INFINITY '2010

Singapore

21st September 2010

IMITATOR II

A Tool for Solving the Good Parameters Problem in Timed Automata

Étienne André

Laboratoire Spécification et Vérification
LSV, ENS de Cachan & CNRS, France

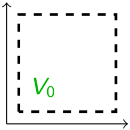
The Good Parameters Problem

- Context: Verification of timed systems
  - Use of timing parameters (unknown constants)
  - Model of Parametric Timed Automata (PTA)

The Good Parameters Problem

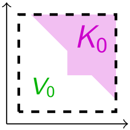
- Context: Verification of timed systems
  - Use of timing parameters (unknown constants)
  - Model of Parametric Timed Automata (PTA)

- The good parameters problem: [Frehse et al., 2008]
  - "Given a bounded parameter domain V_0, find a dense set of points (timing parameters) of good behavior in V_0 (ideally the largest one)"



The Good Parameters Problem

- Context: Verification of timed systems
  - Use of timing parameters (unknown constants)
  - Model of Parametric Timed Automata (PTA)
- The good parameters problem: [Frehse et al., 2008]
  - "Given a bounded parameter domain V_0, find a dense set of points (timing parameters) of good behavior in V_0 (ideally the largest one)"



Classical approaches

- Verification of the property on a set of discrete points
  - Drawback: would need an infinite number of verifications to obtain a dense set of points
- Computation of all the reachable states of a PTA, and intersection with the set of bad states [Alur et al., 1995]
  - Drawback: too costly in practice
- Approach based on CEGAR [Clarke et al., 2000, Frehse et al., 2008]
  - Drawback: underapproximation

Classical approaches

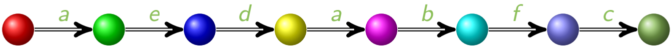
- Verification of the property on a set of discrete points
  - Drawback: would need an infinite number of verifications to obtain a dense set of points
- Computation of all the reachable states of a PTA, and intersection with the set of bad states [Alur et al., 1995]
  - Drawback: too costly in practice
- Approach based on CEGAR [Clarke et al., 2000, Frehse et al., 2008]
  - Drawback: underapproximation

- New approach implemented in IMITATOR II
  - Method of behavioral cartography

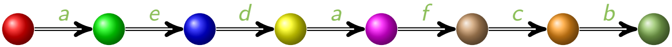
Good and Bad Traces

- Trace over a PTA: finite alternating sequence of locations and actions (time-abstract run)

- A trace is said to be good if it verifies a given property
  - Example of property ϕ: "b always occurs before c"
  - Example of good trace w.r.t. ϕ



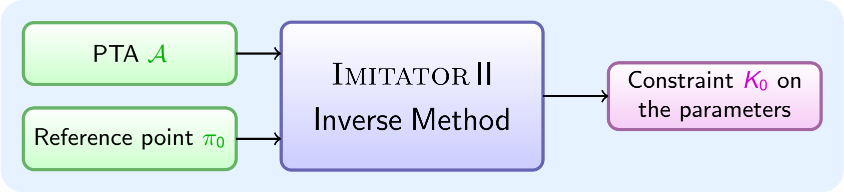
  - Example of bad trace w.r.t. ϕ



Outline

Outline

The Inverse Method (1/2)

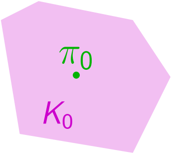


The Inverse Method (2/2)

