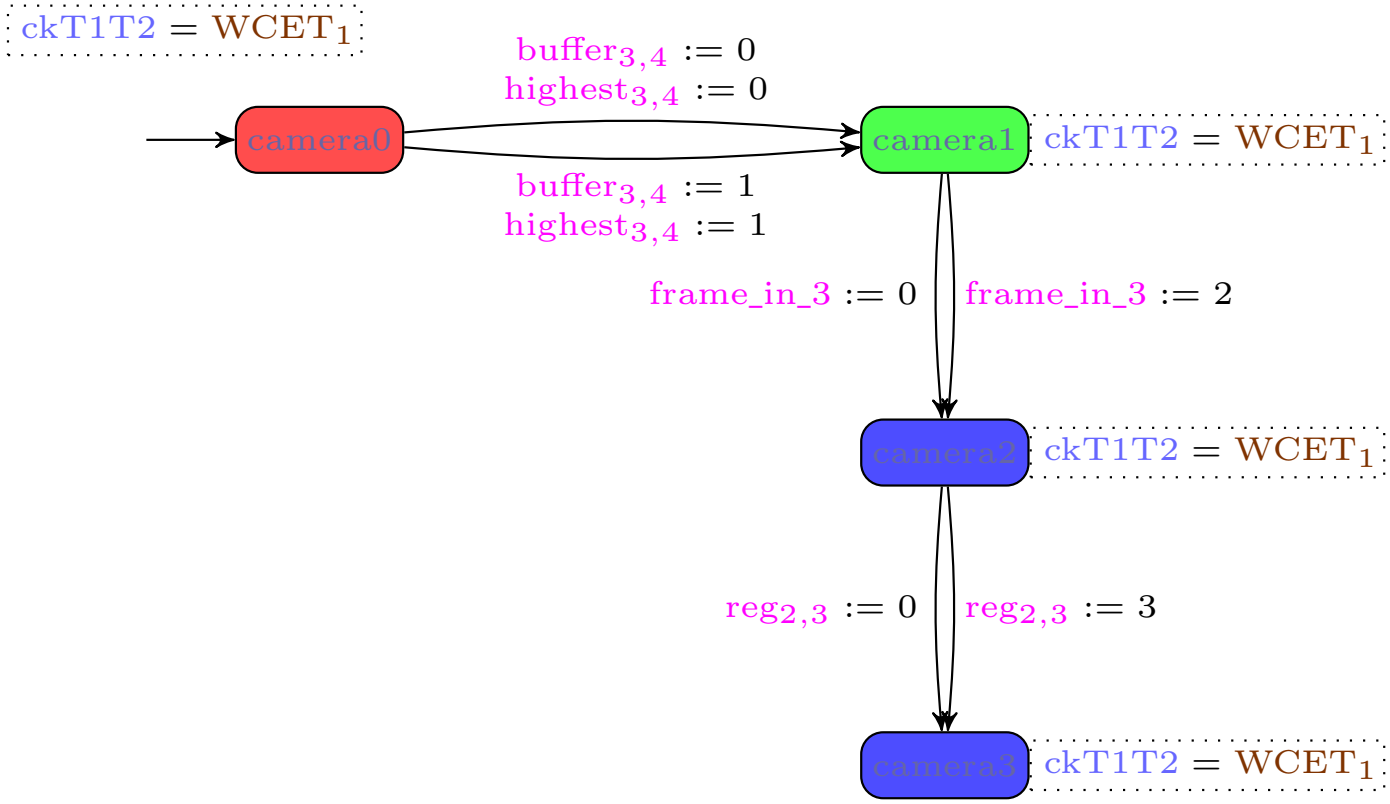
The initialization automaton



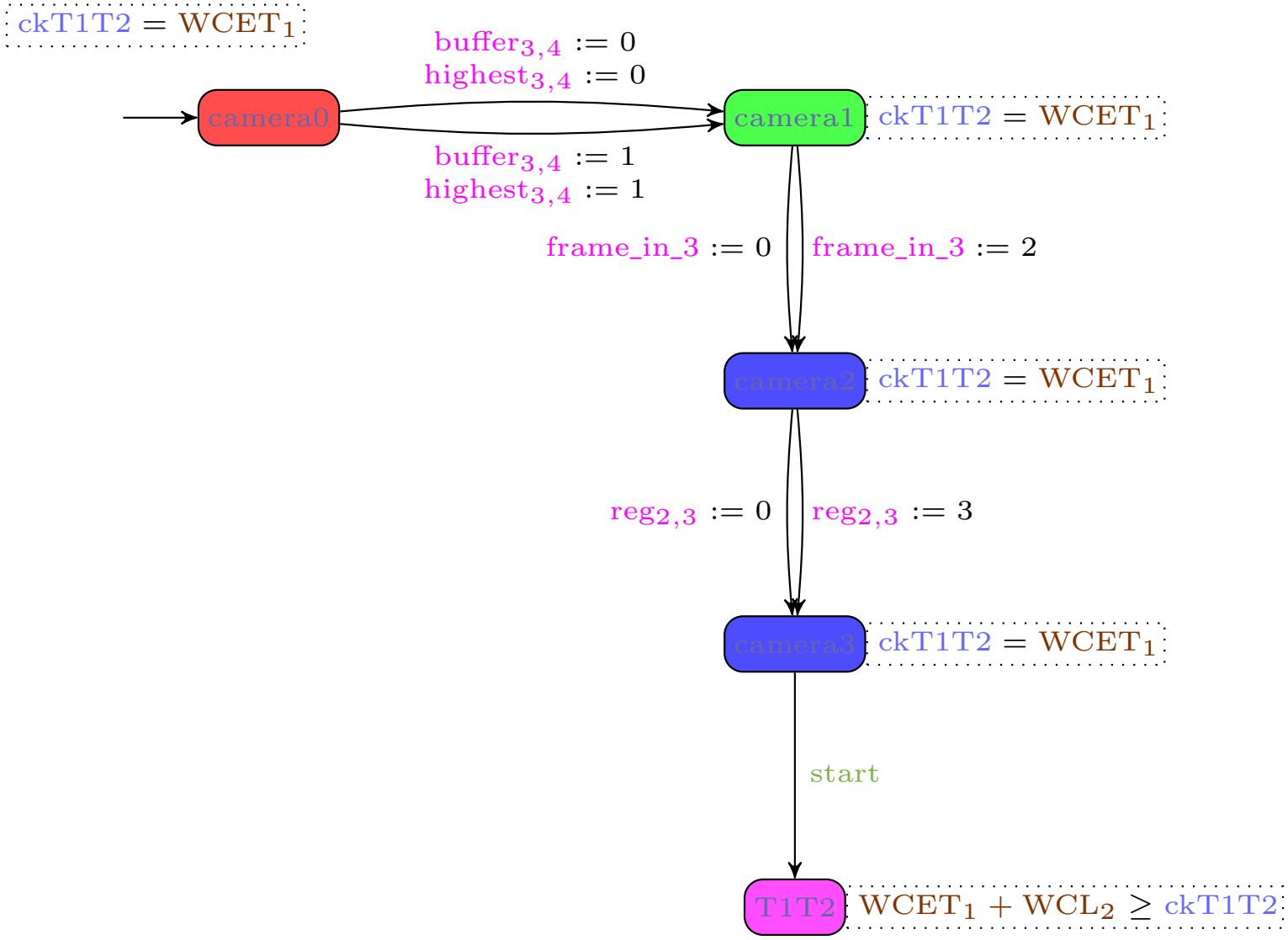
The initialization automaton



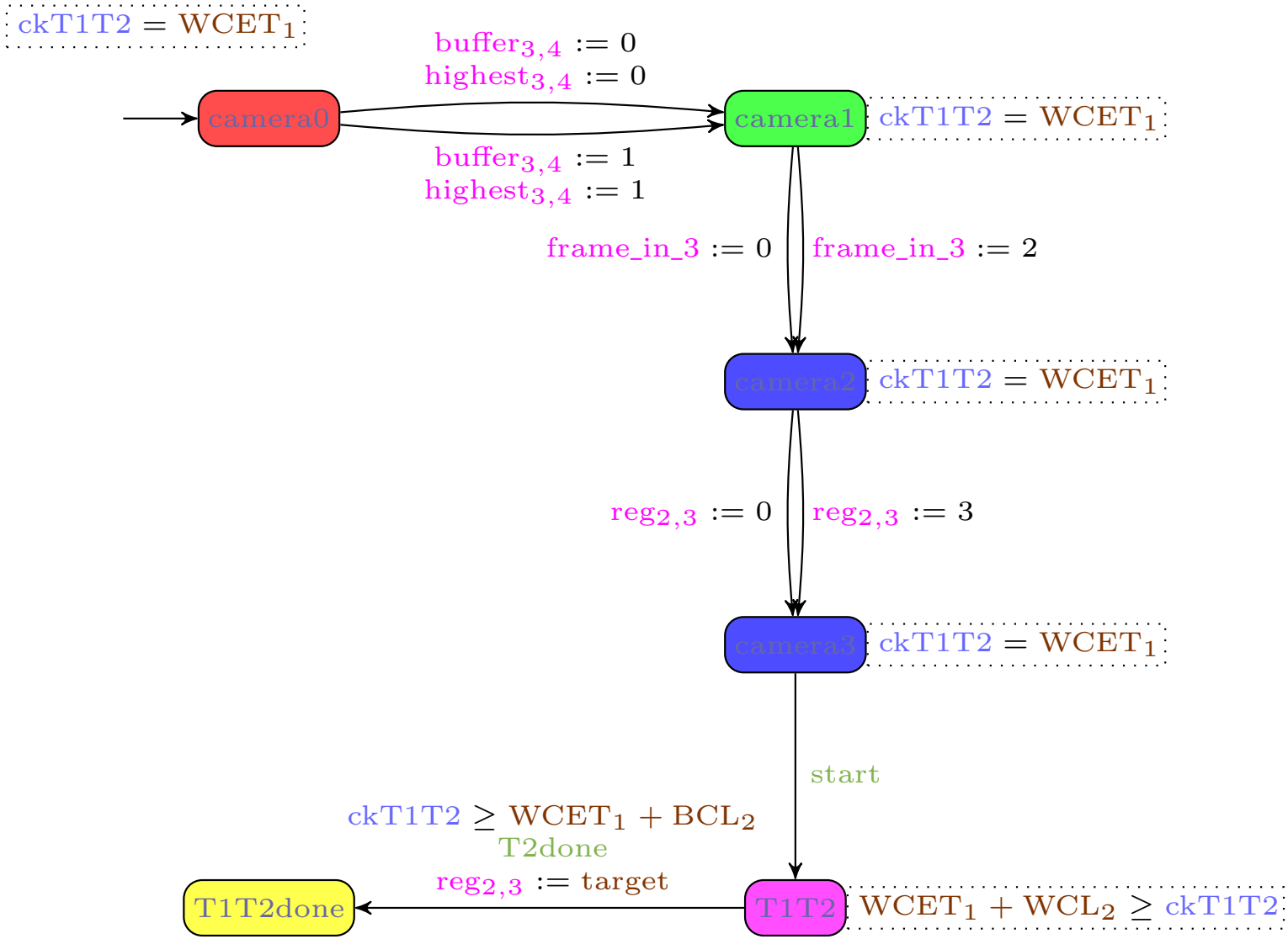
