FTSCS 2013

30th October 2013
Queenstown, New Zealand

Parametric Schedulability Analysis of Fixed Priority Real-Time Distributed Systems

Youcheng Sun[1], Romain Soulat[2], Giuseppe Lipari[1,2], Étienne André[3], Laurent Fribourg[2]

[1]Scuola Superiore Sant'Anna, Pisa, Italy
[2]LSV, ENS Cachan & CNRS, France
[3]LIPN, Université Paris 13, Sorbonne Paris Cité, France

Context: Hard Real-Time Embedded Systems

- Modern hard real-time embedded systems are distributed in nature.
- Many of them have critical timing requirements:
  - automotive systems (modern cars have 10-20 embedded boards connected by one or more CAN bus)
  - avionics systems (several distributed control boards connected by one or more dedicated networks)

- To analyse the schedulability of such systems, it is very important to estimate the (worst-case) computation times of the tasks.
- Estimating WCET is very difficult in modern architectures.

Context: Parameter Synthesis

In distributed systems, an imprecision in one timing parameter can compromise schedulability of the entire system.

$\rightsquigarrow$ Perform parametric analysis with respect to those parameters that have much variability (e.g. computation times)
Useful for:

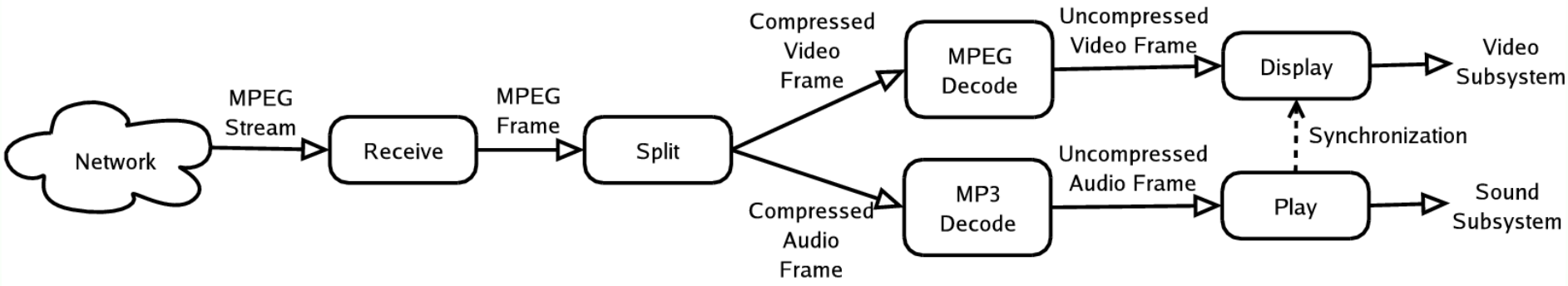
- robustness analysis;
- design space exploration.

Outline

Outline

Real-Time pipelines

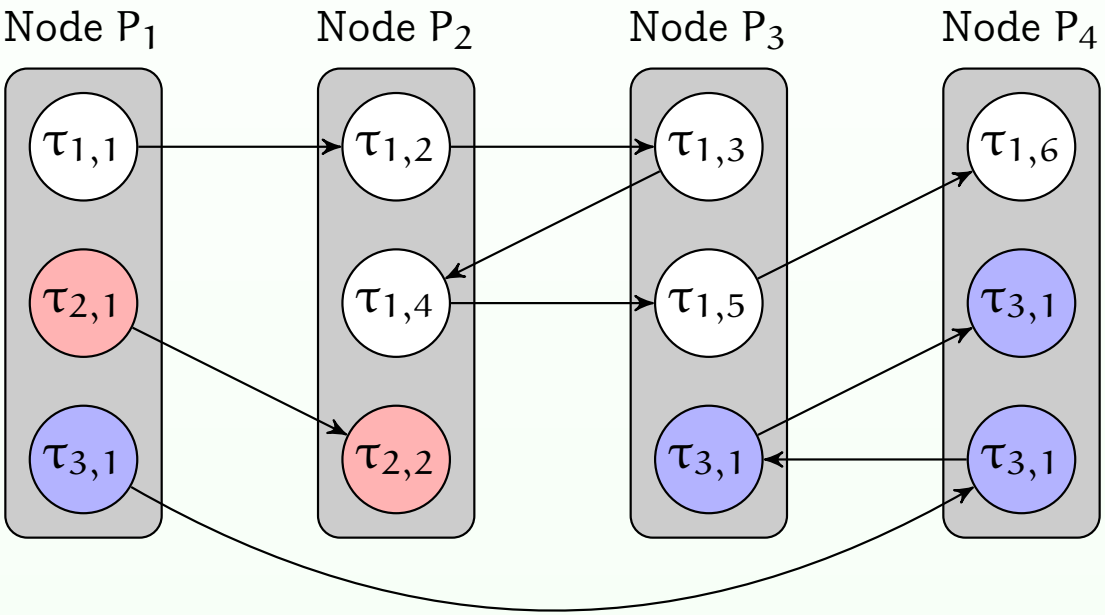
- Many real-time applications can be modeled as *pipelines* of tasks



- Executed on a distributed (or multicore) system
- Activated cyclically (periodic or sporadic)
  - In many cases, period is smaller than the end-to-end deadline
- In this paper we focus on *pipelines* (also called transactions)

Model

- A set of pipelines $\{\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(p)}\}$ distributed over m nodes
- Each pipeline $\mathcal{P}^{(i)}$ is a chain of n_i tasks $\{\tau_{i,1}, \ldots, \tau_{i,n_i}\}$
- Pipeline $\mathcal{P}^{(i)}$ has an end-to-end (E2E) deadline D_i and period T_i

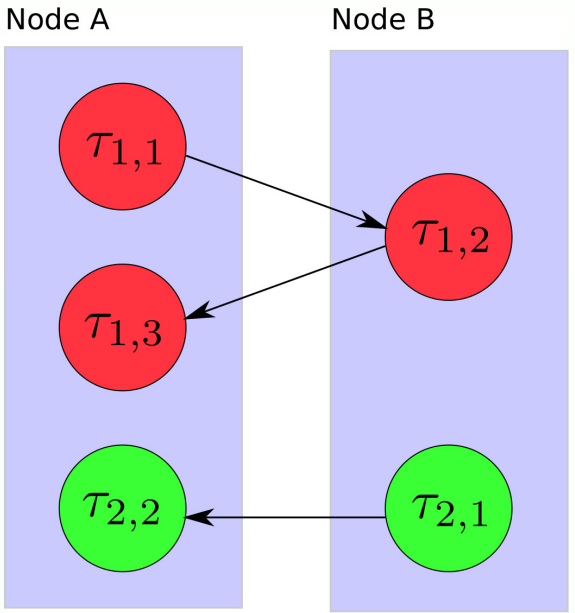


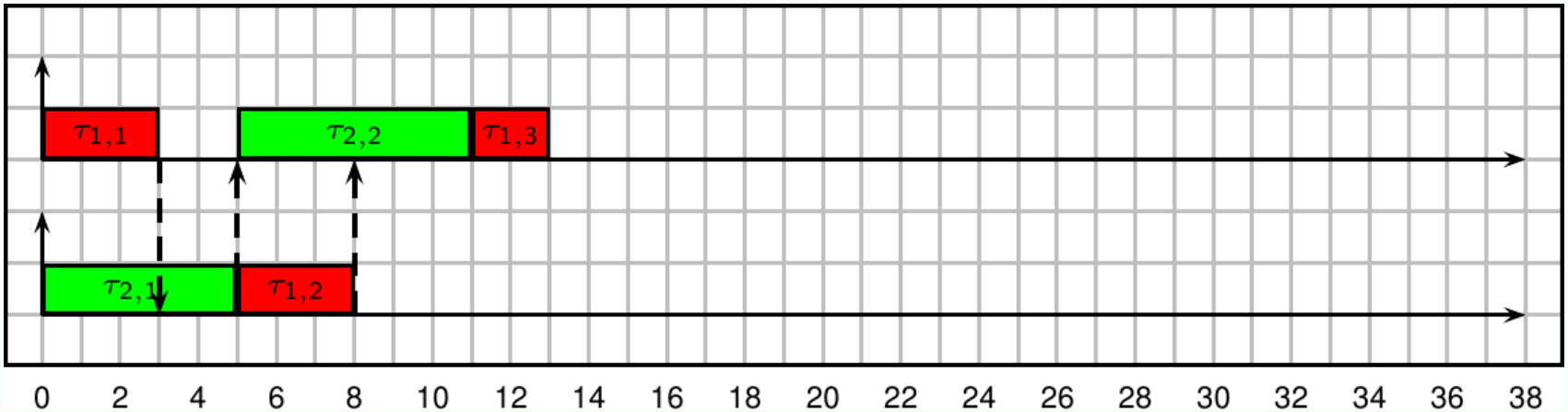
- Scheduling Problem: Guarantee that all pipelines complete before their E2E deadlines

Activations, jitter, deadline

- An example

Task	Per.	E2E	Comp.	Resp.	Jitter	prio
$\tau_{1,1}$	15		3	3	0	HI
$\tau_{1,2}$	-		3	8	0-2	LO
$\tau_{1,3}$	-	15	2	13	3	LO
$\tau_{2,1}$	12		5	6	0	HI
$\tau_{2,2}$	-	12	6	?	0-1	ME

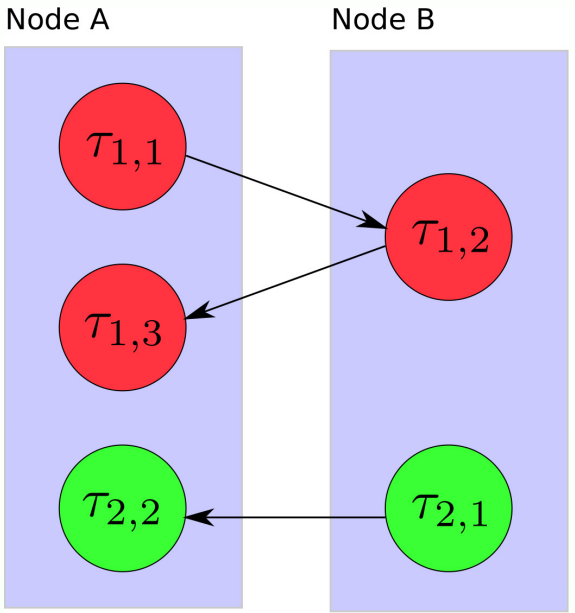


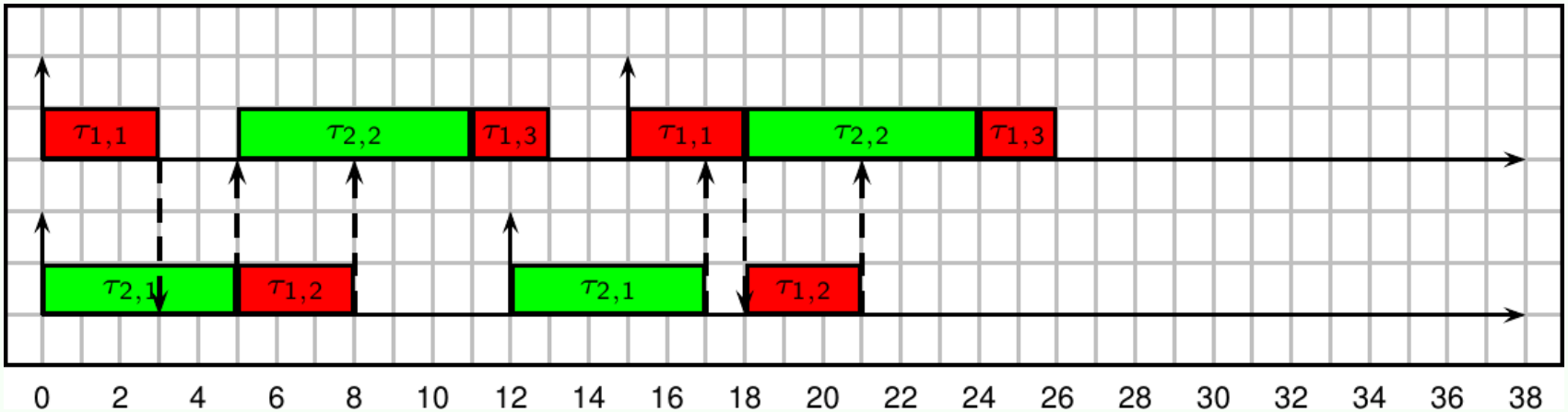


Activations, jitter, deadline

- An example

Task	Per.	E2E	Comp.	Resp.	Jitter	prio
$\tau_{1,1}$	15		3	3	0	HI
$\tau_{1,2}$	-		3	8	0-2	LO
$\tau_{1,3}$	-	15	2	13	3	LO
$\tau_{2,1}$	12		5	6	0	HI
$\tau_{2,2}$	-	12	6	?	0-1	ME

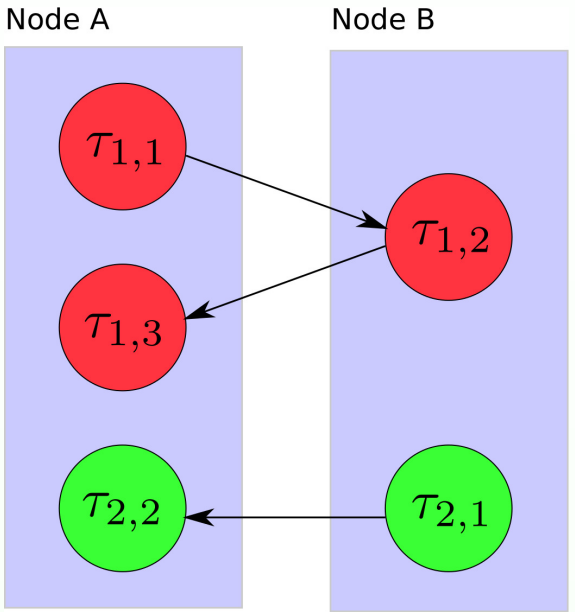


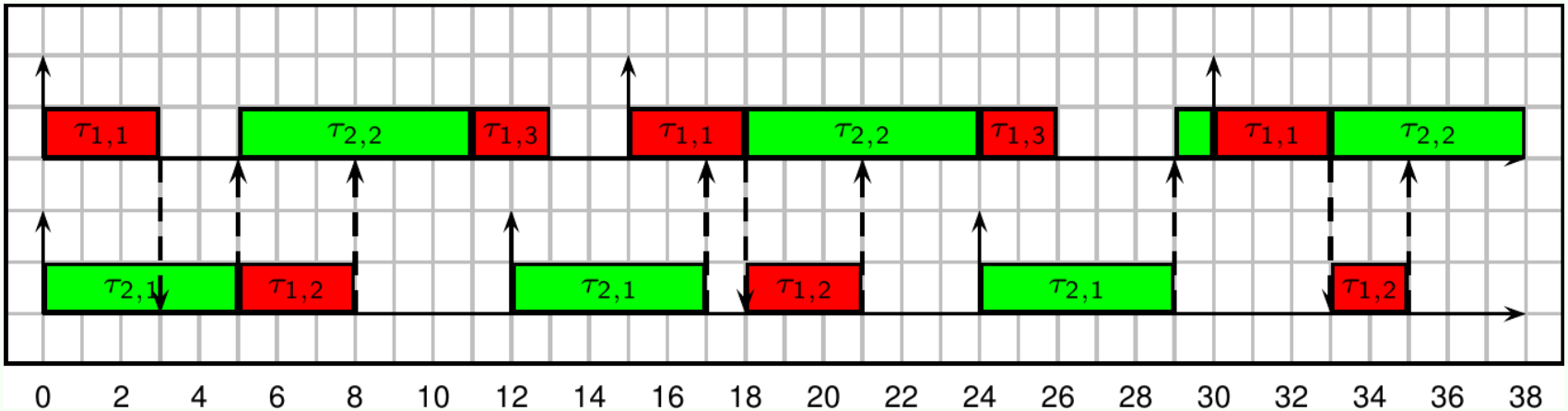


Activations, jitter, deadline

- An example

Task	Per.	E2E	Comp.	Resp.	Jitter	prio
$\tau_{1,1}$	15		3	3	0	HI
$\tau_{1,2}$	-		3	8	0-2	LO
$\tau_{1,3}$	-	15	2	13	3	LO
$\tau_{2,1}$	12		5	6	0	HI
$\tau_{2,2}$	-	12	6	**14**	0-1	ME