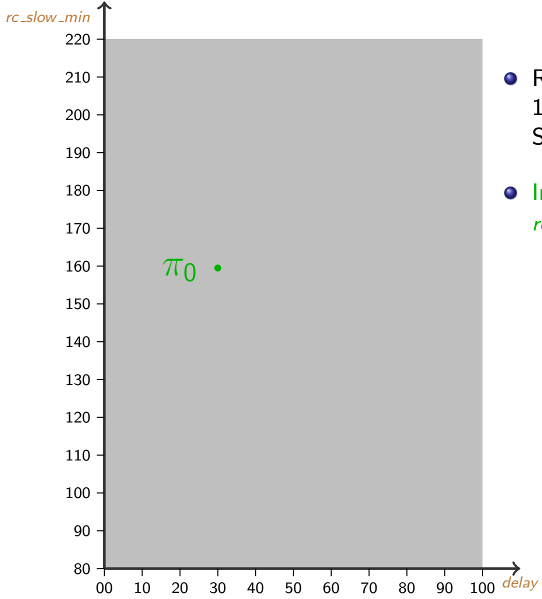
- Input
  - A PTA $\mathcal{A}$
  - A reference valuation π_0 of all the parameters of $\mathcal{A}$

The Inverse Method (2/2)

- Input
  - A PTA $\mathcal{A}$
  - A reference valuation π_0 of all the parameters of $\mathcal{A}$
- Output: tile K_0
  - Convex constraint on the parameters such that
    - $\pi_0 \models K_0$
    - For all point $\pi \models K_0$, $\mathcal{A}$ under π has the same trace set as for π_0 [André et al., 2009]



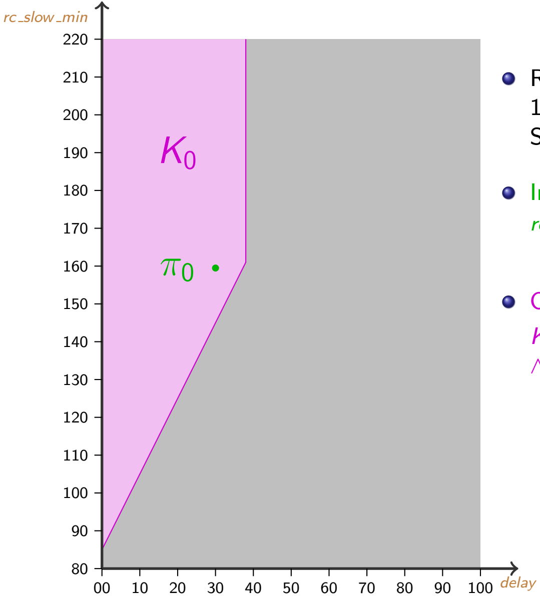
Application to the Root Contention Protocol



- Root contention protocol of the IEEE 1394 ("FireWire") High Performance Serial Bus [Hune et al., 2002]
- Input: IEEE reference valuation

$$rc\_slow\_min = 159ns$$
$$delay = 30ns$$

Application to the Root Contention Protocol



- Root contention protocol of the IEEE 1394 ("FireWire") High Performance Serial Bus [Hune et al., 2002]
- Input: IEEE reference valuation

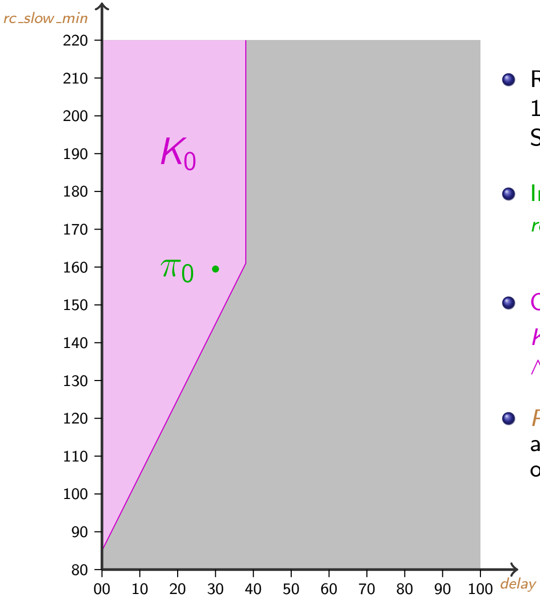
  $rc\_slow\_min = 159ns$

  $delay = 30ns$
- Output:

  $K_0 : 2delay < 76$

  $\wedge \quad 2delay + 85 < rc\_slow\_min$

Application to the Root Contention Protocol



- Root contention protocol of the IEEE 1394 ("FireWire") High Performance Serial Bus [Hune et al., 2002]
- Input: IEEE reference valuation
  $rc\_slow\_min = 159ns$
  $delay = 30ns$
- Output:
  $K_0 : 2delay < 76$
  $\wedge \ 2delay + 85 < rc\_slow\_min$
- $Prop_3$: The minimum probability that a leader is elected after three rounds or less is greater or equal to 0.75
  - For all $\pi \models K_0$, $Prop_3$ is satisfied

Outline

The Behavioral Cartography Algorithm

- Goal: Find the maximal set of points corresponding to a good behavior

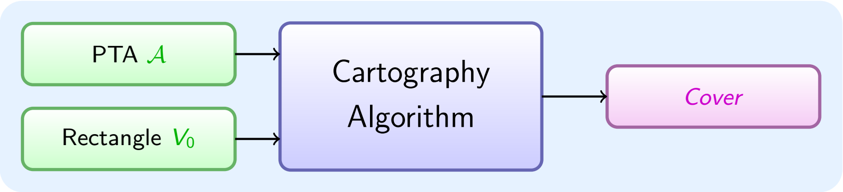
The Behavioral Cartography Algorithm

- Goal: Find the maximal set of points corresponding to a good behavior

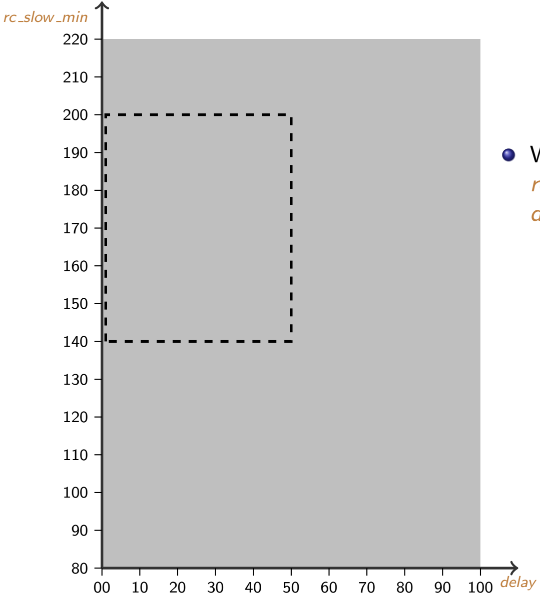
- Method: Iterate the inverse method for all the integer points of a given rectangle V_0

The Behavioral Cartography Algorithm

- Goal: Find the maximal set of points corresponding to a good behavior
- Method: Iterate the inverse method for all the integer points of a given rectangle V_0
- Output: set of tiles for all the integer points of V_0
  - $\rightsquigarrow$ behavioral cartography of the parameter space [André and Fribourg, 2010]

