Université Paris 13
Institut Galilée
Master 2 PLS

SITH – Partie 2.2

Systèmes temporisés paramétrés

Étienne André

Etienne.Andre (à) univ-paris13.fr

Version : January 28, 2018 (diapositives à trous pour Web)

Partie 2.2: SITH – Plan

1. Parametric Timed Automata
2. Decidability results
3. Parameter synthesis
4. Conclusion

Plan: Parametric Timed Automata

1 Parametric Timed Automata
- Syntax
- Specification with PTA
- Semantics

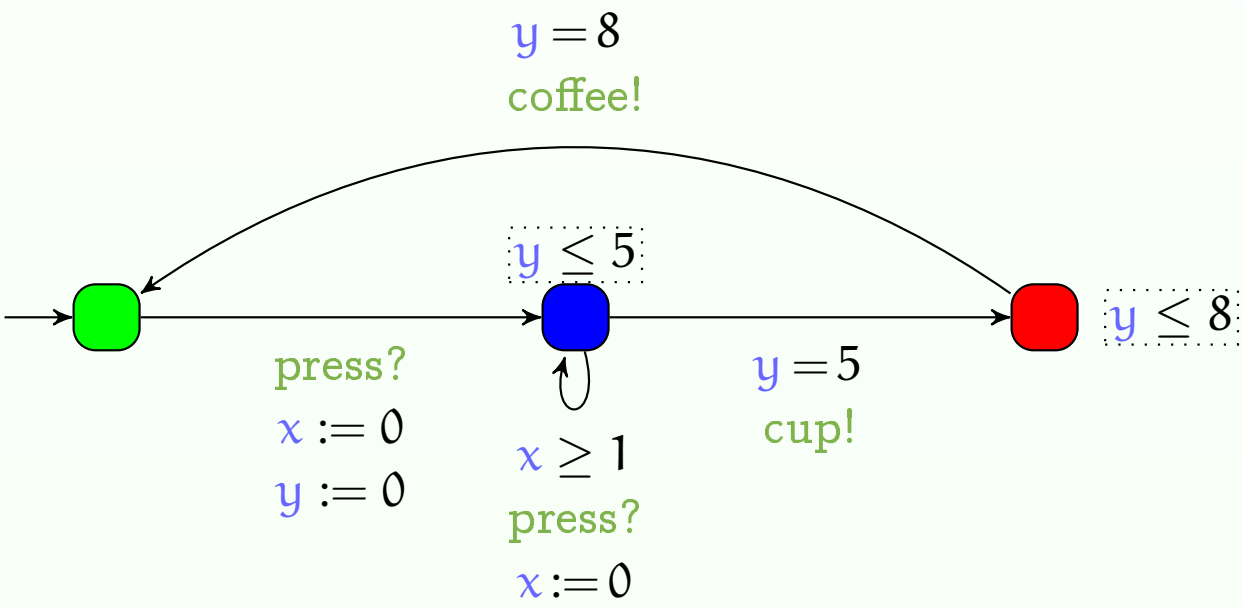
2 Decidability results

3 Parameter synthesis

4 Conclusion

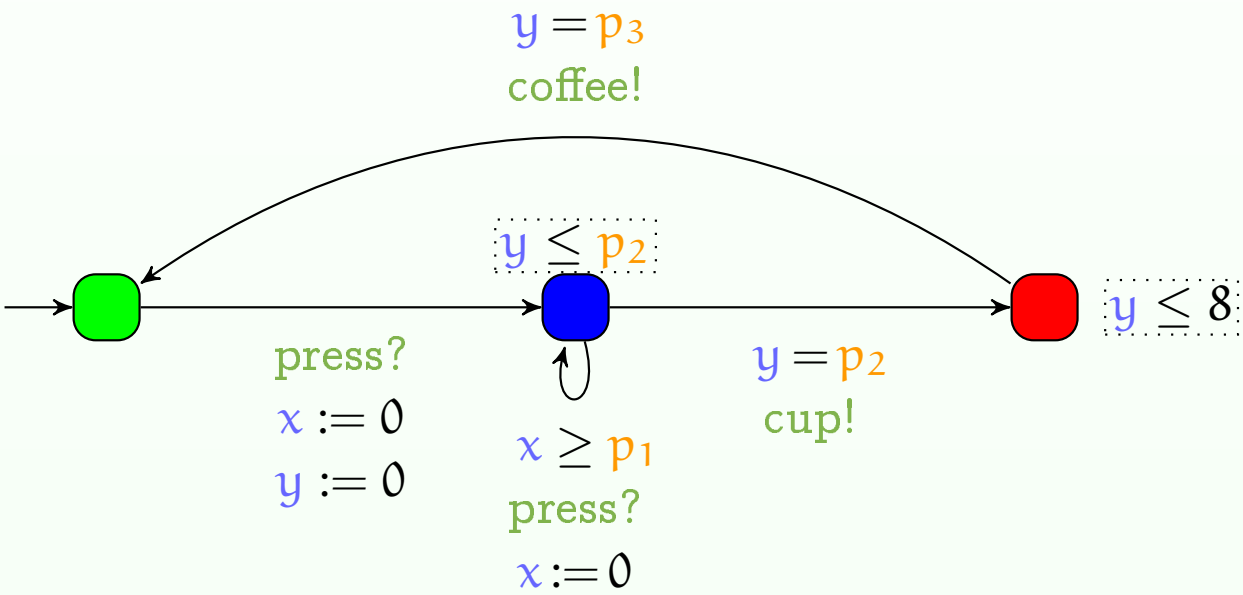
Parametric Timed Automaton (PTA)

- 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



Formal definition

Définition (Parametric timed automaton)

A *parametric timed automaton (PTA)* $\mathcal{A}$ is an 8-tuple of the form $\mathcal{A} = (L, \Sigma, l_I, X, P, K, I, \rightarrow)$, where

- L is a finite set of locations, $l_I \in L$ is the initial location,
- Σ is a finite set of actions,
- X is a set of clocks, P is a set of parameters,
- K is a constraint on the parameters of P,
- I is the invariant, assigning to every $l \in L$ a constraint $I(l)$ on the clocks and the parameters, and
- $\rightarrow$ is a step (or "transition") relation consisting of elements of the form $e = (l, g, a, R, l')$, also denoted by $l \xrightarrow{g,a,R} l'$, where $l, l' \in L$, $a \in \Sigma$, $R \subseteq X$ is a set of clock variables to be reset by the step, and g (the step guard) is a constraint on the clocks and the parameters.

Example 1

Draw the PTA $\mathcal{A} = (L, \Sigma, l_2, X, P, p_1 \leq p_2 + 2p_3, I, \rightarrow)$ such that

- $L = \{l_1, l_2, l_3, l_4\}$,
- $\Sigma = \{a_1, a_2, a_3\}$,
- $X = \{x_1, x_2\}$,
- $P = \{p_1, p_2, p_3\}$,
- $I(l_1) = x_1 \leq 3$, and $I(l_4) = x_2 \geq 2p_1 - p_2$,
- $\rightarrow = \{(l_1, x_2 \leq 1, a_1, \{\}, l_3)$,
  $(l_2, x_2 = p_1, a_3, \{x_2\}, l_2)$,
  $(l_2, \texttt{true}, a_2, \{x_1, x_2\}, l_4)$,
  $(l_3, x_1 \geq p_2, a_1, \{x_1\}, l_2)$,
  $(l_3, \texttt{true}, a_2, \{x_2\}, l_3)$,
  $(l_4, x_2 > p_3, a_3, \{\}, l_3)\}$

Example 1: solution

Example 2

Give the formal PTA corresponding to the parametric timed coffee vending machine.

Example 2

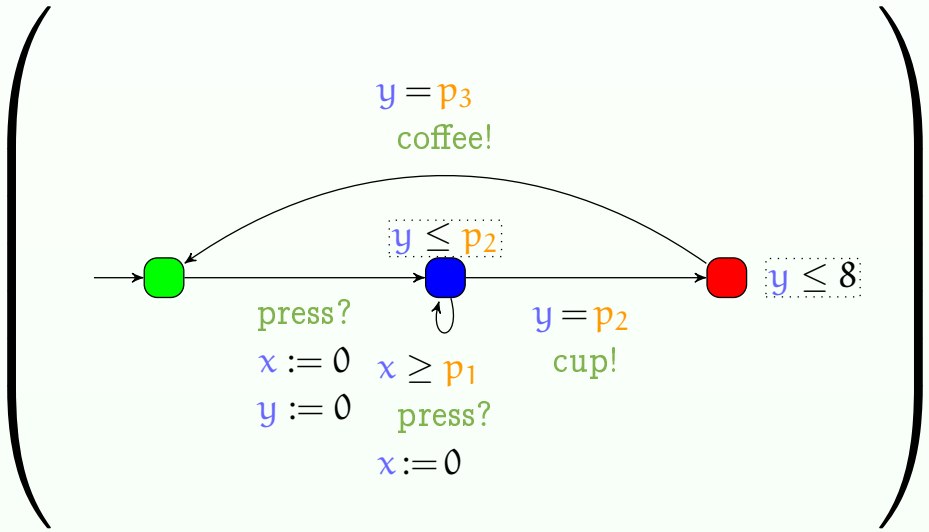
Give the formal PTA corresponding to the parametric timed coffee vending machine.

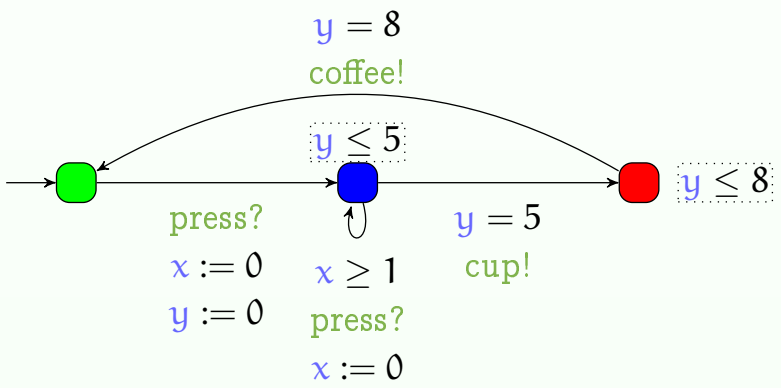
Valuation of a PTA

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

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 

$=$ 

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

Valuation of a PTA: Example

Consider the parametric timed coffee vending machine $\mathcal{A}$.

Let v be a parameter valuation such that $v(p_1) = 3$, $v(p_2) = 4$, and $v(p_3) = 2$.

Draw $v(\mathcal{A})$.

Valuation of a PTA: Example

Consider the parametric timed coffee vending machine $\mathcal{A}$.

Let v be a parameter valuation such that $v(p_1) = 3$, $v(p_2) = 4$, and $v(p_3) = 2$.

Draw $v(\mathcal{A})$.

Example: Railroad gate controller [Alur et al., 1993]

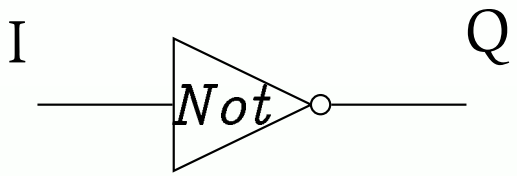
Design three PTAs in parallel:

1. The train: once it is approaching (action approach), it will come in (action in) after at least a time units, then go out (action out) and finally exit (action exit) after at most b time units
2. The gate: upon reception of a lower signal, starts to lower; once it is down, and upon reception of a raise signal, the gate raises again; the time to lower and raise the gate is an interval $[c, d]$
3. The controller: once a train approaches (action approach), it triggers the lower signal within $[e, f]$ time units; then, once the train exits (action exit), it triggers the raise signal again within $[e, f]$ time units

All PTA are cyclic, i.e., repeat the same behavior forever.

Example: Railroad gate controller (cont.)

Example: A hardware gate



The output Q reacts to the change of the input I after a delay $[N^-, N^+]$

Example: A nuclear power plant

Design a PTA modeling a nuclear power plant:
At first, the plant is in normal mode. Suddenly, it may start to heat (action startHeating). At that point, a timer is set; after p_2 time units, the timer will trigger an alarm (action alarm). Then, p_3 time units later, a watering system (action watering) starts. This watering system lasts for p_4 time units, after which the plant is cool again and goes back to the normal mode. However, p_1 time units after the plant starts to heat, the plant may explode at any time (action boom) – unless of course it is cool again.

Example: A nuclear power plant (cont.)

Symbolic states for timed automata

- Objective: group all concrete states reachable by the same sequence of discrete actions
- Symbolic state: a location l and a (infinite) set of states Z
- For timed automata, Z can be represented by a convex polyhedron with a special form called zone, with constraints

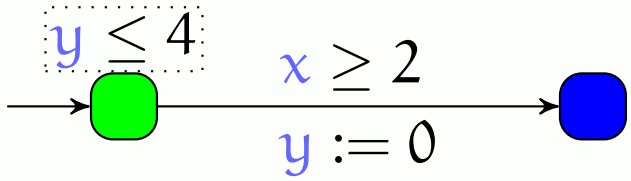
$$-d_{0i} \leq x_i \leq d_{i0} \text{ and } x_i - x_j \leq d_{ij}$$

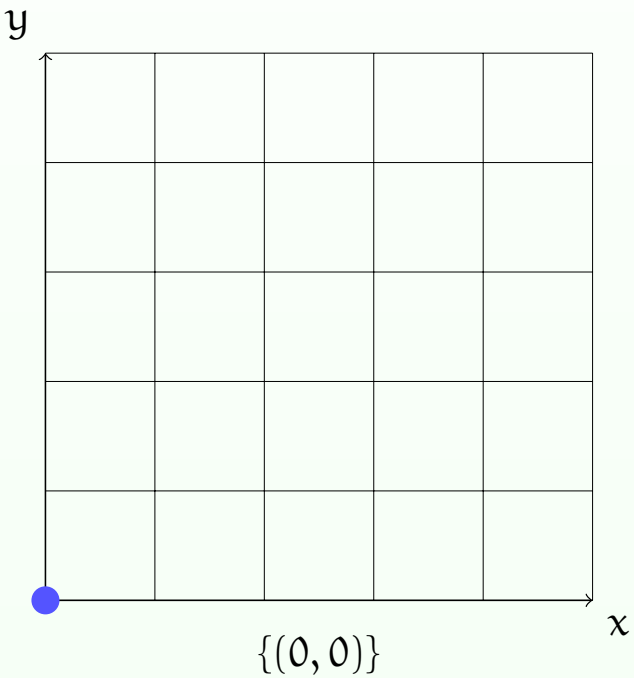
- Computation of successive reachable symbolic states can be performed symbolically with polyhedral operations: for edge $e = (l, a, g, R, l')$:

$$\text{Succ}((l, Z), e) = (l', (Z \cap g)[R] \cap I(l'))^{\nearrow} \cap I(l'))$$

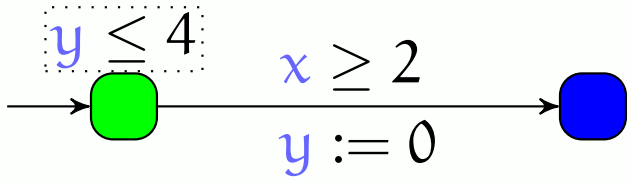
- With an additional technicality, there is a finite number of reachable zones in a TA.

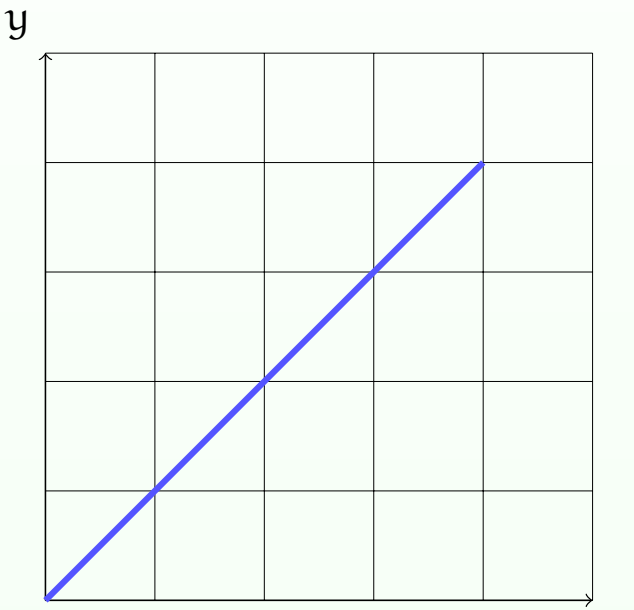
Symbolic states for timed automata: Example





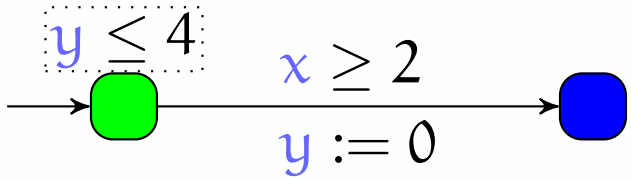
Symbolic states for timed automata: Example