Model of the Network

- Tasks executing on processors are scheduled by Full Preemptive Fixed Priority Scheduler (FPFPS)
- Tasks send messages over the network to signal finishing of instance and activating the subsequent task
- Messages are scheduled over the network according to Non-Preemptive Fixed Priority
  - In the industrial domain, the CAN Bus features such a policy

Outline

Parametric analysis

- Previous work:
  - Holistic analysis in fixed priority [Tindell and Clark, 1994]
  - Event stream and Real-Time Calculus models
- Compute upper bounds on response times of tasks by computing the fixed point of a set of non-linear recursive equations
- Parametric analysis:
  - For which values of the computation times the system is schedulable?
  - Useful for robustness analysis and exploration of design space
- How it can be done:
  - Repeat holistic analysis for a range of parameters
  - But does not scale very well for > 2 parameters

Goal

Goal

Provide a full-parametric analysis of a distributed real-time system with respect to the computation times of the tasks

- Two methodologies of analysis
  1. Use an extension of the holistic analysis to make it parametric
  2. Use Parametric timed automata

Outline

Parametric model

- A task is identified by three parameters:
  - C_i is the worst-case computation time (or worst-case transmission time, in case it models a message)
  - R_i is the task worst-case response time, i.e. the worst case finishing time of any task instance relative to the activation of its pipeline.
  - J_i is the task worst-case activation jitter, i.e. the greatest time since its activation that a task must wait for all preceding tasks to complete their execution
- The only one of interest is the computation time

Formula for single processor

Theorem

A task τ_i *is schedulable if*
$\forall h = 1, \ldots, \frac{H_n}{T_i}, \quad \exists \mathbf{n} \in \mathbb{P}^{i-1}((h-1)T_i + D_i)$ *such that*

- $B_i + (h-1)C_i + \sum_{j=1}^{i-1} n_j C_j \leq n_l T_l - J_l \quad \forall l = 1, \ldots, i-1$;
- $B_i + (h-1)C_i + \sum_{j=1}^{i-1} n_j C_j \leq (h-1)T_i + R_i - C_i - J_i$;
- $R_i \leq D_i$ *and* $B_i \leq C_j - 1$ *for all* $j > i$.

where $\mathbf{n}$ *is a vector of* $i-1$ *integers,* $\mathbb{P}^{i-1}(D_i)$ *is the set of scheduling points, and* B_i *is the blocking time due to non-preemptive scheduling.*

- This is a disjunction (OR) of conjunctions of inequalities (AND)
- The final polyhedron is non-convex

Distributed system

1. For each processor and network, we build the constraint system corresponding to the theorem.
   - Notice that the set of constraints for the individual single processor systems are independent of each other (because they are constraints on different tasks).

2. For each pipeline $\mathcal{T}^a$:
   - two successive tasks $\tau_{a,i}$ and $\tau_{a,i+1}$ must fulfill the constraint $R_{a,i} \leq J_{a,i+1}$;
   - for the initial task we impose $J_{a,1} = 0$.

3. After integration, all parameters $R_{a,i}$ and $J_{a,i}$ can be eliminated by projection

Advantages and Drawbacks

Advantage

- This method directly generates a set of constraints that defines the space of parameters C_i;

Drawbacks

- Since the equations give upper bounds, the space describes a subset of the real-space of constraints
- If too many parameters are specified or too many scheduling points, complexity may explode

In the experimental part, we will evaluate the scalability of this approach

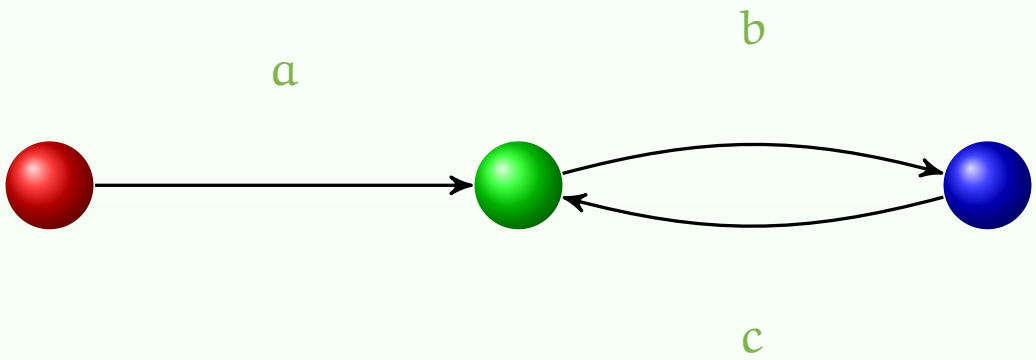
Outline

Timed Automaton

- Finite state automaton (sets of locations)

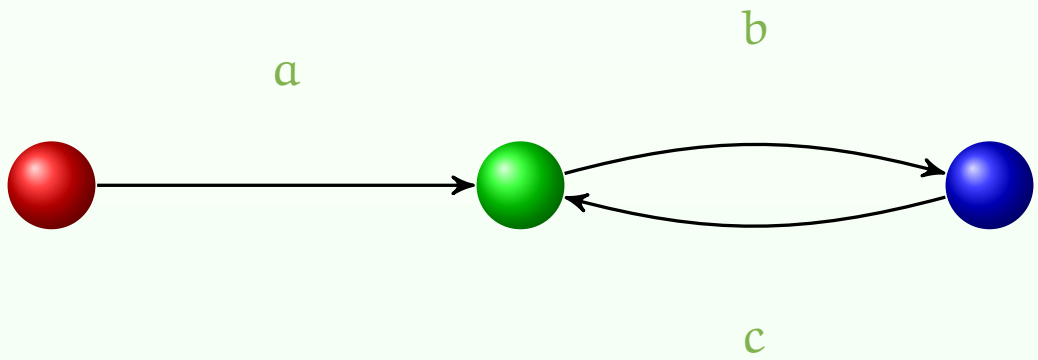
Timed Automaton

- Finite state automaton (sets of locations and actions)



Timed Automaton

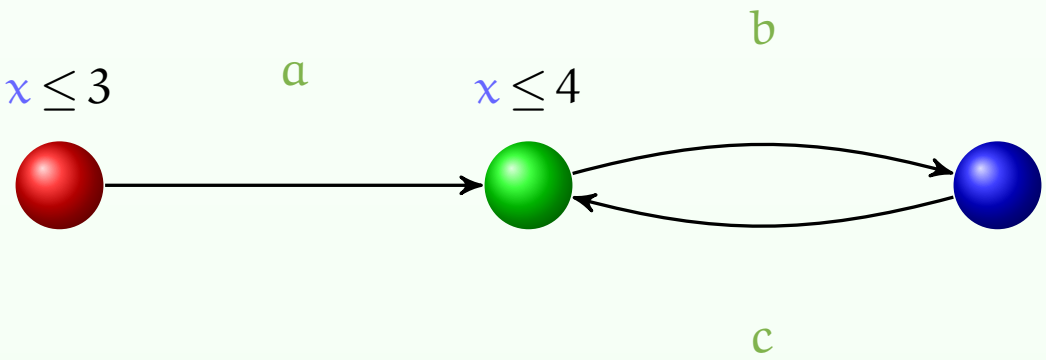
- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate [Alur and Dill, 1994])



Timed Automaton

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

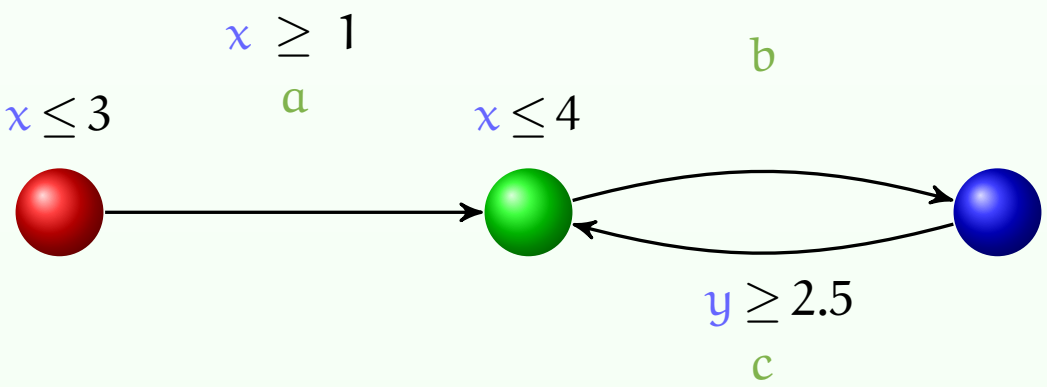
- Features
  - Location invariant: property to be verified to stay at a location



Timed Automaton

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

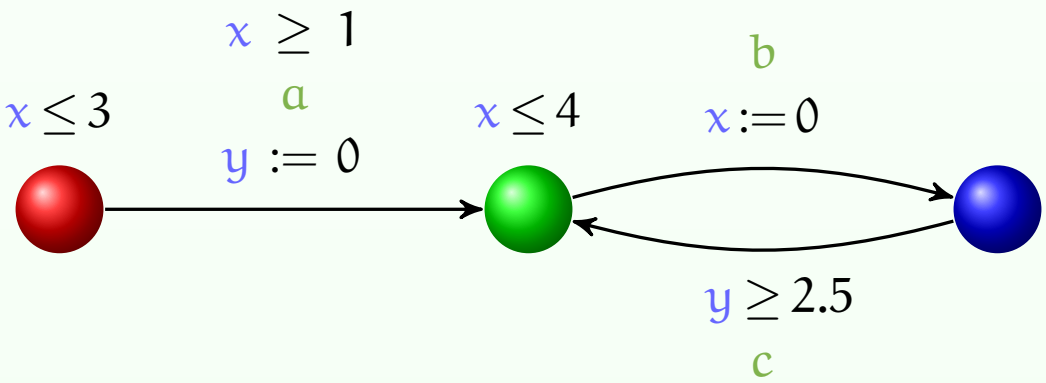
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition



Timed Automaton

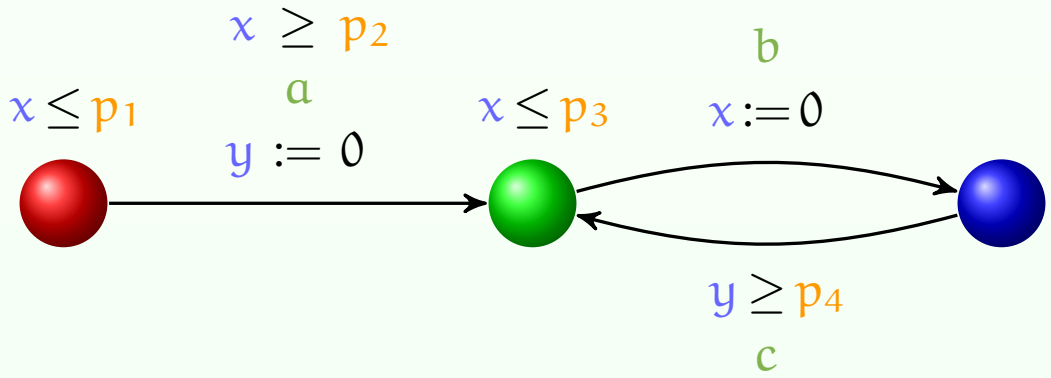
- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate [Alur and Dill, 1994])

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

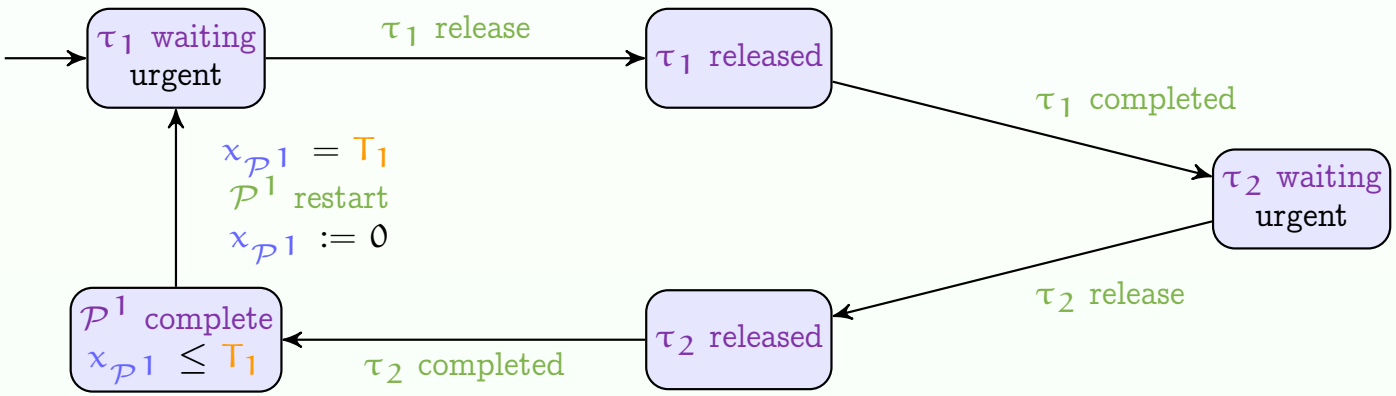


Parametric Timed Automaton (PTA)