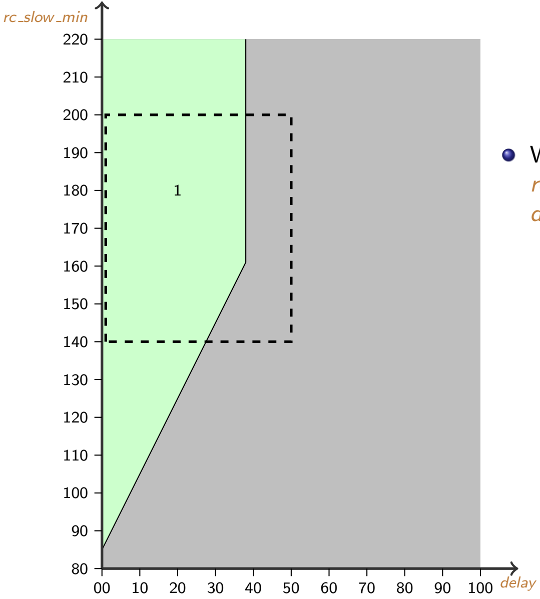


The Root Contention Protocol: Cartography



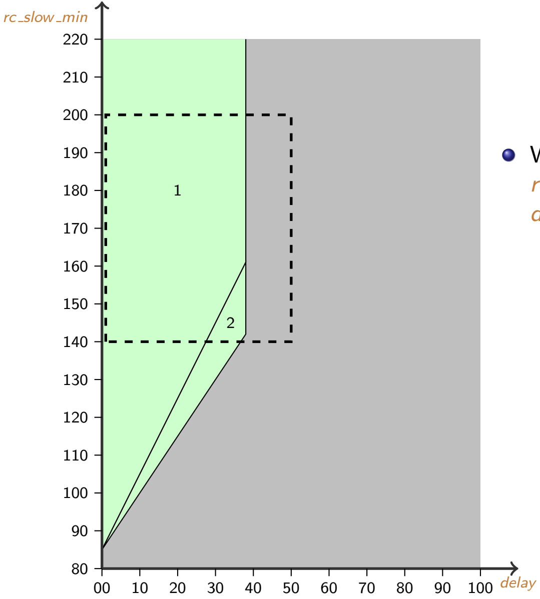
- We consider the following V_0 : *rc_slow_min* $\in [140; 200]$ and *delay* $\in [1; 50]$

The Root Contention Protocol: Cartography



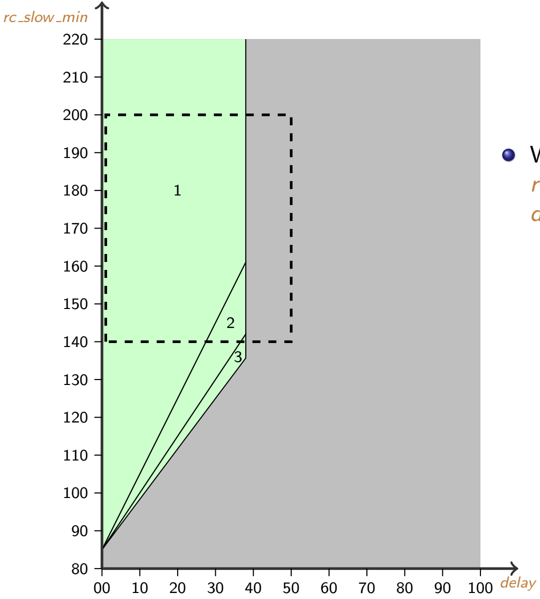
- We consider the following V_0 : $rc\_slow\_min \in [140; 200]$ and $delay \in [1; 50]$

The Root Contention Protocol: Cartography



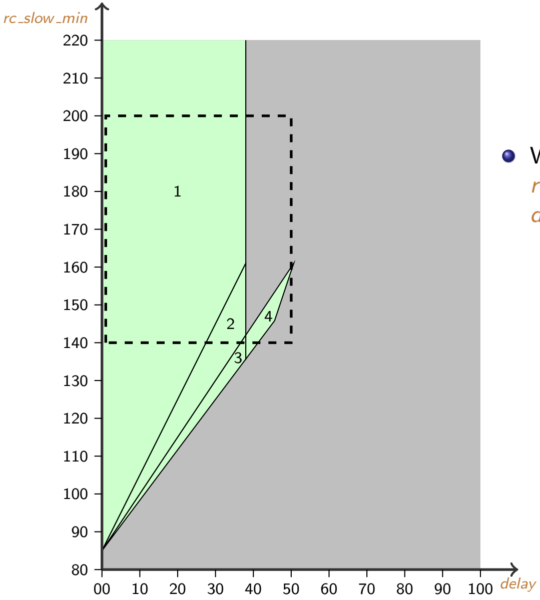
- We consider the following V_0 : *rc_slow_min* $\in [140; 200]$ and *delay* $\in [1; 50]$

