Séminaire à l'IRIT

Jeudi 19 mai 2011

Synthesis of Timing Parameters for the Verification of Hardware Components and Communication Protocols

Étienne ANDRÉ

National University of Singapore

Context: Model Checking Timed Systems (1/2)

- Input

A timed concurrent system

Context: Model Checking Timed Systems (1/2)

- Input

A timed concurrent system

A good behavior expected for the system

Context: Model Checking Timed Systems (1/2)

- Input

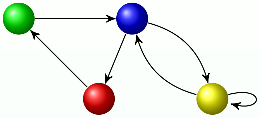
A timed concurrent system

A good behavior expected for the system

- Question: does the system always behave well?

Context: Model Checking Timed Systems (2/2)

- Use of formal methods

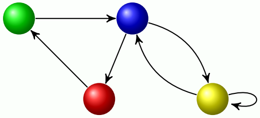


A finite model of the system

A formula to be satisfied

Context: Model Checking Timed Systems (2/2)

- Use of formal methods



$\overset{?}{\models}$

$AG\neg$ 

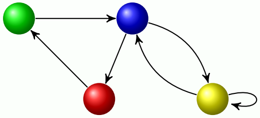
A finite model of the system

A formula to be satisfied

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

Context: Model Checking Timed Systems (2/2)

- Use of formal methods



$\stackrel{?}{\models}$

$AG\neg$

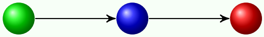
A finite model of the system

A formula to be satisfied

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

Yes

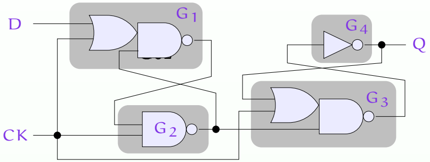
No

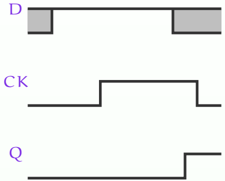


Counterexample

An Example of Flip-Flop Circuit

- An asynchronous circuit [Clarisó and Cortadella, 2007]

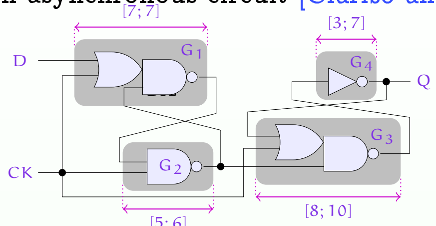


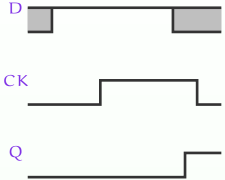


  - Concurrent behavior
    - 4 elements: G_1, G_2, G_3, G_4
    - 2 input signals (D and CK), 1 output signal (Q)

An Example of Flip-Flop Circuit

- An asynchronous circuit [Clarisó and Cortadella, 2007]

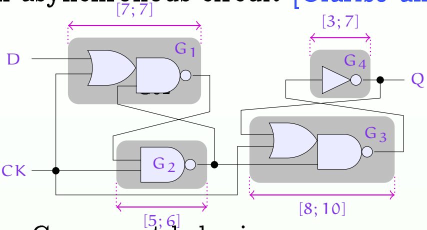


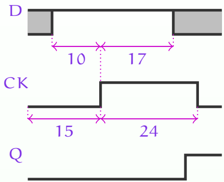


  - Concurrent behavior
    - 4 elements: G_1, G_2, G_3, G_4
    - 2 input signals (D and CK), 1 output signal (Q)
  - Timing delays
    - Traversal delays of the gates: one interval per gate

An Example of Flip-Flop Circuit

- An asynchronous circuit [Clarisó and Cortadella, 2007]

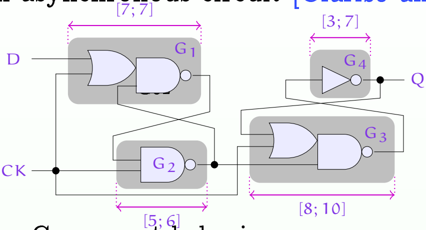


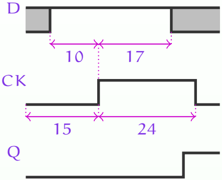


- Concurrent behavior
  - 4 elements: G_1, G_2, G_3, G_4
  - 2 input signals (D and CK), 1 output signal (Q)
- Timing delays
  - Traversal delays of the gates: one interval per gate
  - Environment timing constants

An Example of Flip-Flop Circuit

- An asynchronous circuit [Clarisó and Cortadella, 2007]



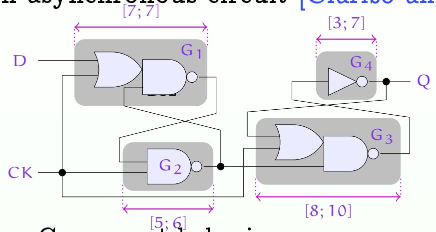


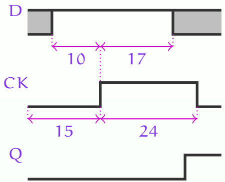
- Concurrent behavior
  - 4 elements: G_1, G_2, G_3, G_4
  - 2 input signals (D and CK), 1 output signal (Q)
- Timing delays
  - Traversal delays of the gates: one interval per gate
  - Environment timing constants

- 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]





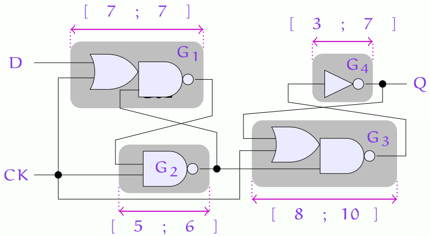
  - Concurrent behavior
    - 4 elements: G_1, G_2, G_3, G_4
    - 2 input signals (D and CK), 1 output signal (Q)
  - Timing delays
    - Traversal delays of the gates: one interval per gate
    - Environment timing constants

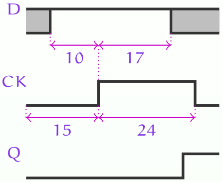
- Question
  - For these timing delays, does the rise of Q always occur before the fall of CK?
  - Timed model checking gives the answer: yes

Parameter Synthesis

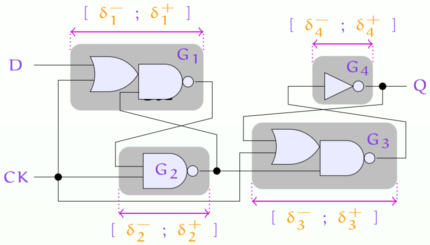
- More difficult problem: find values of the timing delays for which the system behaves well
- Idea: reason with unknown constants or parameters
- Interesting applications
  - Ensure the robustness of the system
  - Allow the designer to optimize timing delays
  - Allow to scale down large timing constants
- Difficult problem
  - Both concurrent behavior and timed behavior
  - Undecidable in general

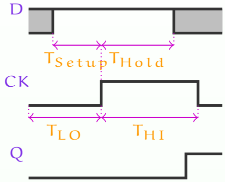
Flip-Flop Circuit: Timing Parameters





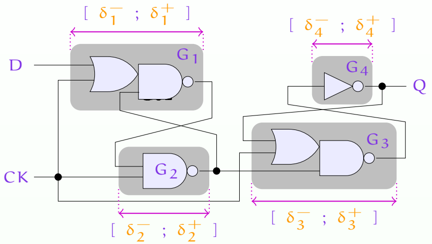
Flip-Flop Circuit: Timing Parameters

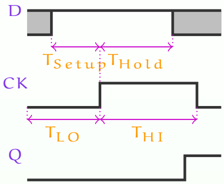




- Timing parameters
  - Traversal delays of the gates: one interval per gate
  - 4 environment parameters: T_{LO}, T_{HI}, T_{Setup} and T_{Hold}

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?

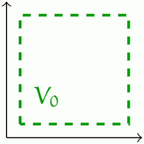
Related Work

- Approaches based on bad state avoidance
  - Computation of all the reachable states, and intersection with the bad states [Henzinger and Wong-Toi, 1996]
  - Definition of parametric structures [Hune et al., 2002]
  - Use of approximations [Clarisó and Cortadella, 2007]
  - Refinement of the model based on CEGAR [Frehse et al., 2008]

- We present here a good state-based method

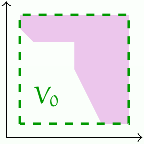
Problems

- The good parameters problem
  - "Given a bounded parameter domain V_0, find a set of parameter valuations of good behavior in V_0"



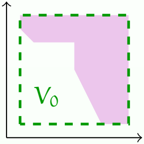
Problems

- The good parameters problem
  - "Given a bounded parameter domain V_0, find a set of parameter valuations of good behavior in V_0"

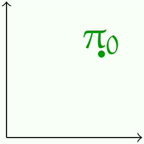


Problems

- The good parameters problem
  - "Given a bounded parameter domain V_0, find a set of parameter valuations of good behavior in V_0"

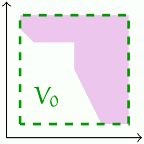


- The inverse problem
  - "Given a reference parameter valuation π_0, find other valuations around π_0 of same behavior"

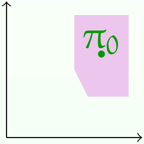


Problems

- The good parameters problem
  - "Given a bounded parameter domain V_0, find a set of parameter valuations of good behavior in V_0"



- The inverse problem
  - "Given a reference parameter valuation π_0, find other valuations around π_0 of same behavior"



Outline

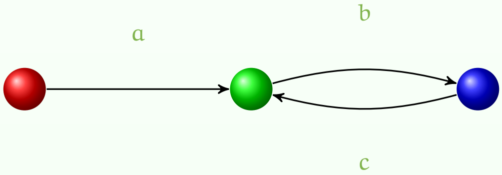
Outline

Timed Automaton (TA)

- Finite state automaton (sets of locations)

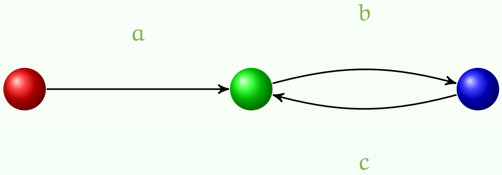
Timed Automaton (TA)

- Finite state automaton (sets of locations and actions)



Timed Automaton (TA)