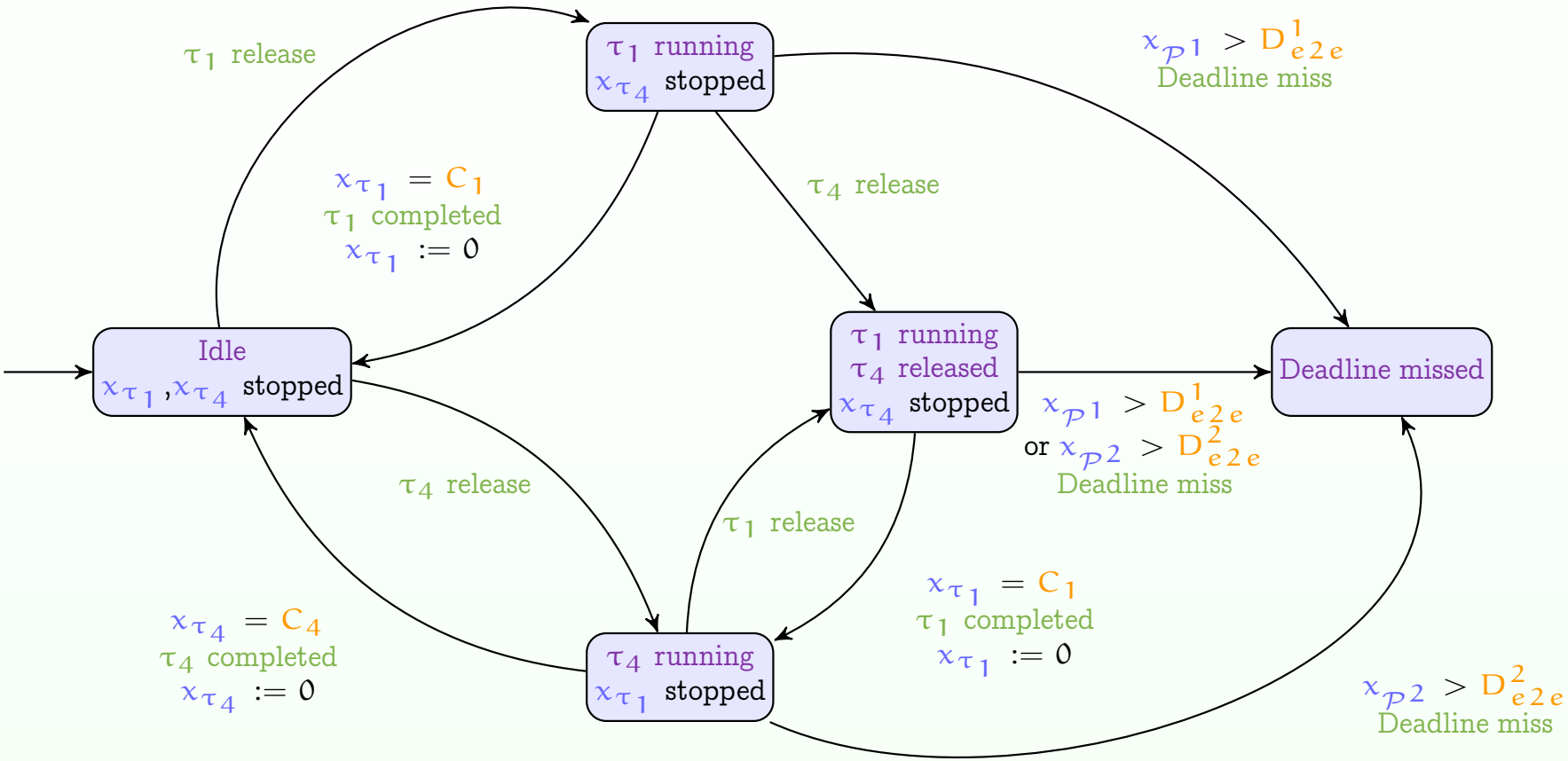
- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate [Alur and Dill, 1994])
  - A set P of parameters (i.e., unknown constants), used in guards and invariants [Alur et al., 1993]
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 at each transition



Modeling a Task / Pipeline



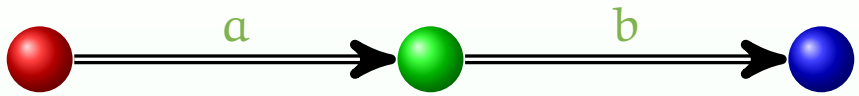
Modeling the Fixed Priority Scheduler (Preemptive)



Actually a PTA extended with stopwatches [Adbeddaïm and Maler, 2002]

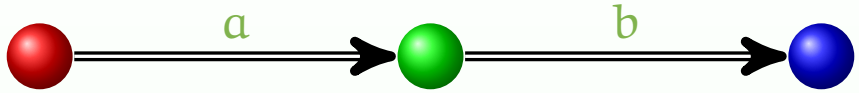
Traces

- Trace over a PTA: time-abstract path
  - Finite alternating sequence of locations and actions

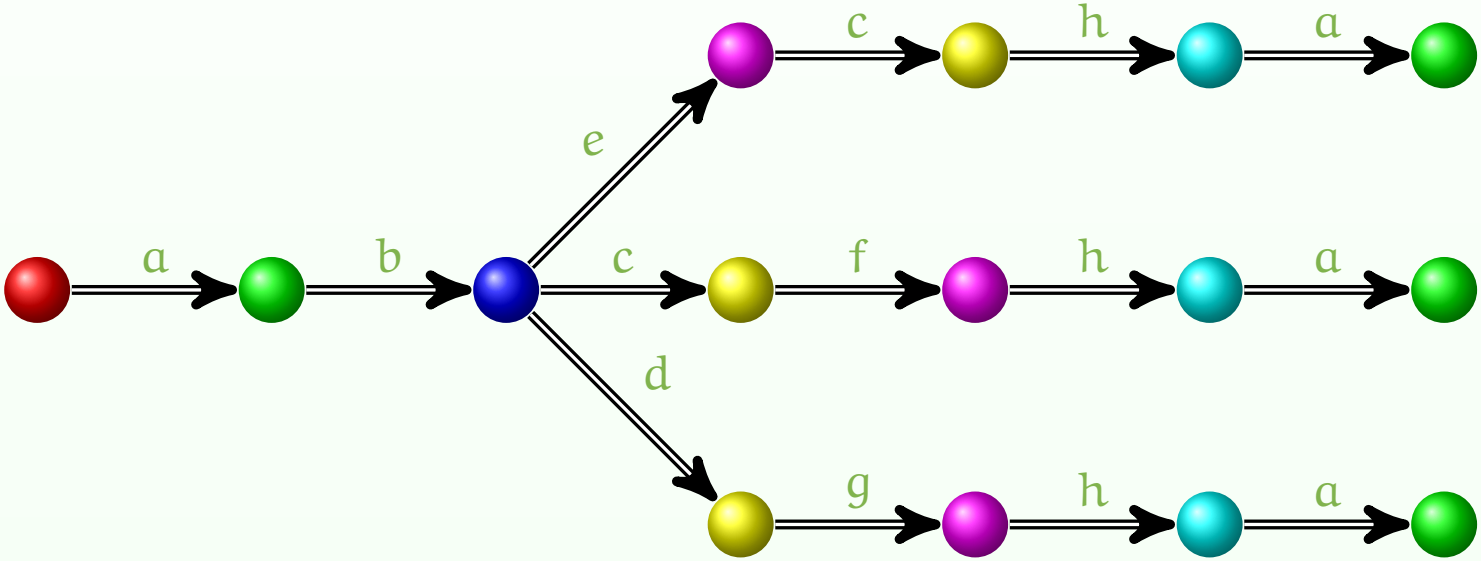


Traces

- Trace over a PTA: time-abstract path
  - Finite alternating sequence of locations and actions



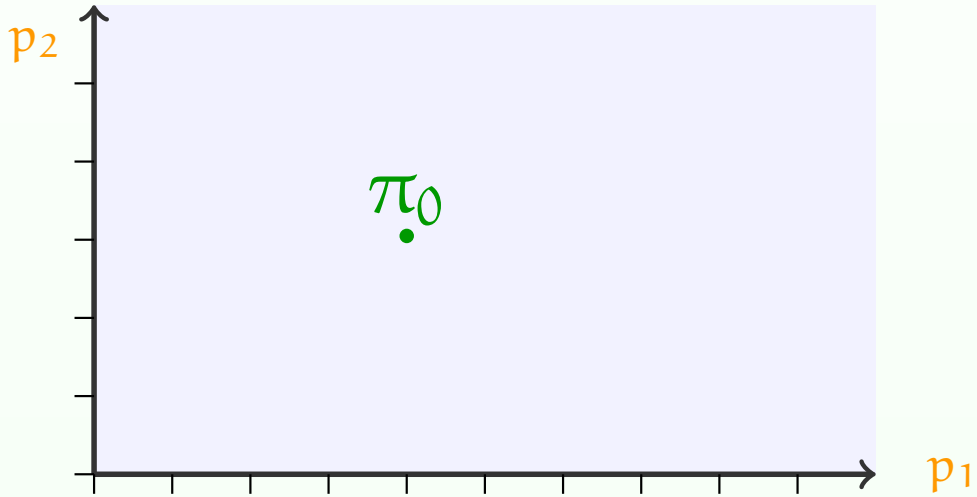
- Trace set of a PTA: set of all its traces



The Inverse Method *IM*

- Algorithm for the synthesis of parameters in PTA
  [A., Chatain, Encrenaz, Fribourg, 2009]

- Takes as input a reference parameter valuation π_0, and outputs a constraint K generalizing this reference valuation

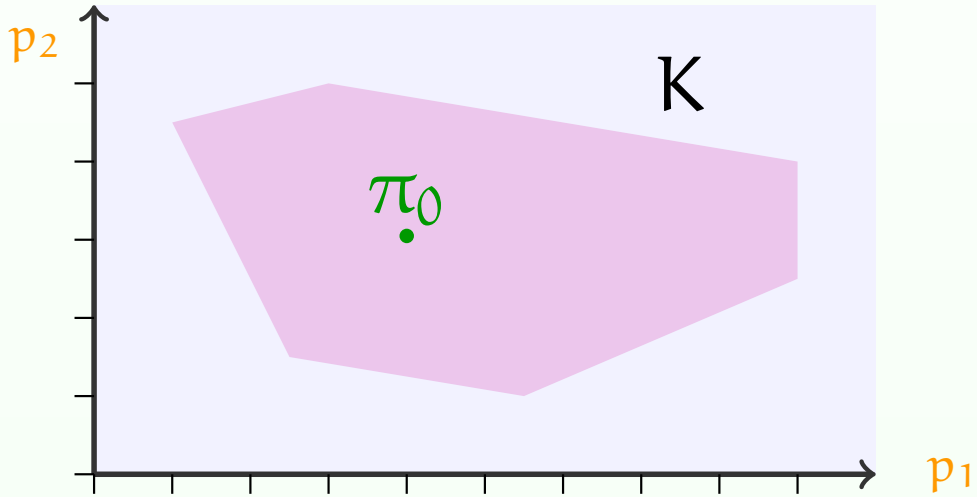


  - For all $\pi \models K$, the trace set is the same as for π_0

The Inverse Method *IM*

- Algorithm for the synthesis of parameters in PTA
  [A., Chatain, Encrenaz, Fribourg, 2009]

- Takes as input a reference parameter valuation π_0, and outputs a constraint K generalizing this reference valuation



  - For all $\pi \models K$, the trace set is the same as for π_0

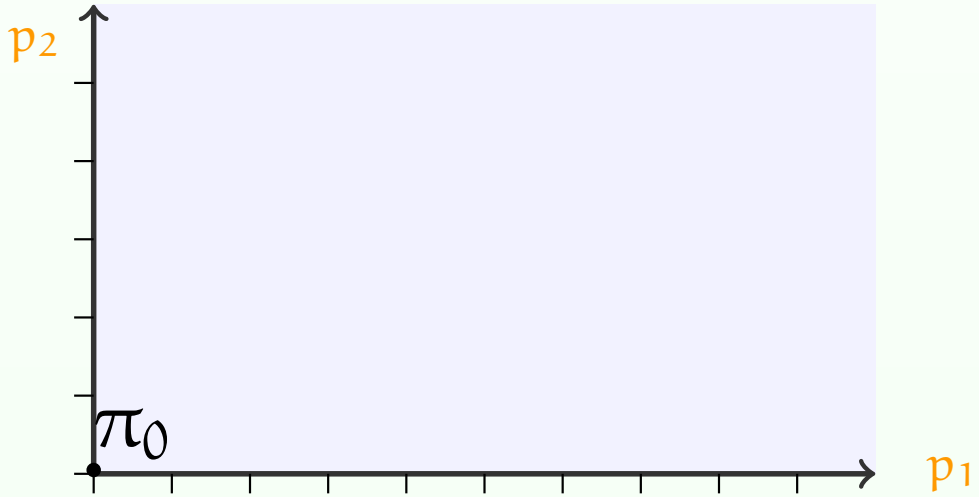
The Inverse Method: Implementation in IMITATOR

- IMITATOR 2.6 [A., Fribourg, Kühne, Soulat, 2012]
  - "Inverse Method for Inferring Time AbstracT BehaviOR"
- Features
  - Synthesis of timing parameters
  - With respect to various properties (using observer patterns)
  - Numerous analysis options
- Inside the tool
  - 10,000 lines of OCaml code
  - Uses the PPL library for operations on polyhedra [Bagnara et al., 2008]
- Use it!
  - Open source: Available under the GNU-GPL license
  - Now integrated in the *CosyVerif* platform [AHHKLLP13]

`http://www.lsv.ens-cachan.fr/Software/imitator/`

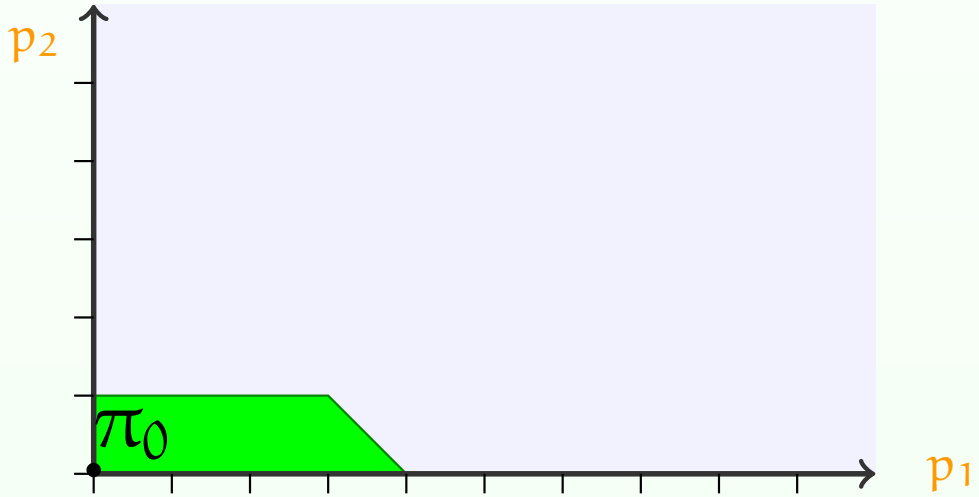
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