The Root Contention Protocol: Cartography



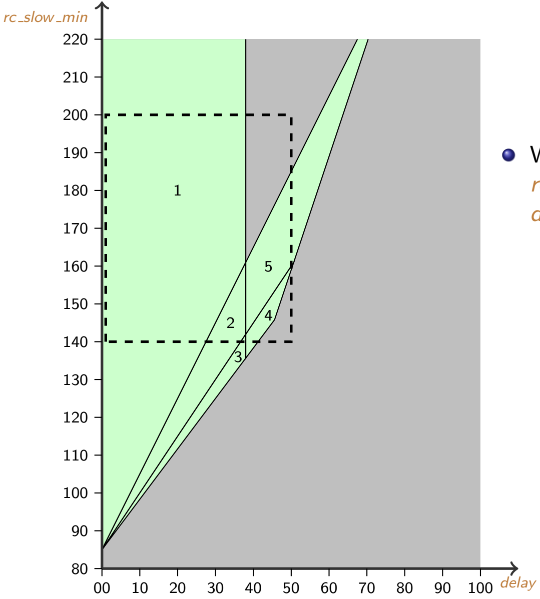
- We consider the following V_0 : $rc\_slow\_min \in [140; 200]$ and $delay \in [1; 50]$

The Root Contention Protocol: Cartography



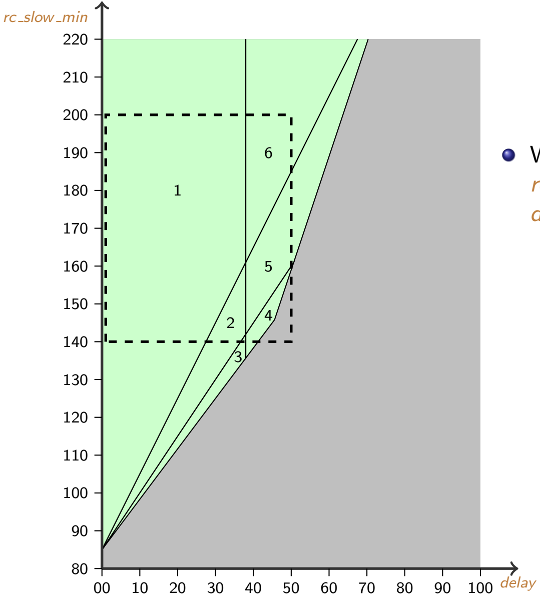
- We consider the following V_0 : *rc_slow_min* $\in [140; 200]$ and *delay* $\in [1; 50]$

The Root Contention Protocol: Cartography



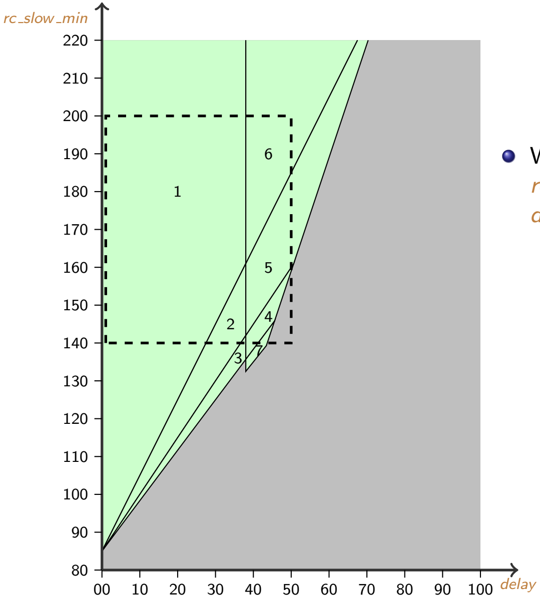
- We consider the following V_0 : *rc_slow_min* $\in [140; 200]$ and *delay* $\in [1; 50]$