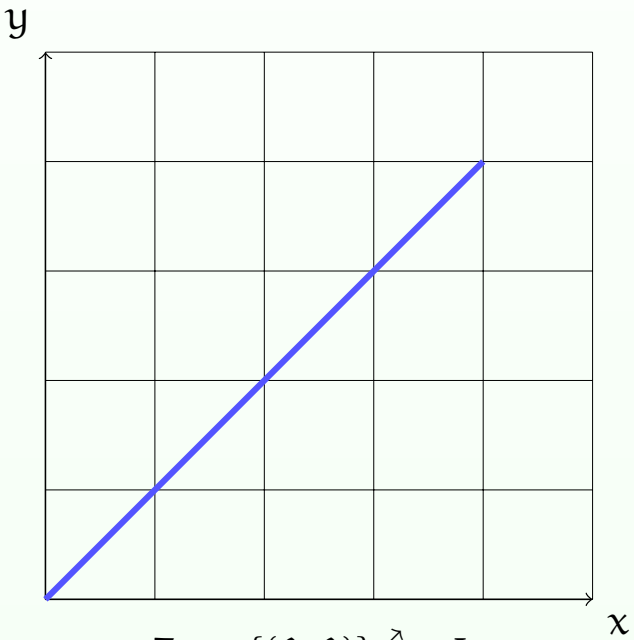




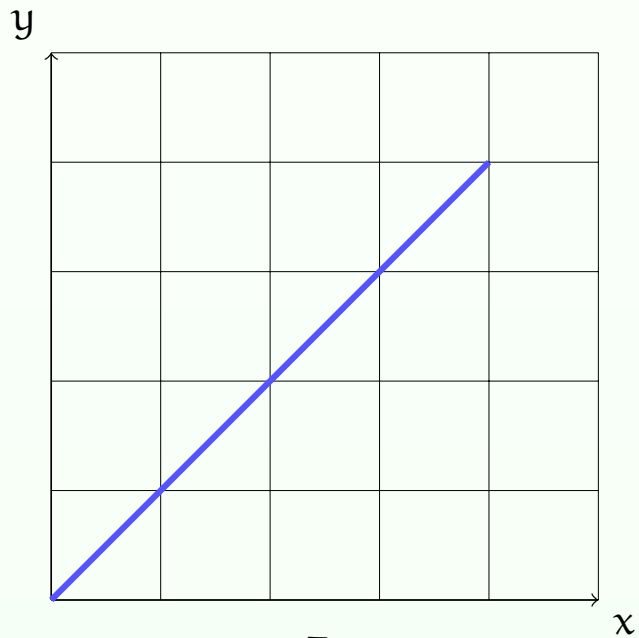
$Z_0 = \{(0,0)\}^{\nearrow} \cap I$

Symbolic states for timed automata: Example



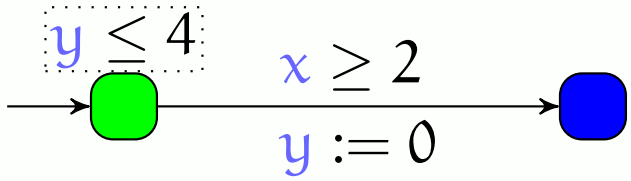


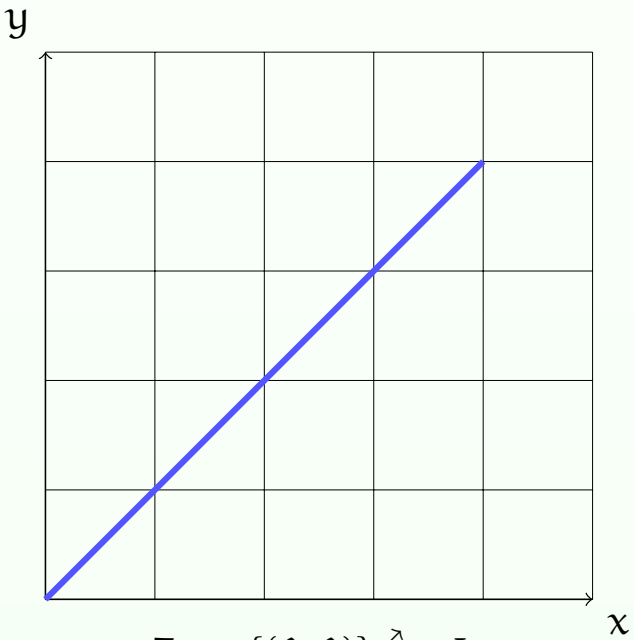
$Z_0 = \{(0,0)\}^{\nearrow} \cap I$



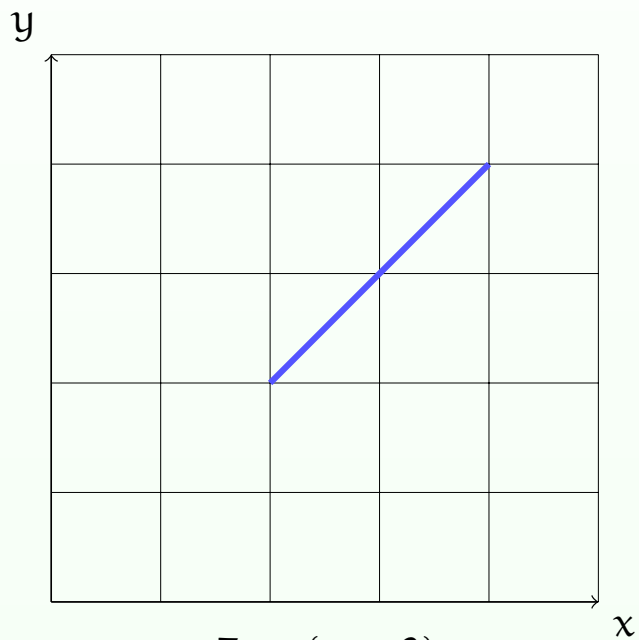
Z_0

Symbolic states for timed automata: Example



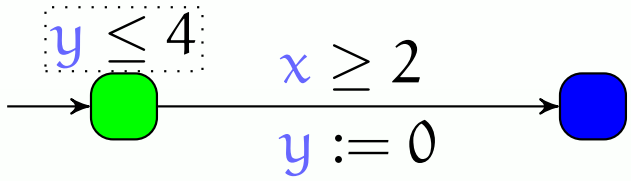


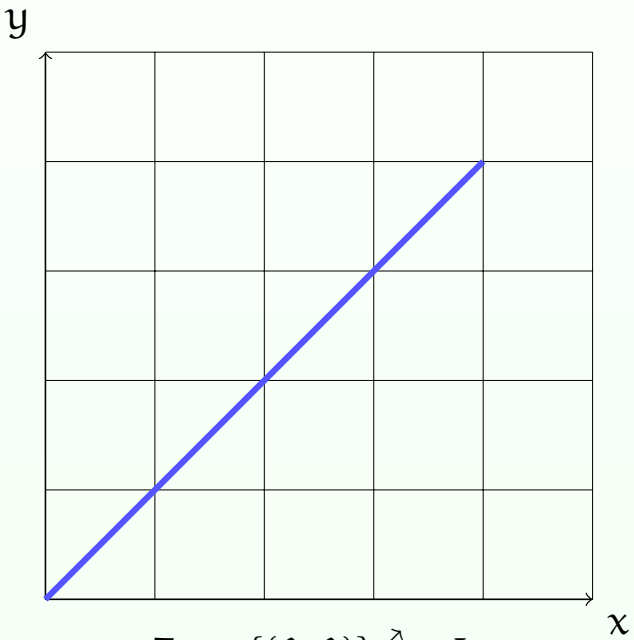
$Z_0 = \{(0,0)\}^{\nearrow} \cap I$



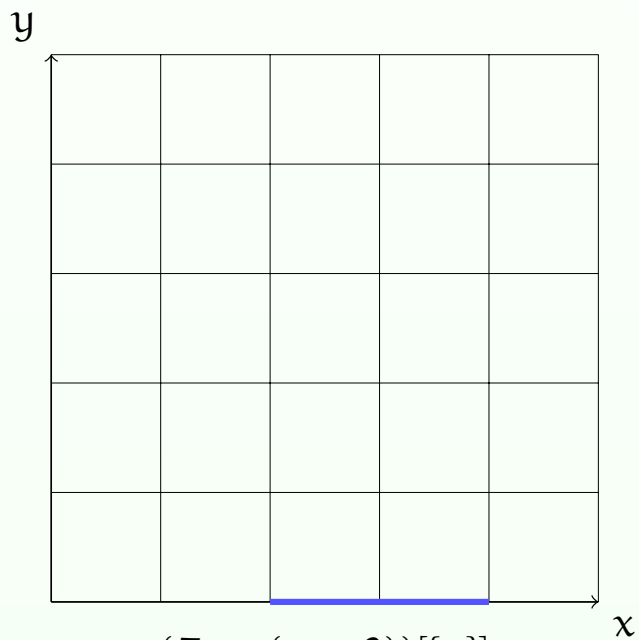
$Z_0 \cap (x \geq 2)$

Symbolic states for timed automata: Example



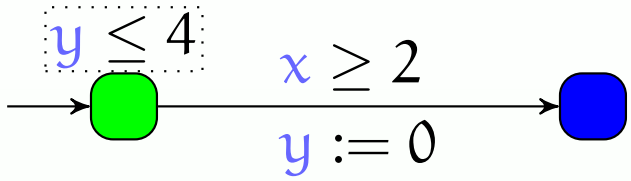


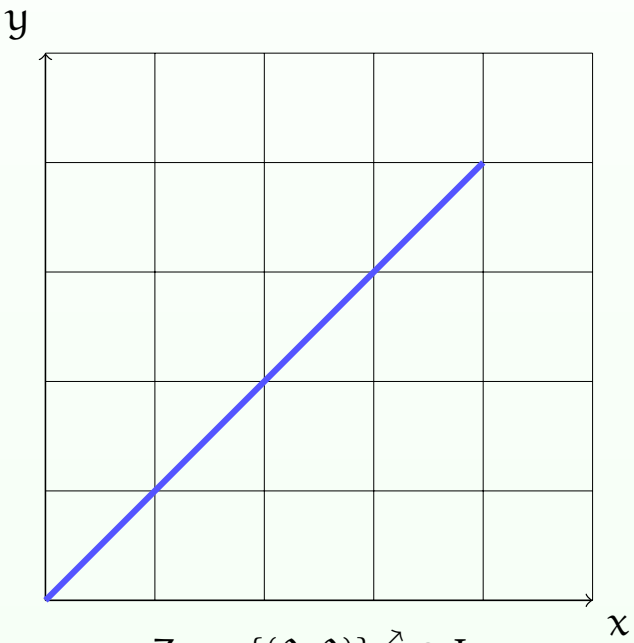
$Z_0 = \{(0,0)\}^{\nearrow} \cap I$



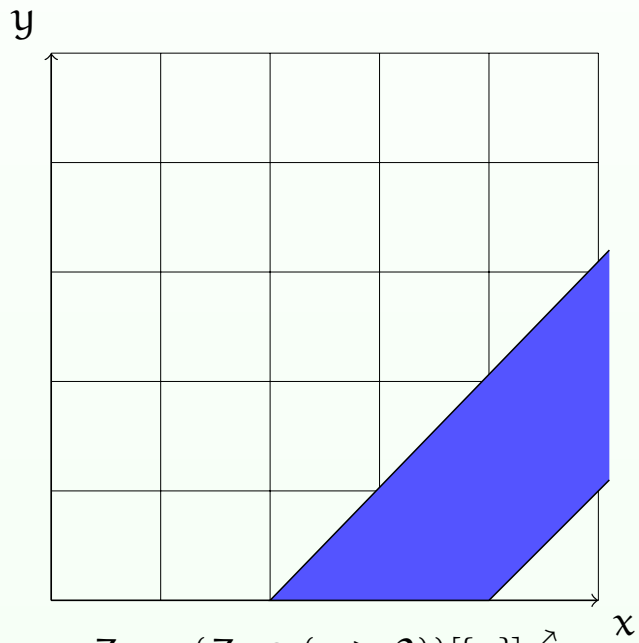
$(Z_0 \cap (x \geq 2))[\{y\}]$

Symbolic states for timed automata: Example





$Z_0 = \{(0,0)\}^{\nearrow} \cap I$



$Z_1 = (Z_0 \cap (x \geq 2))[\{y\}]^{\nearrow}$

Symbolic semantics of parametric timed automata

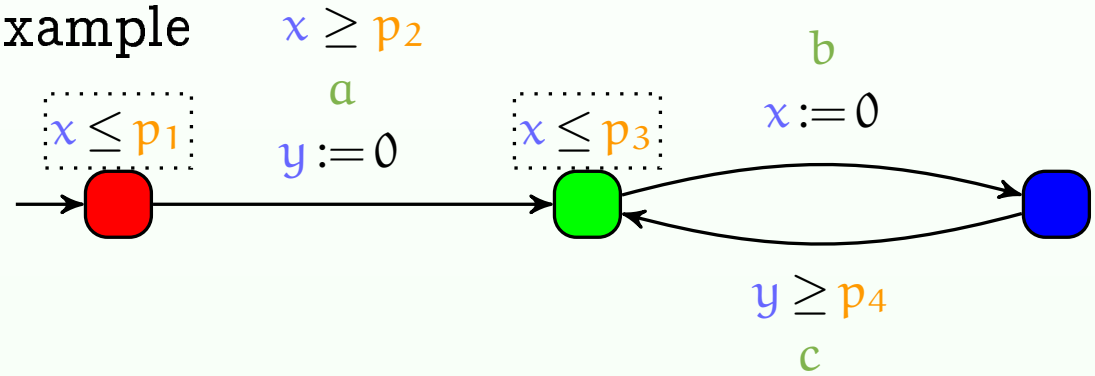
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]

Symbolic semantics of parametric timed automata

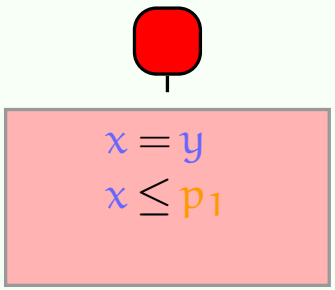
- Symbolic state of a PTA: pair (l, C), where
  - l is a location,
  - C is a convex polyhedron over X and P with a special form, called parametric zone [Hune et al., 2002]
- Symbolic run: alternating sequence of symbolic states and actions

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

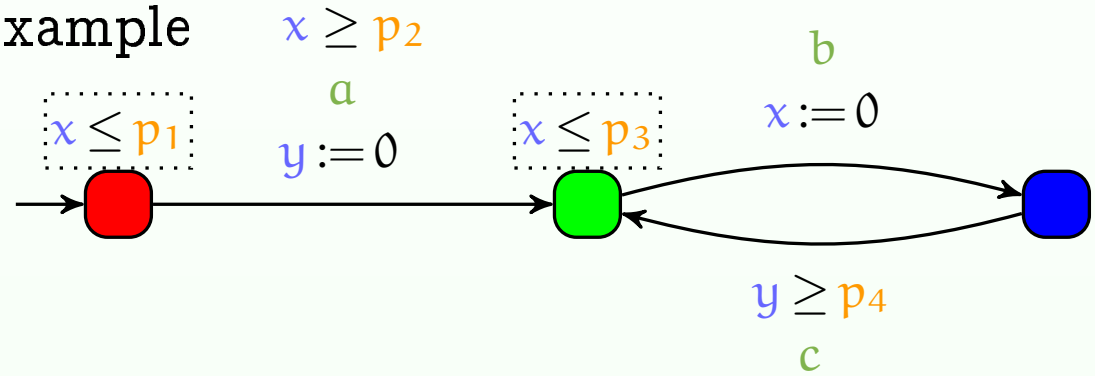


  - Possible symbolic run for this PTA

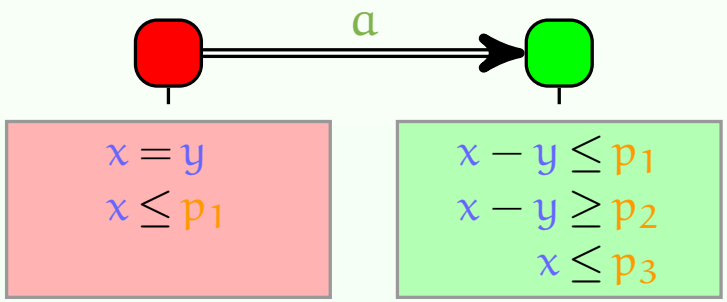


Symbolic semantics of parametric timed automata

- 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

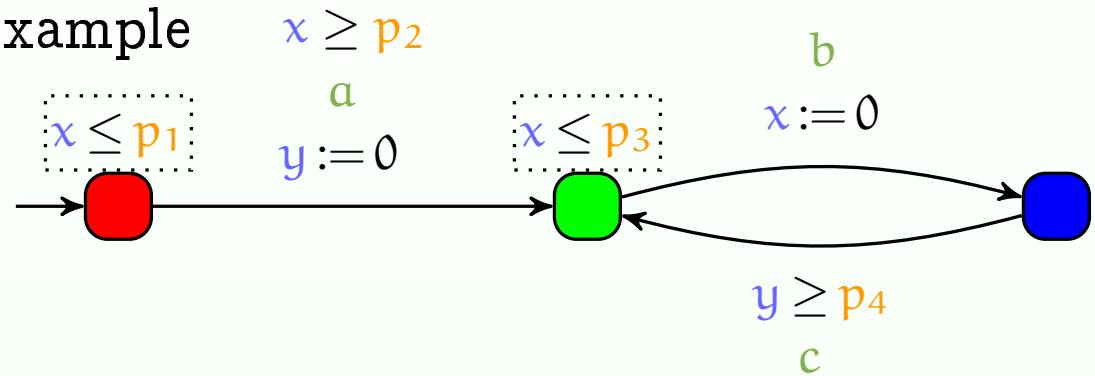


  - Possible symbolic run for this PTA

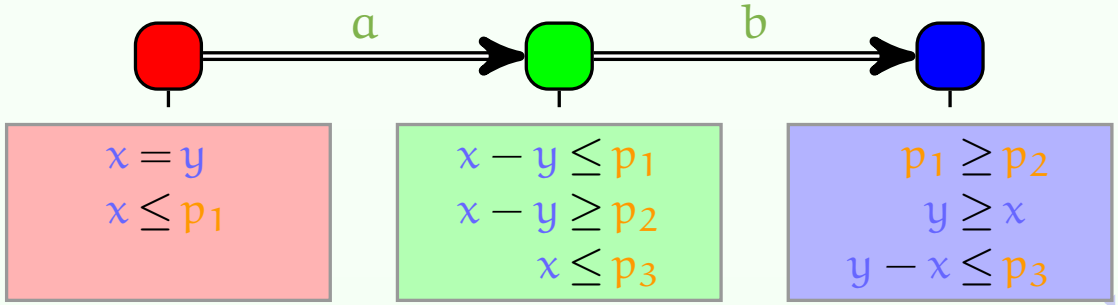


Symbolic semantics of parametric timed automata

- 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 (1/2)

Définition (State)

Let $\mathcal{A} = (\Sigma, L, l_0, X, P, K, I, \rightarrow)$ be a PTA. A *state* s of $\mathcal{A}$ is a pair (l, C) where $l \in L$ is a location, and $C \in \mathcal{L}(X \cup P)$ its associated constraint.

Symbolic semantics of PTA (1/2)

Définition (State)

Let $\mathcal{A} = (\Sigma, L, l_0, X, P, K, I, \rightarrow)$ be a PTA. A *state* s of $\mathcal{A}$ is a pair (l, C) where $l \in L$ is a location, and $C \in \mathcal{L}(X \cup P)$ its associated constraint.

The *initial state* of $\mathcal{A}(K)$ is $s_0 = (l_0, C_0)$, where

$$C_0 = K \wedge I(l_0) \wedge \bigwedge_{i=1}^{H-1} x_i = x_{i+1} \wedge x_i \geq 0$$

Symbolic semantics of PTA (1/2)

Définition (State)

Let $\mathcal{A} = (\Sigma, L, l_0, X, P, K, I, \rightarrow)$ be a PTA. A *state* s of $\mathcal{A}$ is a pair (l, C) where $l \in L$ is a location, and $C \in \mathcal{L}(X \cup P)$ its associated constraint.

The *initial state* of $\mathcal{A}(K)$ is $s_0 = (l_0, C_0)$, where

$$C_0 = K \wedge I(l_0) \wedge \bigwedge_{i=1}^{H-1} x_i = x_{i+1} \wedge x_i \geq 0$$

In this expression, K is the initial constraint over the parameters, $I(l_0)$ is the invariant of the initial state, and the rest of the expression lets clocks evolve from the same initial value.

Symbolic semantics of PTA (2/2)

Définition (Semantics of PTAs)

Let $\mathcal{A} = (\Sigma, L, l_0, X, P, K, I, \rightarrow)$ be a PTA. The *semantics of* $\mathcal{A}$ is $\mathcal{LTS}(\mathcal{A}) = (\Sigma, S, S_0, \Rightarrow)$ where

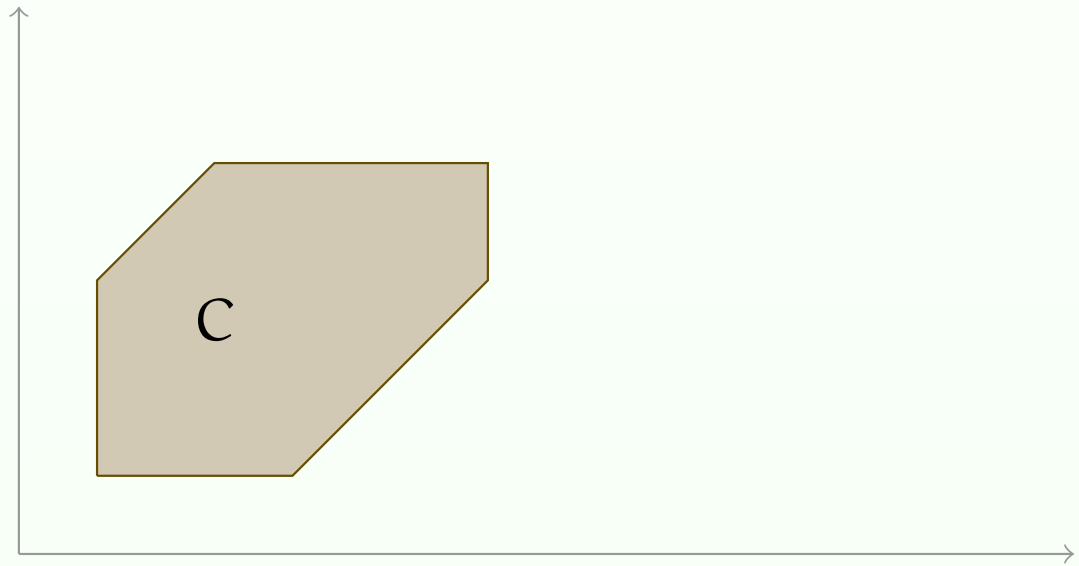
$$S = \{(l, C) \in L \times \mathcal{L}(X \cup P) \mid C \subseteq I(l)\},$$
$$S_0 = \{s_0\}$$

and a transition $(l, C) \overset{a}{\Longrightarrow} (l', C')$ belongs to $\Rightarrow$ if $\exists e = (l, a, g, R, l')$ s.t.:

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

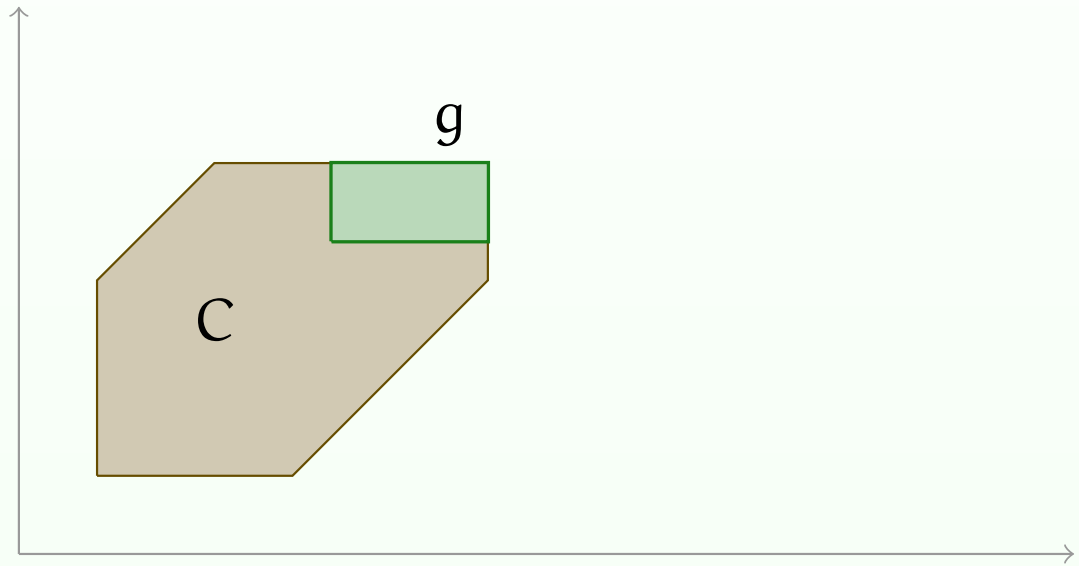
Symbolic semantics of PTA: Illustration

$$C' = (C \cap g)[R] \cap I(l')) \nearrow \cap I(l'))$$



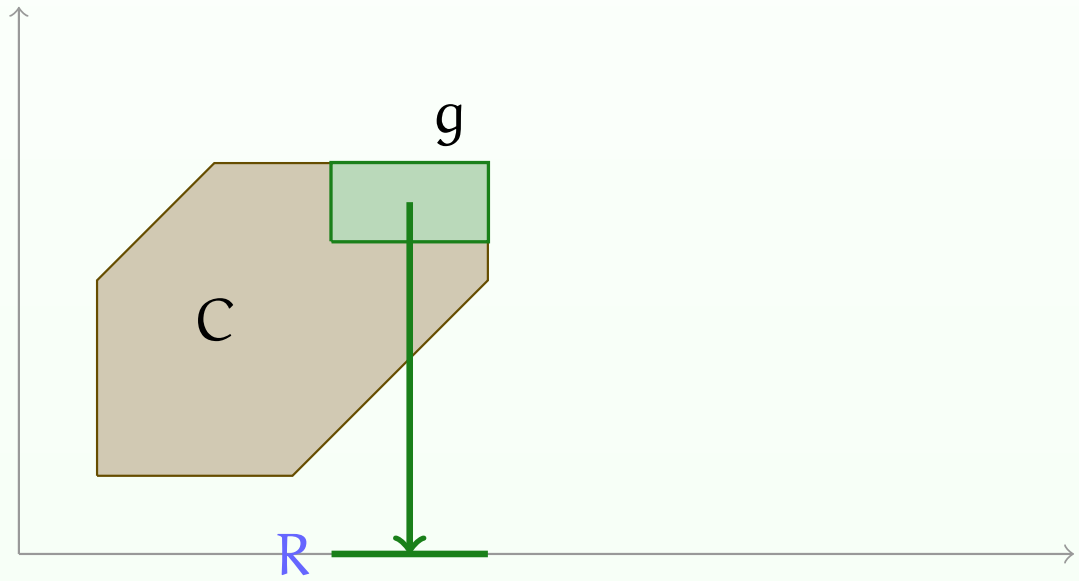
Symbolic semantics of PTA: Illustration

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



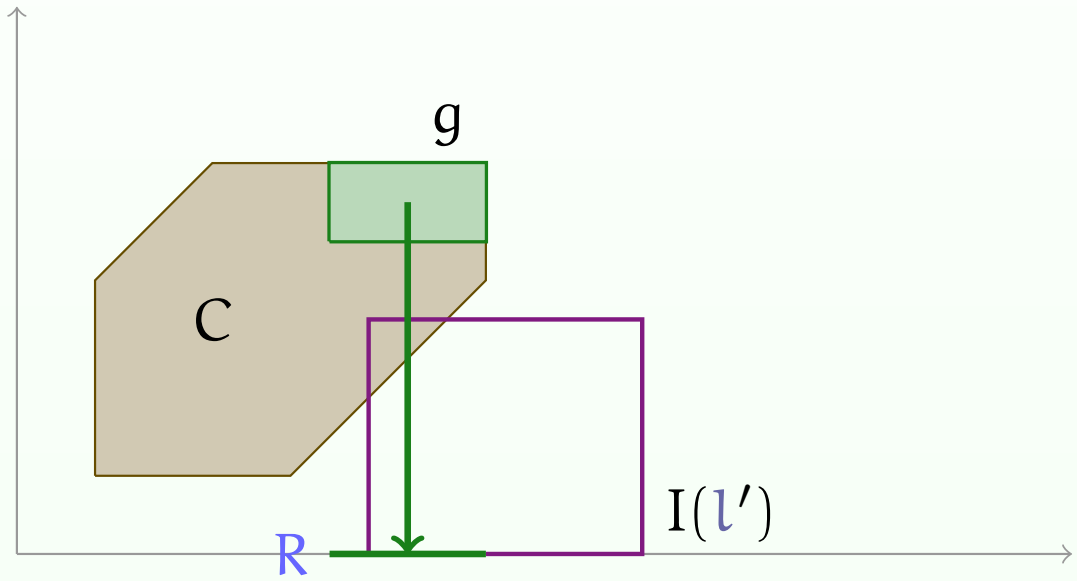
Symbolic semantics of PTA: Illustration

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



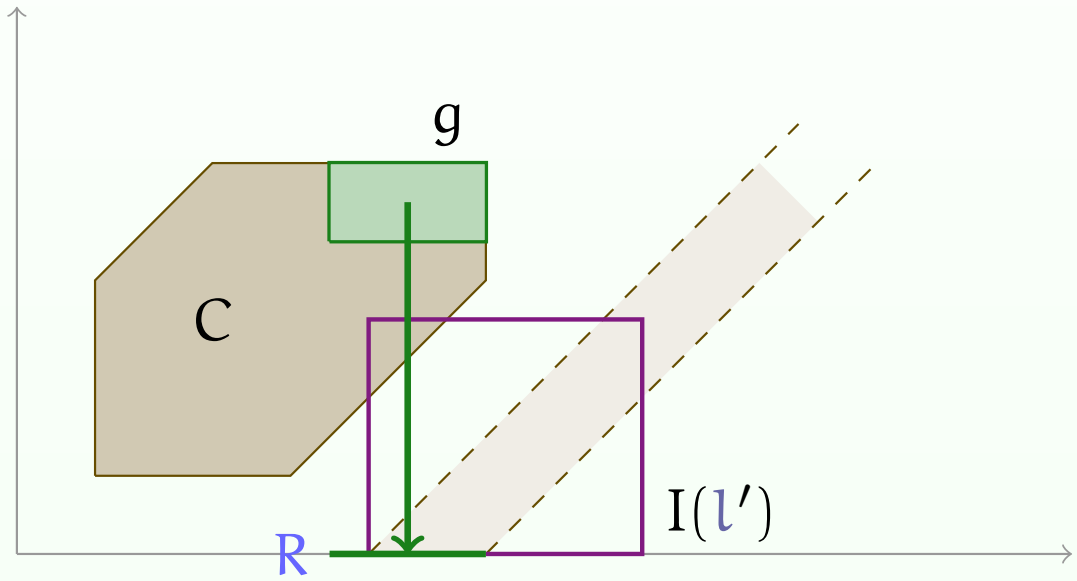
Symbolic semantics of PTA: Illustration

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



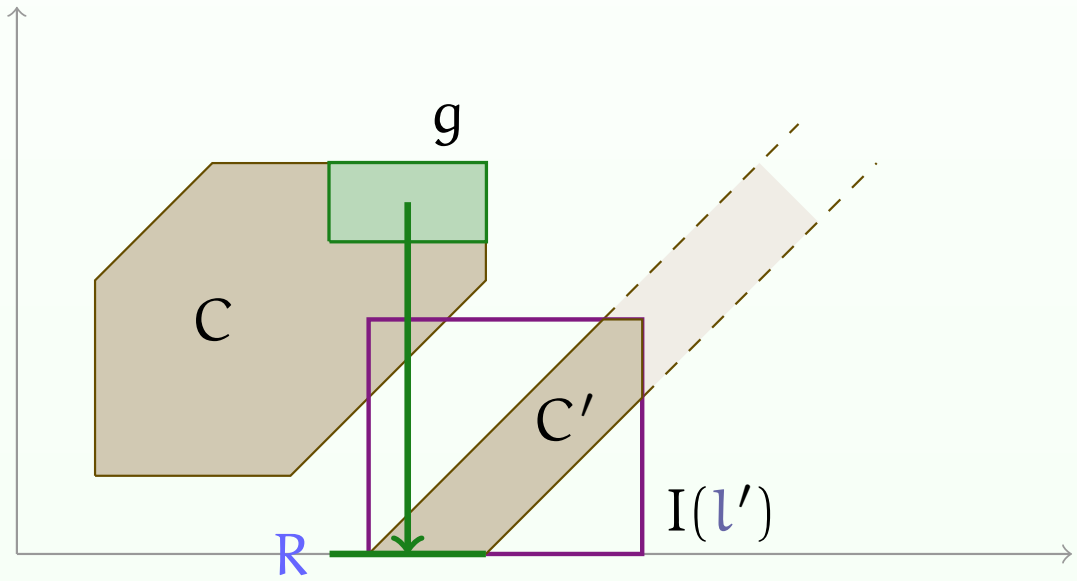
Symbolic semantics of PTA: Illustration

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

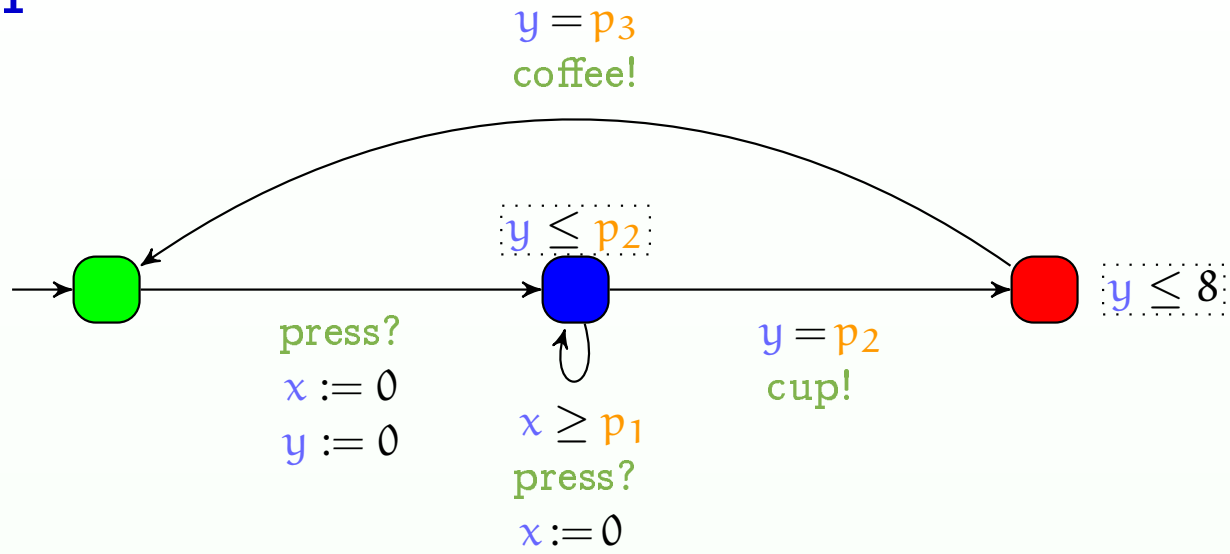


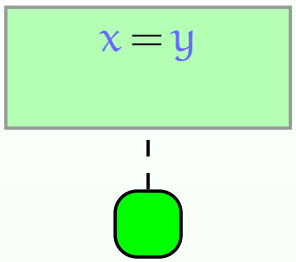
Symbolic semantics of PTA: Illustration

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

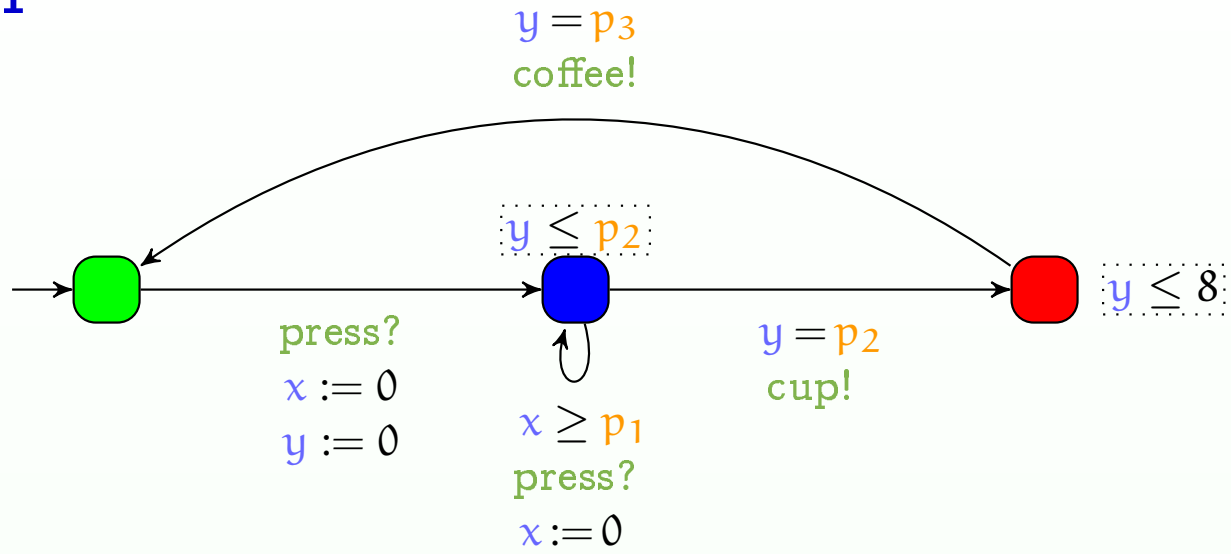


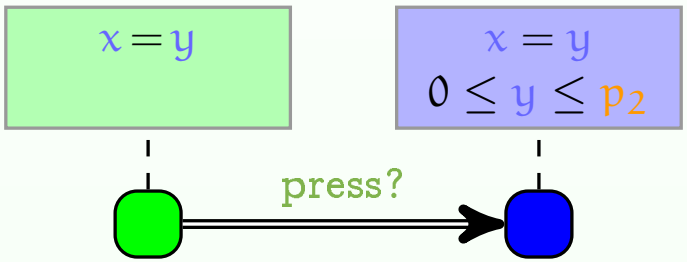
Symbolic exploration: Coffee machine



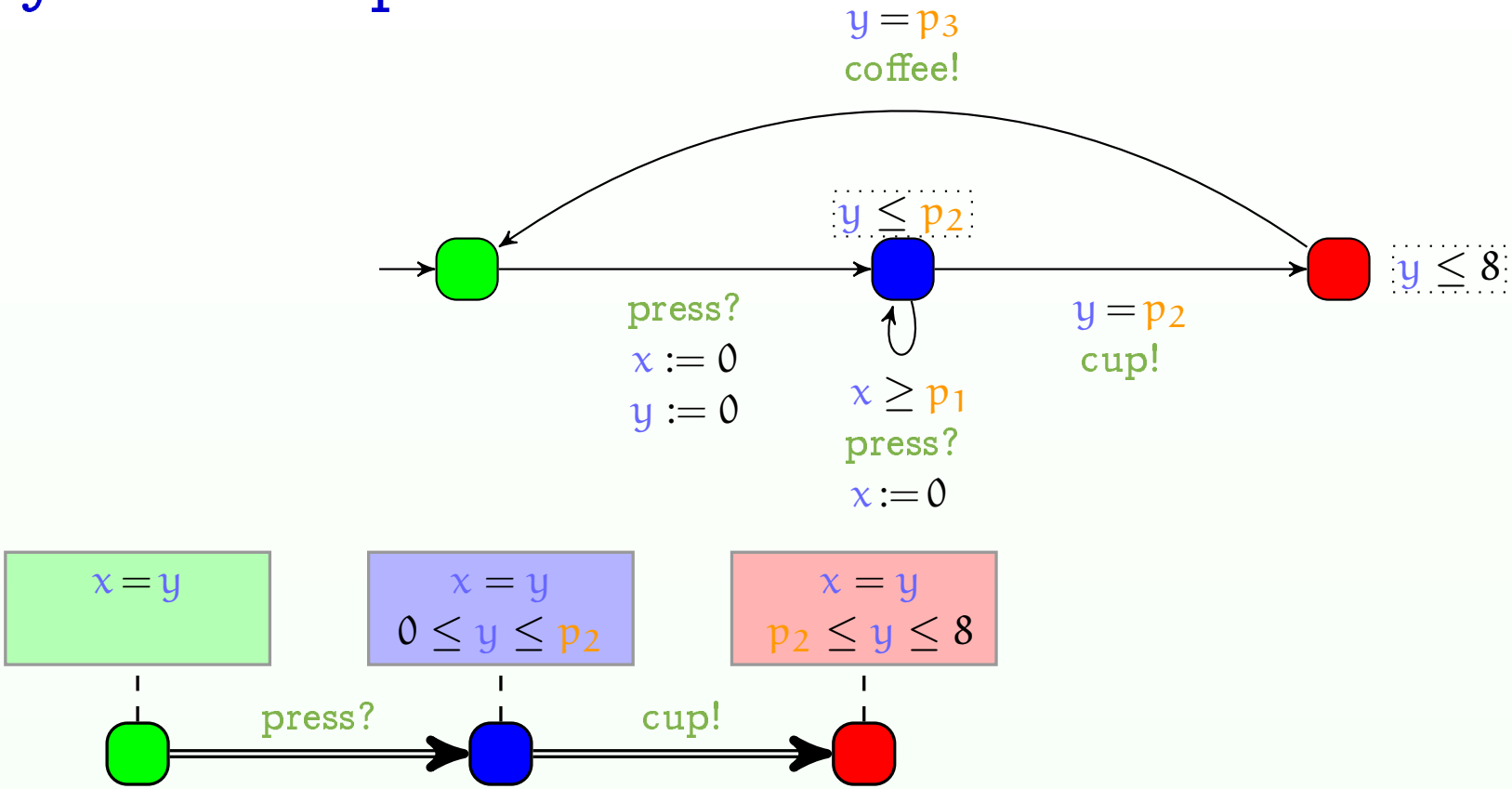


Symbolic exploration: Coffee machine

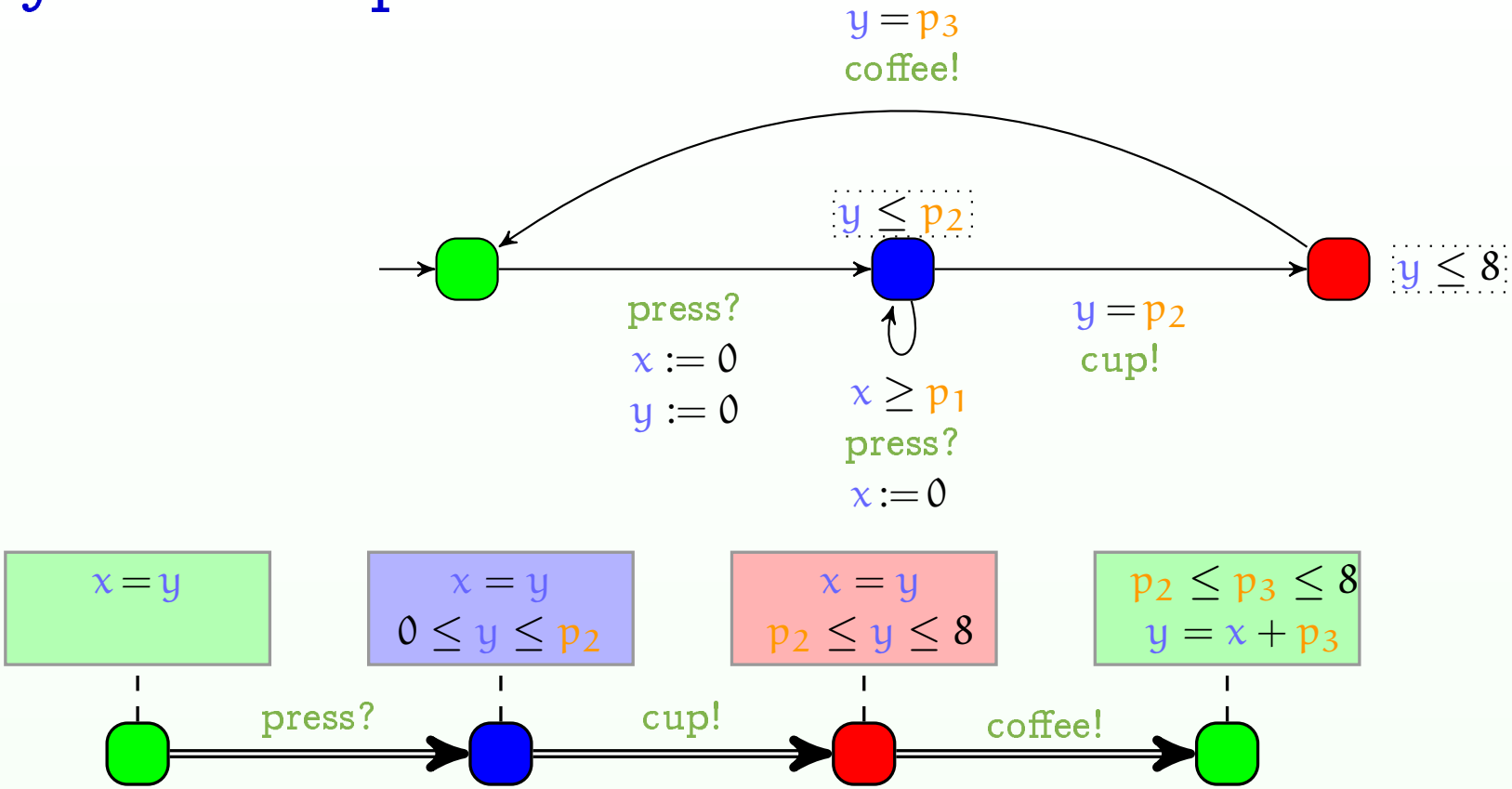




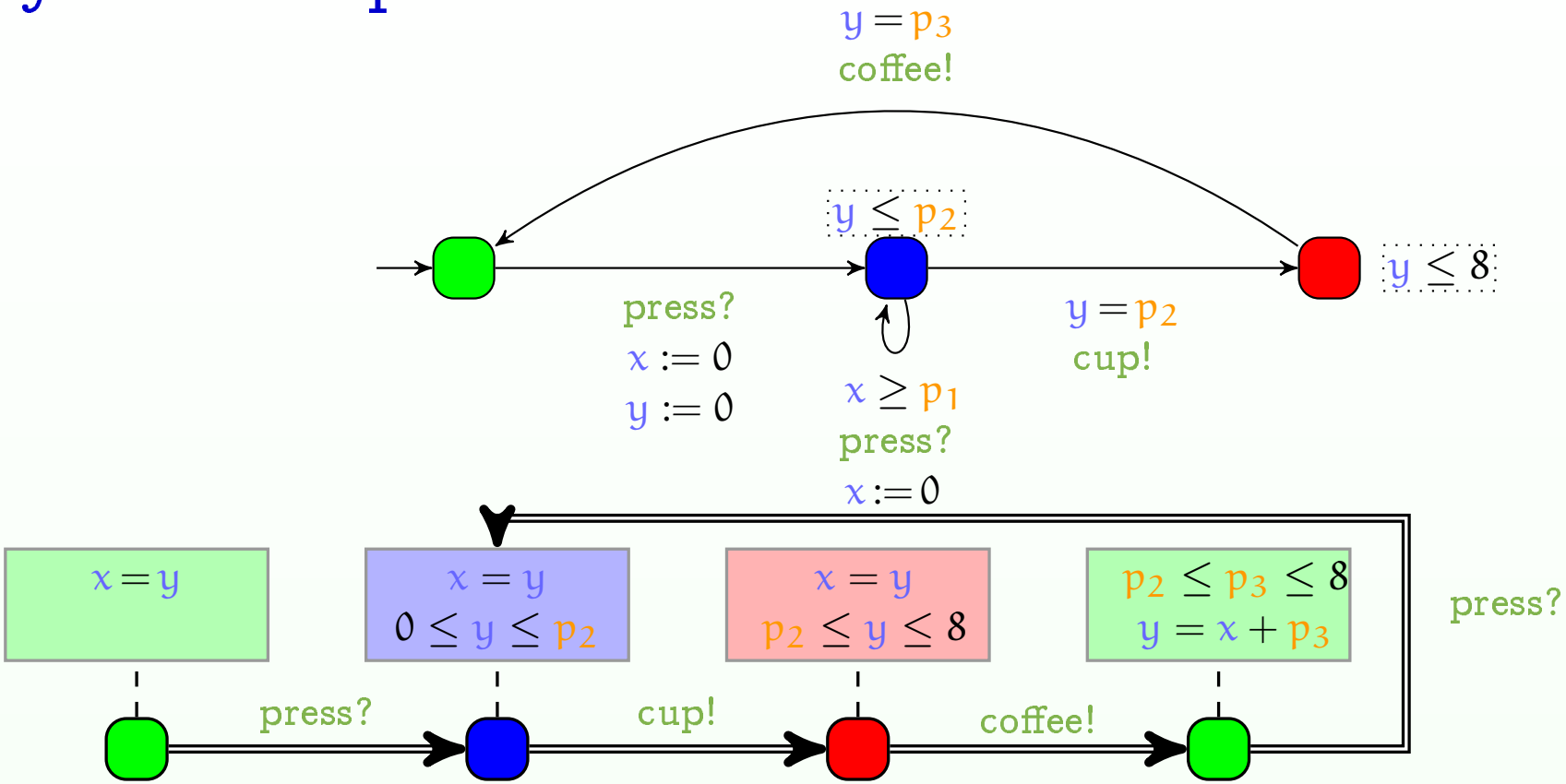
Symbolic exploration: Coffee machine



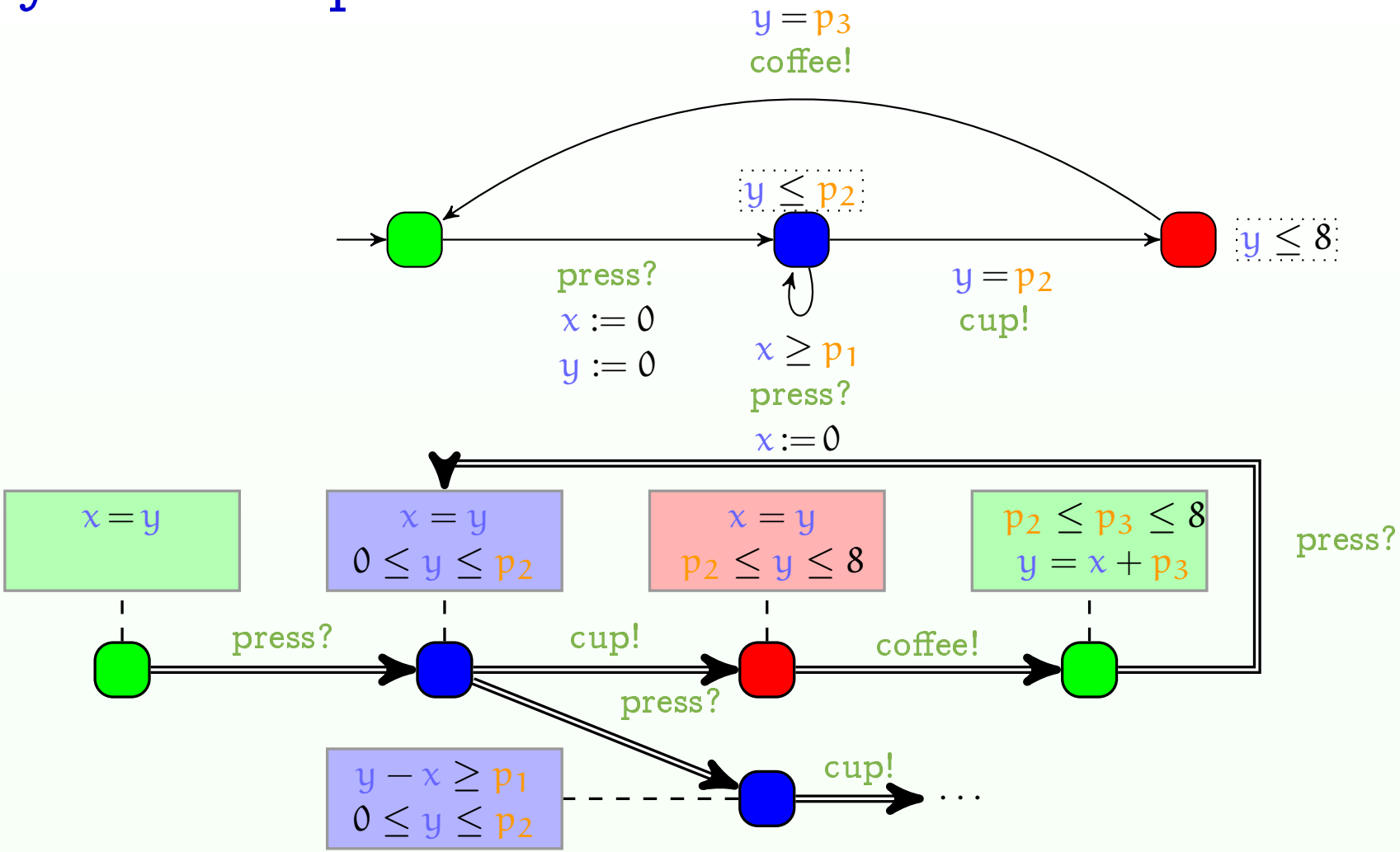
Symbolic exploration: Coffee machine



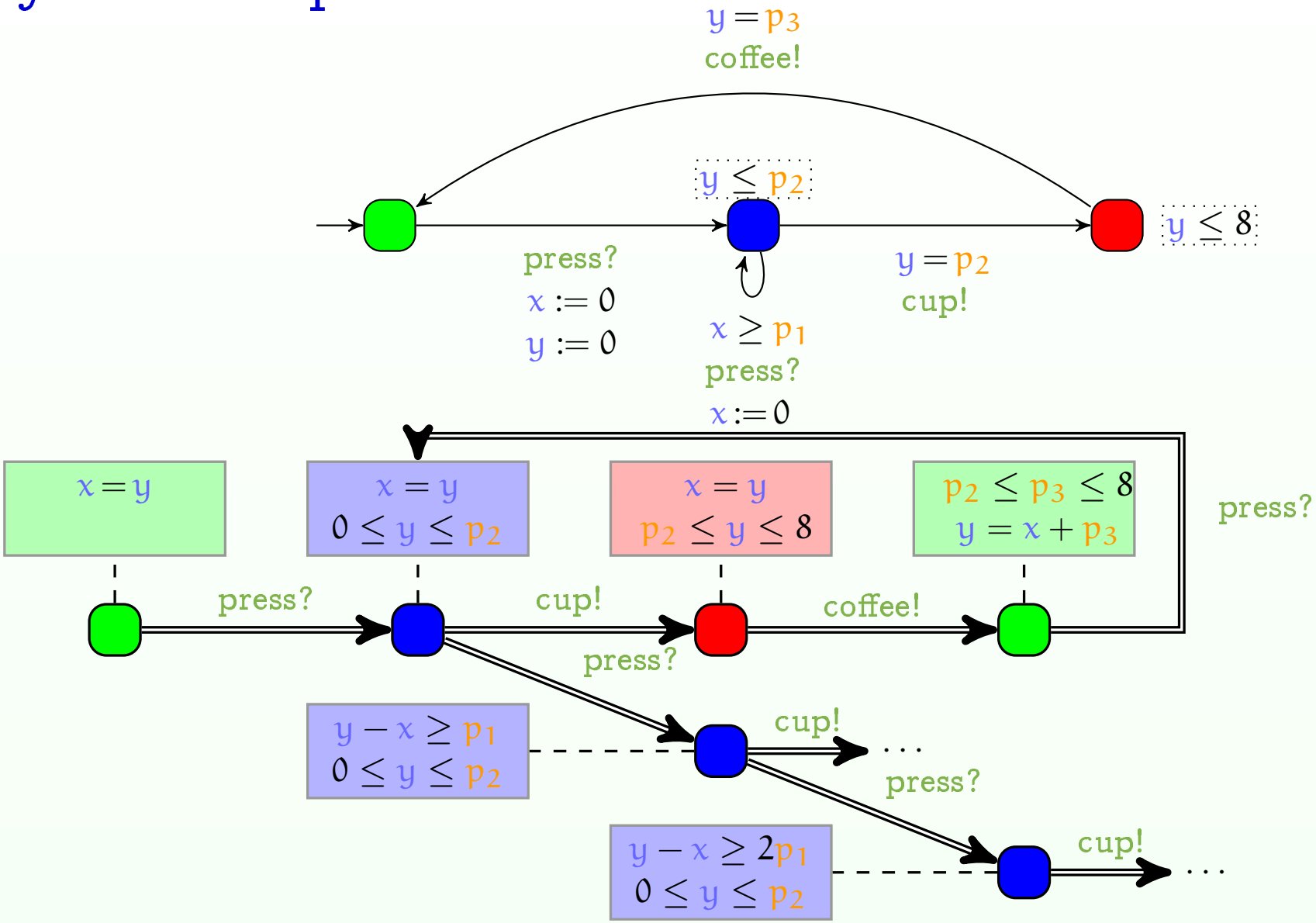
Symbolic exploration: Coffee machine



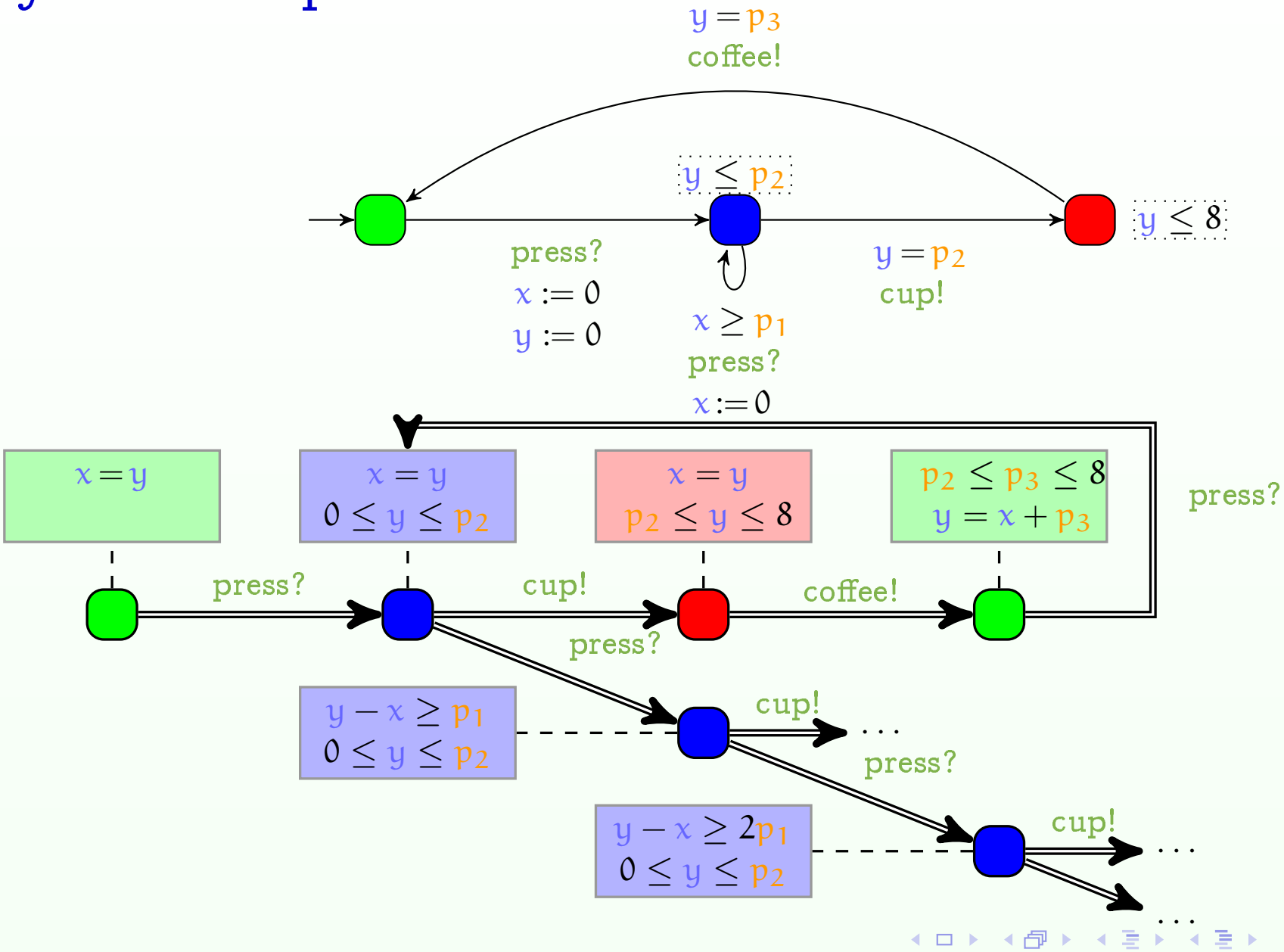
Symbolic exploration: Coffee machine



Symbolic exploration: Coffee machine



Symbolic exploration: Coffee machine



Plan: Decidability results

1 Parametric Timed Automata

2 Decidability results
- Definitions
- General case
- L/U-PTAs

3 Parameter synthesis

4 Conclusion

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?"

- Preservation of the untimed language "Given a parameter valuation, does there exist another valuation with the same untimed language?"

  Example: "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviours?"

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?"

- Preservation of the untimed language "Given a parameter valuation, does there exist another valuation with the same untimed language?"

  Example: "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviours?"

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?"

- Preservation of the untimed language "Given a parameter valuation, does there exist another valuation with the same untimed language?"

  Example: "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviours?"

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?"

- Preservation of the untimed language "Given a parameter valuation, does there exist another valuation with the same untimed language?"

  Example: "Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviours?"

Computation problems for PTA

- EF-Synthesis "Find all parameter valuations for which a given location l is reachable"

  Example: "What are all parameter valuations such that one may eventually get a coffee?"

- and so on

Computation problems for PTA

- EF-Synthesis "Find all parameter valuations for which a given location l is reachable"

  Example: "What are all parameter valuations such that one may eventually get a coffee?"