The initialization automaton



The initialization automaton

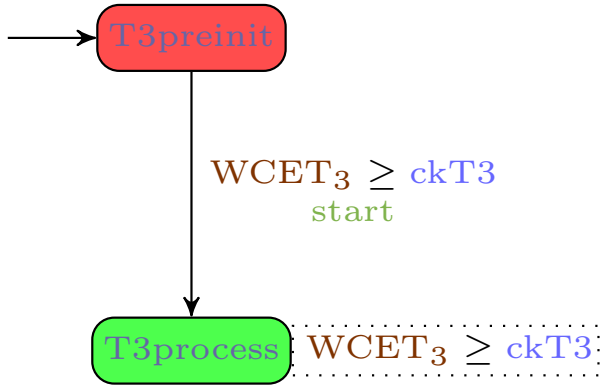


The initialization automaton

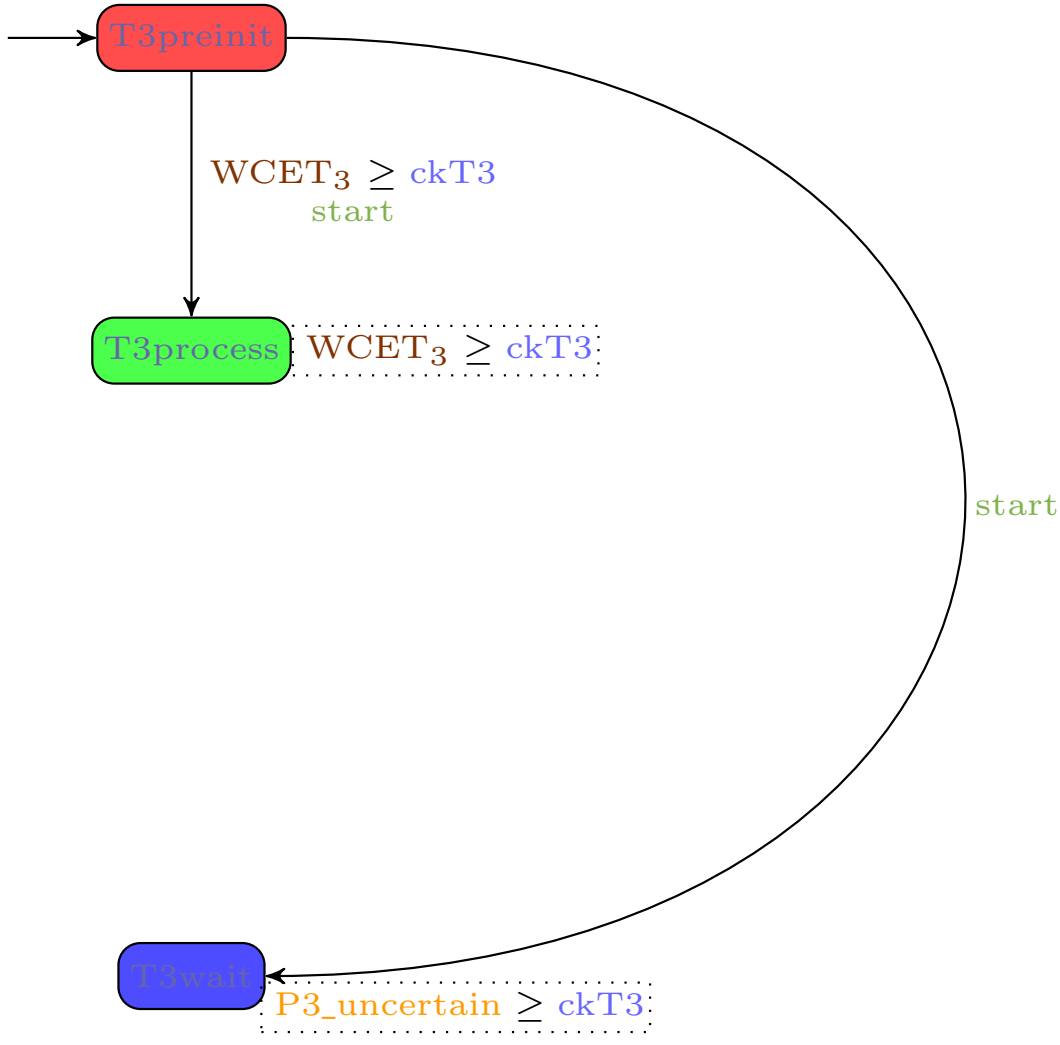


Task T3

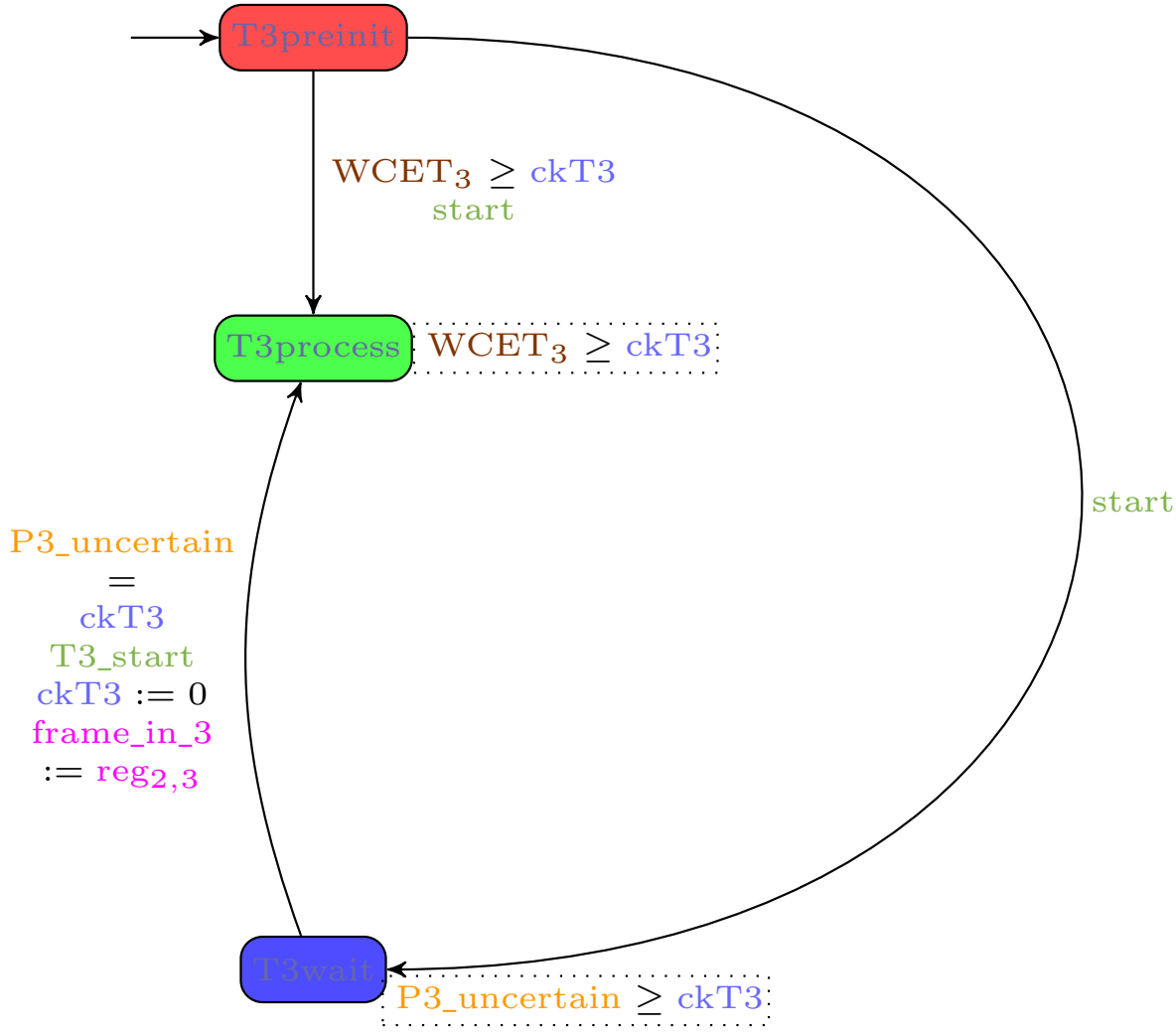
Task T3


T3preinit
$\mathrm{WCET}_3 \geq \mathrm{ckT3}$
start
T3process
$\mathrm{WCET}_3 \geq \mathrm{ckT3}$