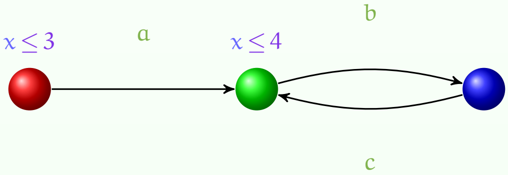
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate



Timed Automaton (TA)

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

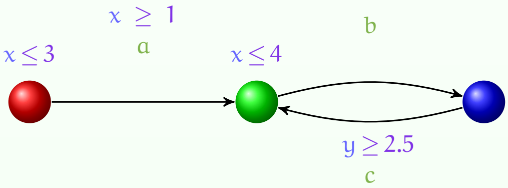
- Features
  - Location invariant: property to be verified to stay at a location



Timed Automaton (TA)

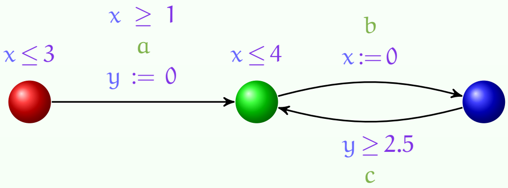
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate

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



Timed Automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 at each transition



Semantics of Timed Automata

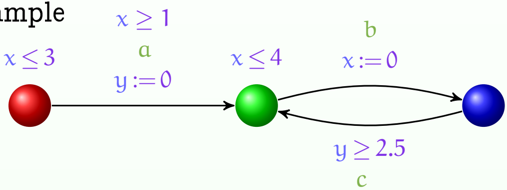
- Concrete state of a TA: couple (q, w), where
  - q is a location,
  - w is a valuation of each clock

Semantics of Timed Automata

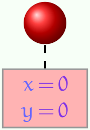
- Concrete state of a TA: couple (q, w), where
  - q is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions

Semantics of Timed Automata

- Concrete state of a TA: couple (q, w), where
  - q is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions
- Example

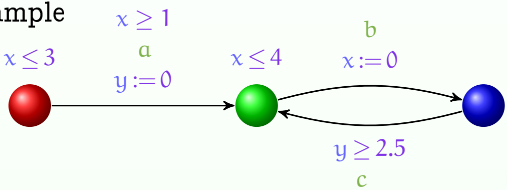


  - Possible concrete run for this TA

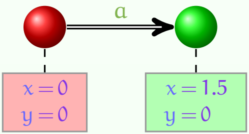


Semantics of Timed Automata

- Concrete state of a TA: couple (q, w), where
  - q is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions
- Example

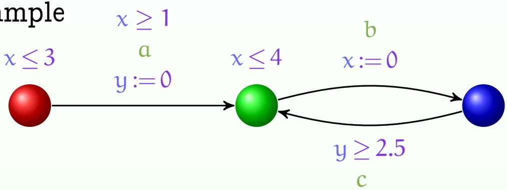


  - Possible concrete run for this TA

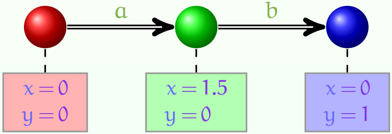


Semantics of Timed Automata

- Concrete state of a TA: couple (q, w), where
  - q is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions
- Example

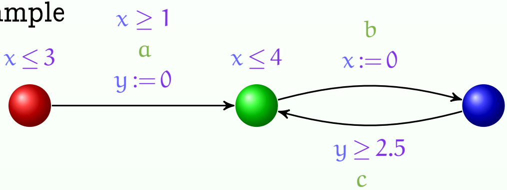


  - Possible concrete run for this TA

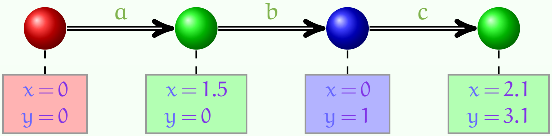


Semantics of Timed Automata

- Concrete state of a TA: couple (q, w), where
  - q is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions
- Example

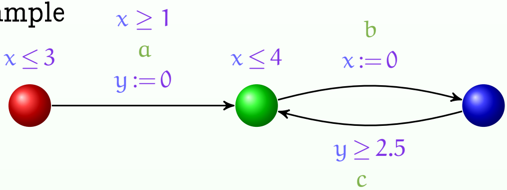


  - Possible concrete run for this TA

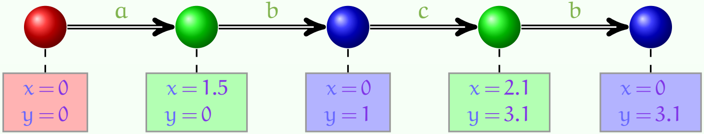


Semantics of Timed Automata

- Concrete state of a TA: couple (q, w), where
  - q is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions
- Example

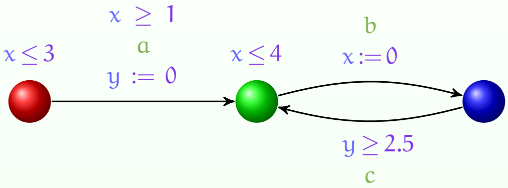


  - Possible concrete run for this TA



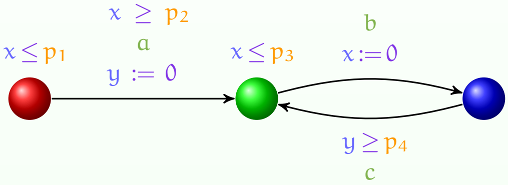
Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks)



Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [Alur et al., 1993]
    - Unknown constants used in guards and invariants



Semantics of Parametric Timed Automata

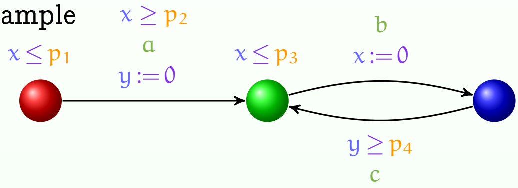
- Symbolic state of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over X and P

Semantics of Parametric Timed Automata

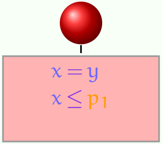
- Symbolic state of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over X and P
- Symbolic run: alternating sequence of symbolic states and actions

Semantics of Parametric Timed Automata

- Symbolic state of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over X and P
- Symbolic run: alternating sequence of symbolic states and actions
- Example

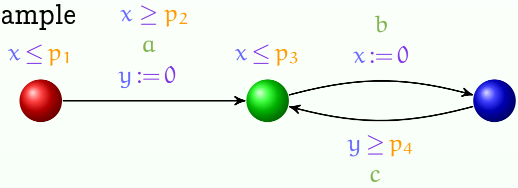


  - Possible symbolic run for this PTA

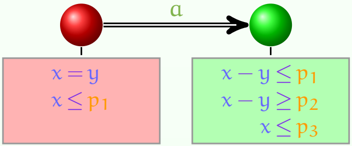


Semantics of Parametric Timed Automata

- Symbolic state of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over X and P
- Symbolic run: alternating sequence of symbolic states and actions
- Example

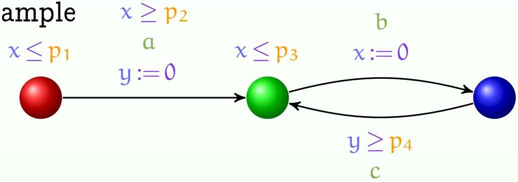


  - Possible symbolic run for this PTA

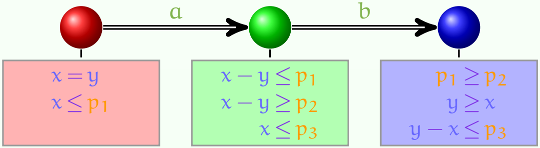


Semantics of Parametric Timed Automata

- Symbolic state of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over X and P
- Symbolic run: alternating sequence of symbolic states and actions
- Example

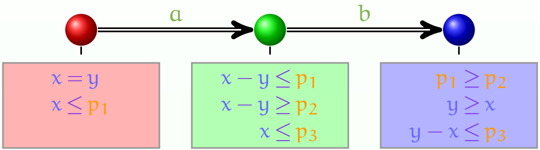


  - Possible symbolic run for this PTA



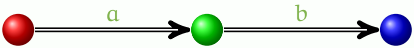
Good and Bad Traces

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



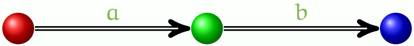
Good and Bad Traces

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

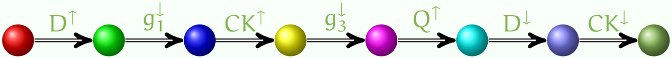


Good and Bad Traces

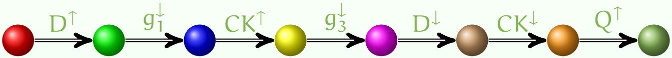
- Trace over a PTA: time-abstract run
  - Finite alternating sequence of locations and actions



- A trace is said to be good if it verifies a given property
  - Example of good trace for the flip-flop ($Q^{\uparrow}$ occurs before $CK^{\downarrow}$)

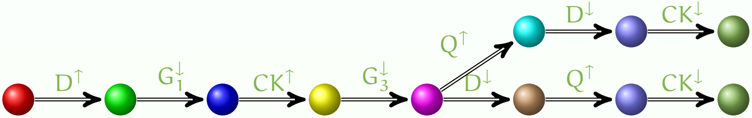


  - Example of bad trace for the flip-flop



Notation

- Trace set: set of all traces of a PTA
  - Example of trace set for the flip-flop example



- A valuation π of all the parameters of P is called a point
- Given a PTA $\mathcal{A}$ and a point π, we denote by $\mathcal{A}[\pi]$ the (non-parametric) timed automaton where all parameters are instantiated by π

Outline

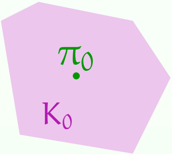
The Inverse Problem

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

π_0

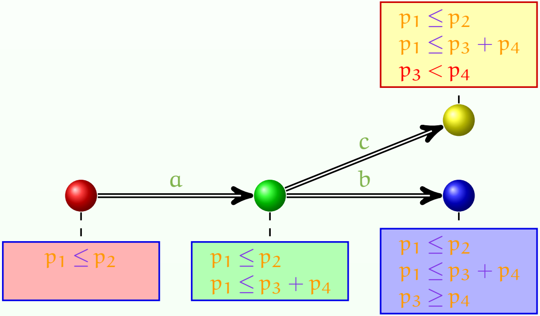
The Inverse Problem

- 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 points $\pi \models K_0$, $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ have the same trace sets