- and so on

Undecidability

- The symbolic state space is infinite in general
- No finite abstraction exists like for timed automata

Undecidability

- The symbolic state space is infinite in general
- No finite abstraction exists like for timed automata

Bad news

Decidability results for PTA (1/2)

- EF-emptiness problem

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

  undecidable [Alur et al., 1993, Beneš et al., 2015]

Decidability results for PTA (1/2)

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

- EF-universality problem
  "Do all parameter valuations allow to reach a given location l?"
  undecidable [André et al., 2016]

Decidability results for PTA (1/2)

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

- EF-universality problem
  "Do all parameter valuations allow to reach a given location l?"
  undecidable [André et al., 2016]

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

Decidability results for PTA (2/2)

- AF-universality problem

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

  undecidable [André et al., 2016]

Decidability results for PTA (2/2)

- AF-universality problem
  "Do all parameter valuations allow to reach a given location l for all runs?"
  undecidable [André et al., 2016]

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

Decidability results for PTA (2/2)

- AF-universality problem
  "Do all parameter valuations allow to reach a given location l for all runs?"
  undecidable [André et al., 2016]

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

In fact most interesting problems for PTAs are undecidable [André, 2016]

Limiting the number of clocks

Undecidability of EF-emptiness is achieved for a single parameter [Miller, 2000, Beneš et al., 2015]

However, reducing the number of clocks yields decidability of the EF-emptiness problem:

Limiting the number of clocks

Undecidability of EF-emptiness is achieved for a single parameter [Miller, 2000, Beneš et al., 2015]

However, 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]

Limiting the number of clocks

Undecidability of EF-emptiness is achieved for a single parameter [Miller, 2000, Beneš et al., 2015]

However, 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]

Limiting the number of clocks

Undecidability of EF-emptiness is achieved for a single parameter [Miller, 2000, Beneš et al., 2015]

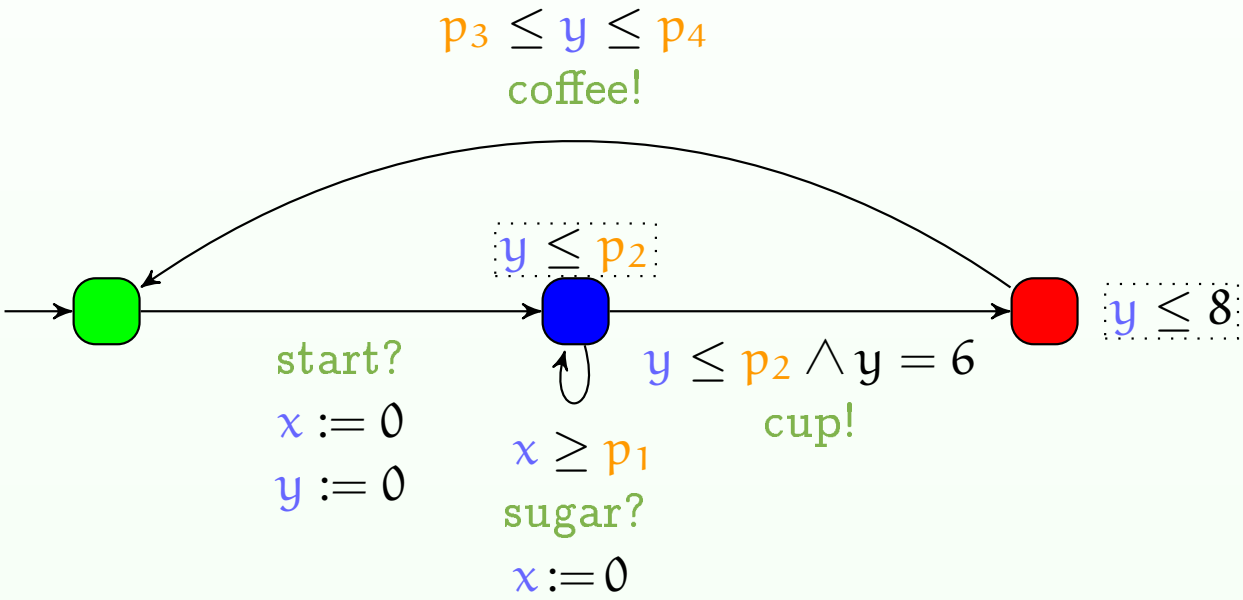
However, reducing the number of clocks yields decidability of the EF-emptiness problem:

- √ 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]

L/U-PTAs

Définition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter p is always compared with clocks as an upper bound or always as a lower bound.



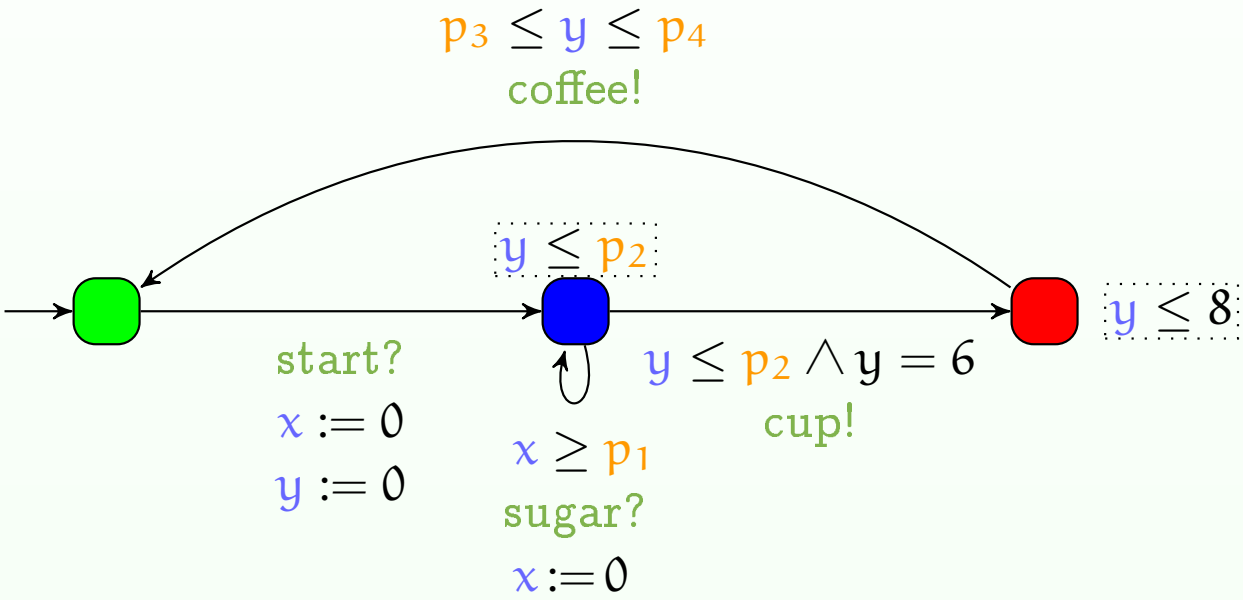
Lower-bound parameters:

Upped-bound parameters:

L/U-PTAs

Définition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter p is always compared with clocks as an upper bound or always as a lower bound.



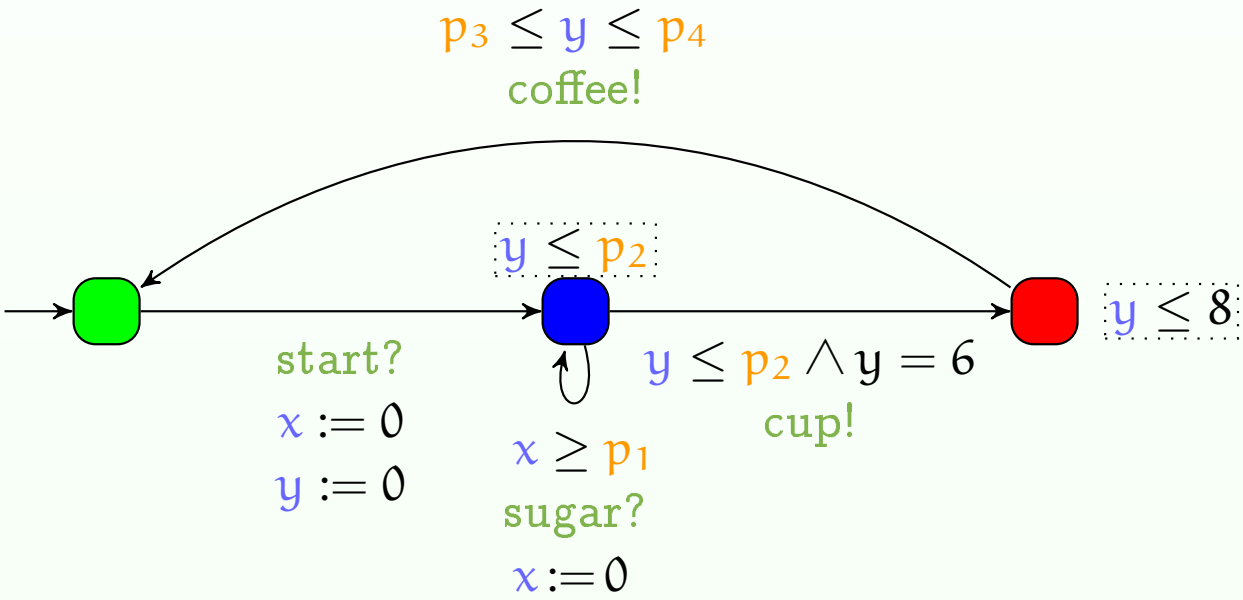
Lower-bound parameters:

Upped-bound parameters:

L/U-PTAs

Définition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter p is always compared with clocks as an upper bound or always as a lower bound.



Lower-bound parameters:

Upped-bound parameters:

Decidable problems for L/U-PTA

- EF-emptiness problem

  "Does there exist a parameter valuation for which a given location l is reachable?"

  decidable [Hune et al., 2002]

Decidable problems for L/U-PTA

- EF-emptiness problem
  "Does there exist a parameter valuation for which a given location l is reachable?"
  decidable [Hune et al., 2002]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location l?"
  decidable [Bozzelli and La Torre, 2009]

Decidable problems for L/U-PTA

- EF-emptiness problem
  "Does there exist a parameter valuation for which a given location l is reachable?"
  decidable [Hune et al., 2002]

- EF-universality problem
  "Do all parameter valuations allow to reach a given location l?"
  decidable [Bozzelli and La Torre, 2009]

- EF-finiteness problem
  "Is the set of parameter valuations allowing to reach a given location l finite?"
  decidable (for integer valuations) [Bozzelli and La Torre, 2009]

Undecidable problems for L/U-PTA

- AF-emptiness problem
  "Does there exist a parameter valuation for which a given location l is always eventually reachable?"
  undecidable [Jovanović et al., 2015]

Undecidable problems for L/U-PTA

- AF-emptiness problem
  "Does there exist a parameter valuation for which a given location l is always eventually reachable?"
  undecidable [Jovanović et al., 2015]

- AF-universality problem
  "Are all valuations such that a given location l is always eventually reachable?"
  undecidable (but...) [André and Lime, 2017]

Undecidable problems for L/U-PTA

- AF-emptiness problem
  "Does there exist a parameter valuation for which a given location l is always eventually reachable?"
  undecidable [Jovanović et al., 2015]

- AF-universality problem
  "Are all valuations such that a given location l is always eventually reachable?"
  undecidable (but...) [André and Lime, 2017]

- language preservation emptiness problem
  "Given a parameter valuation ν, can we find another valuation with the same untimed language?"
  undecidable [André and Markey, 2015]

What can we do with L/U-PTA?

In an L/U PTA, can we syntactically...

- use an equality ($=$) in a guard or invariant?

- use an equality $x = p$ in a guard or invariant?

What can we do with L/U-PTA?

In an L/U PTA, can we syntactically...

- use an equality ($=$) in a guard or invariant?
- use an equality $x = p$ in a guard or invariant?

What can we do with L/U-PTA?

In an L/U PTA, can we syntactically...

- use an equality ($=$) in a guard or invariant?
- use an equality $x = p$ in a guard or invariant?

What fits into the class of L/U-PTA?

- Any model with parametric delays given in the form of intervals
  - E.g.: $[p_{min}, p_{max}]$
- Many communication protocols
- All hardware circuits modeled using a bi-bounded inertial delay model

Plan: Parameter synthesis

1 Parametric Timed Automata

2 Decidability results

3 Parameter synthesis
- EF-synthesis
- AG-synthesis
- Untimed language preservation
- Software supporting PTA

4 Conclusion

The projection operator

Définition

Given a parametric zone C, we denote by $C{\downarrow_P}$ its projection onto the set P.

This can be achieved using variable elimination techniques (e.g., using Fourier-Motzkin).

The projection operator

Définition

Given a parametric zone C, we denote by $C{\downarrow_P}$ its projection onto the set P.
This can be achieved using variable elimination techniques (e.g., using Fourier-Motzkin).

Example:
$(2 < x \leq p_1 \wedge p_1 \leq p_2){\downarrow_P}$
$=$

The projection operator

Définition

Given a parametric zone C, we denote by $C{\downarrow_P}$ its projection onto the set P.
This can be achieved using variable elimination techniques (e.g., using Fourier-Motzkin).

Example:
$(2 < x \leq p_1 \wedge p_1 \leq p_2){\downarrow_P}$
$=$

The projection operator: examples

$(x_2 = 1 \wedge x_1 > x_2 \wedge x_1 \leq p_1 \wedge p_2 = 4)\downarrow_P$

$=$

The projection operator: examples

$(x_2 = 1 \wedge x_1 > x_2 \wedge x_1 \leq p_1 \wedge p_2 = 4)\downarrow_P$

$=$

The projection operator: examples

$(x_2 = 1 \wedge x_1 > x_2 \wedge x_1 \leq p_1 \wedge p_2 = 4)\downarrow_P$
$=$

$(x_1 < x_2 \wedge x_1 = x_3 \wedge x_1 > p_2 \wedge p_1 = p_2)\downarrow_P$
$=$

The projection operator: examples

$(x_2 = 1 \wedge x_1 > x_2 \wedge x_1 \leq p_1 \wedge p_2 = 4)\downarrow_P$
$=$

$(x_1 < x_2 \wedge x_1 = x_3 \wedge x_1 > p_2 \wedge p_1 = p_2)\downarrow_P$
$=$

The projection operator: examples

$(x_2 = 1 \wedge x_1 > x_2 \wedge x_1 \leq p_1 \wedge p_2 = 4)\downarrow_P$
$=$

$(x_1 < x_2 \wedge x_1 = x_3 \wedge x_1 > p_2 \wedge p_1 = p_2)\downarrow_P$
$=$

$(x_1 < x_2 \wedge x_1 < x_3 \wedge x_1 < p_2 \wedge p_1 = x_3)\downarrow_P$
$=$

The projection operator: examples

$(x_2 = 1 \wedge x_1 > x_2 \wedge x_1 \leq p_1 \wedge p_2 = 4)\downarrow_P$
$=$

$(x_1 < x_2 \wedge x_1 = x_3 \wedge x_1 > p_2 \wedge p_1 = p_2)\downarrow_P$
$=$

$(x_1 < x_2 \wedge x_1 < x_3 \wedge x_1 < p_2 \wedge p_1 = x_3)\downarrow_P$
$=$

Restriction of the parametric constraint

Théorème

Let $(l, C) \stackrel{a}{\Longrightarrow} (l', C')$.
Then $C'{\downarrow_P} \subseteq C{\downarrow_P}$

Meaning:

Proof:

Corollary (cycles):

Reachability condition

Théorème

Let (l, C) be a symbolic state of a PTA $\mathcal{A}$.

Let v be a parameter valuation.

Then l is reachable in $v(\mathcal{A})$ if and only if $v \models C\downarrow_P$.

Meaning:

EF-synthesis: Reachability

Input : a PTA $\mathcal{A}$, a set G of locations of $\mathcal{A}$

Output : a constraint K over the parameters such that, for any $v \models K$, then at least one of the locations of G is reachable in $v(\mathcal{A})$

EF-synthesis: An algorithm

Algorithm EFsynth

input : A PTA $\mathcal{A}$, a symbolic state $s = (l, C)$, a set of target locations G, a set S of passed states on the current path

output :

EF-synthesis: An algorithm

Algorithm EFsynth

input : A PTA $\mathcal{A}$, a symbolic state $s = (l, C)$, a set of target locations G, a set S of passed states on the current path

output :

1 **if** $l \in G$ **then**

EF-synthesis: An algorithm

Algorithm EFsynth

input : A PTA $\mathcal{A}$, a symbolic state $s = (l, C)$, a set of target locations G, a set S of passed states on the current path

output :

```
1 if l ∈ G then                    ;
2 else
3 |   K ← false;
4 |   if s ∉ S then
5 |   |   for each outgoing e from l in 𝒜 do
  |   |   |
```

EF-synthesis: An algorithm

Algorithm EFsynth

input : A PTA $\mathcal{A}$, a symbolic state $s = (l, C)$, a set of target locations G, a set S of passed states on the current path

output :

```
1 if l ∈ G then                    ;
2 else
3     K ← false;
4     if s ∉ S then
5         for each outgoing e from l in 𝒜 do
              
7 return
```

EF-synthesis: An algorithm

Algorithm EFsynth

input : A PTA $\mathcal{A}$, a symbolic state $s = (l, C)$, a set of target locations G, a set S of passed states on the current path

output :

```
1 if l ∈ G then                    ;
2 else
3     K ← false;
4     if s ∉ S then
5         for each outgoing e from l in 𝒜 do
              
7 return
```

EF-synthesis: Correctness and completeness

Théorème ([Jovanović et al., 2015])

Assume EFsynth

EF-synthesis: Correctness and completeness

Théorème ([Jovanović et al., 2015])

Assume EFsynth terminates with result K.
Then

EF-synthesis: Correctness and completeness

Théorème ([Jovanović et al., 2015])

Assume EFsynth terminates with result K.
Then

Now, what happens if EFsynth does not terminate?

AG-synthesis: Global invariant

Input : a PTA $\mathcal{A}$, a set G of locations of $\mathcal{A}$

Output : a constraint K over the parameters such that, for any $\nu \models K$, then all runs permanently stay within G in $\nu(\mathcal{A})$

AG-synthesis: First method

Idea: use EFsynth!

AG-synthesis: First method

Idea: use EFsynth!

Drawback:

- EFsynth must terminate

AG-synthesis: Second method

Algorithm AGsynth

input : A PTA $\mathcal{A}$, a symbolic state $s = (l, C)$, a set of target locations G, a set S of passed states on the current path

output :

AG-synthesis: Second method

Algorithm AGsynth

---

input : A PTA $\mathcal{A}$, a symbolic state $s = (l, C)$, a set of target locations G, a set S of passed states on the current path

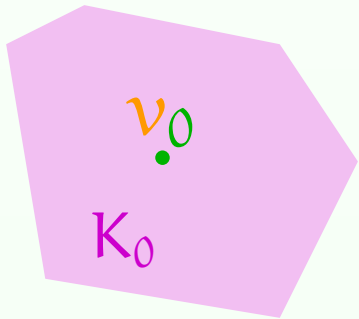
output :

The untimed language preservation problem

- Input
  - A PTA $\mathcal{A}$
  - A reference valuation v_0

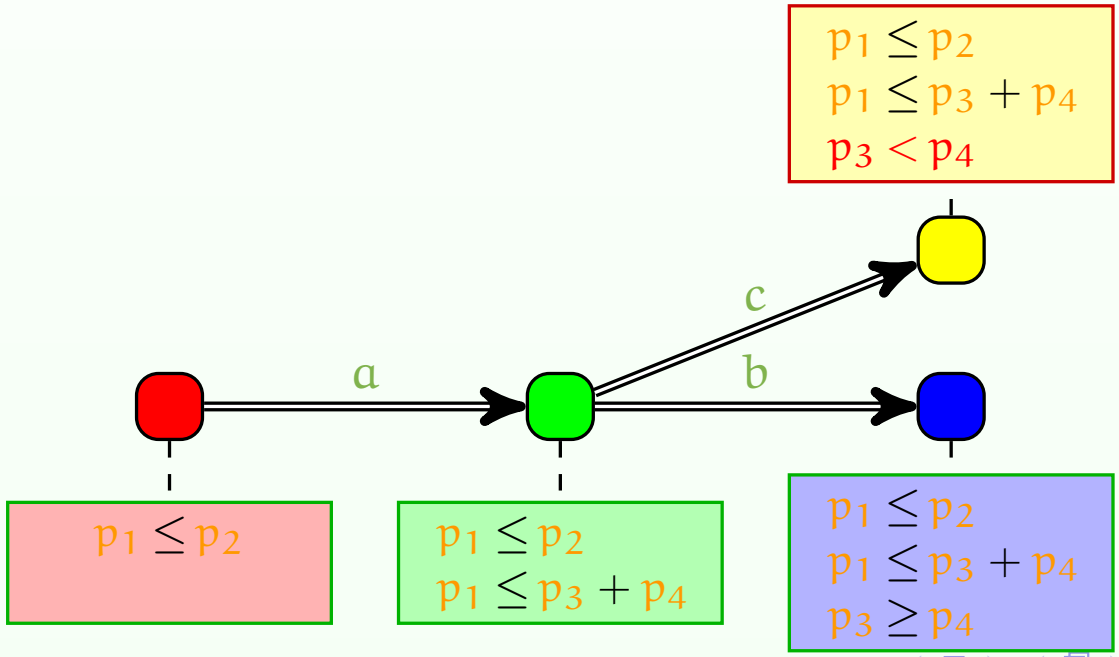
The untimed language preservation problem

- Input
  - A PTA $\mathcal{A}$
  - A reference valuation v_0
- Output: constraint K_0
  - Constraint on the parameters such that
    - $v_0 \models K_0$
    - For all points $v \models K_0$, $v(\mathcal{A})$ and $v_0(\mathcal{A})$ have the same untimed language (hence, same untimed behavior)