The Root Contention Protocol: Cartography



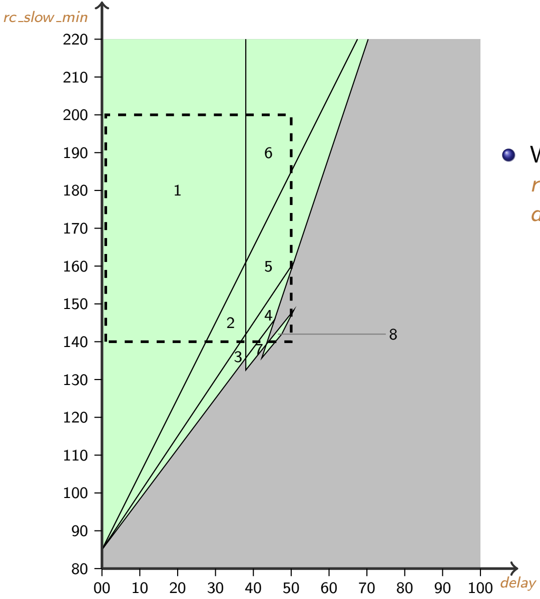
- We consider the following V_0 : *rc_slow_min* $\in [140; 200]$ and *delay* $\in [1; 50]$

The Root Contention Protocol: Cartography



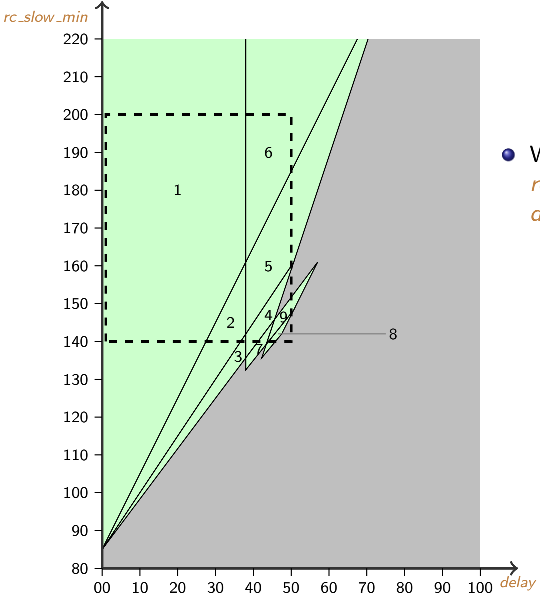
- We consider the following V_0 : *rc_slow_min* $\in [140; 200]$ and *delay* $\in [1; 50]$

The Root Contention Protocol: Cartography



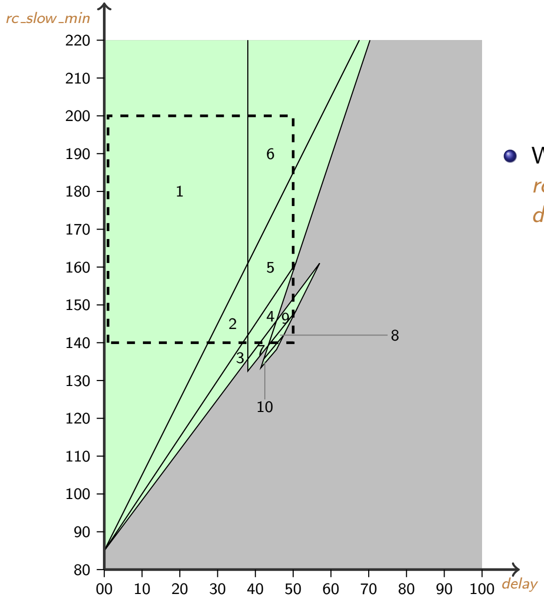
- We consider the following V_0 : $rc\_slow\_min \in [140; 200]$ and $delay \in [1; 50]$