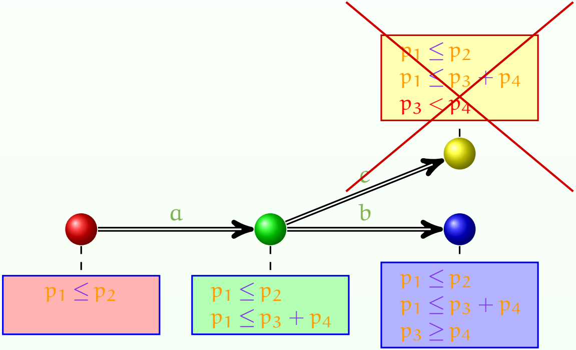
The Inverse Method IM: General Idea

- Our idea [A., Chatain, Encrenaz, Fribourg, IJFCS, 2009]
  - CEGAR-like approach
  - Instead of negating bad states, we remove π_0-incompatible states



The Inverse Method IM: General Idea

- Our idea [A., Chatain, Encrenaz, Fribourg, IJFCS, 2009]
  - CEGAR-like approach
  - Instead of negating bad states, we remove π_0-incompatible states



The Inverse Method IM: Simplified Algorithm

Start with $K_0 = \mathtt{true}$

REPEAT

1. Compute a set S of reachable symbolic states under K_0
2. Project the constraints onto the parameters
3. Refine K_0 by removing a π_0-incompatible state from S
    - Select a π_0-incompatible state (q, C) within S (i.e., $\pi_0 \not\models C$)
    - Select a π_0-incompatible inequality J within C (i.e., $\pi_0 \not\models J$)
    - Add $\neg J$ to K_0

UNTIL no more π_0-incompatible state in S

Application to the Flip-Flop Circuit

π_0 :

δ_1^-	$= 7$	δ_1^+	$= 7$	T_{HI}	$= 24$
δ_2^-	$= 5$	δ_2^+	$= 6$	T_{LO}	$= 15$
δ_3^-	$= 8$	δ_3^+	$= 10$	T_{Setup}	$= 10$
δ_4^-	$= 3$	δ_4^+	$= 7$	T_{Hold}	$= 17$

$K_0 = \texttt{true}$

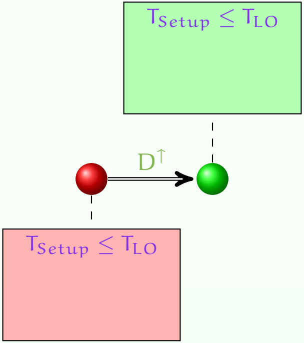
$T_{Setup} \leq T_{LO}$

Application to the Flip-Flop Circuit

π_0 :

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

$K_0 = \texttt{true}$

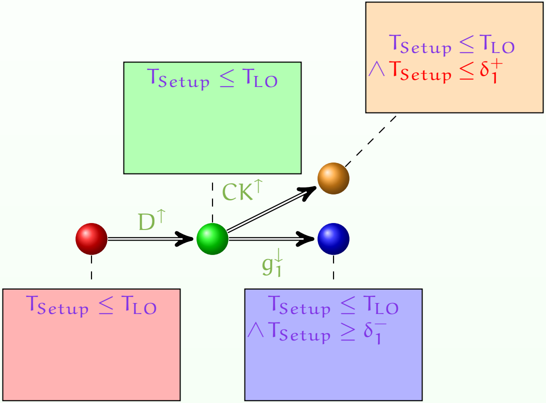


Application to the Flip-Flop Circuit

π_0 :

δ_1^-	$= 7$	δ_1^+	$= 7$	T_{HI}	$= 24$
δ_2^-	$= 5$	δ_2^+	$= 6$	T_{LO}	$= 15$
δ_3^-	$= 8$	δ_3^+	$= 10$	T_{Setup}	$= 10$
δ_4^-	$= 3$	δ_4^+	$= 7$	T_{Hold}	$= 17$

$K_0 = \texttt{true}$



Application to the Flip-Flop Circuit

π_0 :

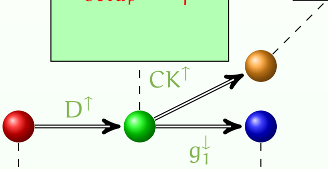
δ_1^-	$= 7$	δ_1^+	$= 7$	T_{HI}	$= 24$
δ_2^-	$= 5$	δ_2^+	$= 6$	T_{LO}	$= 15$
δ_3^-	$= 8$	δ_3^+	$= 10$	T_{Setup}	$= 10$
δ_4^-	$= 3$	δ_4^+	$= 7$	T_{Hold}	$= 17$

$K_0 =$

$T_{Setup} > \delta_1^+$

$T_{Setup} \leq T_{LO}$
$\wedge\, T_{Setup} \leq \delta_1^+$

$T_{Setup} \leq T_{LO}$
$\wedge\, T_{Setup} > \delta_1^+$



$T_{Setup} \leq T_{LO}$
$\wedge\, T_{Setup} > \delta_1^+$

$T_{Setup} \leq T_{LO}$
$\wedge\, T_{Setup} > \delta_1^+$

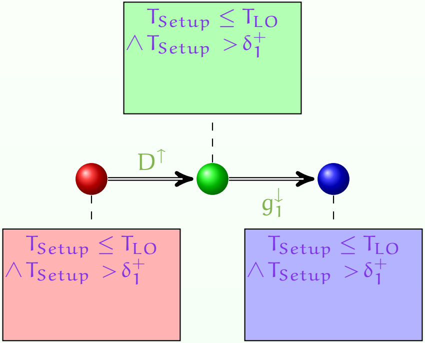
Application to the Flip-Flop Circuit

π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$



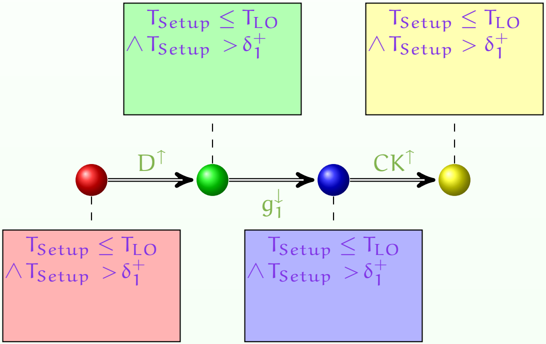
Application to the Flip-Flop Circuit

π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$



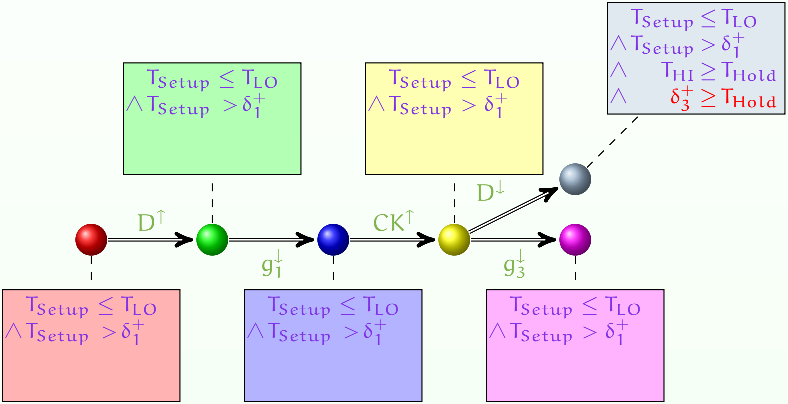
Application to the Flip-Flop Circuit

π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$

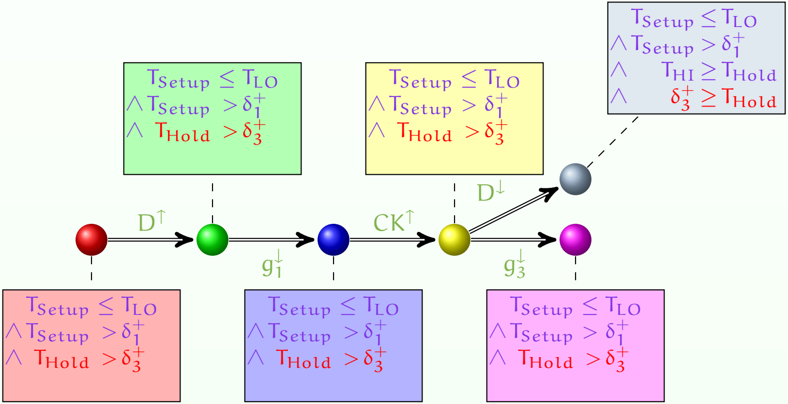


Application to the Flip-Flop Circuit

π_0 :

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

$$K_0 = T_{Setup} > \delta_1^+ \wedge T_{Hold} > \delta_3^+$$

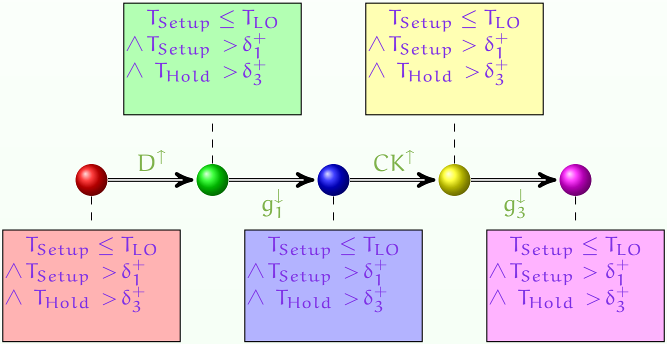


Application to the Flip-Flop Circuit

π_0 :

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

$$K_0 = \quad T_{Setup} > \delta_1^+ \;\wedge\; T_{Hold} > \delta_3^+$$

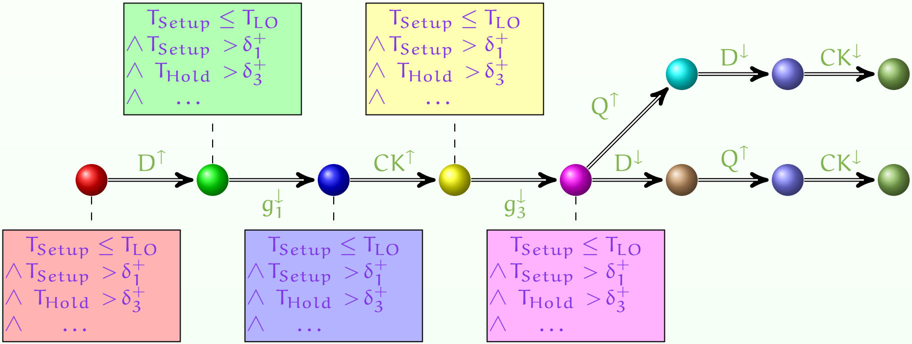


Application to the Flip-Flop Circuit

π_0 :