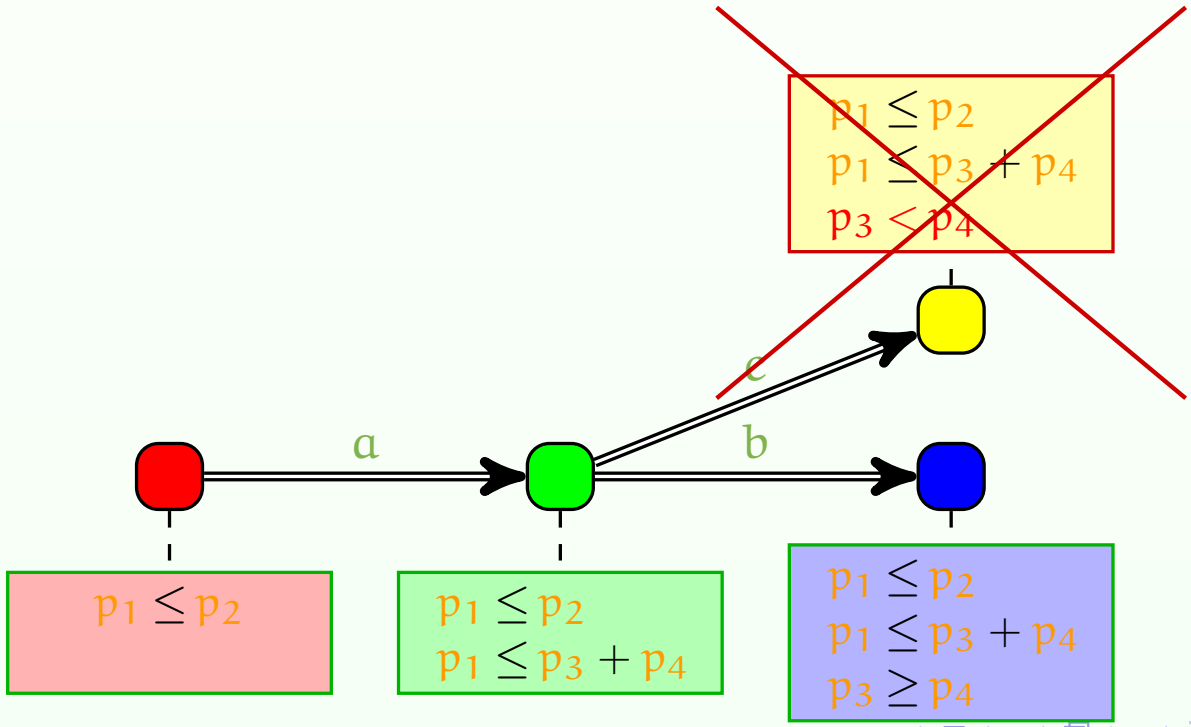
Trace preservation synthesis TPsynth: General idea

- The idea [André et al., 2009]
  - CEGAR-like approach
    - CEGAR: counter-example guided abstraction refinement
  - Instead of negating bad states, we remove ν_0-incompatible states



Trace preservation synthesis TPsynth: General idea

- The idea [André et al., 2009]
  - CEGAR-like approach
    - CEGAR: counter-example guided abstraction refinement
  - Instead of negating bad states, we remove ν_0-incompatible states



TPsynth ("inverse method"): Simplified algorithm

Start with $K_0 = \texttt{true}$

REPEAT

1. Compute a set S of reachable symbolic states under K_0
2. Refine K_0 by removing a ν_0-incompatible state from S
   - Select a ν_0-incompatible state (l, C) within S (i.e., $\nu_0 \not\models C$)
   - Add

TPsynth ("inverse method"): Simplified algorithm

Start with $K_0 = \texttt{true}$

REPEAT

1. Compute a set S of reachable symbolic states under K_0
2. Refine K_0 by removing a ν_0-incompatible state from S
   - Select a ν_0-incompatible state (l, C) within S (i.e., $\nu_0 \not\models C$)
   - Add to K_0

UNTIL no more ν_0-incompatible state in S

RETURN

TPsynth ("inverse method"): Simplified algorithm

Start with $K_0 = \texttt{true}$

REPEAT

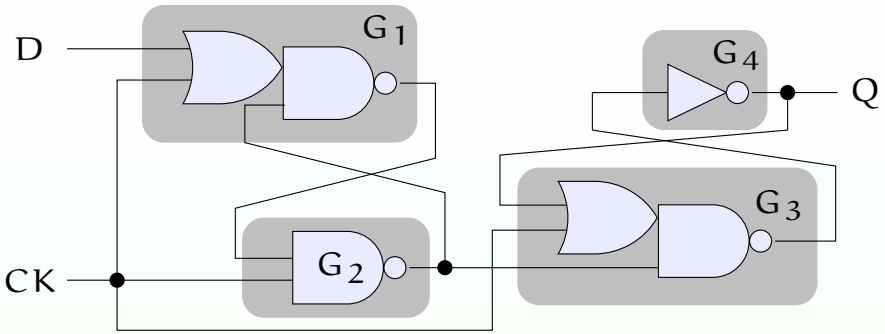
1. Compute a set S of reachable symbolic states under K_0
2. Refine K_0 by removing a ν_0-incompatible state from S
   - Select a ν_0-incompatible state (l, C) within S (i.e., $\nu_0 \not\models C$)
   - Add to K_0

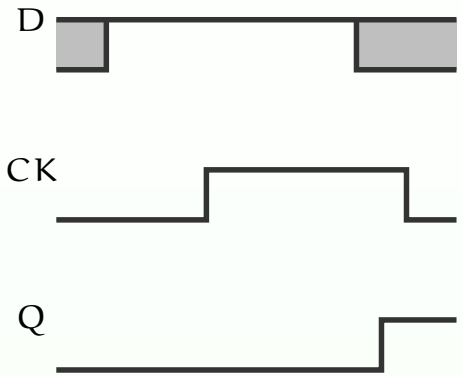
UNTIL no more ν_0-incompatible state in S

RETURN

An example of flip-flop circuit

- An asynchronous circuit [Clarisó and Cortadella, 2007]



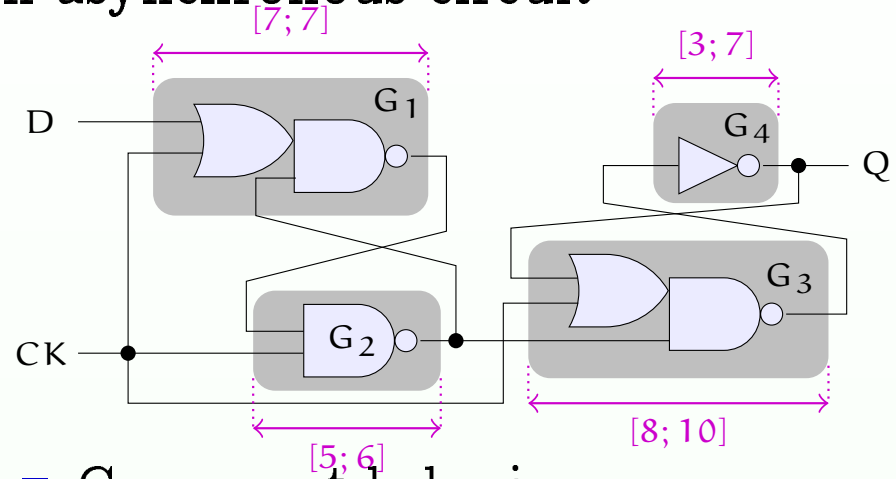


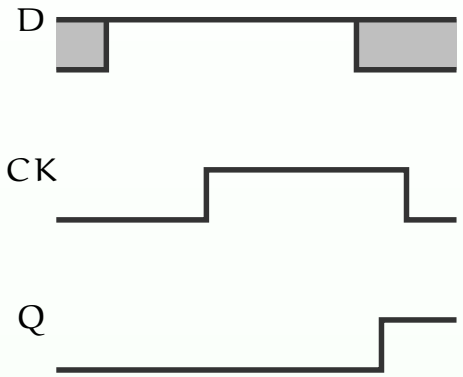
  - Concurrent behavior
    - 4 elements: G_1, G_2, G_3, G_4
    - 2 input signals (D and CK), 1 output signal (Q)

An example of flip-flop circuit

- An asynchronous circuit

[Clarisó and Cortadella, 2007]



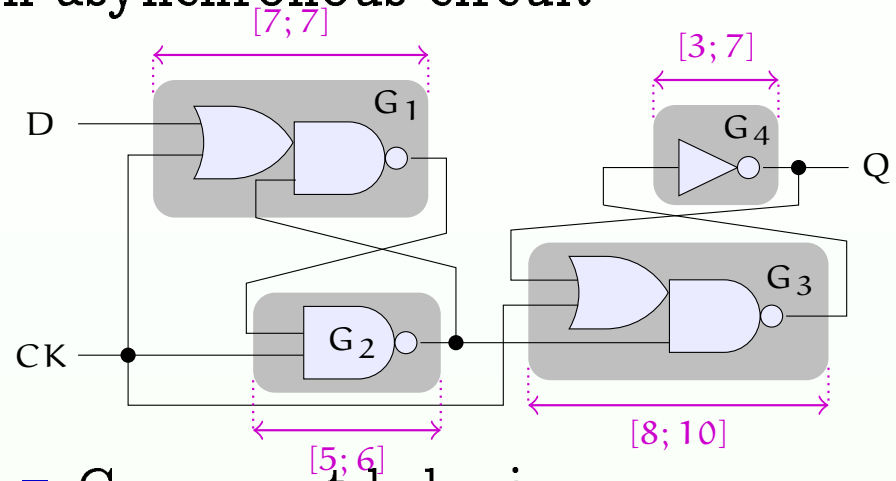


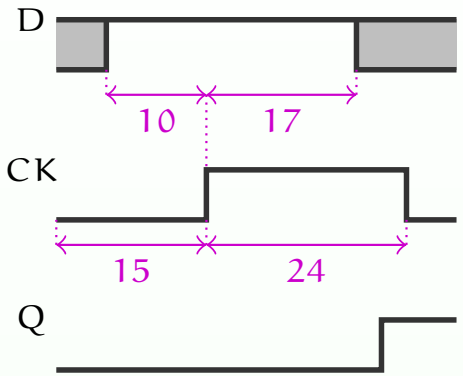
- Concurrent behavior
  - 4 elements: G_1, G_2, G_3, G_4
  - 2 input signals (D and CK), 1 output signal (Q)
- Timing delays
  - Traversal delays of the gates: one interval per gate

An example of flip-flop circuit

- An asynchronous circuit

[Clarisó and Cortadella, 2007]

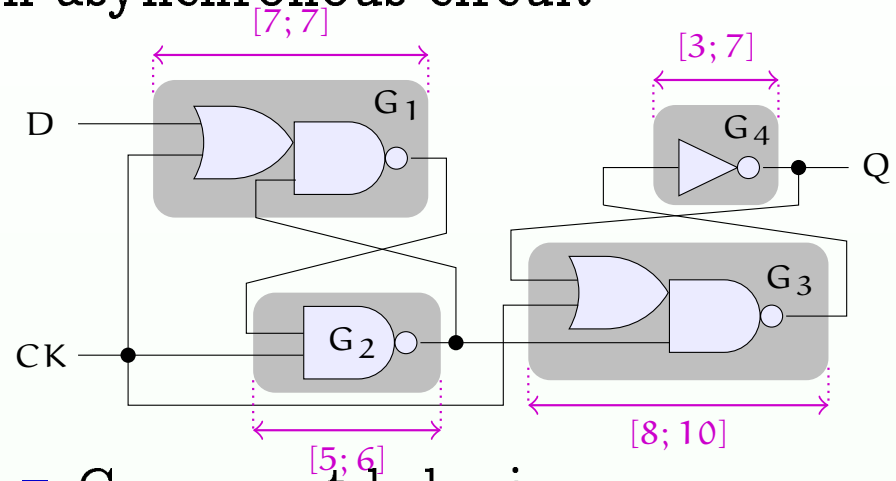


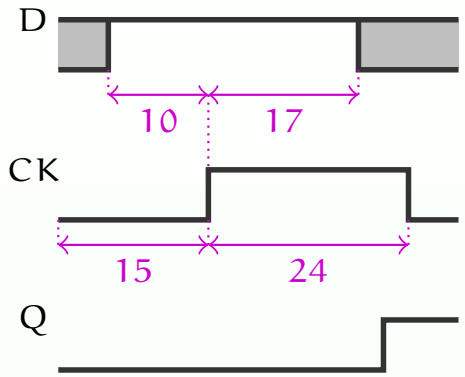


- Concurrent behavior
  - 4 elements: G_1, G_2, G_3, G_4
  - 2 input signals (D and CK), 1 output signal (Q)
- Timing delays
  - Traversal delays of the gates: one interval per gate
  - Environment timing constants

An example of flip-flop circuit

- An asynchronous circuit [Clarisó and Cortadella, 2007]



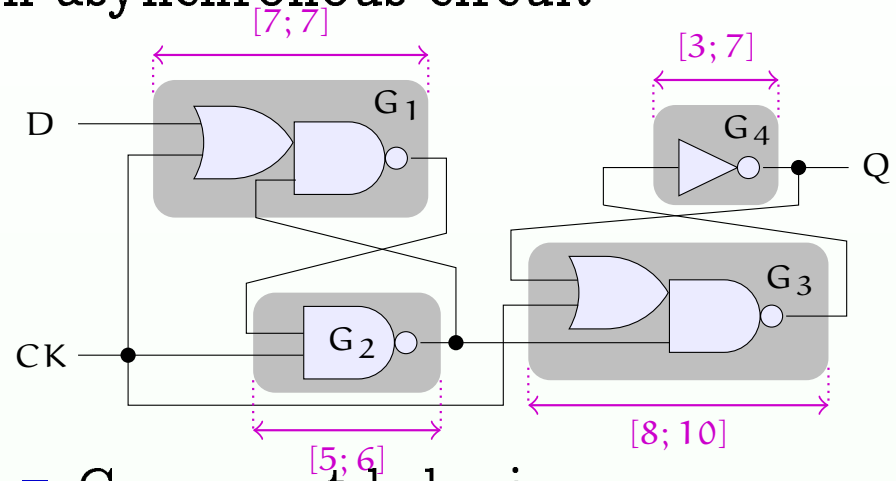


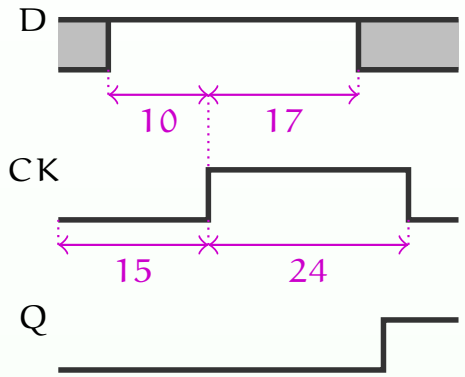
  - Concurrent behavior
    - 4 elements: G_1, G_2, G_3, G_4
    - 2 input signals (D and CK), 1 output signal (Q)
  - Timing delays
    - Traversal delays of the gates: one interval per gate
    - Environment timing constants

- Question
  - For these timing delays, does the rise of Q always occur before the fall of CK?

An example of flip-flop circuit

- An asynchronous circuit [Clarisó and Cortadella, 2007]

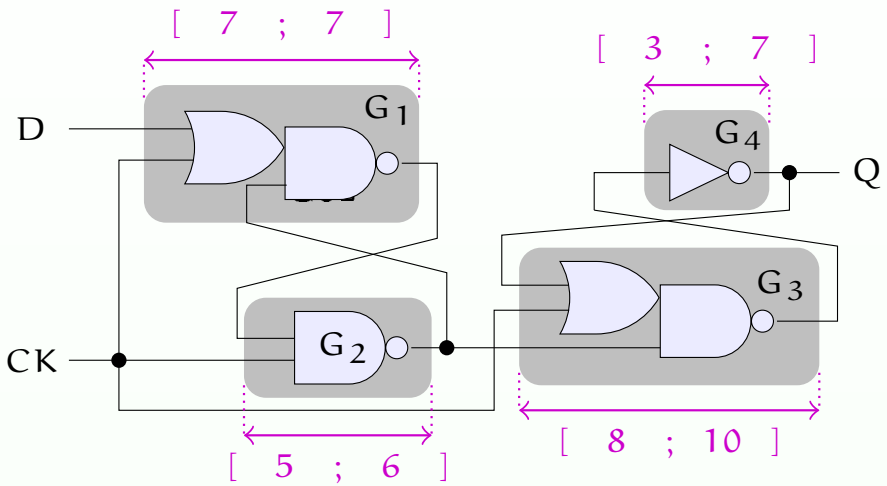


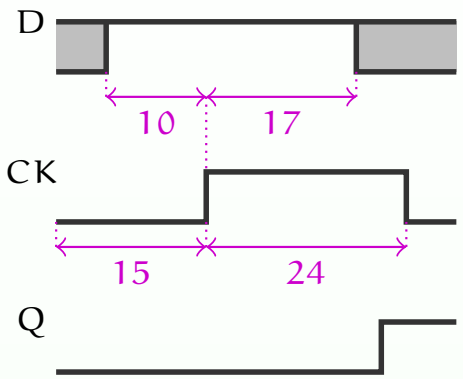


  - Concurrent behavior
    - 4 elements: G_1, G_2, G_3, G_4
    - 2 input signals (D and CK), 1 output signal (Q)
  - Timing delays
    - Traversal delays of the gates: one interval per gate
    - Environment timing constants

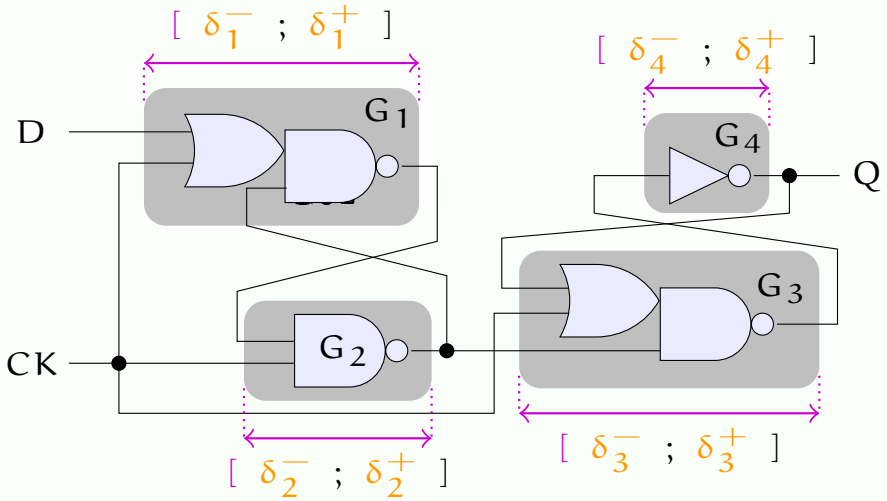
- Question
  - For these timing delays, does the rise of Q always occur before the fall of CK?
  - Timed model checking gives the answer: yes

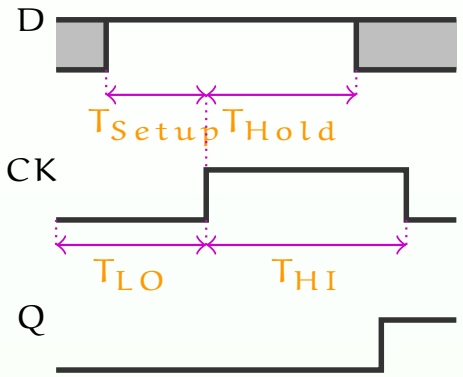
Flip-flop circuit: Timing parameters





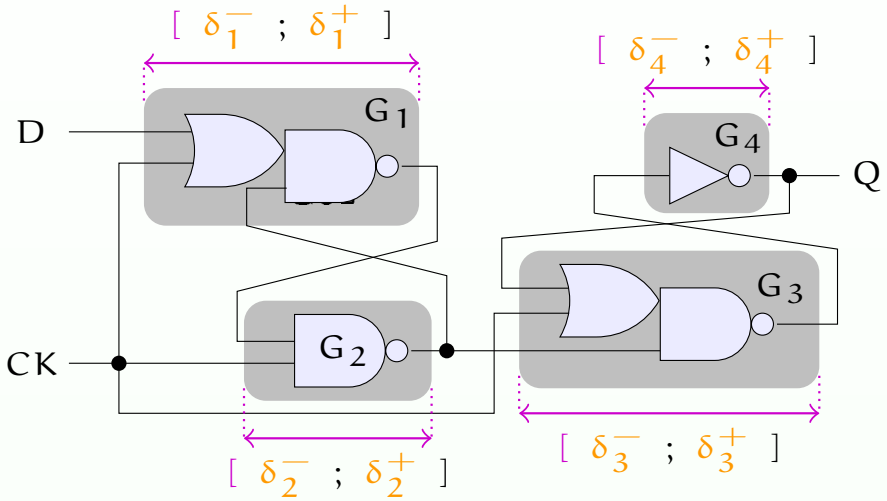
Flip-flop circuit: Timing parameters

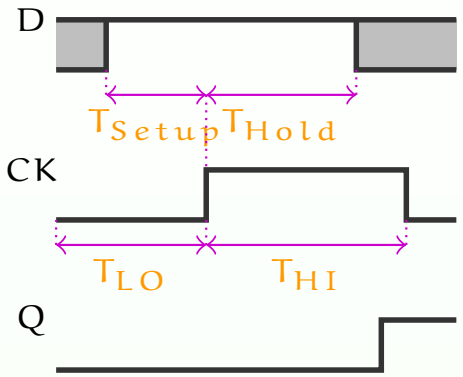




- Timing parameters
  - Traversal delays of the gates: one interval per gate
  - 4 environment parameters: T_{LO}, T_{HI}, T_{Setup} and T_{Hold}

Flip-flop circuit: Timing parameters



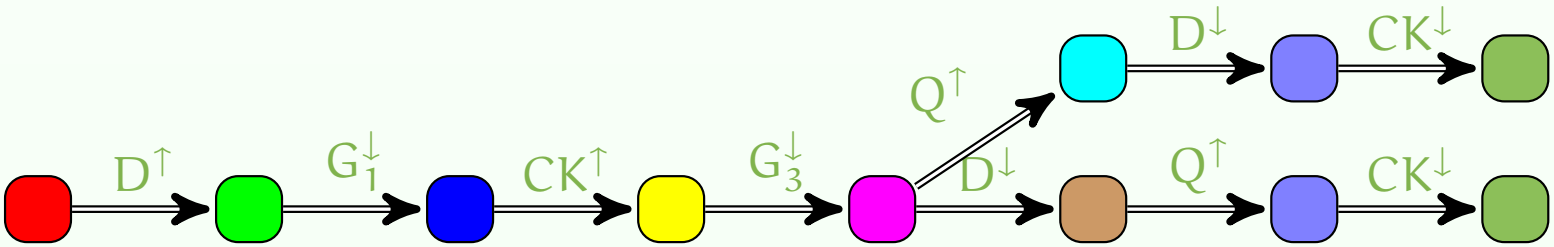


- Timing parameters
  - Traversal delays of the gates: one interval per gate
  - 4 environment parameters: T_{LO}, T_{HI}, T_{Setup} and T_{Hold}

- Question: for which values of the parameters does the rise of Q always occur before the fall of CK?