The Root Contention Protocol: Cartography



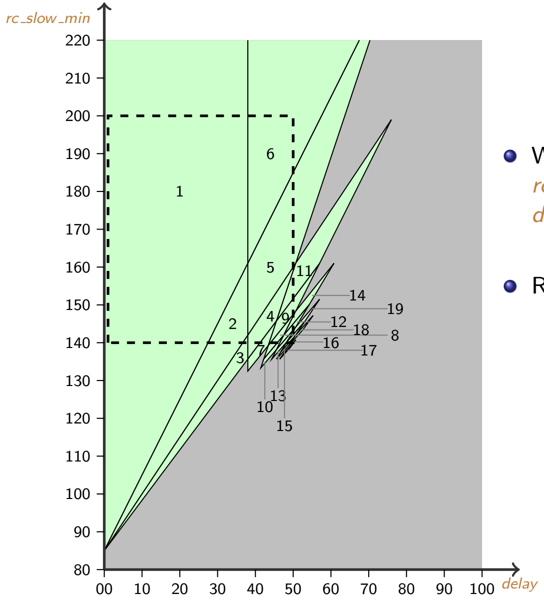
- We consider the following V_0 : *rc_slow_min* $\in [140; 200]$ and *delay* $\in [1; 50]$

The Root Contention Protocol: Cartography



- We consider the following V_0 : *rc_slow_min* $\in [140; 200]$ and *delay* $\in [1; 50]$

The Root Contention Protocol: Cartography

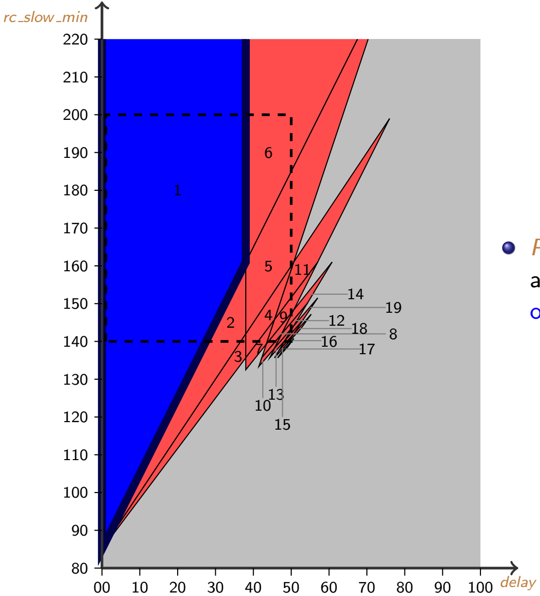


- We consider the following V_0 : *rc_slow_min* $\in [140; 200]$ and *delay* $\in [1; 50]$
- Remarks
  - Tiles 1 and 6 are infinite towards one dimension
  - The cartography does not cover the whole real-valued space within V_0 (holes in the lower right corner of V_0)

Partition into Good and Bad Tiles

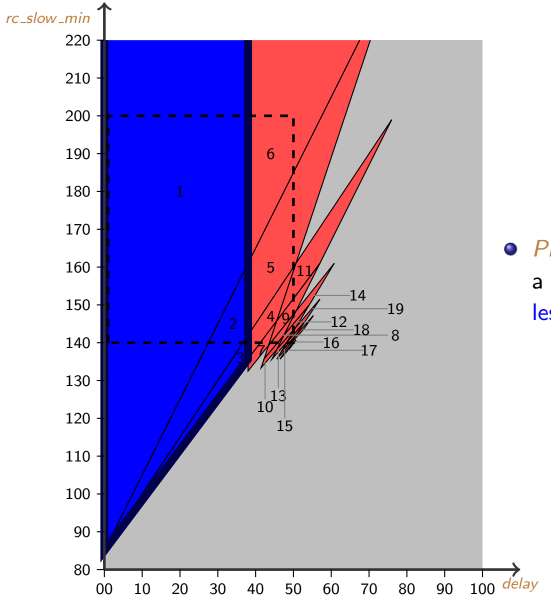
- A tile is said to be good if all its corresponding traces are good
- According to the nature of the trace sets, we can partition the tiles into good and bad ones