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

$$
\begin{array}{llll}
K_0 = & & & \\
 & T_{Setup} > \delta_1^+ & \wedge & \delta_3^+ + \delta_4^+ \geq T_{Hold} \\
\wedge & T_{Hold} > \delta_3^+ & \wedge & \delta_3^+ + \delta_4^+ < T_{HI} \\
\wedge & T_{Setup} \leq T_{LO} & \wedge & \delta_3^- + \delta_4^- \leq T_{Hold} \\
\wedge & \delta_1^- > 0 & &
\end{array}
$$



Correctness of IM

Theorem (Correctness)

Suppose that $IM(\mathcal{A}, \pi_0)$ *terminates with output* K_0. *We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Termination

- Parameter synthesis undecidable in general for PTAs
- However, we give sufficient condition for the termination of IM

Proposition (Termination)

IM *terminates if the set of traces of* $\mathcal{A}[\pi_0]$ *contains no cyclic trace (trace passing twice by the same location).*

- Remarks
  - Many case studies fall into this class
  - Termination can be shown in more cases
  - In practice, IM also terminates for most of our “cyclic” case studies

Implementation

- IMITATOR II [André, INFINITY'10]
  - IMITATOR: "Inverse Method for Inferring Time AbstracT BehaviOR"
  - 10 000 lines of code
  - Program written in OCaml
  - Makes use of the PPL library for handling polyhedra
- Available on the Web
  - http://www.lsv.ens-cachan.fr/~andre/IMITATOR2

Inverse Method: Case Studies

- Hardware circuits
- Communication protocols
  - Bounded Retransmission Protocol
  - CSMA/CD Protocol
  - Root Contention Protocol
- SIMOP (Farman project LSV – LURPA): manufacturing system with sensors and controllers communicating through a network
- SPSMALL: real memory circuit (ST-Microelectronics)

Summary of the Inverse Method

- Advantages
  - Useful to optimize timing delays of systems
  - Gives a criterion of robustness to the system
  - Independent of the property one wants to check
  - Terminates often in practice
  - Efficient: allows to handle dozens of parameters

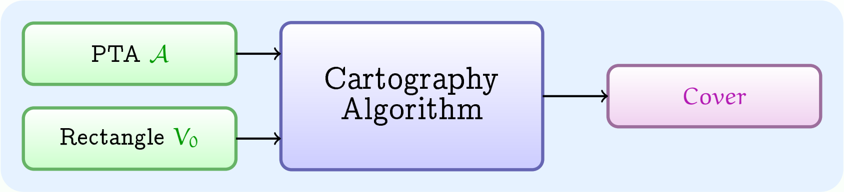
- Remarks
  - The constraint K_0 synthesized is not maximal: there are points $\pi \notin K_0$ which give the same trace set as π_0
  - For a given property φ, there may be different trace sets satisfying φ

Outline

Beyond the Inverse Method

- Goal: Find the maximal set of points corresponding to a good behavior
- Method: iterate IM for all integer points of a given rectangle V_0
- Output: set of behavioral tiles
  - $\rightsquigarrow$ behavioral cartography of the parameter space [A., Fribourg, RP'10]

The Behavioral Cartography Algorithm



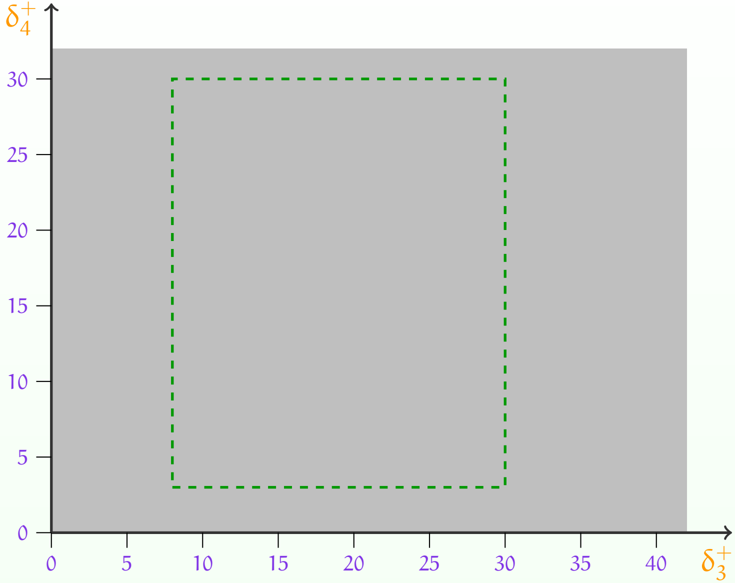
---

repeat
    select an integer point $\pi \in V_0$;
    if $\pi \notin \text{Cover}$ **then**
        $\text{Cover} \leftarrow \text{Cover} \cup \text{IM}(\mathcal{A}, \pi)$;
until Cover *contains all the integer points of* V_0;

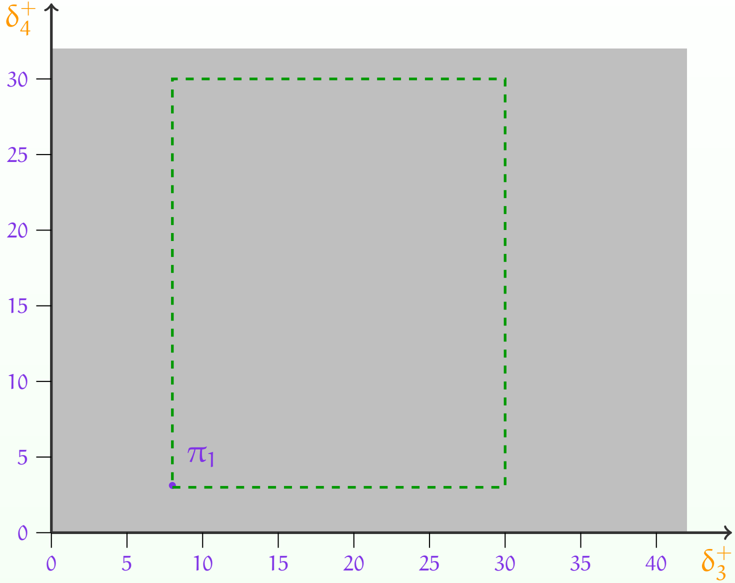
Application to the Flip-Flop Example

- Goal
  - Find the maximal set of values for δ_3^+ and δ_4^+ such that the flip-flop has a good behavior
- Method
  - Perform the behavioral cartography of the flip-flop circuit according to δ_3^+ and δ_4^+
  - The other parameters are instantiated