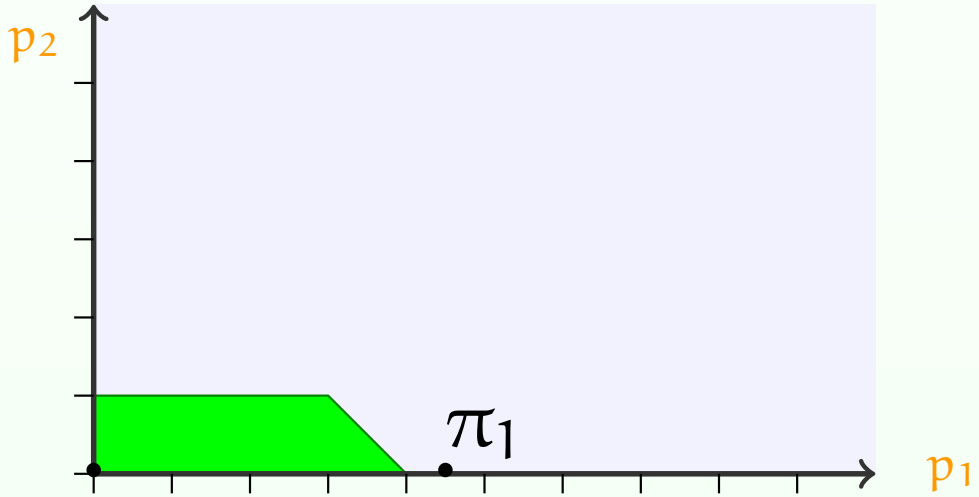
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



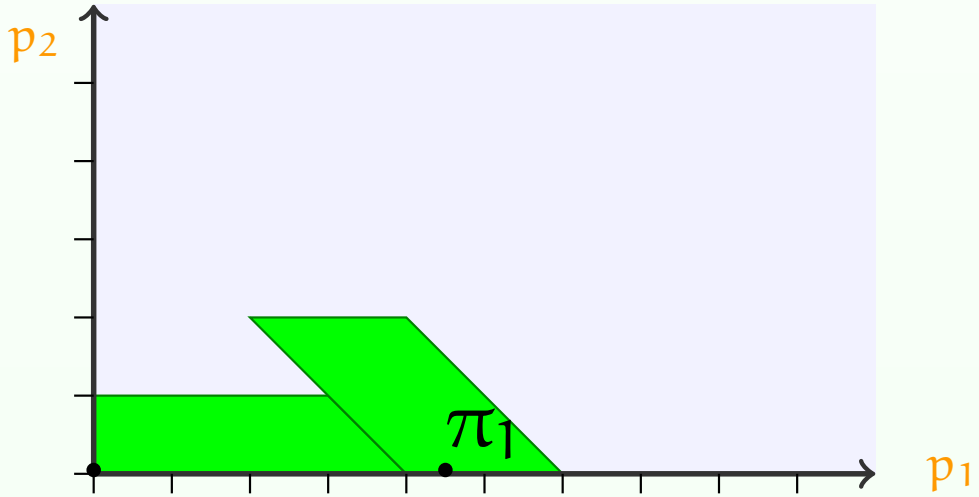
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



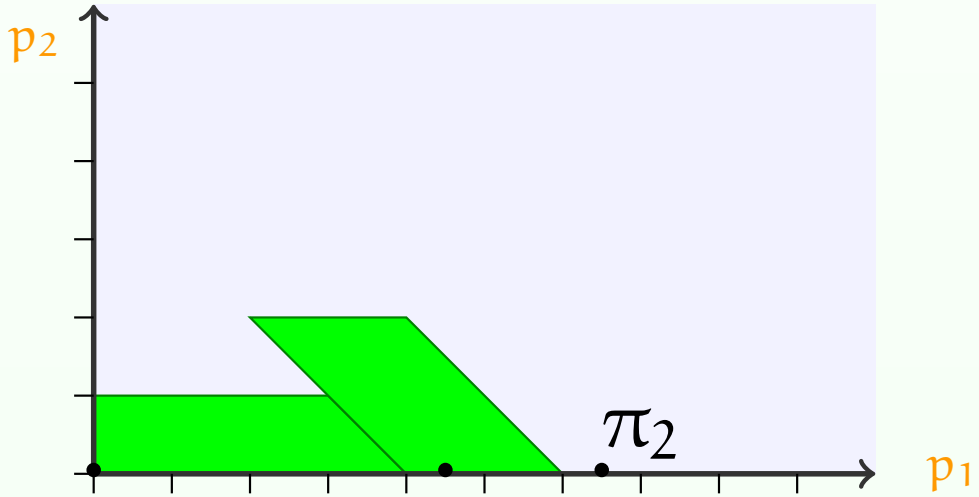
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



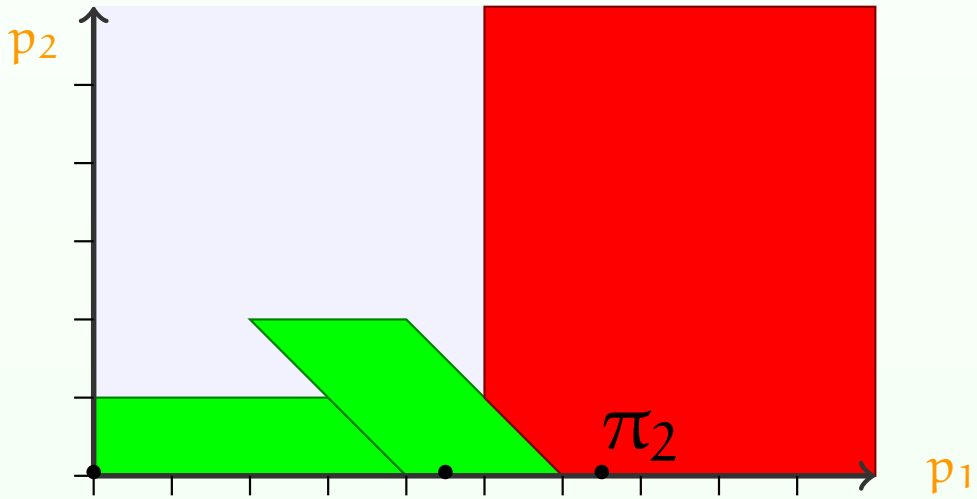
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



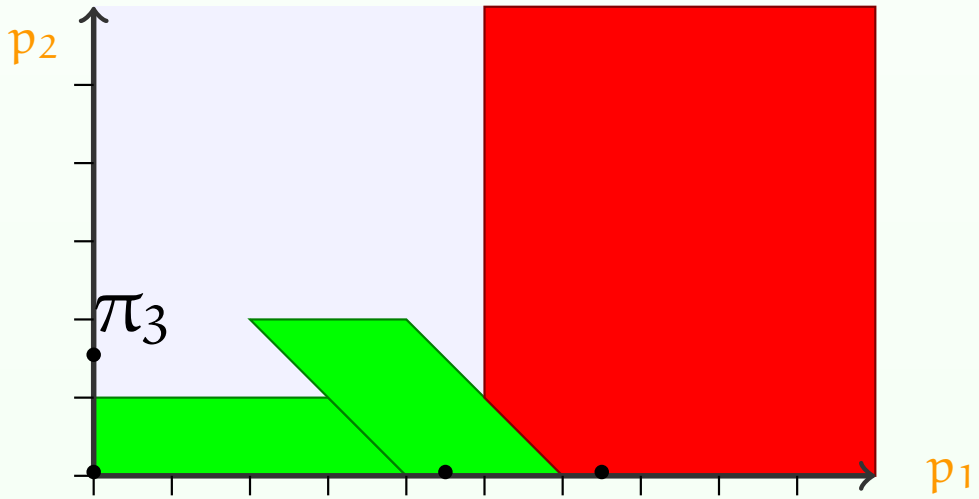
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



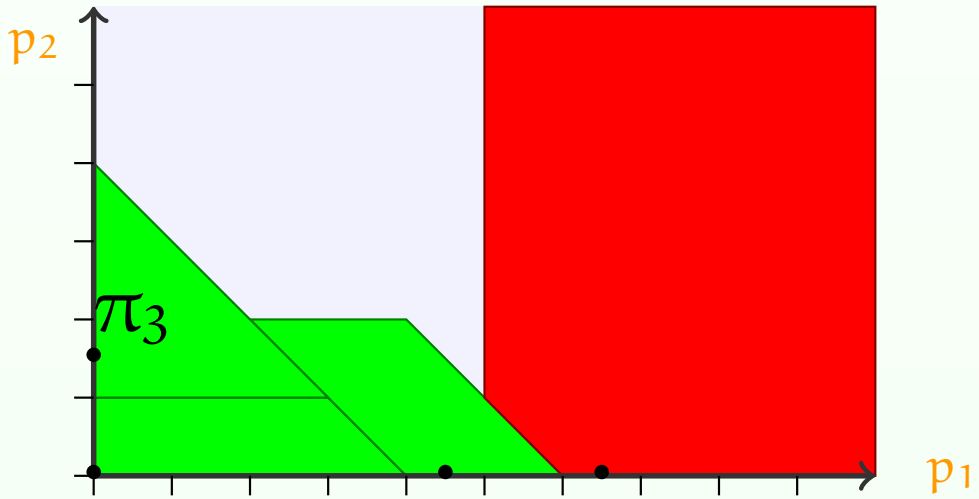
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



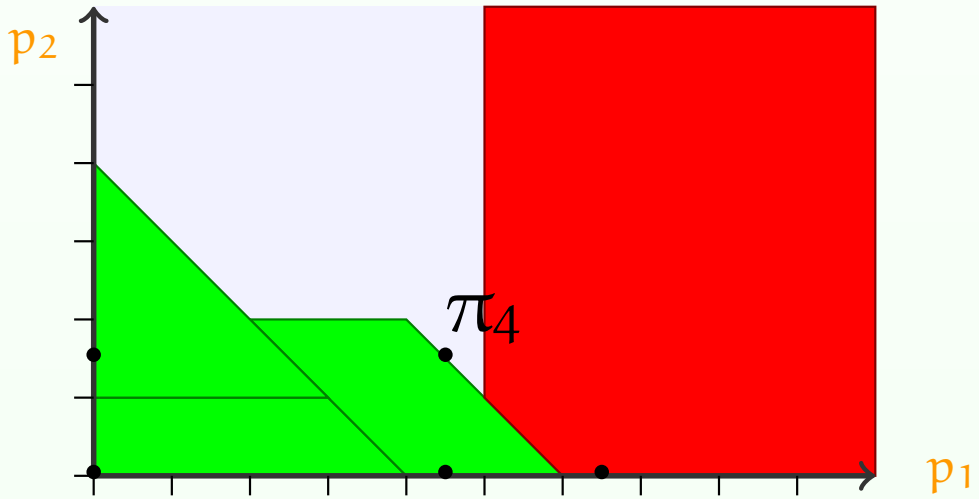
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



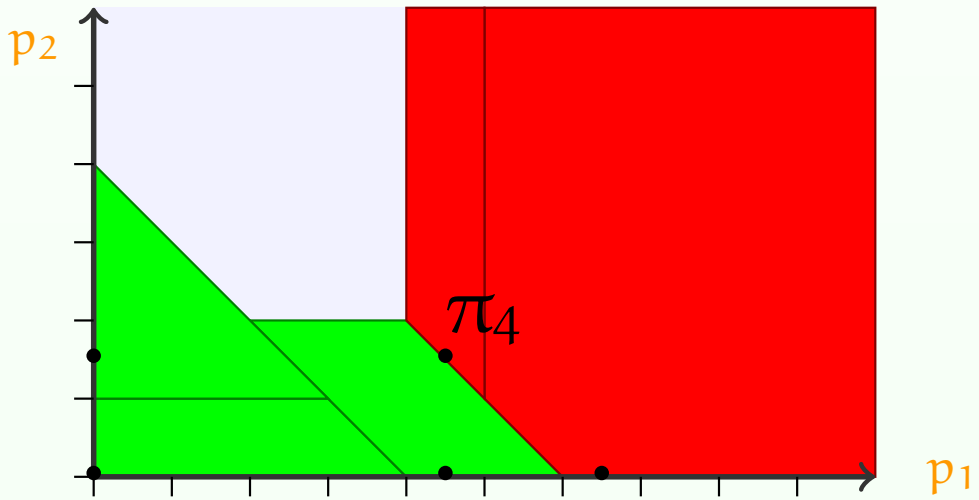
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



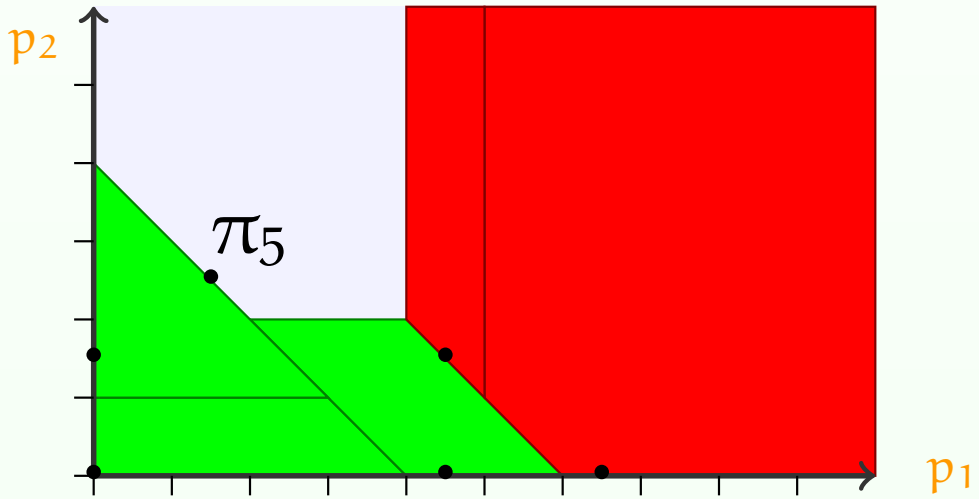
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