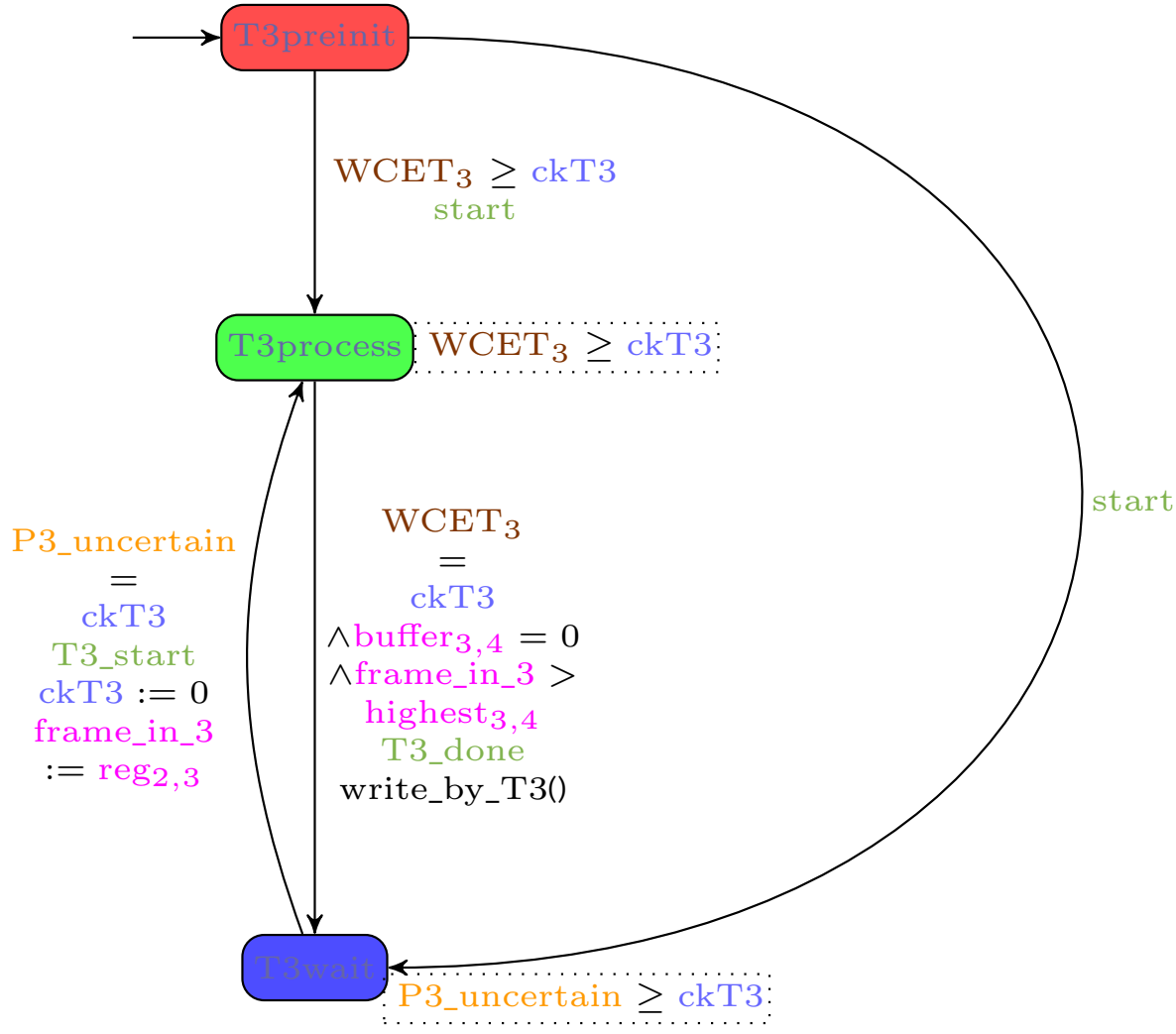
Task T3


T3preinit
$WCET_3 \geq ckT3$
start
T3process
$WCET_3 \geq ckT3$
start
T3wait
$P3\_uncertain \geq ckT3$