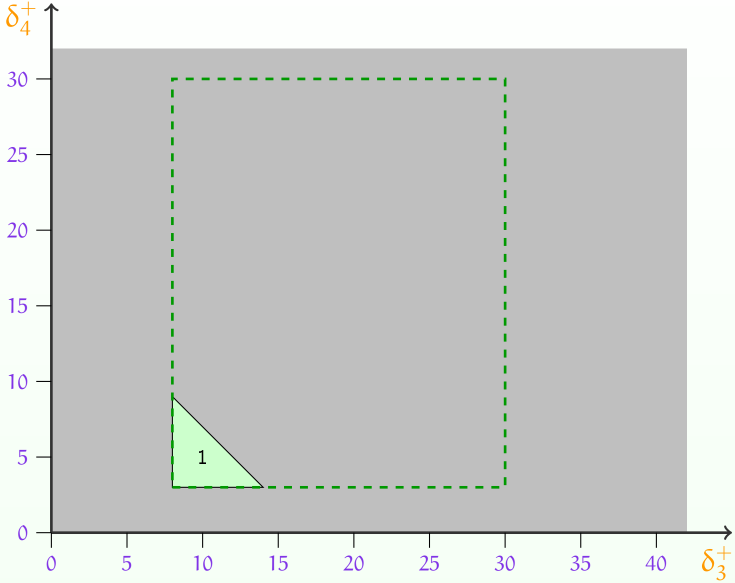
Behavioral Cartography of the Flip-Flop



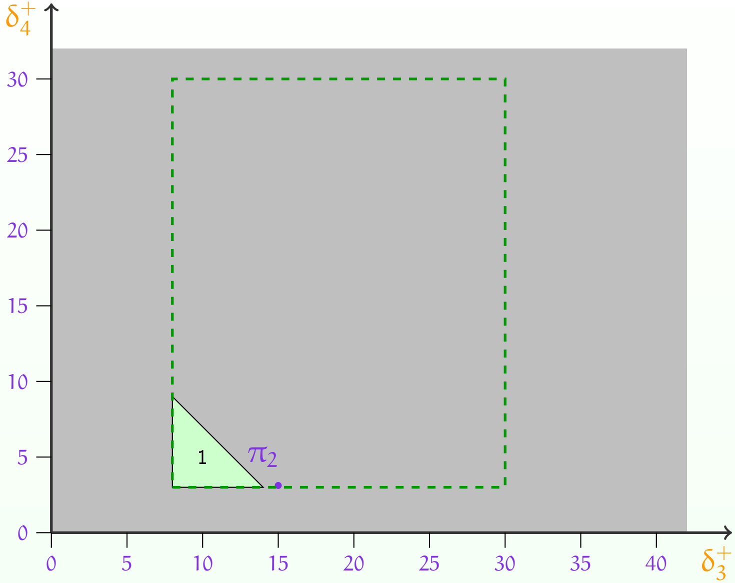
Behavioral Cartography of the Flip-Flop



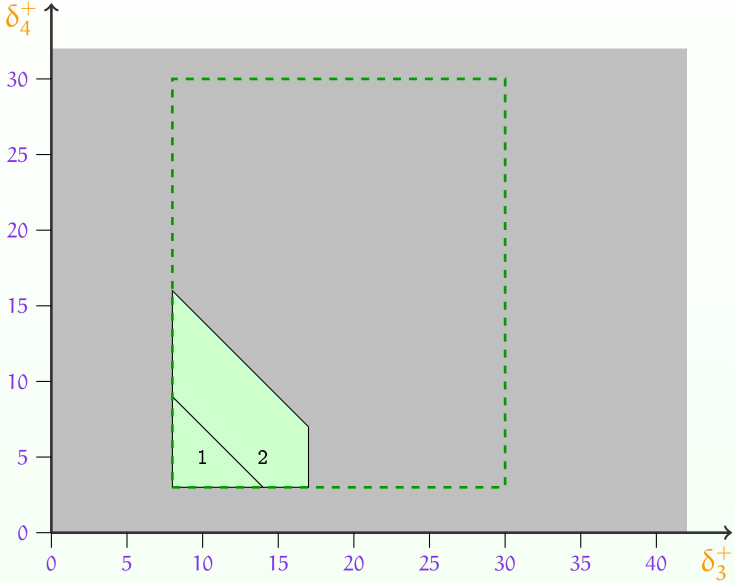
Behavioral Cartography of the Flip-Flop



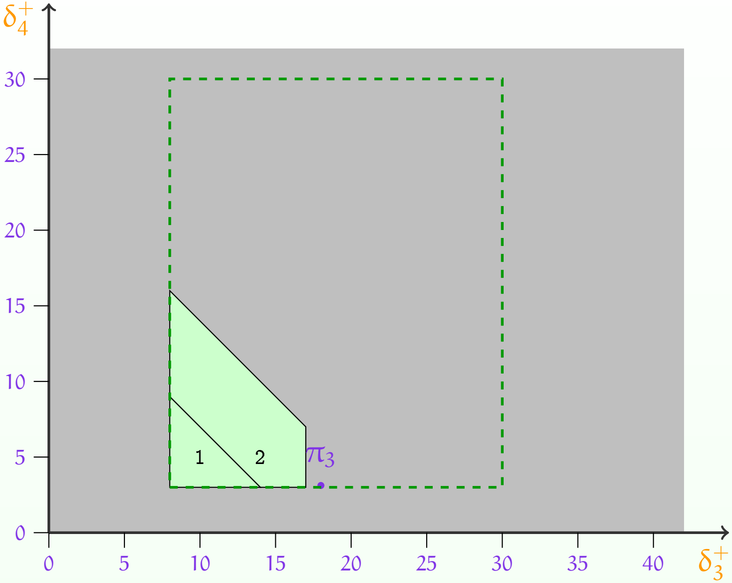
Behavioral Cartography of the Flip-Flop



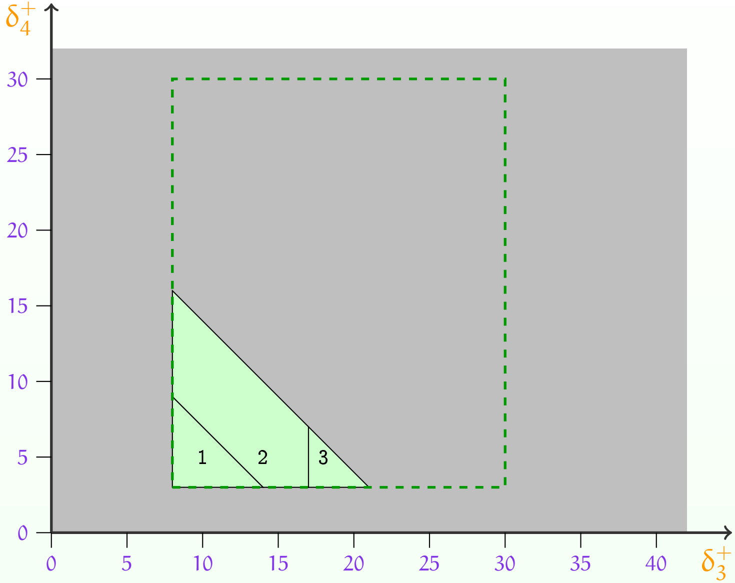
Behavioral Cartography of the Flip-Flop



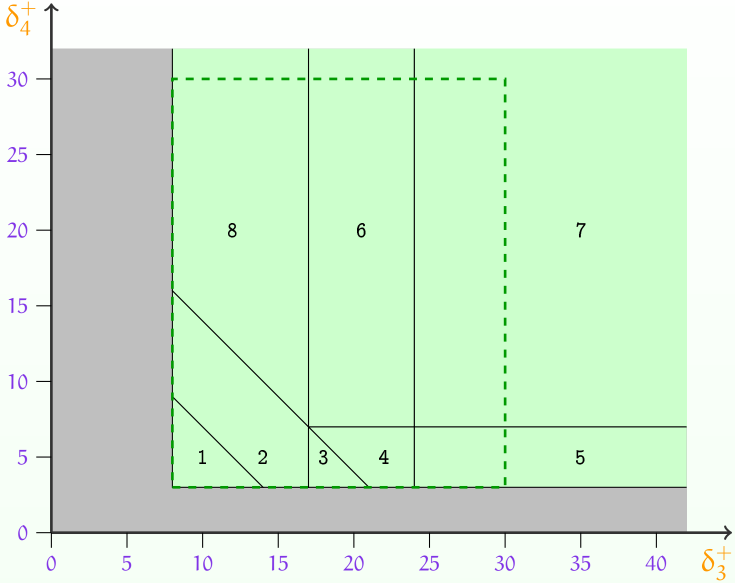
Behavioral Cartography of the Flip-Flop



Behavioral Cartography of the Flip-Flop



Behavioral Cartography of the Flip-Flop



Behavioral Cartography Algorithm: Full Coverage

Proposition

For acyclic PTAs, the full coverage of the whole parametric space is ensured for a grid fine enough.

Grid: points (integers or rationals) on which IM may be called

Behavioral Cartography Algorithm: Full Coverage

Proposition

For acyclic PTAs, the full coverage of the whole parametric space is ensured for a grid fine enough.

Grid: points (integers or rationals) on which IM may be called

Idea of the proof:

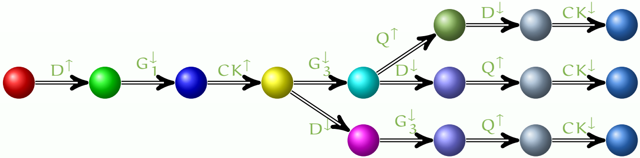
- Based on the finiteness of the number of possible tiles

Partition into Good and Bad Tiles

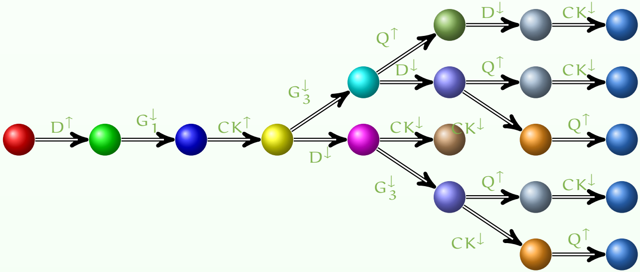
- A tile is said to be a good tile if all its corresponding traces are good traces
- According to the nature of the trace sets, we can partition the tiles into good and bad ones

Example of Good and Bad Tiles for the Flip-flop

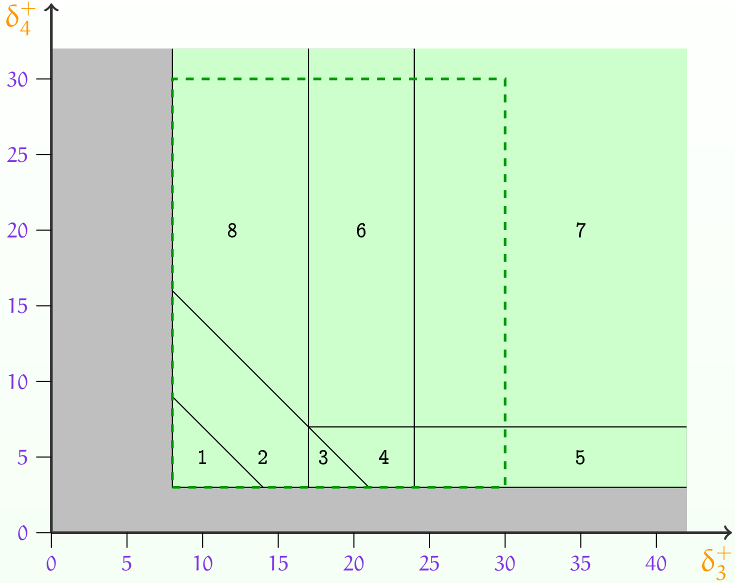
- Good tile 3



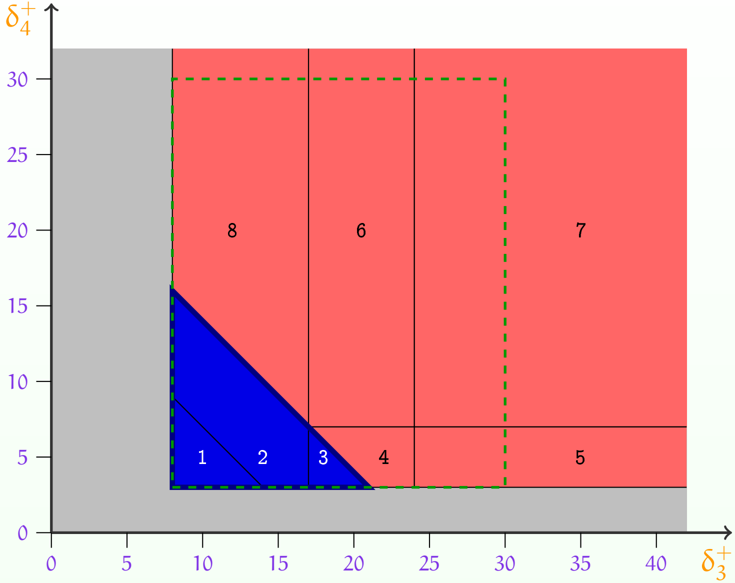
- Bad tile 7



Behavioral Cartography of the Flip-flop: Partition



Behavioral Cartography of the Flip-flop: Partition



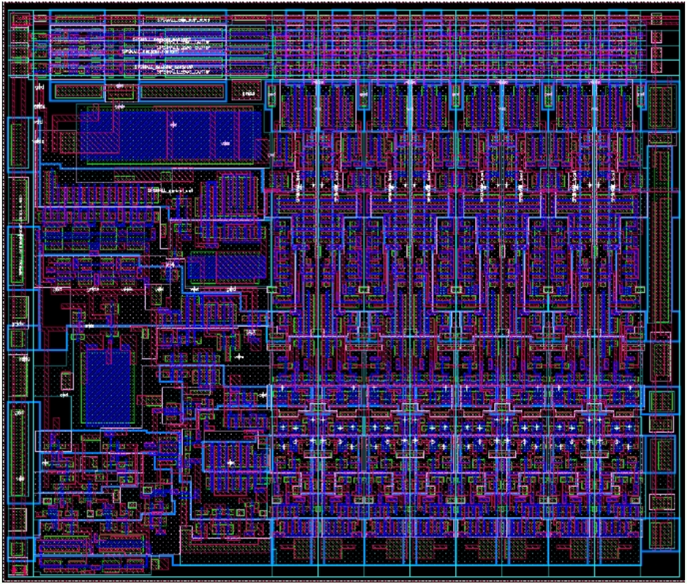
Behavioral Cartography of the Flip-flop: Remarks