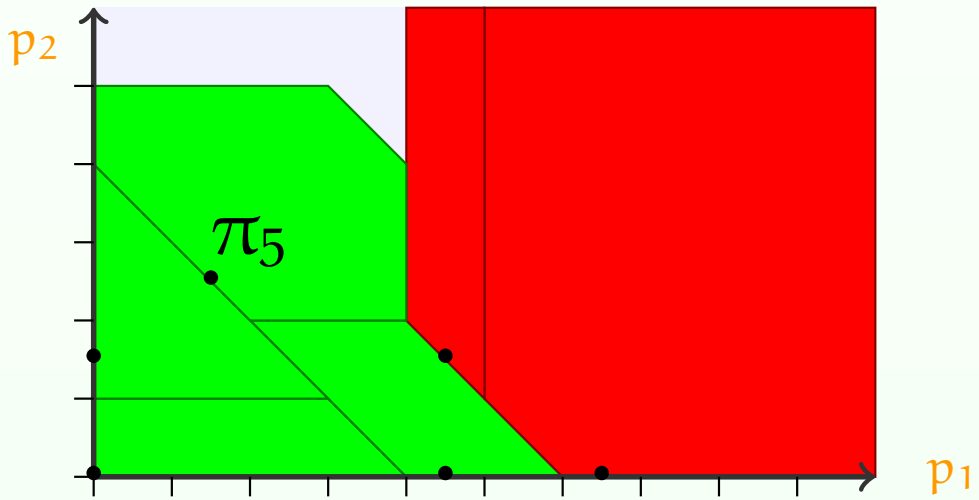
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



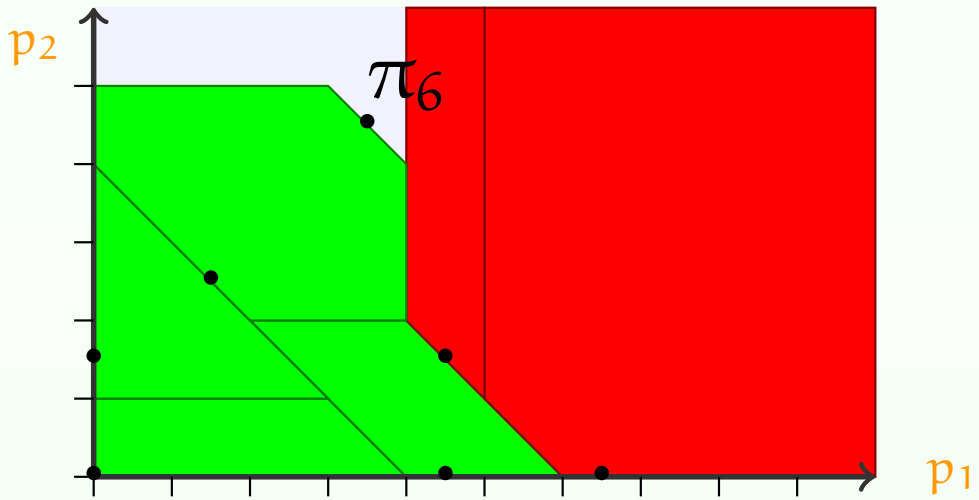
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



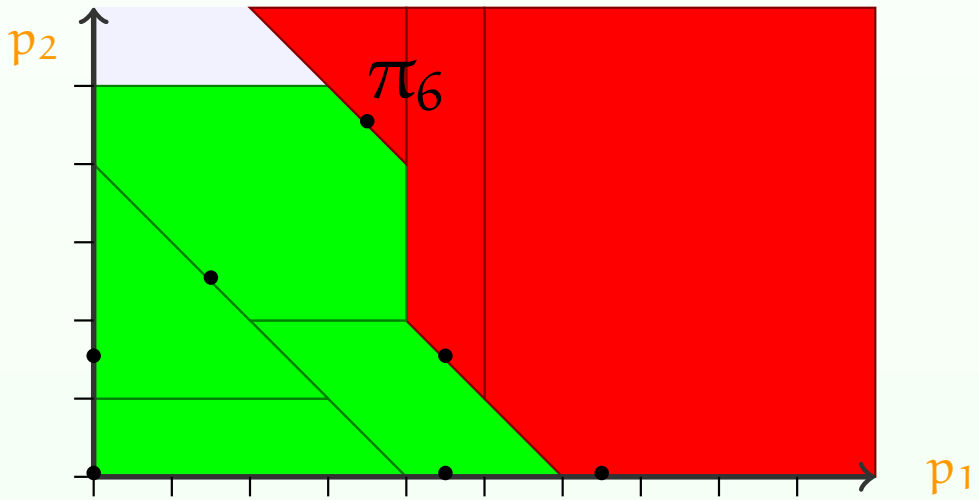
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



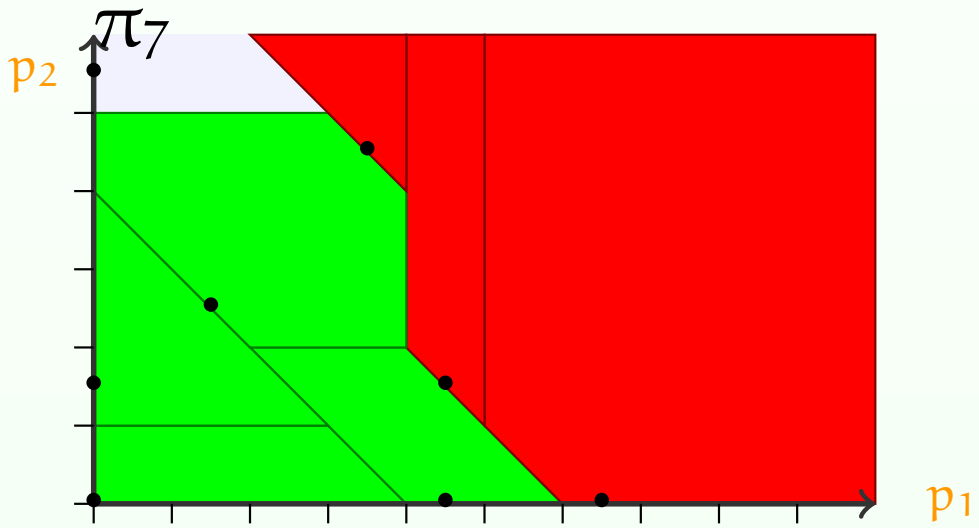
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



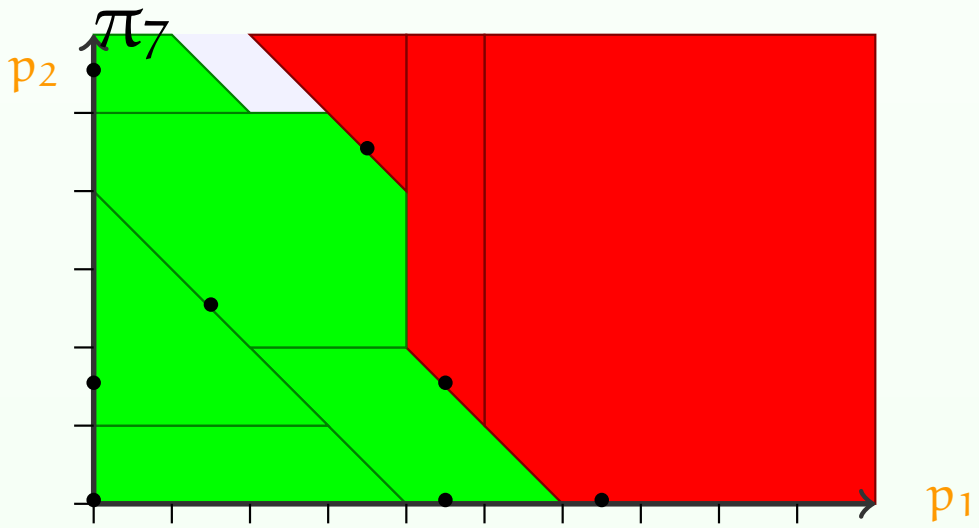
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



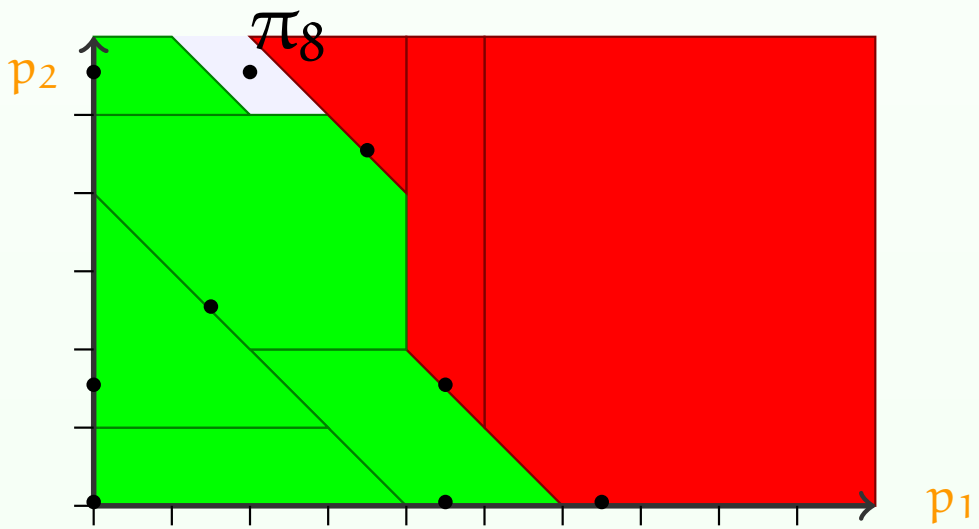
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



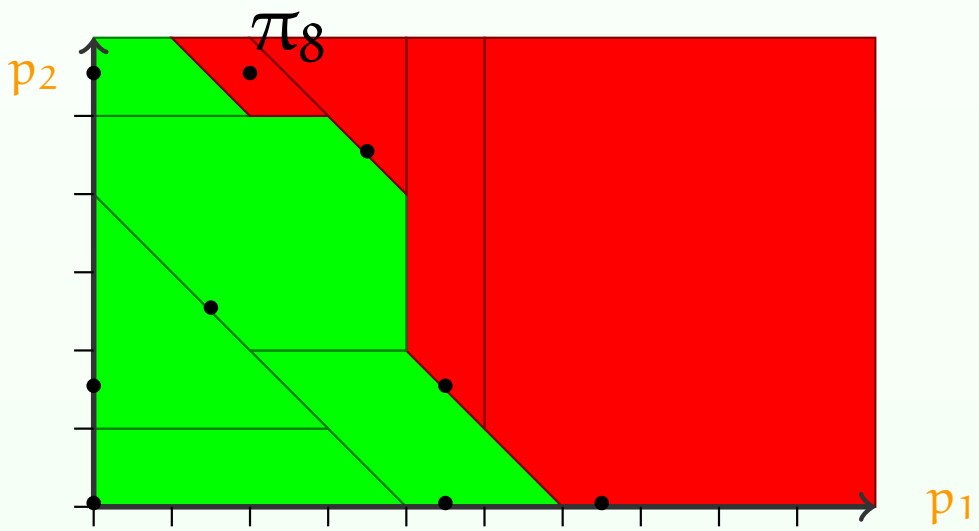
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



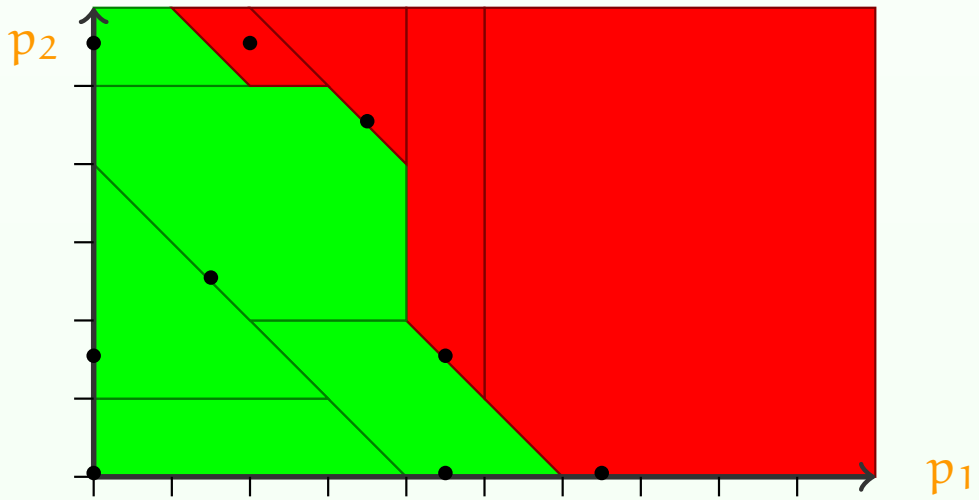
Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



Behavioral Cartography of a Model

- Principle: Tiling of the parametric space by iteratively calling IM [A. and Fribourg, 2010]
- Each tile has the same trace set
- Good or bad depending on the presence or not of deadline misses
- Good values for the parameters: union of all regions



Outline

Comparison

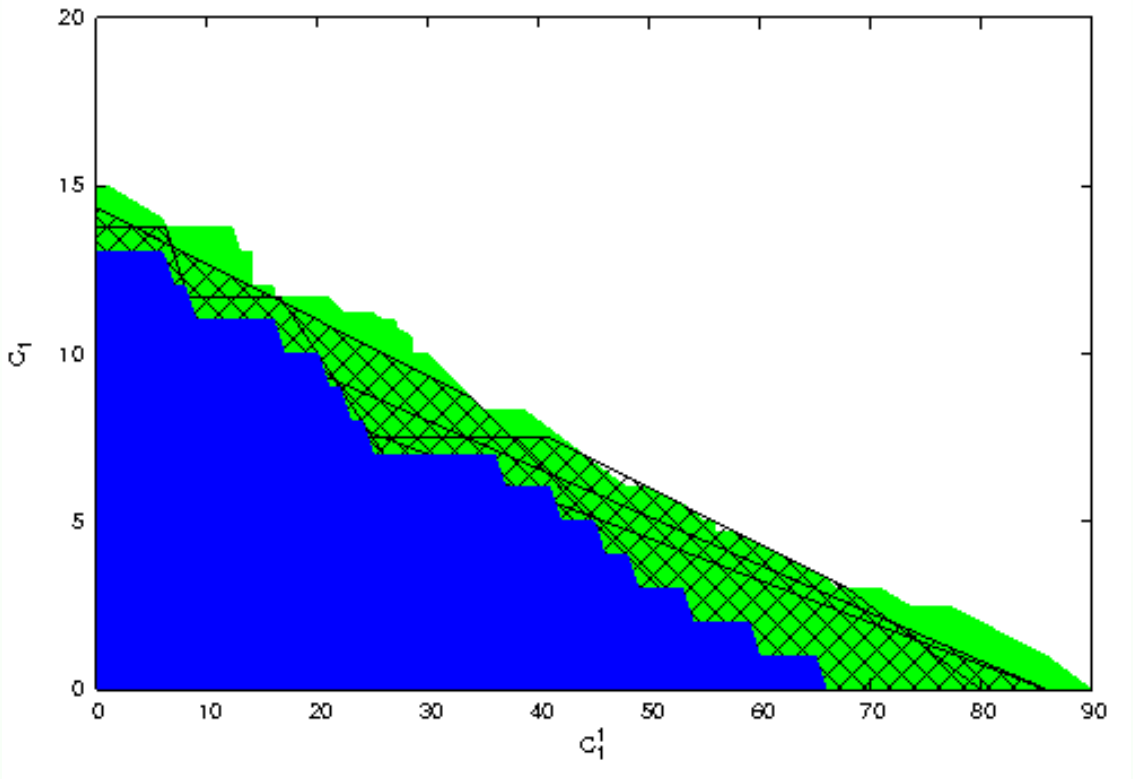
- We evaluate the effectiveness and the running time of the two proposed tools on two case studies.
  - Inverse method implemented in IMITATOR
  - Extension to the holistic analysis implemented in RTSCAN
- As a baseline comparison, we choose to also run the same kind of analysis using MAST [González Harbour et al., 2001]
  - Implements classical holistic analysis
  - Parametric analysis realized by computing schedulability for each possible combination of the values of parameters