Untimed language: Trace set

- Trace set: set of all traces of a PTA
- Graphical representation under the form of a tree
  - Does not give any information on the branching behavior though
  - Example of trace set for the flip-flop example
- Example: trace set of the flip-flop circuit for the original valuation ν_0



Application of TPsynth to the flip-flop circuit

ν_0 :

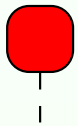
$\delta_1^- = 7$ $\quad \delta_1^+ = 7$ $\quad T_{HI} = 24$

$\delta_2^- = 5$ $\quad \delta_2^+ = 6$ $\quad T_{LO} = 15$

$\delta_3^- = 8$ $\quad \delta_3^+ = 10$ $\quad T_{Setup} = 10$

$\delta_4^- = 3$ $\quad \delta_4^+ = 7$ $\quad T_{Hold} = 17$

$K_0 = \texttt{true}$



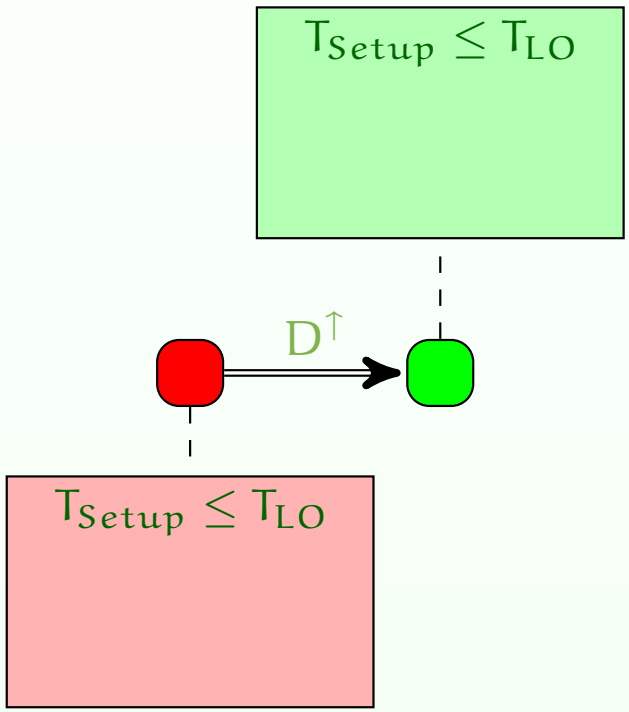
$T_{Setup} \leq T_{LO}$

Application of TPsynth to the flip-flop circuit

ν_0 :

$\delta_1^- = 7$	$\delta_1^+ = 7$	$T_{HI} = 24$
$\delta_2^- = 5$	$\delta_2^+ = 6$	$T_{LO} = 15$
$\delta_3^- = 8$	$\delta_3^+ = 10$	$T_{Setup} = 10$
$\delta_4^- = 3$	$\delta_4^+ = 7$	$T_{Hold} = 17$

$K_0 = \texttt{true}$

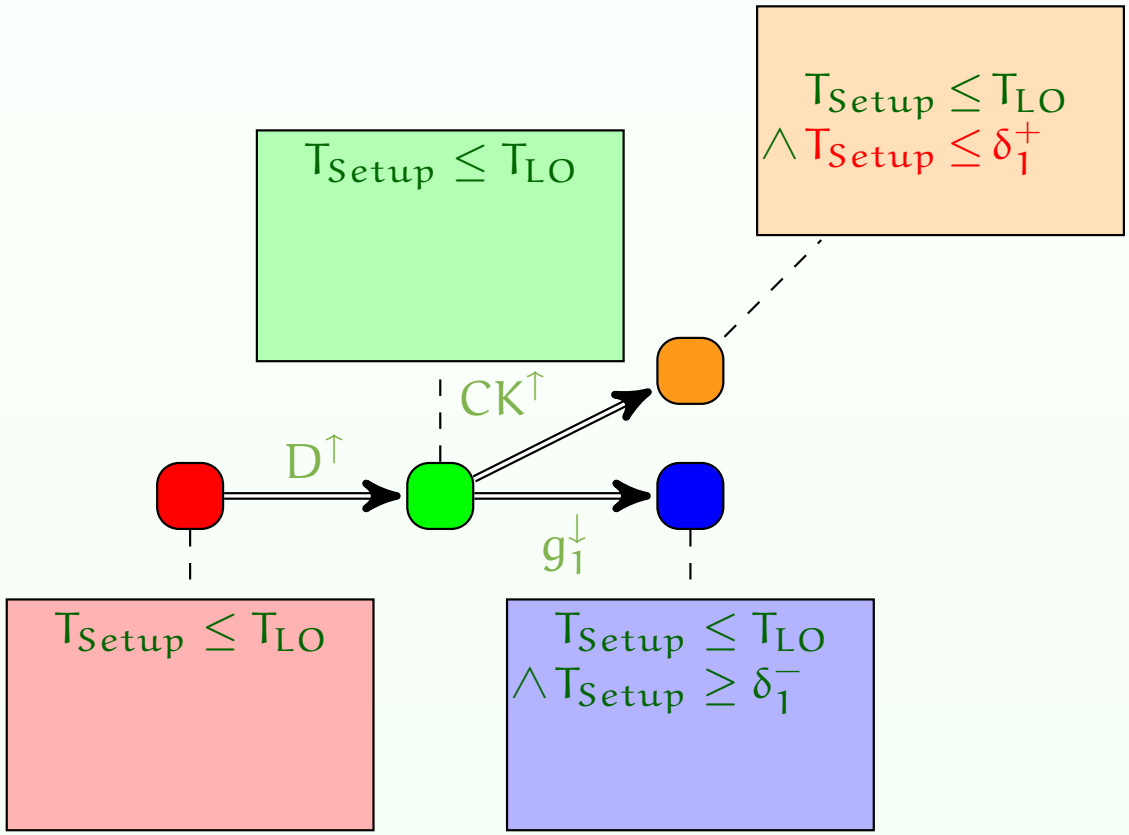


Application of TPsynth to the flip-flop circuit

ν_0 :

$\delta_1^- = 7$	$\delta_1^+ = 7$	$T_{HI} = 24$
$\delta_2^- = 5$	$\delta_2^+ = 6$	$T_{LO} = 15$
$\delta_3^- = 8$	$\delta_3^+ = 10$	$T_{Setup} = 10$
$\delta_4^- = 3$	$\delta_4^+ = 7$	$T_{Hold} = 17$

$K_0 = \texttt{true}$

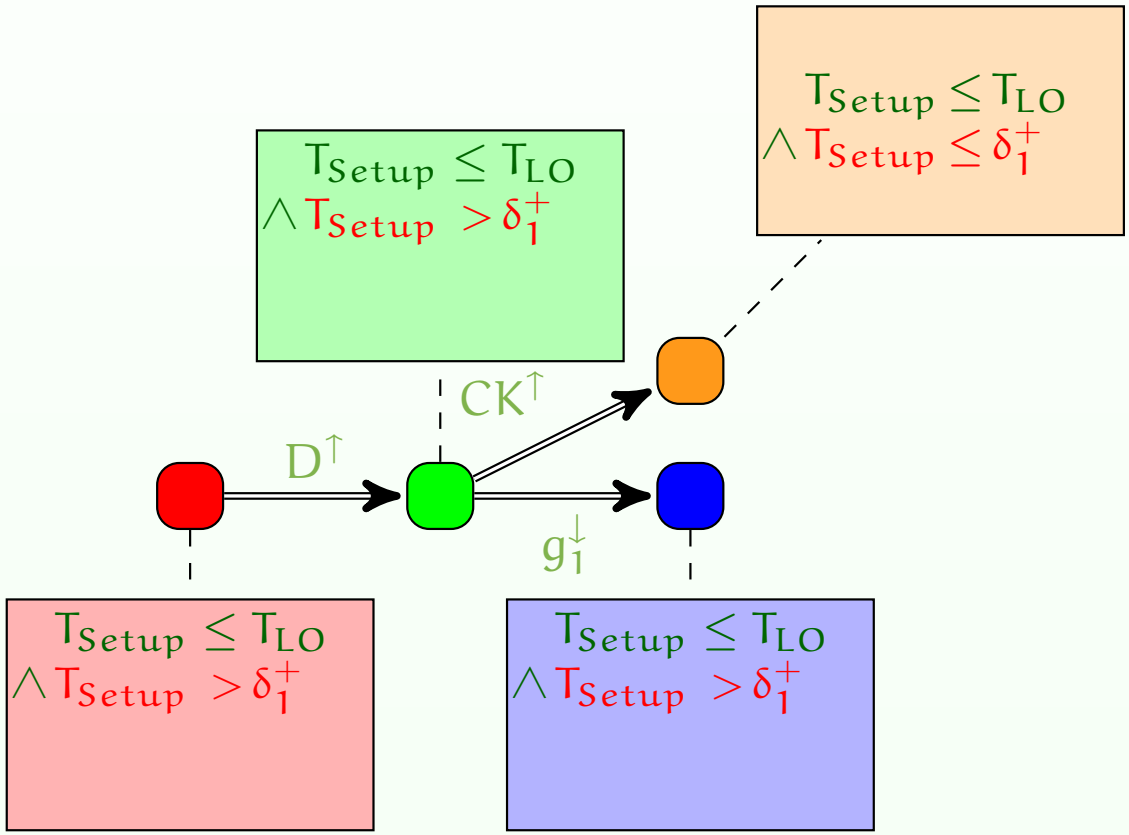


Application of TPsynth to the flip-flop circuit

ν_0 :

$\delta_1^- = 7$	$\delta_1^+ = 7$	$T_{HI} = 24$
$\delta_2^- = 5$	$\delta_2^+ = 6$	$T_{LO} = 15$
$\delta_3^- = 8$	$\delta_3^+ = 10$	$T_{Setup} = 10$
$\delta_4^- = 3$	$\delta_4^+ = 7$	$T_{Hold} = 17$

$K_0 = T_{Setup} > \delta_1^+$



Application of TPsynth to the flip-flop circuit

ν_0 :

$\delta_1^- = 7$ $\quad \delta_1^+ = 7$ $\quad T_{HI} = 24$

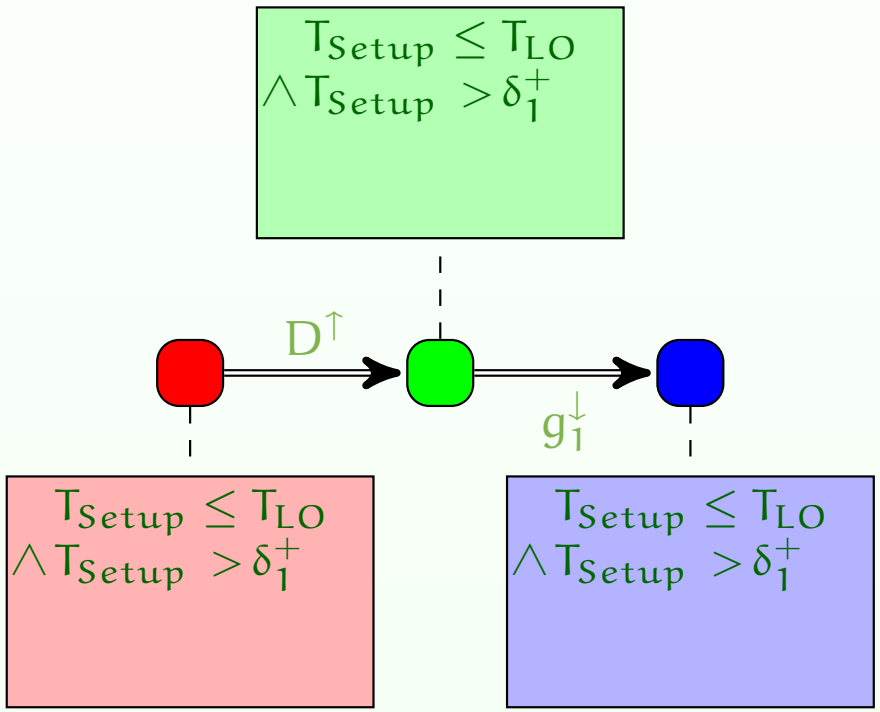
$\delta_2^- = 5$ $\quad \delta_2^+ = 6$ $\quad T_{LO} = 15$

$\delta_3^- = 8$ $\quad \delta_3^+ = 10$ $\quad T_{Setup} = 10$

$\delta_4^- = 3$ $\quad \delta_4^+ = 7$ $\quad T_{Hold} = 17$

$K_0 =$

$T_{Setup} > \delta_1^+$

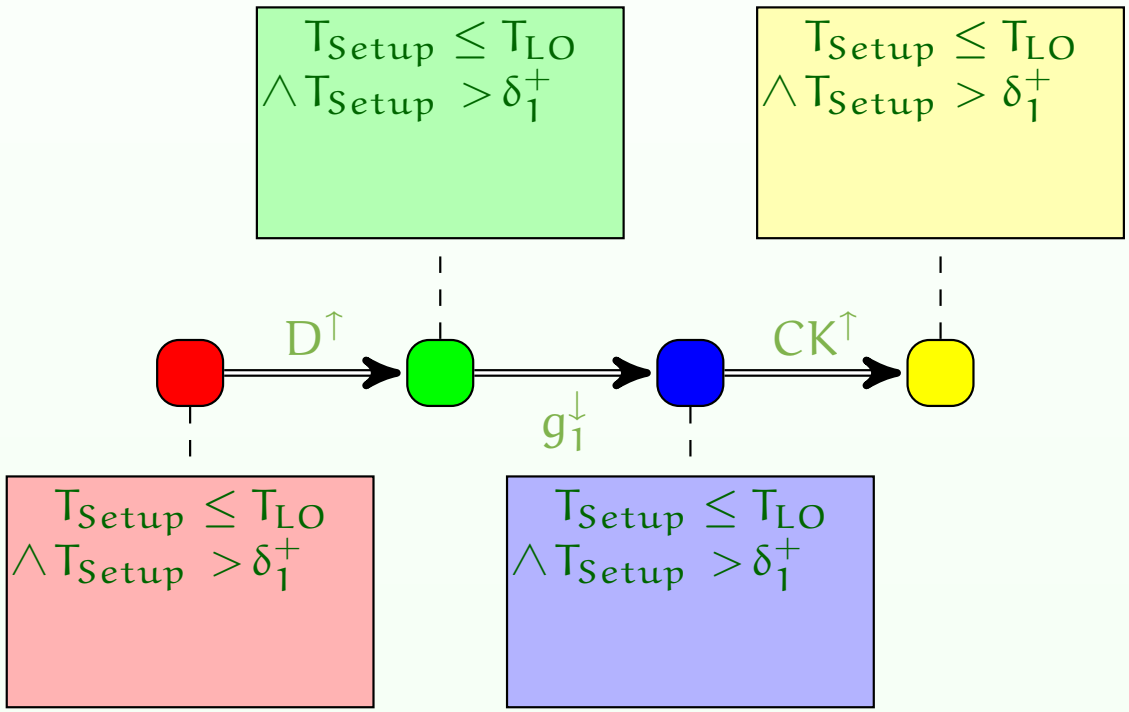


Application of TPsynth to the flip-flop circuit

ν_0 :

$\delta_1^- = 7$	$\delta_1^+ = 7$	$T_{HI} = 24$
$\delta_2^- = 5$	$\delta_2^+ = 6$	$T_{LO} = 15$
$\delta_3^- = 8$	$\delta_3^+ = 10$	$T_{Setup} = 10$
$\delta_4^- = 3$	$\delta_4^+ = 7$	$T_{Hold} = 17$

$K_0 = T_{Setup} > \delta_1^+$



Application of TPsynth to the flip-flop circuit

ν_0 :