The Root Contention Protocol: Partition (1/2)



- *Prop*$_3$: "The minimum probability that a leader is elected after three rounds or less is greater or equal to 0.75"
  - Good tile: 1
  - Bad tiles: 2 to 19

The Root Contention Protocol: Partition (2/2)



- *$Prop_5$*: "The minimum probability that a leader is elected after five rounds or less is greater or equal to 0.75"
  - Good tiles: 1, 2, 3
  - Bad tiles: 4 to 19

Outline

Implementation

- In short
  - IMITATOR II: new improved version of IMITATOR
  - "Inverse Method for Inferring Time AbstracT BehaviOR"
  - 8000 lines of code
  - Program written in OCaml
  - Makes use of the PPL library for handling polyhedra [Bagnara et al., 2008]
- Some features
  - List of tiles with their corresponding trace set under a graphical form
  - Cartography under a graphical form (for 2 parameter dimensions)
- IMITATOR II is available on its Web page
  - `http://www.lsv.ens-cachan.fr/~andre/IMITATOR2`

Case Studies

- Application to various case studies
  - Asynchronous circuits
  - Communication protocols
- Computation time for the cartography algorithm
  - Experiments conducted on an Intel Core2 Duo 2.4 GHz with 2 Gb

Example	PTAs	loc./PTA	$\|X\|$	$\|P\|$	$\|V_0\|$	tiles	states	trans.	Time (s)
SR-latch	3	$[3, 8]$	3	3	1331	6	5	4	0.3
Flip-flop	5	$[4, 16]$	5	2	644	8	15	14	3
Latch circuit	7	$[2, 5]$	8	4	73062	5	21	20	96.3
And–Or	3	$[4, 8]$	4	6	75600	4	64	72	118
CSMA/CD	3	$[3, 8]$	3	3	2000	140	349	545	269
SPSMALL	10	$[3, 8]$	10	2	3149	259	60	61	1194
RCP	5	$[6, 11]$	6	3	186050	19	5688	9312	7018

Outline

Summary

- Implementation of the Inverse Method
  - Modeling of a system with parametric timed automata
  - Starting with a valuation π_0 of the system, we synthesize a constraint K_0 with the same trace set as π_0
  - Gives a criterion of robustness by guaranteeing the same behavior around π_0
- Implementation of the Behavioral cartography
  - Solves the good parameters problem: synthesizes the largest set of points within a rectangle V_0 corresponding to a given good behavior
  - Independent from the property: only the partition does

Future Work

- Automatize the partition into good and bad tiles
  - Make use of the UPPAAL model checker [Larsen et al., 1997]
- Extend the behavioral cartography to hybrid automata
  - Allow to consider different clock rates
- Consider a dynamic cartography
  - Refine the scale in order to fill the whole real-valued V_0
- Consider a weaker property than equality of trace sets
  - Reference trace with partial orders

References I

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1995).
The algorithmic analysis of hybrid systems.
Theoretical Computer Science, 138:3–34.

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.

André, É. and Fribourg, L. (2010).
Behavioral cartography of timed automata.
In *RP'10*, volume 6227 of *LNCS*, pages 76–90. Springer.

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.

Clarke, E. M., Grumberg, O., Jha, S., Lu, Y., and Veith, H. (2000).
Counterexample-guided abstraction refinement.
In *CAV '00*, pages 154–169. Springer-Verlag.

References II

Frehse, G., Jha, S., and Krogh, B. (2008).
A counterexample-guided approach to parameter synthesis for linear hybrid automata.
In *HSCC '08*, volume 4981 of *LNCS*, pages 187–200. Springer.

Hune, T., Romijn, J., Stoelinga, M., and Vaandrager, F. (2002).
Linear parametric model checking of timed automata.
Journal of Logic and Algebraic Programming.

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