First Case Study

- 2 processing nodes (1, 3)
- connected by a single bus (2)
- One pipeline with three tasks and 2 messages
- three other independent tasks (pipelines of length 1)

Pipeline/Task	T	D_{e2e}	Tasks	C	Priority	Processor
τ_1	20	20	-	free	9	1
P^1	150	150	τ^1_1	free	3	1
			τ^1_2	10	9	2
			τ^1_3	8	5	3
			τ^1_4	15	2	2
			τ^1_5	25	2	1
τ_2	30	30	-	6	9	3
τ_3	200	200	-	40	2	3

Results



Time for computing the region:

- RTScan(hatched): 0.27s
- Mast (blue, below): 7s
- Imitator (green, above): 19min42

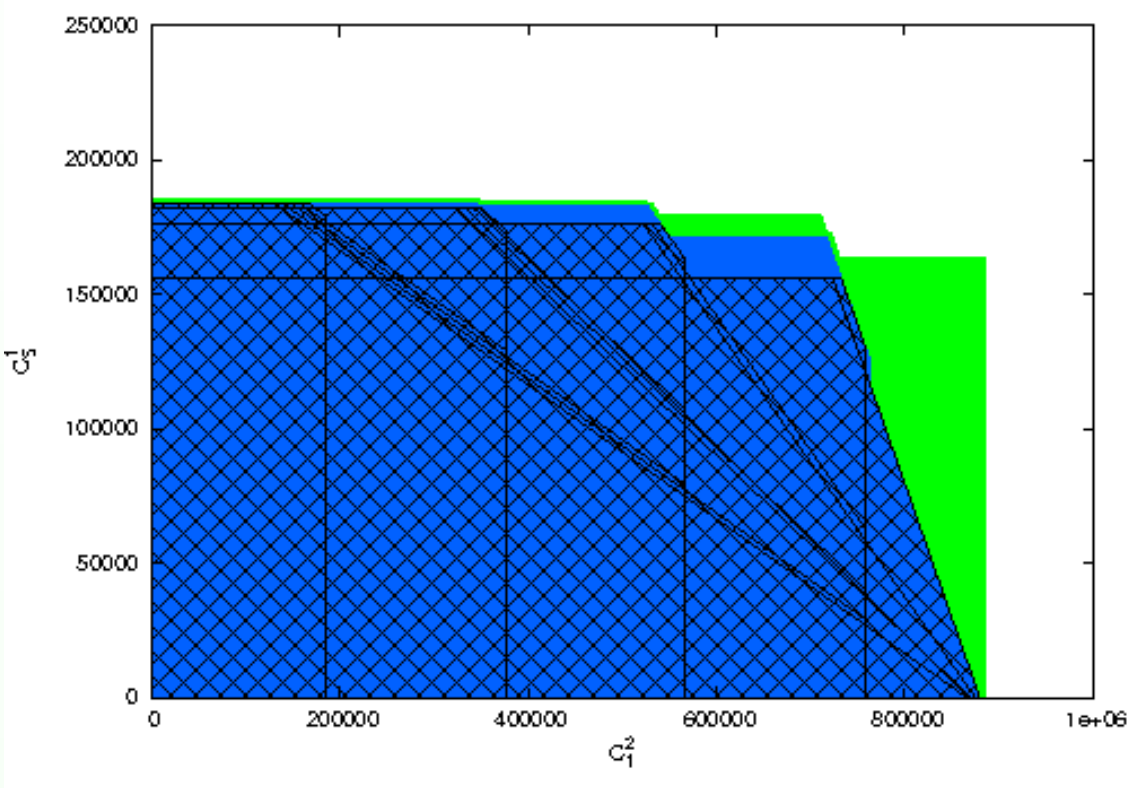
Second Case Study

- 3 processing nodes (1, 3, 4)
- connected by one bus (2)
- Two pipelines

Pipeline	T	D_{e2e}	Tasks	C	Priority	Processor
P^1	200 (30)	200	τ_1^1	4,546	10	1
			τ_2^1	445	10	2
			τ_3^1	9,091	10	4
			τ_4^1	445	9	2
			τ_5^1	free	9	1
P^2	3,000	1,000	τ_1^2	free	9	4
			τ_2^2	889	8	2
			τ_3^2	44,248	10	3
			τ_4^2	889	7	2
			τ_5^2	22,728	8	1

- Periods are in milliseconds, computation times are in microseconds

Results



Time for computing the region:

- RTScan: 0.47s
- Mast: 40min13 (with step of 10µs)
- Imitator: 2h08

Comments

- RTScan: very fast, but uses a lot of memory
  - A test with deadline larger than period easily runs out of memory (on a 8GiB PC)
  - Due to the non-convex nature of the problem
- Mast: no memory problem
  - However, not fully parametric
  - Depends on the tick size (i.e., precision)
- Imitator
  - Each "tile" is analyzed in a few seconds
  - However, a very long time is spent in moving to the next tile
  - We believe that there is much space for improvement here

Outline

Perspectives

- We are confident that the current methodology can be improved and be made more scalable
- Possible extensions:
  - Combine RTSCAN with IMITATOR
    - The first one could find a first approximate solution
    - The second one may refine the initial region by working on the frontier
  - Work on the tile exploration algorithm to improve performance

Bibliography

References I

Adbeddaïm, Y. and Maler, O. (2002).
Preemptive job-shop scheduling using stopwatch automata.
In Katoen, J.-P. and Stevens, P., editors, *Proceedings of the 8th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'02)*, volume 2280 of *Lecture Notes in Computer Science*, 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 *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC'93, pages 592–601, New York, NY, USA. ACM.

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

References II

André, É. and Fribourg, L. (2010).
Behavioral cartography of timed automata.
In Kučera, A. and Potapov, I., editors, *Proceedings of the 4th Workshop on Reachability Problems in Computational Models (RP'10)*, volume 6227 of *Lecture Notes in Computer Science*, pages 76–90. Springer.

André, É., Fribourg, L., Kühne, U., and Soulat, R. (2012).
IMITATOR 2.5: A tool for analyzing robustness in scheduling problems.
In Giannakopoulou, D. and Méry, D., editors, *Proceedings of the 18th International Symposium on Formal Methods (FM'12)*, volume 7436 of *Lecture Notes in Computer Science*, pages 33–36. Springer.

André, É., Hillah, L.-M., Hulin-Hubard, F., Kordon, F., Lembachar, Y., Linard, A., and Petrucci, L. (2013).
CosyVerif: An open source extensible verification environment.
In Liu, Y. and Martin, A., editors, *18th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'13)*, pages 33–36. IEEE Computer Society.

Bagnara, R., Hill, P. M., and Zaffanella, E. (2008).
The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems.
Science of Computer Programming, 72(1–2):3–21.

References III

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

González Harbour, M., Gutiérrez, J., Palencia, J., and Drake, J. (2001).
MAST: Modeling and analysis suite for real-time applications.
In *ECRTS*.

Tindell, K. and Clark, J. (1994).
Holistic schedulability analysis for distributed hard real-time systems.
Microprocess. Microprogram., 40(2-3):117–134.

Additional explanation

Semantics of Parametric Timed Automata

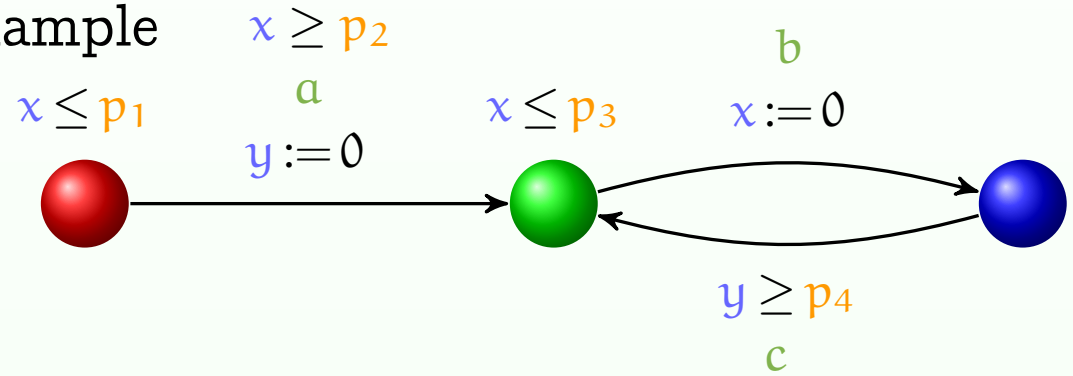
- State of a PTA: pair (l, C), where
  - l is a location (e.g., ),
  - C is a constraint (conjunction of inequalities) over X and P

Semantics of Parametric Timed Automata

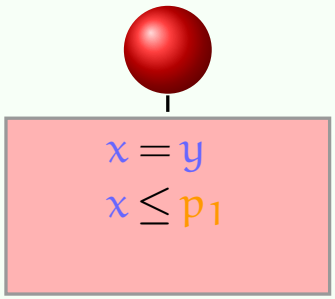
- State of a PTA: pair (l, C), where
  - l is a location (e.g., ),
  - C is a constraint (conjunction of inequalities) over X and P
- Path: alternating sequence of states and actions

Semantics of Parametric Timed Automata

- State of a PTA: pair (l, C), where
  - l is a location (e.g., ),
  - C is a constraint (conjunction of inequalities) over X and P
- Path: alternating sequence of states and actions
- Example

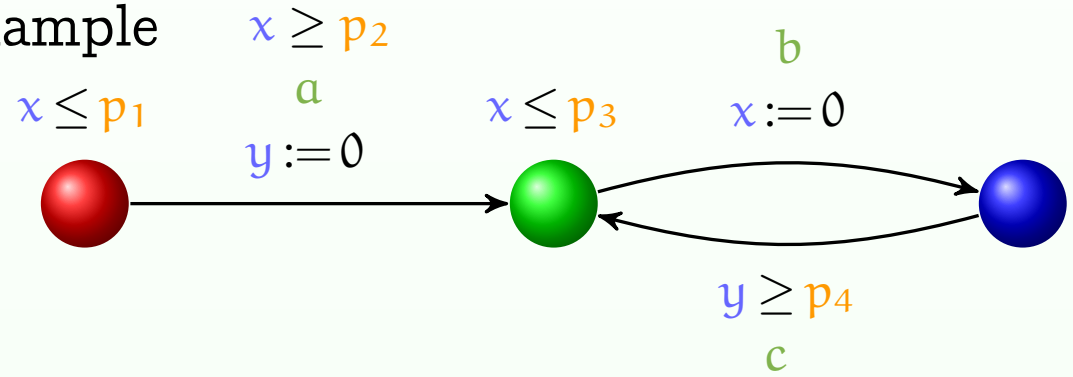


  - Possible path for this PTA

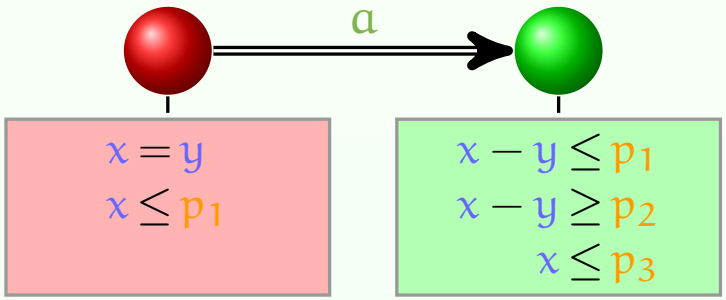


Semantics of Parametric Timed Automata

- State of a PTA: pair (l, C), where
  - l is a location (e.g., ),
  - C is a constraint (conjunction of inequalities) over X and P
- Path: alternating sequence of states and actions
- Example

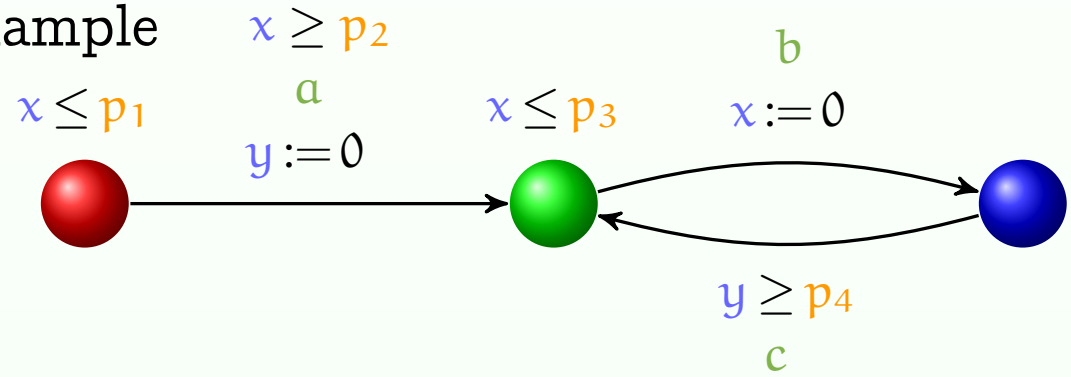


  - Possible path for this PTA

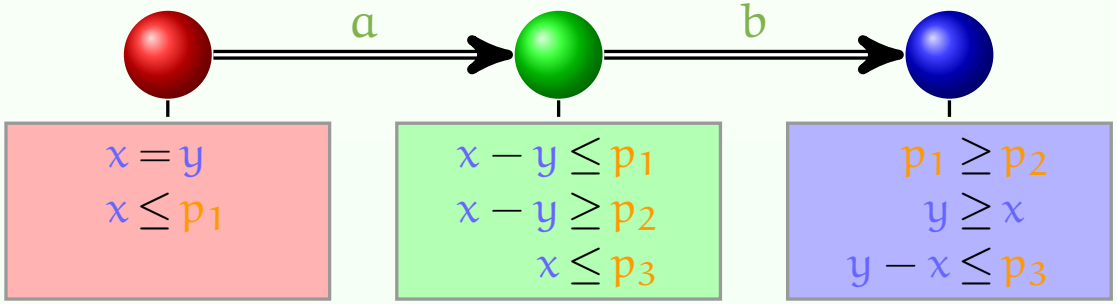


Semantics of Parametric Timed Automata

- State of a PTA: pair (l, C), where
  - l is a location (e.g., ),
  - C is a constraint (conjunction of inequalities) over X and P
- Path: alternating sequence of states and actions
- Example