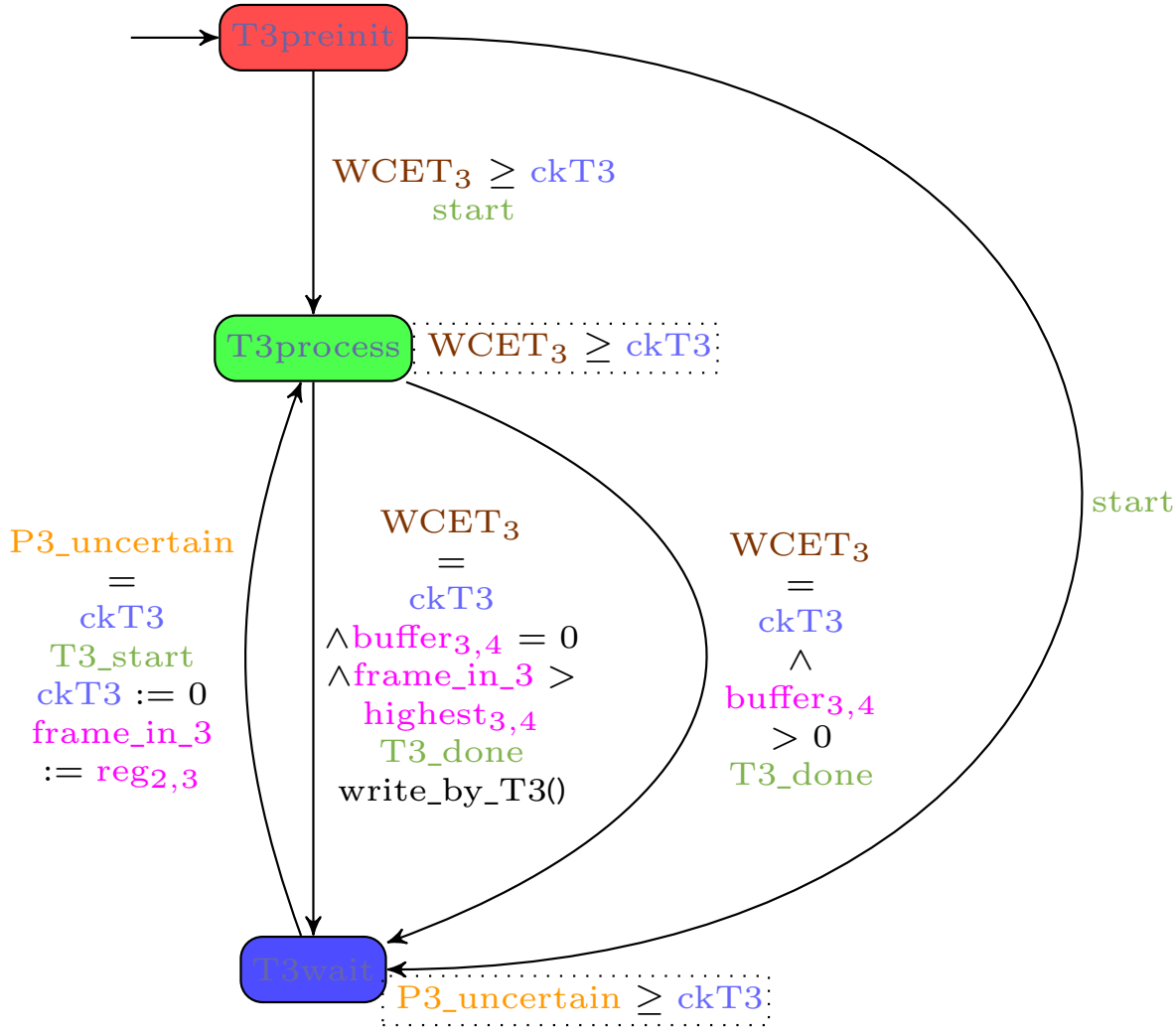
Task T3


T3preinit
WCET3 ≥ ckT3
start
T3process
WCET3 ≥ ckT3
start
P3_uncertain
=
ckT3
T3_start
ckT3 := 0
frame_in_3
:= reg2,3
T3wait
P3_uncertain ≥ ckT3