- Remarks on the cartography
  - For this example, all the real-valued part of the parametric space within and outside V_0 is covered
- The set of good tiles (in blue) corresponds to the maximal set of good values for δ_3^+ and δ_4^+
  - $\delta_3^+ + \delta_4^+ \leq 24 \quad \wedge \quad \delta_3^+ \geq 8 \quad \wedge \quad \delta_4^+ \geq 3$

Cartography Algorithm: Implementation

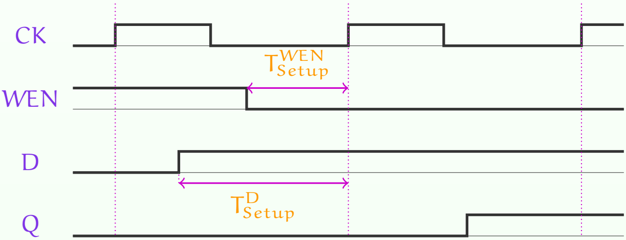
- Implementation in IMITATOR II
- Application to case studies
  - Hardware devices
  - Communication protocols
  - SPSMALL memory
- Allows to solve the good parameters problem
- Comparison of the constraints
  - Constraints synthesized always equal to or better than the ones from the literature

The SPSMALL Memory

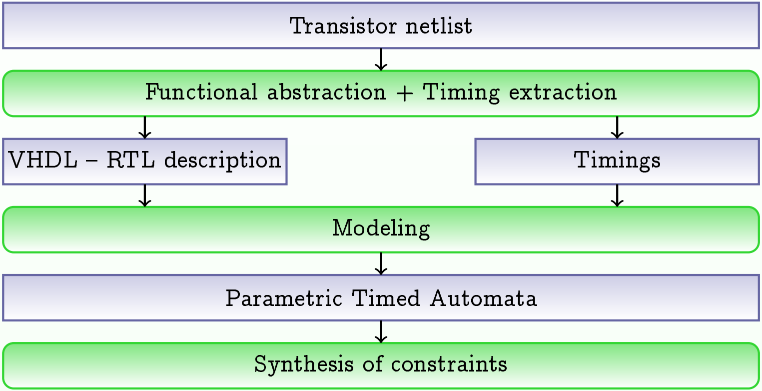


VALMEM Project

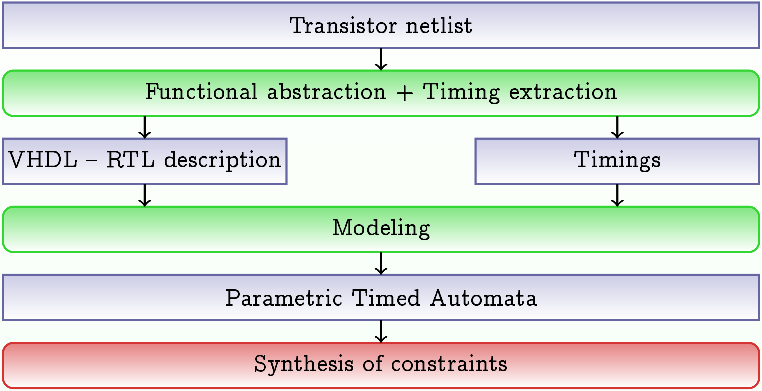
- Memory circuit sold by ST-Microelectronics
- Studied in the ANR VALMEM project
  - LIP6, LSV, ST-Microelectronics
- Goal: minimize timing parameters T^{D}_{Setup} and T^{WEN}_{Setup}
  - Reference valuation in the memory datasheet: $T^{D}_{Setup} = 108$ and $T^{WEN}_{Setup} = 48$



Methodology



Methodology



Cartography Algorithm

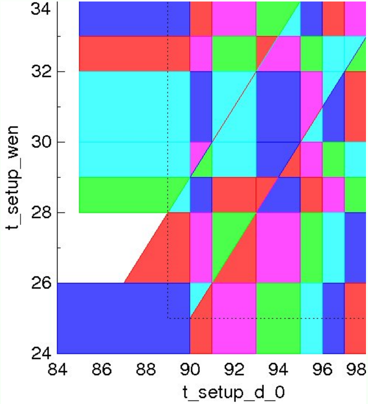
- Cartography of the memory according to T^{D}_{Setup} and T^{WEN}_{Setup}
  - Reference rectangle V_0:

$$T^{D}_{Setup} \in [89; 108] \qquad T^{WEN}_{Setup} \in [25; 48]$$

Cartography Algorithm

- Cartography of the memory according to T^{D}_{Setup} and T^{WEN}_{Setup}
  - Reference rectangle V_0:

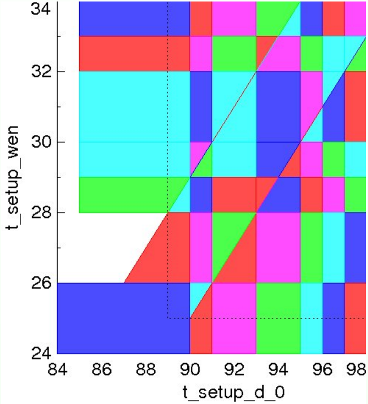
$$T^{D}_{Setup} \in [89; 108] \qquad T^{WEN}_{Setup} \in [25; 48]$$



Cartography Algorithm

- Cartography of the memory according to T^{D}_{Setup} and T^{WEN}_{Setup}
  - Reference rectangle V_0:

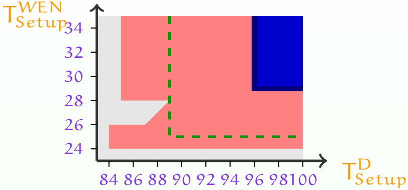
$$T^{D}_{Setup} \in [89; 108] \qquad T^{WEN}_{Setup} \in [25; 48]$$



$\Rightarrow$ Full coverage of V_0

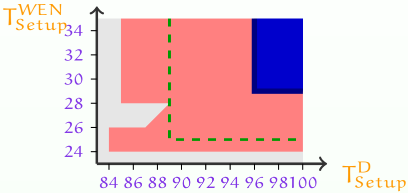
Minimization of Timings

- Partition into good and bad tiles
  - Using the property of good behavior specified by the datasheet



Minimization of Timings

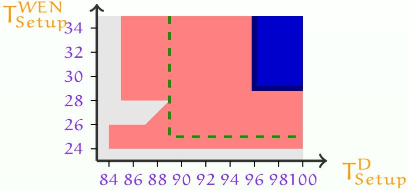
- Partition into good and bad tiles
  - Using the property of good behavior specified by the datasheet



- Minimization of timing delays
  - $T^{D}_{Setup} = 108$
  - $T^{WEN}_{Setup} = 48$

Minimization of Timings

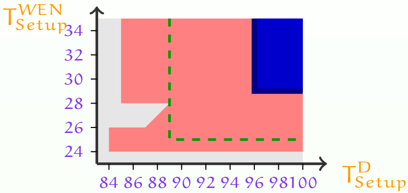
- Partition into good and bad tiles
  - Using the property of good behavior specified by the datasheet



- Minimization of timing delays
  - $T^{D}_{Setup} = 108 \rightsquigarrow 96$ (decrease of $11.1\,\%$)
  - $T^{WEN}_{Setup} = 48 \rightsquigarrow 29$ (decrease of $39.6\,\%$)

Minimization of Timings

- Partition into good and bad tiles
  - Using the property of good behavior specified by the datasheet



- Minimization of timing delays
  - $T^{D}_{Setup} = 108 \rightsquigarrow 96$ (decrease of 11.1 %)
  - $T^{WEN}_{Setup} = 48 \rightsquigarrow 29$ (decrease of 39.6 %)
- Practical interest: allows to work in a faster environment
  - Optimization of the datasheet
  - Financial interest

Advantages of the Behavioral Cartography

- Solves the good parameters problem
- Under certain conditions, covers the whole real-valued parametric space
- Independent of the property one wants to check
    - Only the partition depends on the property
    - No need to compute a cartography for each property

Outline

The Root Contention Protocol

- Root contention protocol of the IEEE 1394 (“FireWire”)
  - Election of a leader after a certain number of rounds
  - Protocol mixing time and probabilities
  - Timing delays: $s\_min = 1590\,ns$ and $delay = 300\,ns$
- Computation of minimum or maximum probabilities
  - Example: “Minimum probability that a leader is elected after 5 rounds or less”
  - Use of the PRISM model checker [Hinton et al., 2006]
- Problem
  - PRISM is very sensitive to the size of the timing constants
  - For this valuation ($s\_min = 1590\,ns$ and $delay = 300\,ns$), PRISM does not succeed to compute probabilities

Goal

Goal

Compute constraints on the timing parameters such that the minimum and maximum probabilities of reachability properties remain the same.

Goal

Goal

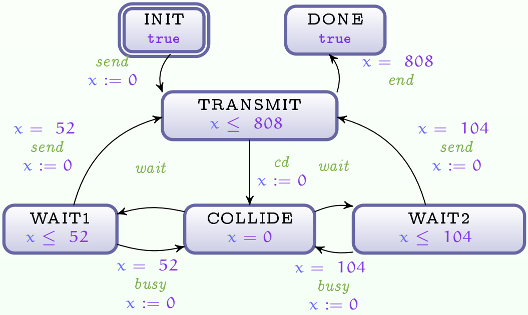
Compute constraints on the timing parameters such that the minimum and maximum probabilities of reachability properties remain the same.