  - Possible path for this PTA



The Inverse Method: Algorithm

Algorithm 1: $IM(\mathcal{A}, \pi)$

input : PTA $\mathcal{A}$ of initial state s_0, parameter valuation π
output: Constraint K over the parameters

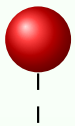
1 $i \leftarrow 0$; $K_c \leftarrow \texttt{true}$; $S_{new} \leftarrow \{s_0\}$; $S \leftarrow \{\}$
2 **while** `true` **do**
3 **while** *there are* π*-incompatible states in* S_{new} **do**
4 Select a π-incompatible state (l, C) of S_{new} (i.e., s.t. $\pi \not\models C$) ;
5 Select a π-incompatible J in $C{\downarrow_P}$ (i.e., s.t. $\pi \not\models J$) ;
6 $K_c \leftarrow K_c \wedge \neg J$; $S \leftarrow \bigcup_{j=0}^{i-1} Post^j_{\mathcal{A}(K_c)}(\{s_0\})$; $S_{new} \leftarrow Post_{\mathcal{A}(K_c)}(S)$;
7 **if** $S_{new} \sqsubseteq S$ **then return** $K \leftarrow \bigcap_{(l,C)\in S} C{\downarrow_P}$
8 $i \leftarrow i+1$; $S \leftarrow S \cup S_{new}$; $S_{new} \leftarrow Post_{\mathcal{A}(K_c)}(S)$

The Inverse Method: Example

π_0 :

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

$K = \texttt{true}$



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

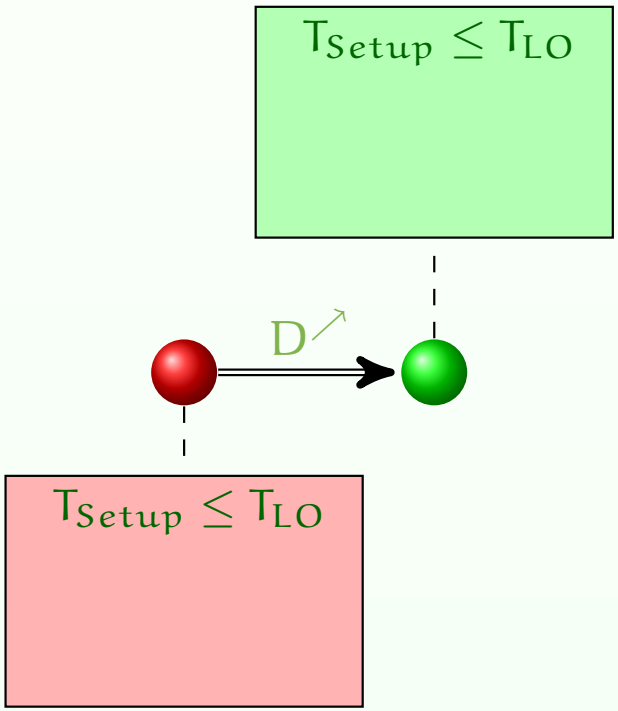
Example of a flip-flop circuit [Clarisó and Cortadella, 2007]

The Inverse Method: Example

π_0 :

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

$K = \texttt{true}$



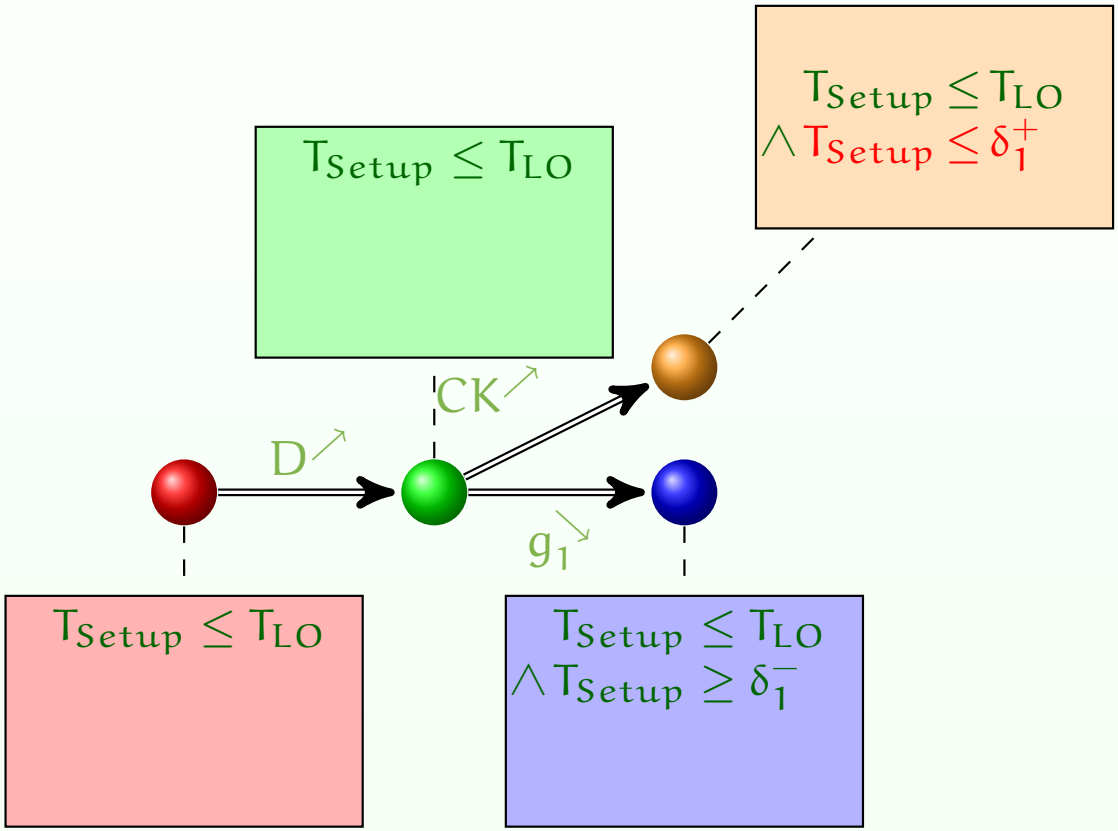
Example of a flip-flop circuit [Clarisó and Cortadella, 2007]

The Inverse Method: Example

π_0 :

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

$K = \texttt{true}$



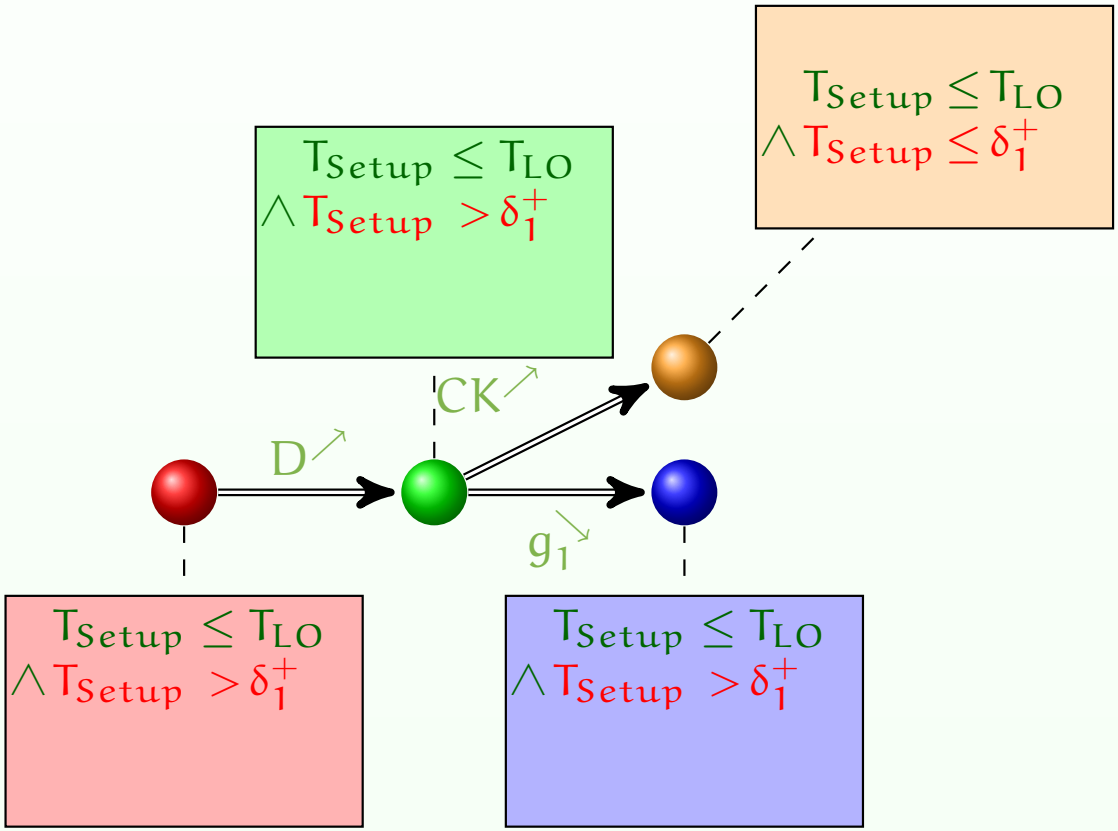
Example of a flip-flop circuit [Clarisó and Cortadella, 2007]

The Inverse Method: Example

π_0 :

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

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



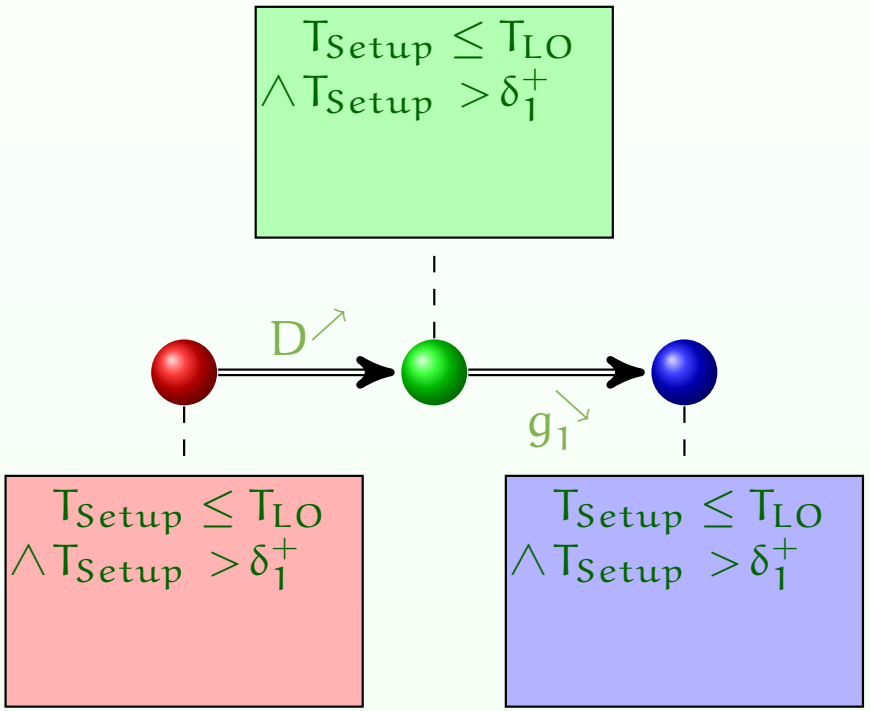
Example of a flip-flop circuit [Clarisó and Cortadella, 2007]

The Inverse Method: Example

π_0 :

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

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



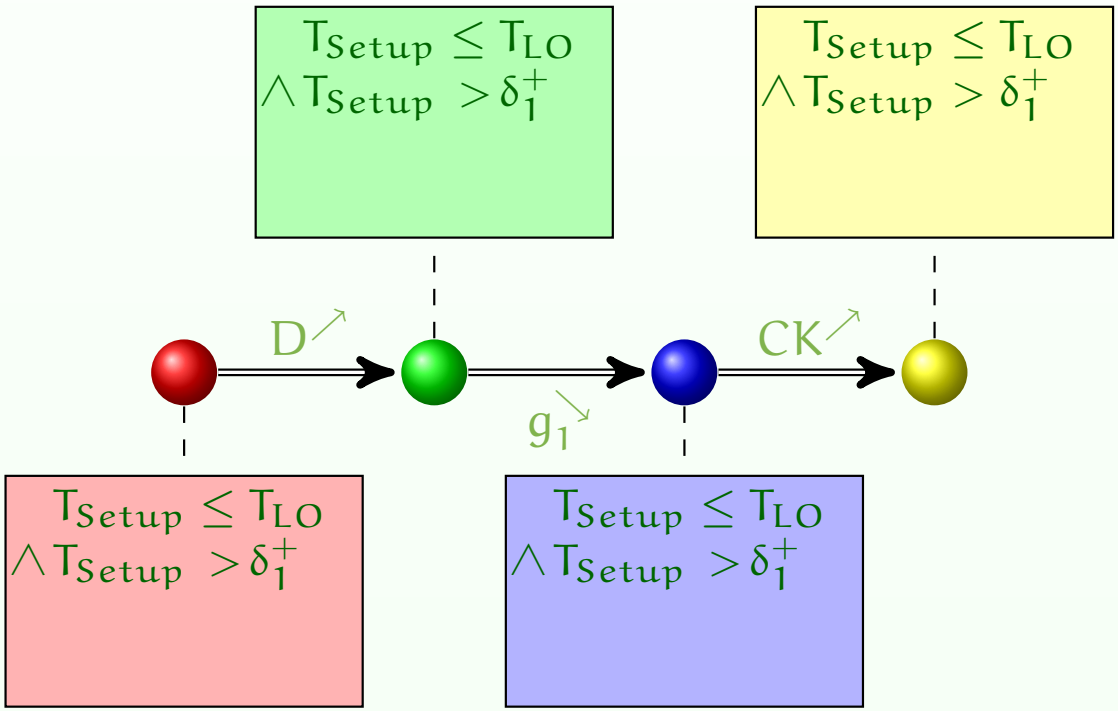
Example of a flip-flop circuit [Clarisó and Cortadella, 2007]

The Inverse Method: Example

π_0 :