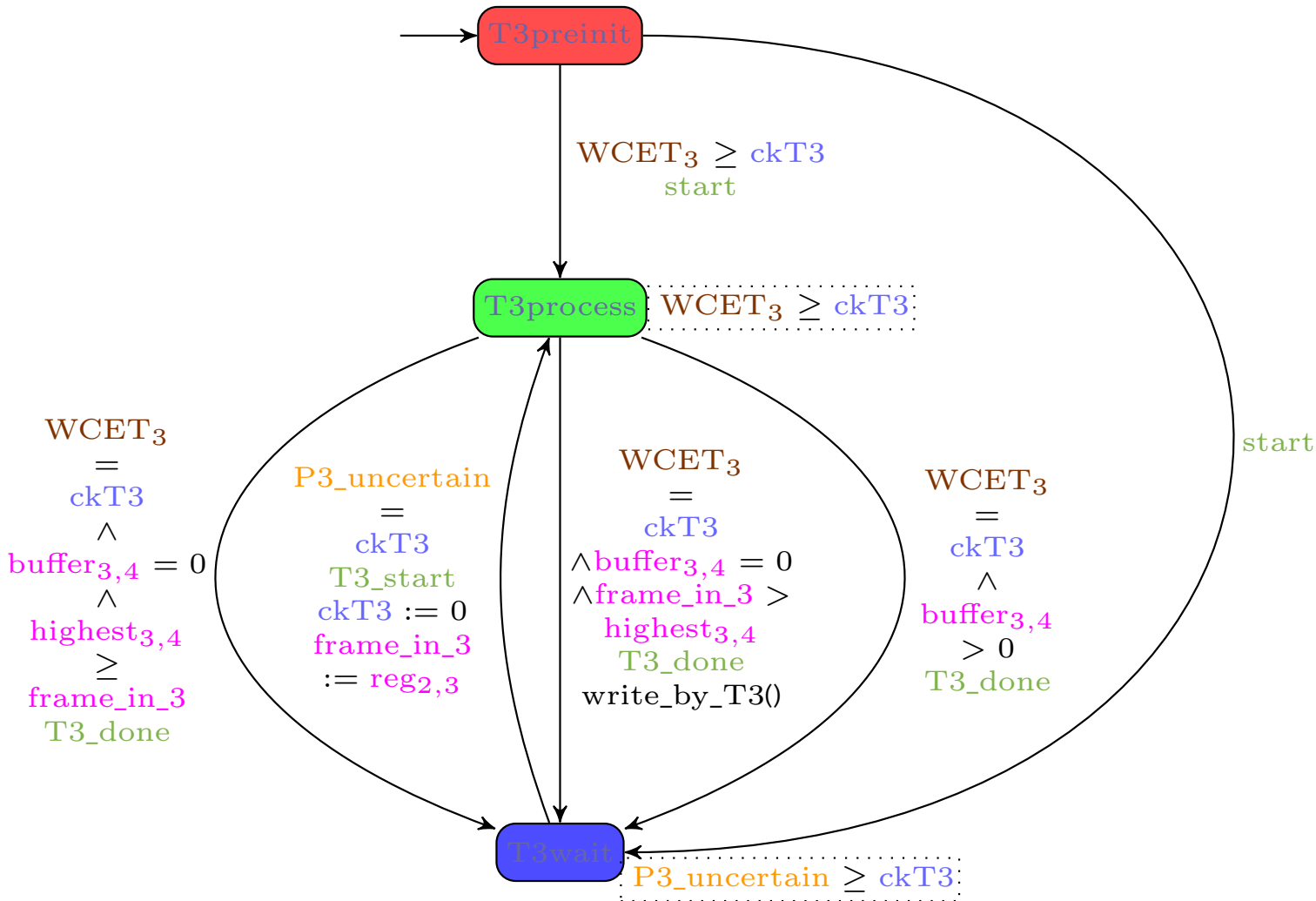
Task T3


T3preinit
WCET3 ≥ ckT3
start
T3process
WCET3 ≥ ckT3
start
P3_uncertain
=
ckT3
T3_start
ckT3 := 0
frame_in_3
:= reg2,3
WCET3
=
ckT3
∧buffer3,4 = 0
∧frame_in_3 >
highest3,4
T3_done
write_by_T3()
T3wait
P3_uncertain ≥ ckT3

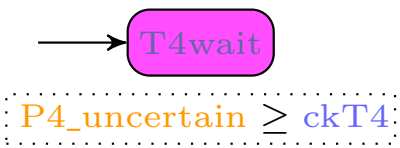
Task T3


T3preinit
WCET3 ≥ ckT3
start
T3process
WCET3 ≥ ckT3
start
P3_uncertain
=
ckT3
T3_start
ckT3 := 0
frame_in_3
:= reg2,3
WCET3
=
ckT3
∧buffer3,4 = 0
∧frame_in_3 >
highest3,4
T3_done
write_by_T3()
WCET3
=
ckT3
∧
buffer3,4
> 0
T3_done
T3wait
P3_uncertain ≥ ckT3

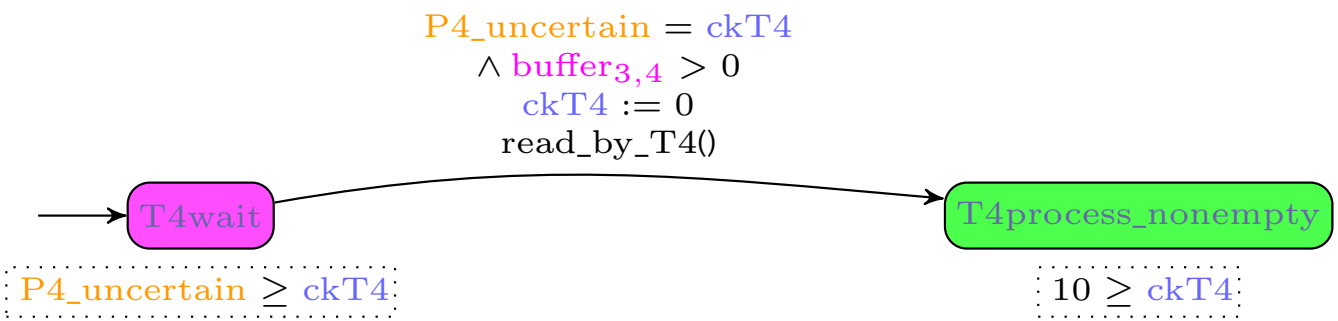
Task T3


T3preinit
WCET3 ≥ ckT3
start
T3process
WCET3 ≥ ckT3
WCET3
=
ckT3
∧
buffer3,4 = 0
∧
highest3,4
≥
frame_in_3
T3_done
P3_uncertain
=
ckT3
T3_start
ckT3 := 0
frame_in_3
:= reg2,3
WCET3
=
ckT3
∧buffer3,4 = 0
∧frame_in_3 >
highest3,4
T3_done
write_by_T3()
WCET3
=
ckT3
∧
buffer3,4
> 0
T3_done
start
T3wait
P3_uncertain ≥ ckT3