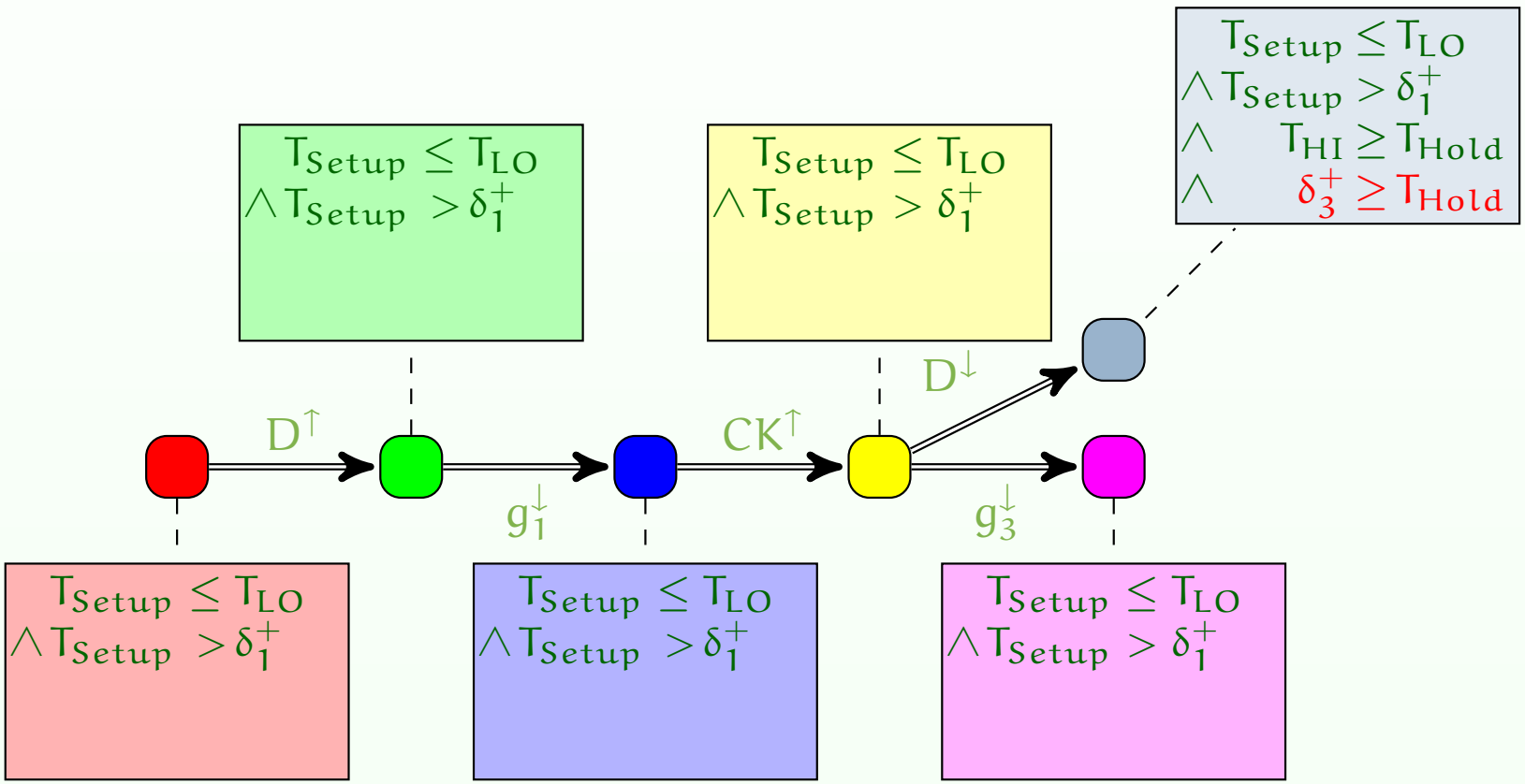
$\delta_1^- = 7$ $\quad \delta_1^+ = 7$ $\quad T_{HI} = 24$

$\delta_2^- = 5$ $\quad \delta_2^+ = 6$ $\quad T_{LO} = 15$

$\delta_3^- = 8$ $\quad \delta_3^+ = 10$ $\quad T_{Setup} = 10$

$\delta_4^- = 3$ $\quad \delta_4^+ = 7$ $\quad T_{Hold} = 17$

$K_0 = T_{Setup} > \delta_1^+$



Application of TPsynth to the flip-flop circuit

ν_0 :

$\delta_1^- = 7$ $\quad \delta_1^+ = 7$ $\quad T_{HI} = 24$

$\delta_2^- = 5$ $\quad \delta_2^+ = 6$ $\quad T_{LO} = 15$

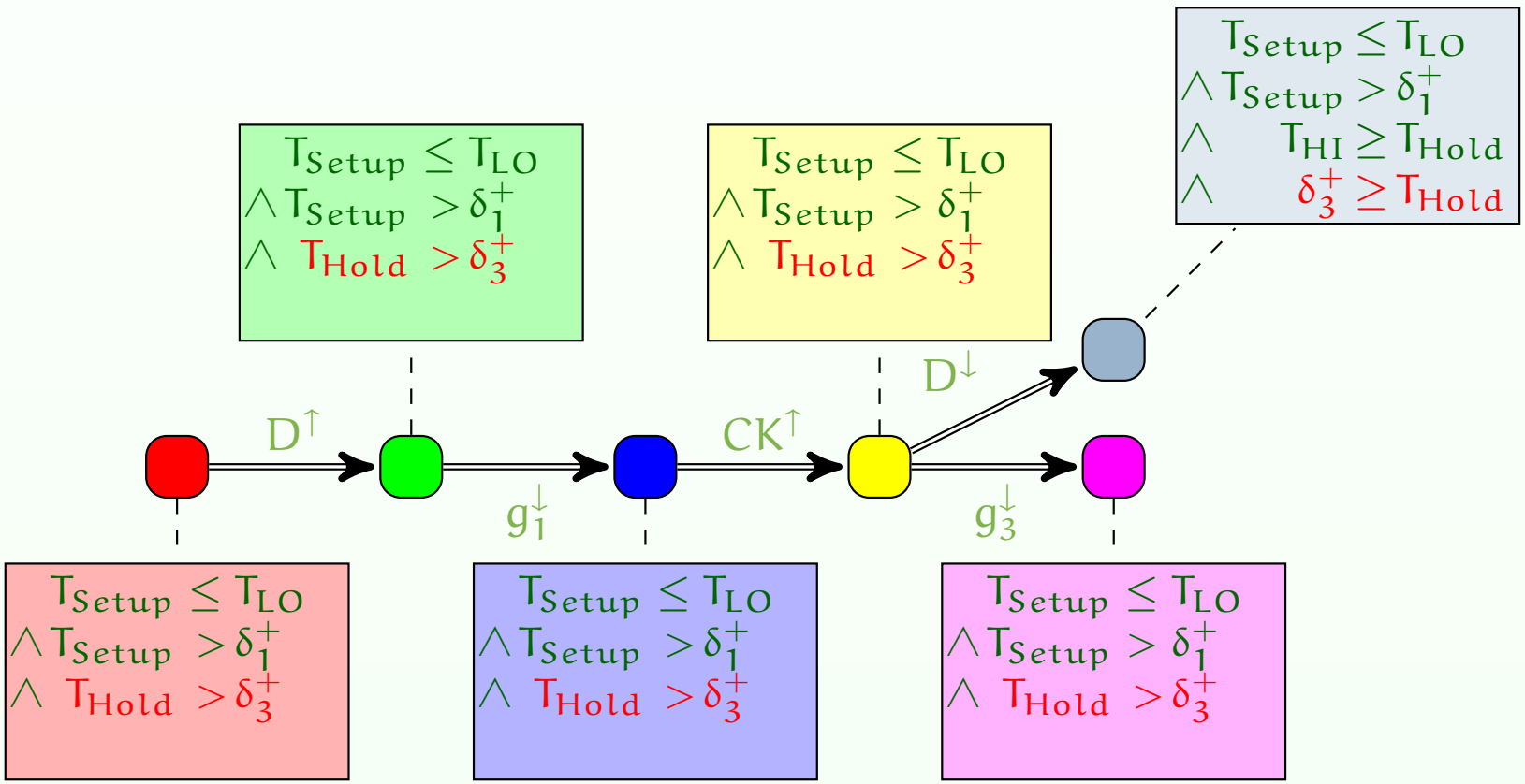
$\delta_3^- = 8$ $\quad \delta_3^+ = 10$ $\quad T_{Setup} = 10$

$\delta_4^- = 3$ $\quad \delta_4^+ = 7$ $\quad T_{Hold} = 17$

$K_0 =$

$T_{Setup} > \delta_1^+$

$\wedge \; T_{Hold} > \delta_3^+$



Application of TPsynth to the flip-flop circuit

ν_0 :

$\delta_1^- = 7$ $\quad \delta_1^+ = 7$ $\quad T_{HI} = 24$

$\delta_2^- = 5$ $\quad \delta_2^+ = 6$ $\quad T_{LO} = 15$

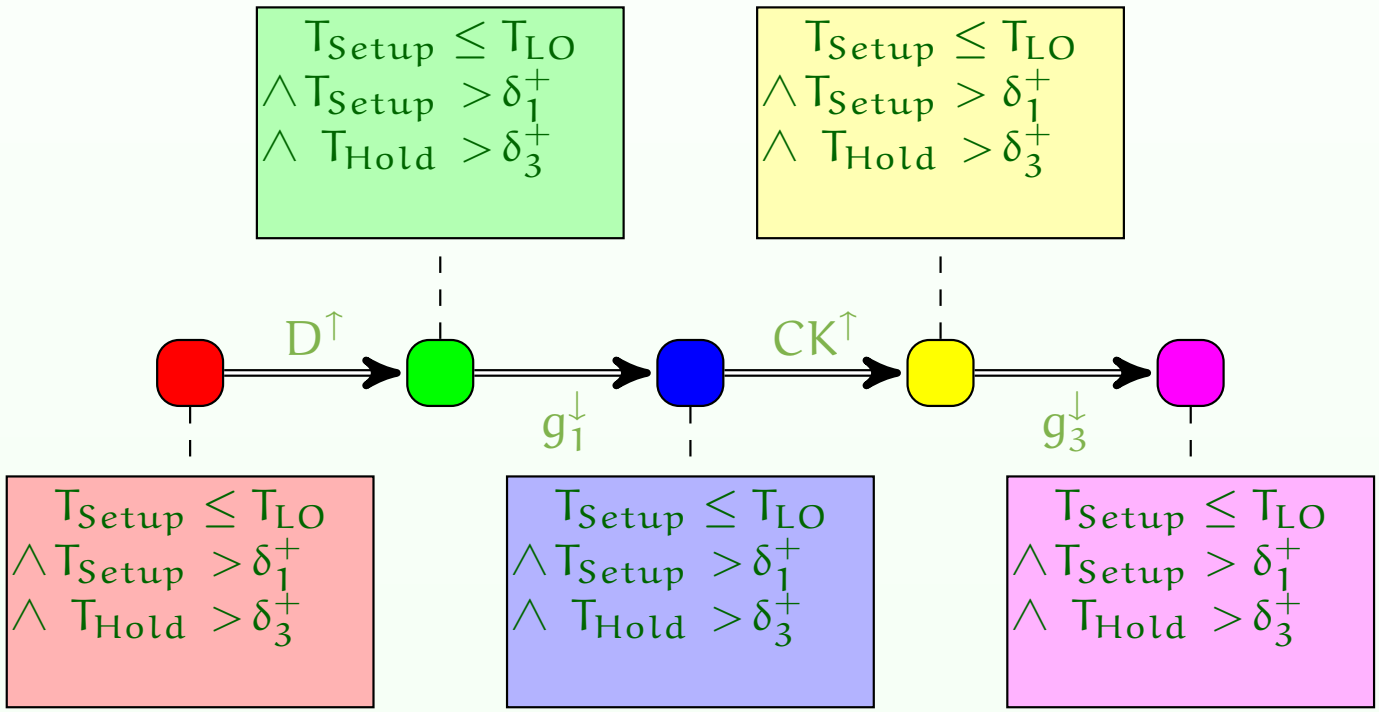
$\delta_3^- = 8$ $\quad \delta_3^+ = 10$ $\quad T_{Setup} = 10$

$\delta_4^- = 3$ $\quad \delta_4^+ = 7$ $\quad T_{Hold} = 17$

$K_0 =$

$T_{Setup} > \delta_1^+$

$\wedge \; T_{Hold} > \delta_3^+$



Application of TPsynth to the flip-flop circuit

ν_0 :

$\delta_1^- = 7$ $\quad \delta_1^+ = 7$ $\quad T_{HI} = 24$

$\delta_2^- = 5$ $\quad \delta_2^+ = 6$ $\quad T_{LO} = 15$

$\delta_3^- = 8$ $\quad \delta_3^+ = 10$ $\quad T_{Setup} = 10$

$\delta_4^- = 3$ $\quad \delta_4^+ = 7$ $\quad T_{Hold} = 17$

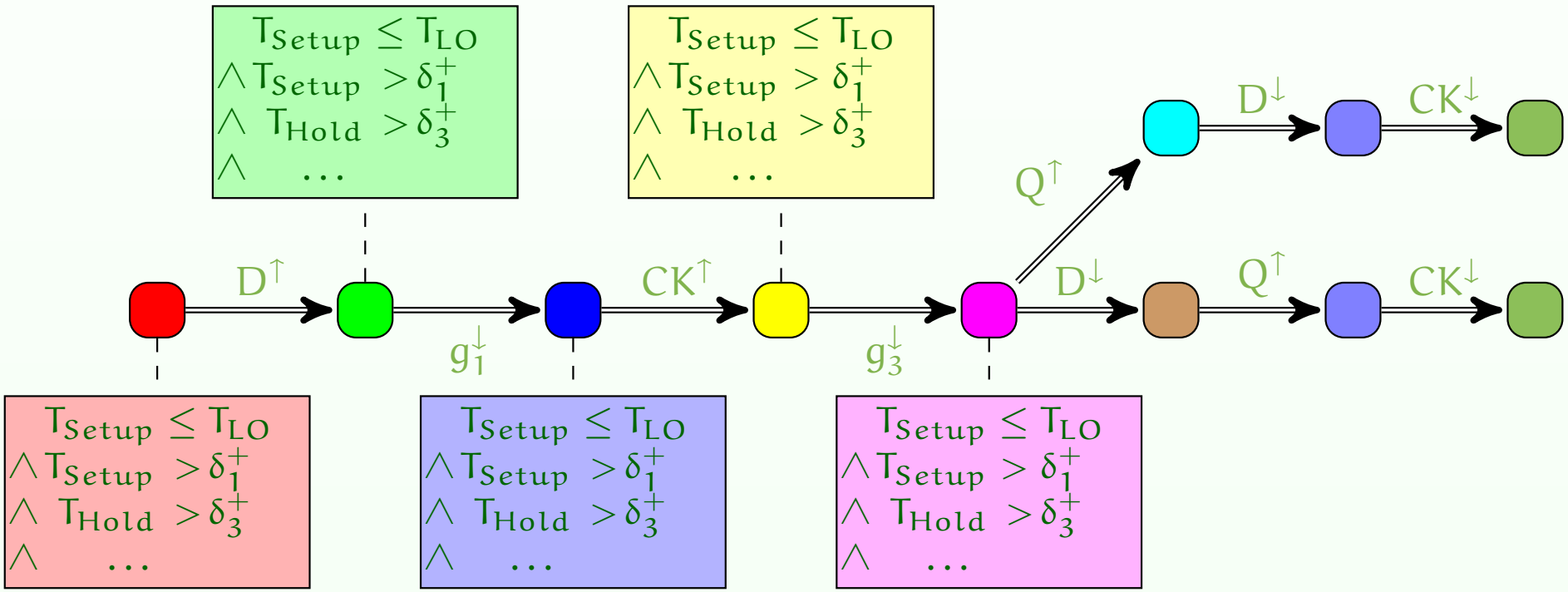
$K_0 =$

$T_{Setup} > \delta_1^+ \quad \wedge \quad \delta_3^+ + \delta_4^+ \geq T_{Hold}$

$\wedge \; T_{Hold} > \delta_3^+ \quad \wedge \quad \delta_3^+ + \delta_4^+ < T_{HI}$

$\wedge \; T_{Setup} \leq T_{LO} \quad \wedge \quad \delta_3^- + \delta_4^- \leq T_{Hold}$

$\wedge \quad \delta_1^- > 0$



Correctness of TPsynth

Théorème ([André et al., 2009])

Suppose that TPsynth$(\mathcal{A}, v_0)$ terminates with output K_0. We have:

1. $v_0 \models K_0$, and
2. for all $v \models K_0$,

Correctness of TPsynth

Théorème ([André et al., 2009])

Suppose that TPsynth$(\mathcal{A}, v_0)$ terminates with output K_0. We have:

1. $v_0 \models K_0$, and
2. for all $v \models K_0$,

What about completeness...?

Software supporting parametric timed automata

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

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

Implementation in 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 with preemption

Under continuous development since 2008 [André et al., 2012]

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

www.imitator.fr

Some success stories using PTAs

- Variants of train controllers [Alur et al., 1993, Hune et al., 2002]
- The root contention protocol [Hune et al., 2002]
- Philip's bounded retransmission protocol [Hune et al., 2002]
- An asynchronous circuit commercialized by ST-Microelectronics [Chevallier et al., 2009]
- A 4-phase handshake protocol [Knapik and Penczek, 2012]
- A distributed prospective architecture for the flight control system of the next generation of spacecrafts designed at ASTRIUM Space Transportation [Fribourg et al., 2012]
- Analysis of music scores [Fanchon and Jacquemard, 2013]
- The alternating bit protocol [Jovanović et al., 2015]
- (non-preemptive) schedulability problems [Jovanović et al., 2015]
- An unmanned aerial video system by Thales

Plan: Conclusion

1 Parametric Timed Automata

2 Decidability results

3 Parameter synthesis

4 Conclusion
- Summary
- Perspectives

Summary

- Finite-state automata
  - ☺ Mostly decidable results
  - ☺ Efficient model checking algorithms
  - ☹ Miss the quantitative aspects
  - ☺ Many powerful tools
- Timed automata
  - ☺ Finite abstract semantics
  - ☺ Some decidable results
  - ☹ Some undecidable results
  - ☺ Several powerful tools
- Parametric timed automata
  - ☺ Very expressive
  - ☹ No finite abstract semantics
  - ☹ Almost only undecidability results
  - ☺ Some powerful tools

Perspectives

- Exhibit decidable subclasses of parametric timed automata
  - Good candidates: L-PTA and U-PTA
- Design efficient semi-algorithms with good termination conditions
  - e.g., reuse the integer hull of [Jovanović et al., 2015]
- Beyond parametric timed automata
  - Hybrid systems with non-linear dynamics
  - Models with costs and probabilities
- Broaden the use of formal methods in the industry

Perspectives

- Exhibit decidable subclasses of parametric timed automata
  - Good candidates: L-PTA and U-PTA
- Design efficient semi-algorithms with good termination conditions
  - e.g., reuse the integer hull of [Jovanović et al., 2015]
- Beyond parametric timed automata
  - Hybrid systems with non-linear dynamics
  - Models with costs and probabilities
- Broaden the use of formal methods in the industry
  - ☺ . . . and save lives!!

Source et références

General References

- The Inverse Method (Étienne André and Romain Soulat), ISTE and Wiley & Sons, 2013
- Video tutorial: Tutorial on Parametric Verification (Étienne André, Didier Lime, Laure Petrucci), YouTube, 2016

  https://www.youtube.com/playlist?list=PL9SOLKoGjbepAll8RnD16FfYngJtN22DY

References I

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. (2016).
What's decidable about parametric timed automata?
In Artho, C. and Ölveczky, P. C., editors, *Proceedings of the 4th International Workshop on Formal Techniques for Safety-Critical Systems (FTSCS'15)*, volume 596 of *Communications in Computer and Information Science*, pages 52–68. Springer.

André, É., Chatain, Th., Encrenaz, E., and Fribourg, L. (2009).
An inverse method for parametric timed automata.
International Journal of Foundations of Computer Science, 20(5):819–836.

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.

References II

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 Roux, O. H. (2016).
Decision problems for parametric timed automata.
In Ogata, K., Lawford, M., and Liu, S., editors, *Proceedings of the 18th International Conference on Formal Engineering Methods (ICFEM'16)*, volume 10009 of *Lecture Notes in Computer Science*, pages 400–416. Springer.

André, É. and Markey, N. (2015).
Language preservation problems in parametric timed automata.
In Sankaranarayanan, S. and Vicario, E., editors, *Proceedings of the 13th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'15)*, volume 9268 of *Lecture Notes in Computer Science*, pages 27–43. 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 III

Beneš, N., Bezděk, P., Larsen, K. G., and Srba, J. (2015).
Language emptiness of continuous-time parametric timed automata.
In Halldórsson, M. M., Iwama, K., Kobayashi, N., and Speckmann, B., editors, *ICALP, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 69–81. Springer.

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 Csuhaj-Varjú, E., Dietzfelbinger, M., and Ésik, Z., editors, *MFCS, Part I*, volume 8634 of *Lecture Notes in Computer Science*, pages 123–134. Springer.

Chevallier, R., Encrenaz-Tiphène, E., Fribourg, L., and Xu, W. (2009).
Timed verification of the generic architecture of a memory circuit using parametric timed automata.
Formal Methods in System Design, 34(1):59–81.

Clarisó, R. and Cortadella, J. (2007).
The octahedron abstract domain.
Science of Computer Programming, 64(1):115–139.

References IV

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.

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.

References V

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

Knapik, M. and Penczek, W. (2012).
Bounded model checking for parametric timed automata.
Transactions on Petri Nets and Other Models of Concurrency, 5:141–159.

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.

Miller, J. S. (2000).
Decidability and complexity results for timed automata and semi-linear hybrid automata.
In Lynch, N. A. and Krogh, B. H., editors, *HSCC*, volume 1790 of *Lecture Notes in Computer Science*, pages 296–309. Springer.

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

Licence de ce document

Auteur : Étienne André

(Source LaTeX disponible sur demande)

UNIVERSITÉ PARIS 13