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

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



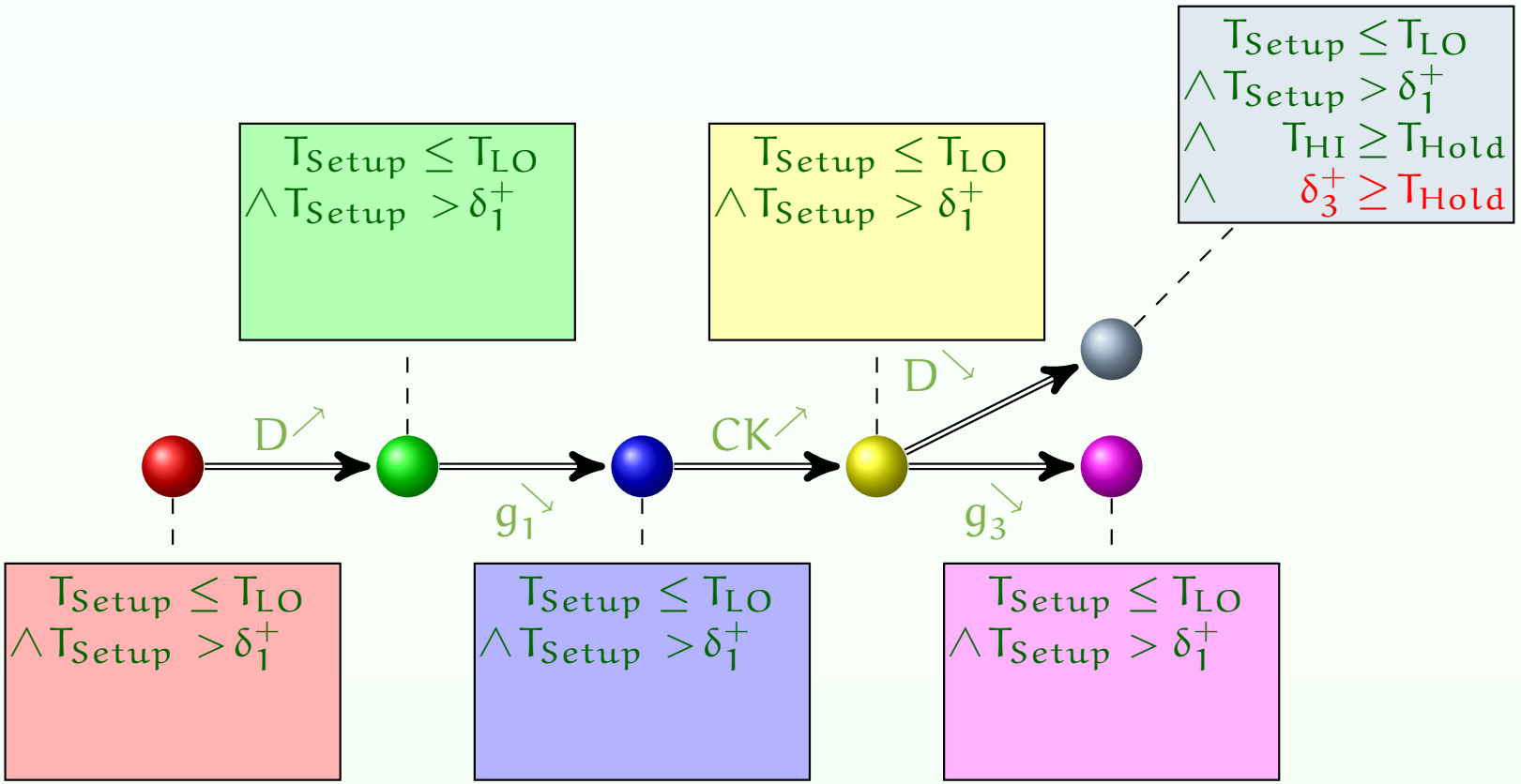
Example of a flip-flop circuit [Clarisó and Cortadella, 2007]

The Inverse Method: Example

π_0 :

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

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



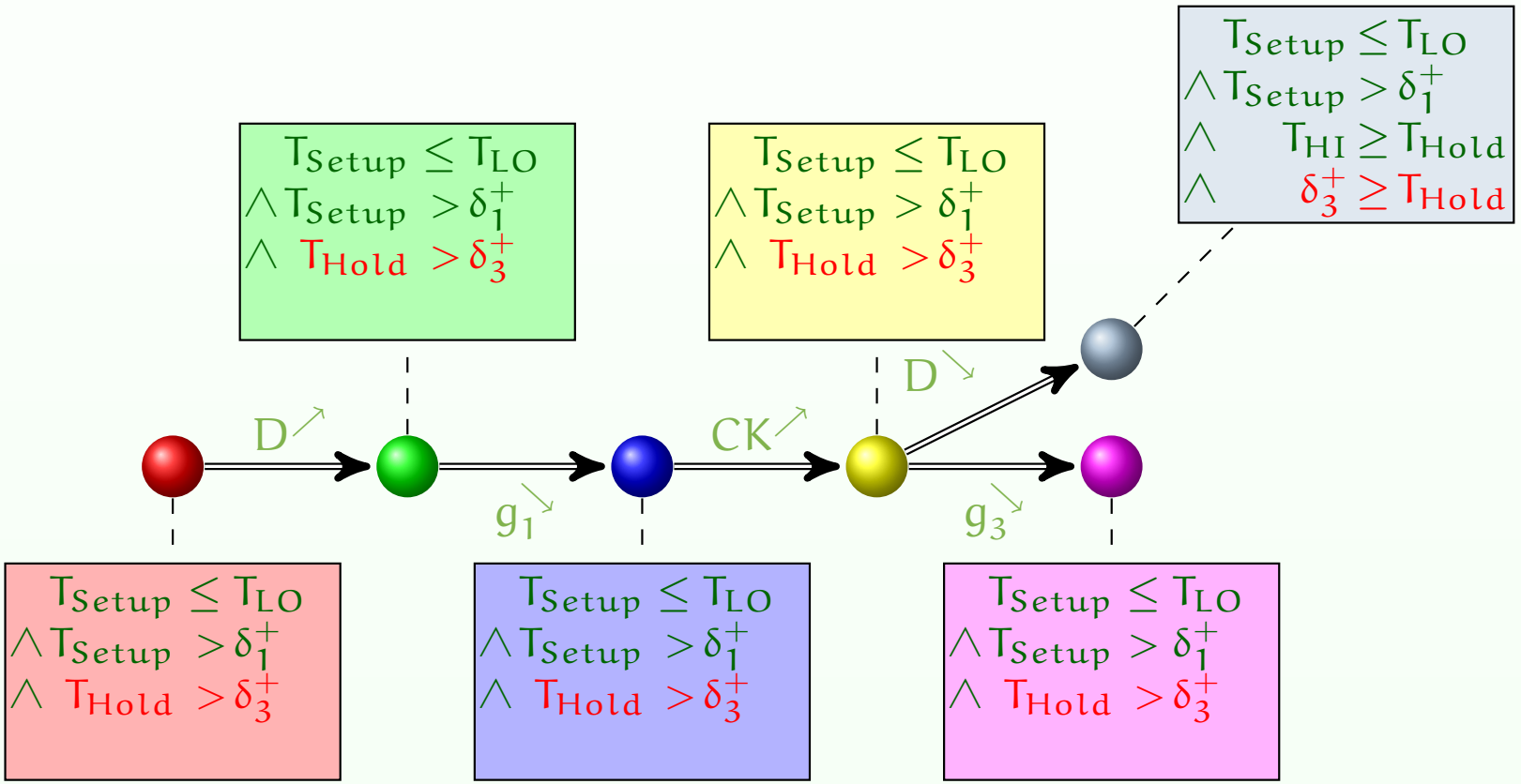
Example of a flip-flop circuit [Clarisó and Cortadella, 2007]

The Inverse Method: Example

π_0 :

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

$K = T_{Setup} > \delta_1^+ \wedge T_{Hold} > \delta_3^+$



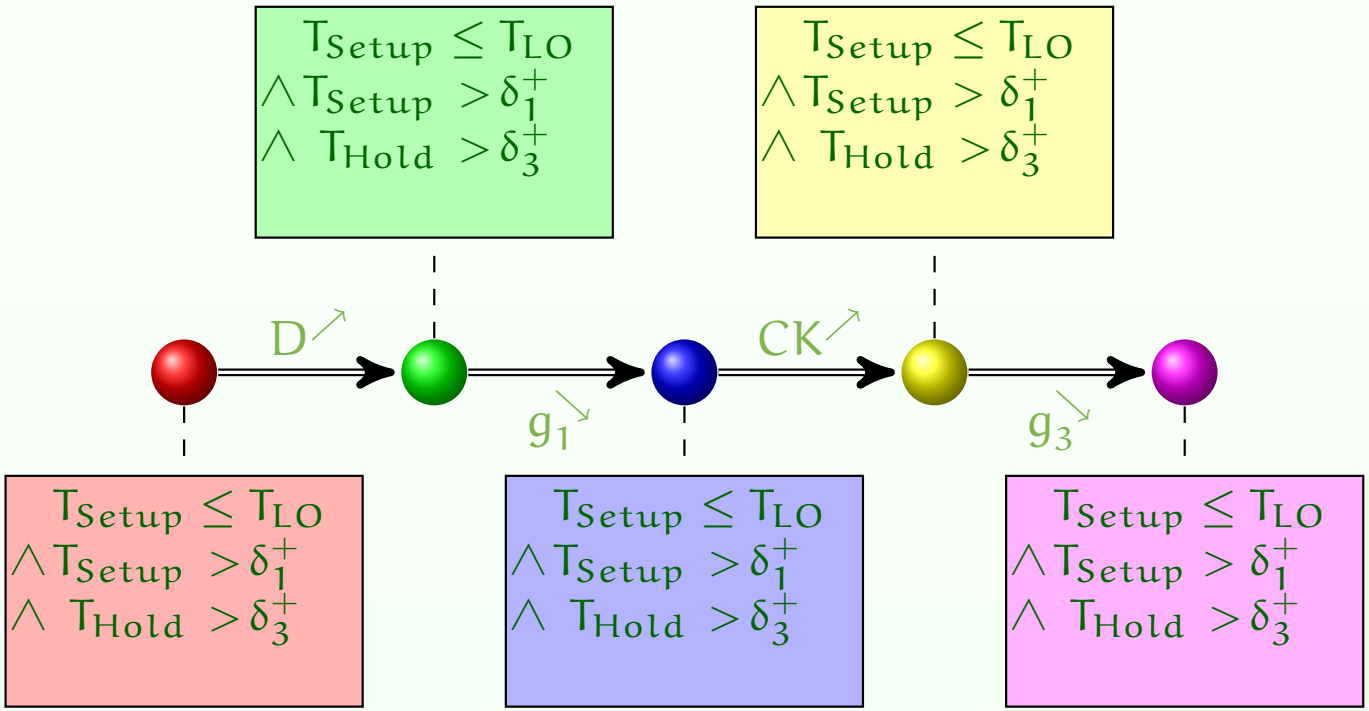
Example of a flip-flop circuit [Clarisó and Cortadella, 2007]

The Inverse Method: Example

π_0 :

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

$K =$
$T_{Setup} > \delta_1^+$
$\wedge\ T_{Hold} > \delta_3^+$



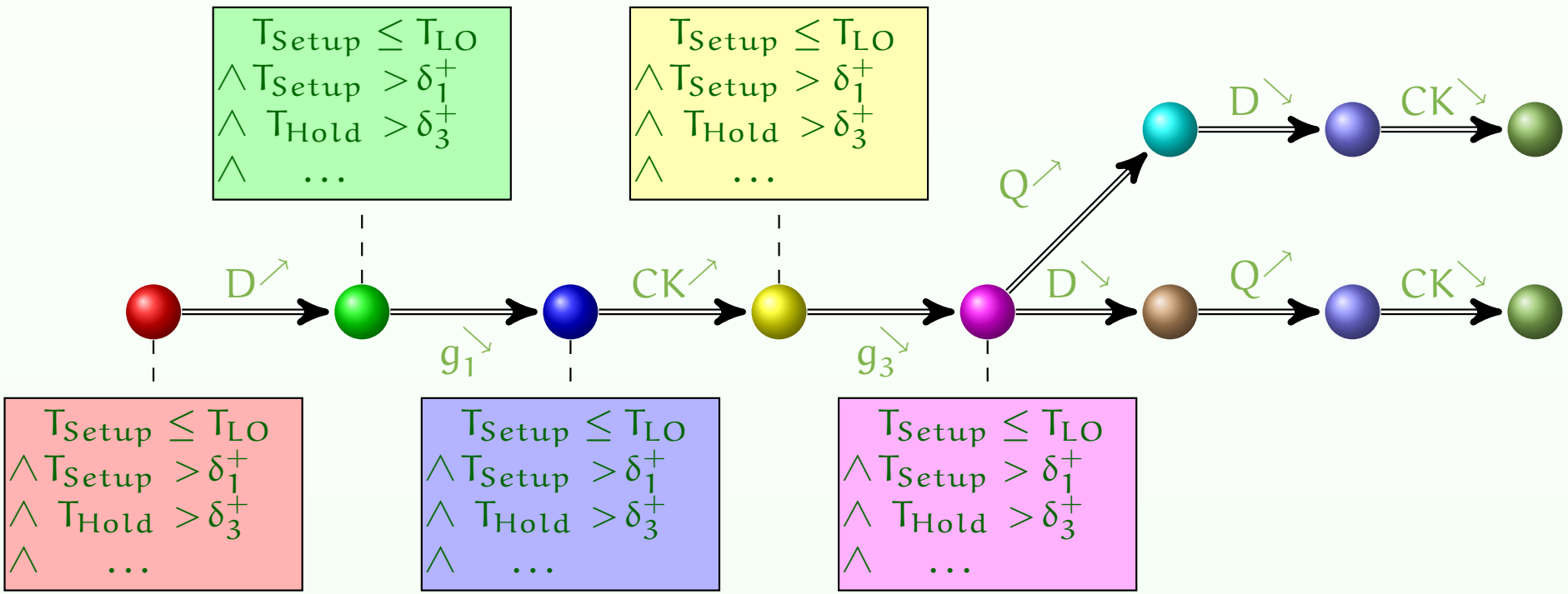
Example of a flip-flop circuit [Clarisó and Cortadella, 2007]

The Inverse Method: Example

π_0 :

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

$$
\begin{aligned}
K = \quad & T_{Setup} > \delta_1^+ \wedge \delta_3^+ + \delta_4^+ \geq T_{Hold} \\
\wedge \; & T_{Hold} > \delta_3^+ \wedge \delta_3^+ + \delta_4^+ < T_{HI} \\
\wedge \; & T_{Setup} \leq T_{LO} \wedge \delta_3^- + \delta_4^- \leq T_{Hold} \\
\wedge \; & \delta_1^- > 0
\end{aligned}
$$



Example of a flip-flop circuit [Clarisó and Cortadella, 2007]

Licensing

Source of the graphics used

Title: Renault Twizy
Author: Citron
Source: https://commons.wikimedia.org/wiki/File:Renault_Twizy.jpg
License: CC BY-SA 3.0

Title: Airbus A380
Author: Axwel
Source: https://commons.wikimedia.org/wiki/File:Airbus_A380_blue_sky.jpg
License: CC BY 2.0

License of this document

Authors: Étienne André and Giuseppe Lipari