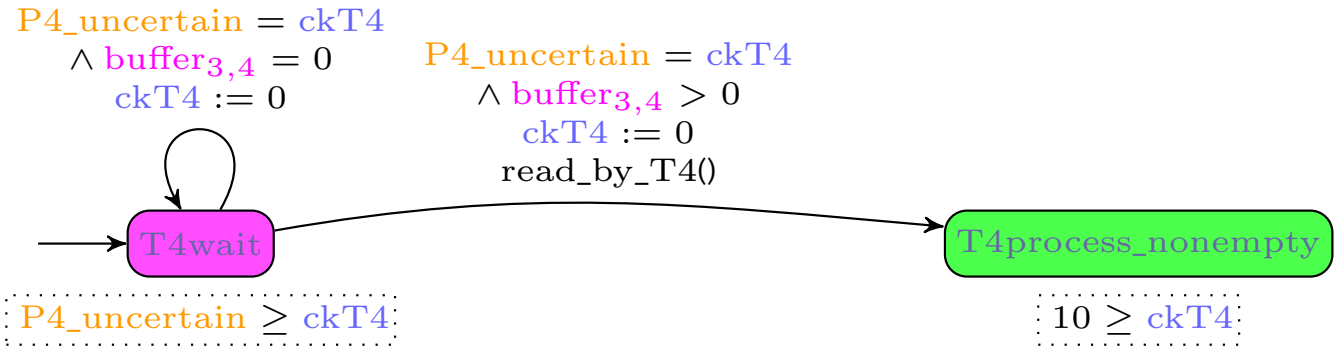
Task T4



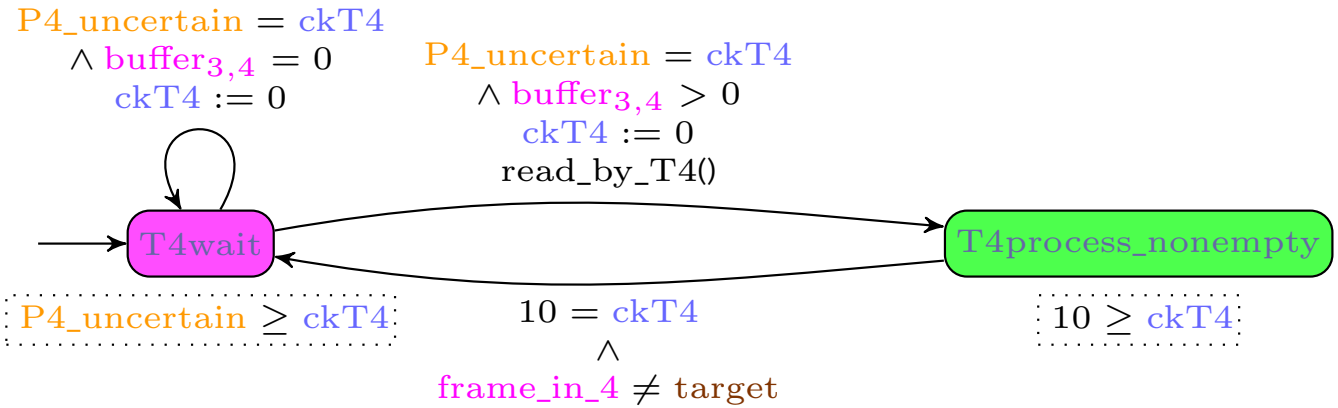
Task T4


P4_uncertain = ckT4
∧ buffer3,4 > 0
ckT4 := 0
read_by_T4()
T4wait
T4process_nonempty
P4_uncertain ≥ ckT4
10 ≥ ckT4

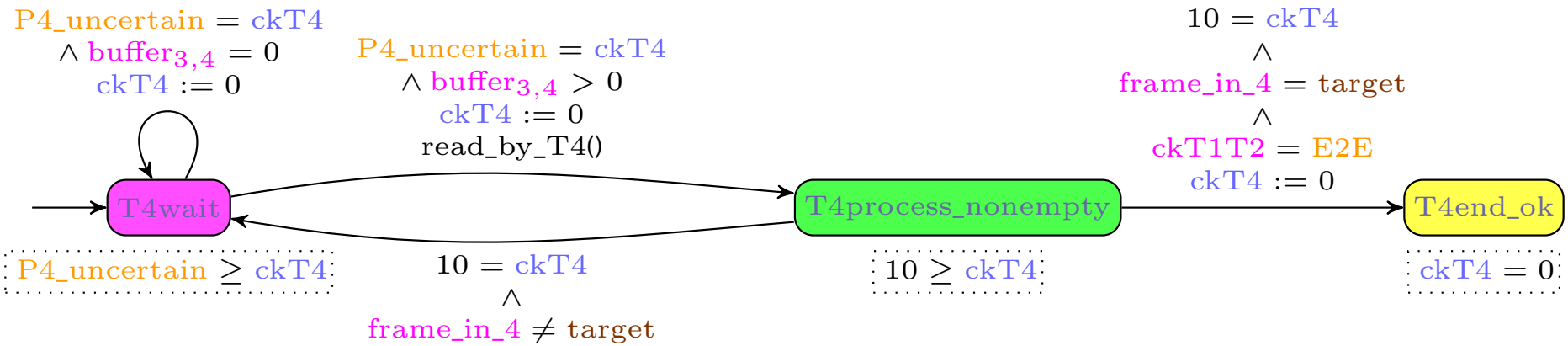
Task T4


P4_uncertain = ckT4
∧ buffer3,4 = 0
ckT4 := 0
P4_uncertain = ckT4
∧ buffer3,4 > 0
ckT4 := 0
read_by_T4()
T4wait
T4process_nonempty
P4_uncertain ≥ ckT4
10 ≥ ckT4

Task T4


P4_uncertain = ckT4
∧ buffer3,4 = 0
ckT4 := 0
P4_uncertain = ckT4
∧ buffer3,4 > 0
ckT4 := 0
read_by_T4()
T4wait
T4process_nonempty
P4_uncertain ≥ ckT4
10 = ckT4
∧
frame_in_4 ≠ target
10 ≥ ckT4

Task T4


P4_uncertain = ckT4
∧ buffer3,4 = 0
ckT4 := 0
P4_uncertain = ckT4
∧ buffer3,4 > 0
ckT4 := 0
read_by_T4()
10 = ckT4
∧
frame_in_4 = target
∧
ckT1T2 = E2E
ckT4 := 0
T4wait
T4process_nonempty
T4end_ok
P4_uncertain ≥ ckT4
10 = ckT4
∧
frame_in_4 ≠ target
10 ≥ ckT4
ckT4 = 0

Results

E2E latency results for $n = 1$ and $n = 3$

	$n = 1$	$n = 3$
min E2E	63 ms	63 ms
max E2E	145.008 ms	225.016 ms

Results obtained using IMITATOR in a few seconds

[ÉA, Lipari, Sun @ WATERS'15]

Bibliography

References I

Abdeddaïm, Y. and Maler, O. (2001).
Job-shop scheduling using timed automata.
In Berry, G., Comon, H., and Finkel, A., editors, *CAV*, volume 2102 of *Lecture Notes in Computer Science*, pages 478–492. Springer.

Adbeddaïm, Y. and Maler, O. (2002).
Preemptive job-shop scheduling using stopwatch automata.
In *TACAS*, volume 2280 of *LNCS*, pages 113–126. Springer-Verlag.

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 *STOC*, pages 592–601. ACM.

André, É. (2017).
A unified formalism for monoprocessor schedulability analysis under uncertainty.
In Cavalcanti, A., Petrucci, L., and Seceleanu, C., editors, *FMICS-AVoCS*, volume 10471 of *Lecture Notes in Computer Science*, pages 100–115. Springer.
Best paper award.

References II


André, É. (2017).
What's decidable about parametric timed automata?
International Journal on Software Tools for Technology Transfer.
To appear.


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


André, É., Hasuo, I., and Waga, M. (2018a).
Offline timed pattern matching under uncertainty.
In *ICECCS*. IEEE.
To appear.


André, É. and Lime, D. (2017).
Liveness in L/U-parametric timed automata.
In Legay, A. and Schneider, K., editors, *ACSD*, pages 9–18. IEEE.


André, É., Lime, D., and Ramparison, M. (2018b).
TCTL model checking lower/upper-bound parametric timed automata without invariants.
In Jansen, D. N. and Prabhakar, P., editors, *FORMATS*, volume 11022 of *Lecture Notes in Computer Science*, pages 1–17. Springer.

References III


André, É., Lime, D., and Ramparison, M. (2018c).
Timed automata with parametric updates.
In Juhás, G., Chatain, T., and Grosu, R., editors, *ACSD*, pages 21–29. IEEE.
To appear.


André, É., Lime, D., and Roux, O. H. (2016).
Decision problems for parametric timed automata.
In Ogata, K., Lawford, M., and Liu, S., editors, *ICFEM*, volume 10009 of *LNCS*, pages 400–416. Springer.


André, É., Lipari, G., and Sun, Y. (2015).
Verification of two real-time systems using parametric timed automata.
In Quinton, S. and Vardanega, T., editors, *WATERS*.


André, É. and Markey, N. (2015).
Language preservation problems in parametric timed automata.
In *FORMATS*, volume 9268 of *LNCS*, pages 27–43. Springer.


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


Beneš, N., Bezděk, P., Larsen, K. G., and Srba, J. (2015).
Language emptiness of continuous-time parametric timed automata.
In *ICALP, Part II*, volume 9135 of *LNCS*, pages 69–81. Springer.

References IV


Bozzelli, L. and La Torre, S. (2009).
Decision problems for lower/upper bound parametric timed automata.
Formal Methods in System Design, 35(2):121–151.


Bundala, D. and Ouaknine, J. (2014).
Advances in parametric real-time reasoning.
In *MFCS*, volume 8634 of *LNCS*, pages 123–134. Springer.


Cimatti, A., Palopoli, L., and Ramadian, Y. (2008).
Symbolic computation of schedulability regions using parametric timed automata.
In *RTSS*, pages 80–89. IEEE Computer Society.


Doyen, L. (2007).
Robust parametric reachability for timed automata.
Information Processing Letters, 102(5):208–213.


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


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 *TIME*, pages 73–80. IEEE Computer Society Press.

References V


Hune, T., Romijn, J., Stoelinga, M., and Vaandrager, F. W. (2002).
Linear parametric model checking of timed automata.
Journal of Logic and Algebraic Programming, 52-53:183–220.


Jovanović, A., Lime, D., and Roux, O. H. (2015).
Integer parameter synthesis for timed automata.
IEEE Transactions on Software Engineering, 41(5):445–461.


Larsen, K. G., Pettersson, P., and Yi, W. (1997).
UPPAAL in a nutshell.
International Journal on Software Tools for Technology Transfer, 1(1-2):134–152.


Lipari, G., Sun, Y., André, É., and Fribourg, L. (2014).
Toward parametric timed interfaces for real-time components.
In Andre, E. and Frehse, G., editors, *SynCoP*, volume 145 of *Electronic Proceedings in Theoretical Computer Science*, pages 49–64.


Luthmann, L., Stephan, A., Bürdek, J., and Lochau, M. (2017).
Modeling and testing product lines with unbounded parametric real-time constraints.
In Cohen, M. B., Acher, M., Fuentes, L., Schall, D., Bosch, J., Capilla, R., Bagheri, E., Xiong, Y., Troya, J., Cortés, A. R., and Benavides, D., editors, *SPLC, Volume A*, pages 104–113. ACM.


Markey, N. (2011).
Robustness in real-time systems.
In *SIES*, pages 28–34. IEEE Computer Society Press.

References VI

Miller, J. S. (2000).
Decidability and complexity results for timed automata and semi-linear hybrid automata.
In *HSCC*, volume 1790 of *LNCS*, pages 296–309. Springer.

Norström, C., Wall, A., and Yi, W. (1999).
Timed automata as task models for event-driven systems.
In *RTCSA*, pages 182–189. IEEE Computer Society.

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

Licensing

Source of the graphics used I

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

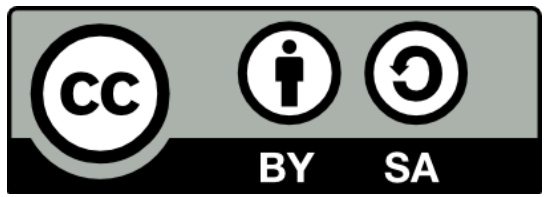
Title: Smiley green alien big eyes (cry)
Author: LadyofHats
Source: `https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg`
License: public domain

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/