Application

- By minimizing the timing delays within this constraint, Prism will be able to compute probabilities more easily

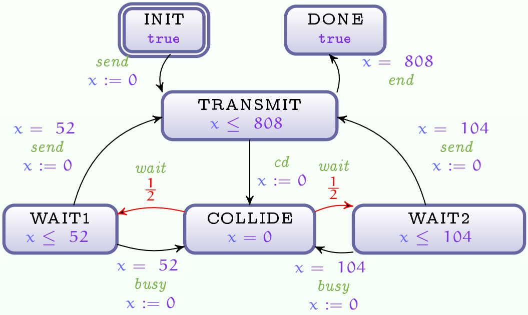
Probabilistic Timed Automaton

- Probabilistic Timed Automaton [Gregersen and Jensen, 1995, Kwiatkowska et al., 2002a]
  - Timed automaton



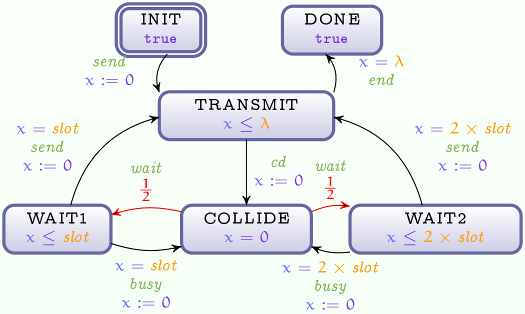
Probabilistic Timed Automaton

- Probabilistic Timed Automaton
  [Gregersen and Jensen, 1995, Kwiatkowska et al., 2002a]
  - Timed automaton with probabilities



Parametric Probabilistic Timed Automaton (PPTA)

- Probabilistic Timed Automaton [Gregersen and Jensen, 1995, Kwiatkowska et al., 2002a]
  - Timed automaton with probabilities
- Augmented with a set of parameters [A., Fribourg, Sproston, AVoCS'09]



Probabilistic Traces

- Probabilistic trace
  - Finite alternating sequence of locations and actions

INIT $\xrightarrow{send}$ TRANSMIT $\xrightarrow{cd}$ COLLIDE $\xrightarrow{wait}$ WAIT1 $\xrightarrow{send}$ TRANSMIT $\longrightarrow \cdots$

Probabilistic Traces

- Probabilistic trace
  - Finite alternating sequence of locations and actions with probabilities

$$\text{INIT} \xrightarrow[1]{send} \text{TRANSMIT} \xrightarrow[1]{cd} \text{COLLIDE} \xrightarrow[\frac{1}{2}]{wait} \text{WAIT1} \xrightarrow[1]{send} \text{TRANSMIT} \longrightarrow \cdots$$

Min / Max Probabilities of Reaching a State

- A scheduler s associates to every state one output distribution
- Given a scheduler, one can associate a probability to the state space
  - In particular: probability of reaching a location
- Minimum and maximum probabilities of reaching a given location
  - Minimum and maximum for all possible schedulers

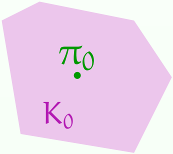
The Inverse Problem for PPTAs

- Inputs
  - A PPTA $\mathcal{A}$
  - A reference valuation π_0 of $\mathcal{A}$

The Inverse Problem for PPTAs

- Inputs
  - A PPTA $\mathcal{A}$
  - A reference valuation π_0 of $\mathcal{A}$

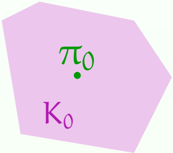
- Output: tile K_0
  - Convex constraint on the parameters such that
    - $\pi_0 \models K_0$
    - For all $\pi \models K_0$, the sets of probabilistic traces of $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are equal



The Inverse Problem for PPTAs

- Inputs
  - A PPTA $\mathcal{A}$
  - A reference valuation π_0 of $\mathcal{A}$

- Output: tile K_0
  - Convex constraint on the parameters such that
    - $\pi_0 \models K_0$
    - For all $\pi \models K_0$, the sets of probabilistic traces of $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are equal



As a consequence, the minimum and maximum probabilities for reachability properties are the same in $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$

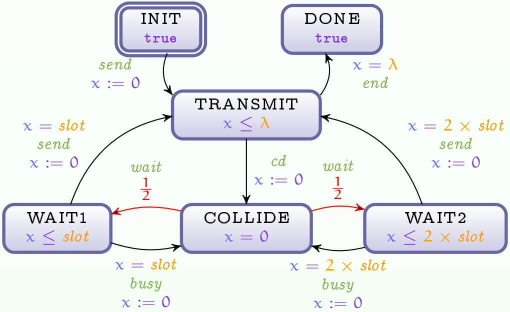
Derandomized PPTA

- Derandomized form $\mathcal{A}^*$ of a PPTA $\mathcal{A}$: replace distributions by non-determinism
  - $\mathcal{A}^*$ becomes a PTA

Derandomized PPTA

- Derandomized form $\mathcal{A}^*$ of a PPTA $\mathcal{A}$: replace distributions by non-determinism
  - $\mathcal{A}^*$ becomes a PTA

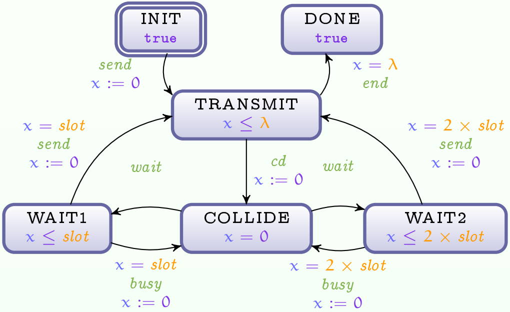
Example:



Derandomized PPTA

- Derandomized form $\mathcal{A}^*$ of a PPTA $\mathcal{A}$: replace distributions by non-determinism
  - $\mathcal{A}^*$ becomes a PTA

Example:



Extension of the Inverse Method to PPTAs

1. Construct a derandomized version $\mathcal{A}^*$ of $\mathcal{A}$
2. Compute $K_0 = IM(\mathcal{A}^*, \pi_0)$

Equality of the Sets of Probabilistic Traces

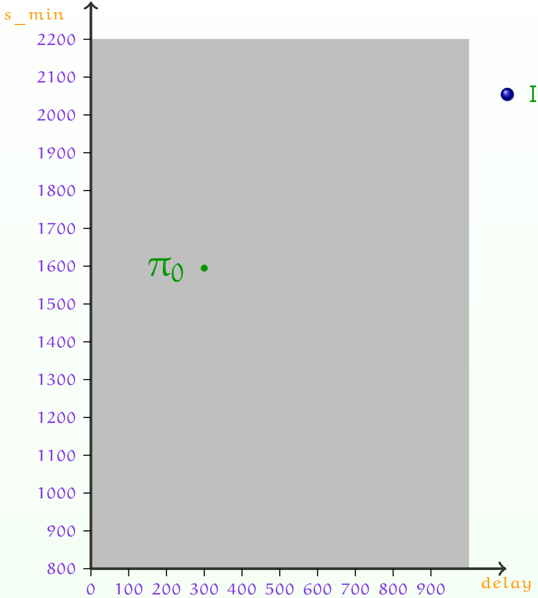
Theorem (Correctness)

Let $\mathcal{A}$ be a PPTA, and π_0 a valuation of the parameters.
Let $K_0 = \mathrm{IM}(\mathcal{A}^*, \pi_0)$.
Then, for all $\pi \models K_0$, the sets of probabilistic traces of $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are equal.

Consequence:

- The minimum and maximum probabilities for reachability properties are the same in $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$

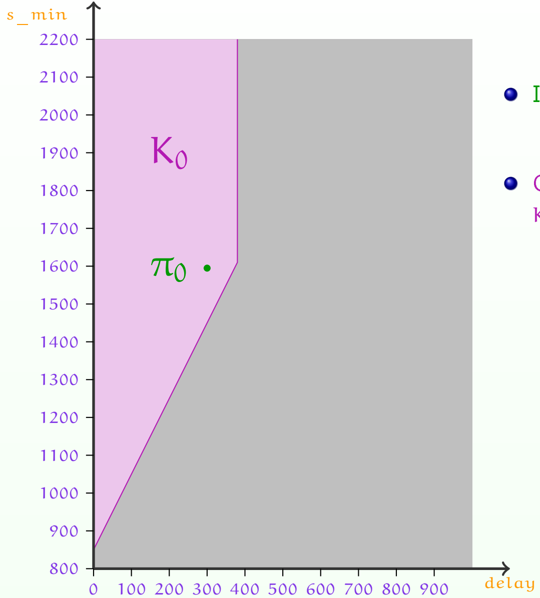
Application to the Root Contention Protocol



- Input: IEEE reference valuation

  $s\_min = 1590\,ns$ $\quad$ $delay = 300\,ns$

Application to the Root Contention Protocol



- **Input**: IEEE reference valuation

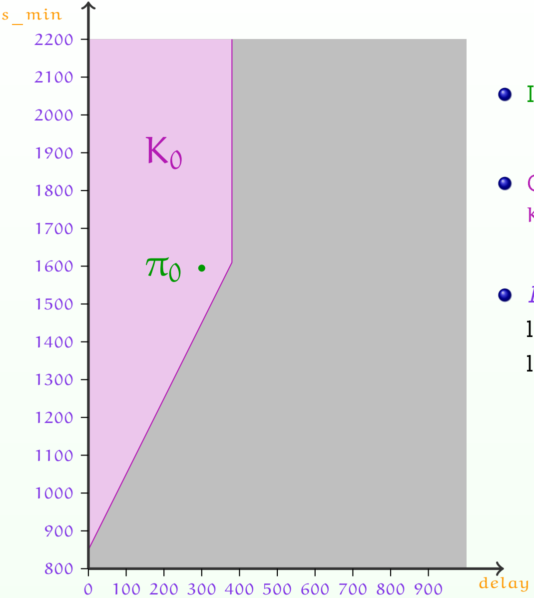
  $s\_min = 1590\,ns \qquad delay = 300\,ns$

- **Output**:

  K_0 : $\quad 2delay < 760$

  $\quad \wedge\ 2delay + 850 < s\_min$

Application to the Root Contention Protocol



- Input: IEEE reference valuation

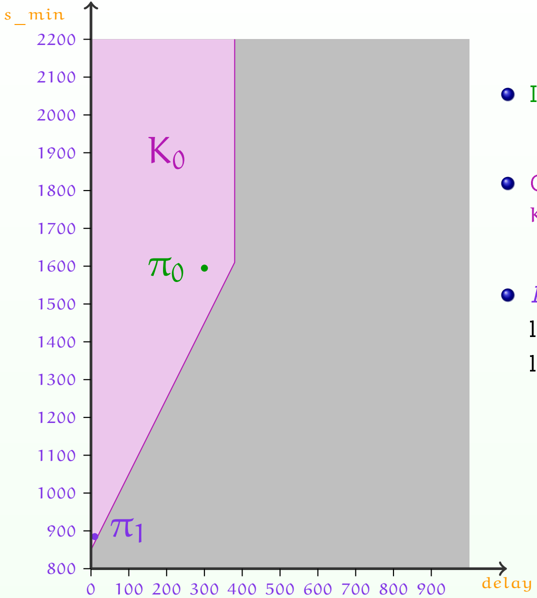
  $s\_min = 1590\, ns \qquad delay = 300\, ns$

- Output:

  $K_0 : \quad 2delay < 760$
  $\wedge\ 2delay + 850 < s\_min$

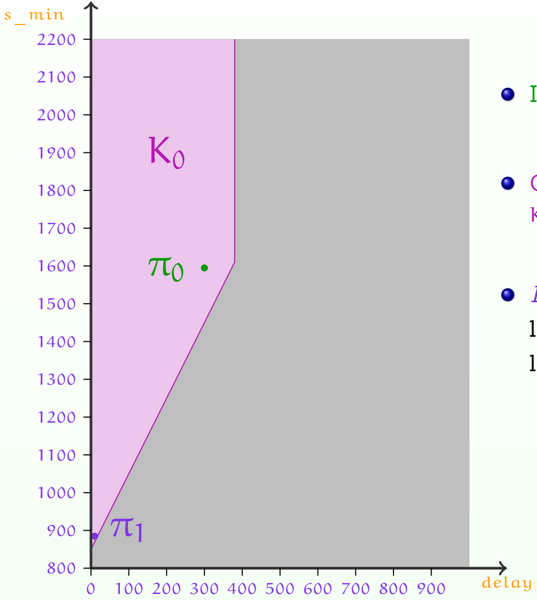
- $Prob_5$: Minimum probability that a leader is elected after 5 rounds or less
  - PRISM does not succeed in computing $Prob_5$ for π_0

Application to the Root Contention Protocol



- Input: IEEE reference valuation
  $s\_min = 1590\,ns \qquad delay = 300\,ns$
- Output:
  K_0 : $2delay < 760$
  $\wedge\ 2delay + 850 < s\_min$
- $Prob_5$: Minimum probability that a leader is elected after 5 rounds or less
  - PRISM does not succeed in computing $Prob_5$ for π_0
  - For a smaller valuation π_1, PRISM computes that $Prob_5 = 0.94$

Application to the Root Contention Protocol

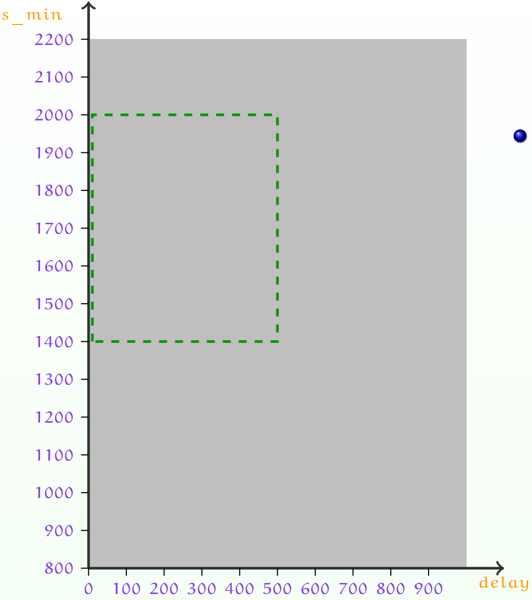


- Input: IEEE reference valuation
  $s\_min = 1590\,ns \qquad delay = 300\,ns$
- Output:
  K_0 : $\quad 2delay < 760$
  $\wedge\ 2delay + 850 < s\_min$
- $Prob_5$: Minimum probability that a leader is elected after 5 rounds or less
  - PRISM does not succeed in computing $Prob_5$ for π_0
  - For a smaller valuation π_1, PRISM computes that $Prob_5 = 0.94$
  - By correctness of our method, $Prob_5 = 0.94$ also for π_0

Extension of the Cartography to PPTAs

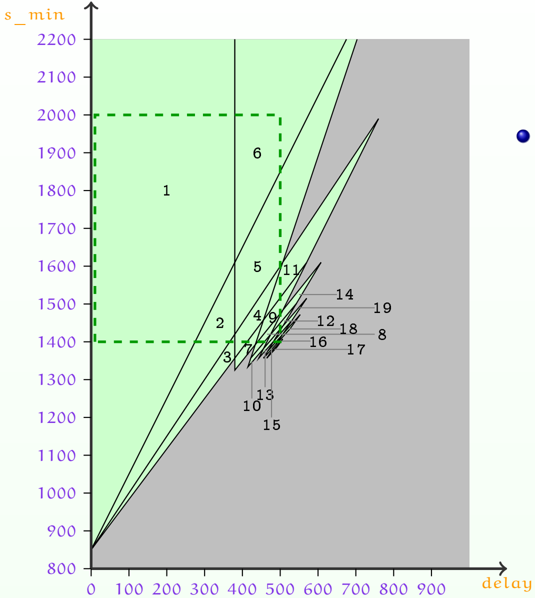
1. Construct a derandomized (non-probabilistic) version $\mathcal{A}^*$ of $\mathcal{A}$
2. Apply the cartography algorithm to $\mathcal{A}^*$ and V_0

The Root Contention Protocol: Cartography



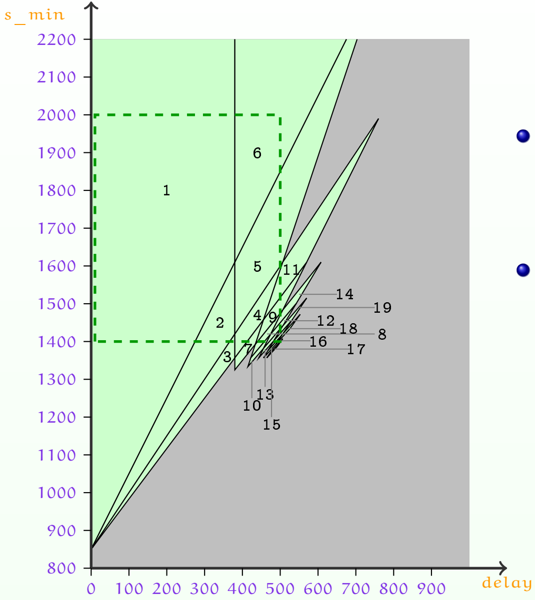
- We consider the following V_0 : $s\_min \in [1400; 2000]$ and $delay \in [10; 500]$

The Root Contention Protocol: Cartography



- We consider the following V_0 : $s\_min \in [1400; 2000]$ and $delay \in [10; 500]$

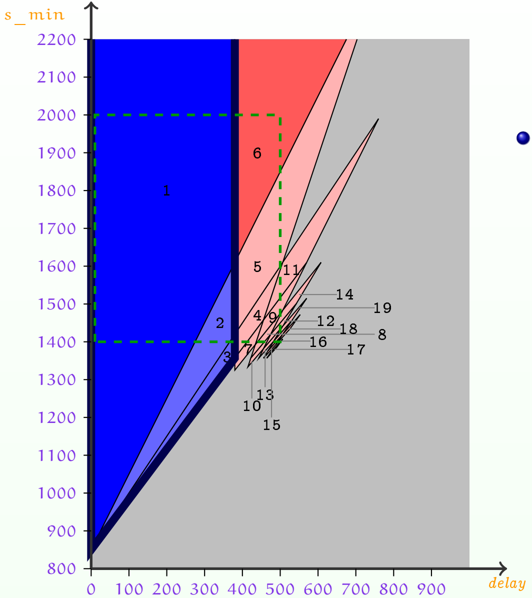
The Root Contention Protocol: Cartography



- We consider the following V_0 : $s\_min \in [1400; 2000]$ and $delay \in [10; 500]$
- Remarks
  - Tiles 1 and 6 are infinite towards one dimension
  - The cartography does not cover the whole real-valued space within V_0

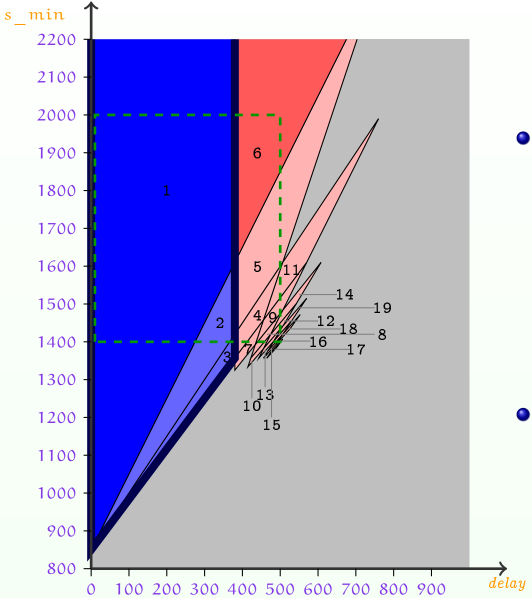
    (holes in the lower right corner of V_0)

The Root Contention Protocol: Partition



- $Prob_5$: "Minimum probability that a leader is elected after five rounds or less"
  - Tile 1: $Prob_5 = 0.94$
  - Tiles 2 and 3: $Prob_5 = 0.79$
  - Tile 6: $Prob_5 = 0.66$
  - Other tiles: $Prob_5 = 0.5$

The Root Contention Protocol: Partition



- $Prob_5$: "Minimum probability that a leader is elected after five rounds or less"
  - Tile 1: $Prob_5 = 0.94$
  - Tiles 2 and 3: $Prob_5 = 0.79$
  - Tile 6: $Prob_5 = 0.66$
  - Other tiles: $Prob_5 = 0.5$
- Find parameter valuations such that $Prob_5 \geq 0.75$
  - Good tiles: 1, 2 and 3

Advantages of the Probabilistic Cartography

- Allows the rescaling of the timing constants
  - Allows a much faster computation of probabilities in practice
- Avoids the repeated computation of probabilities for many different values of the parameters
- Gives a quantitative refinement of the good parameters problem
  - Instead of a partition with a binary criterion (good / bad), partition according to various probabilities

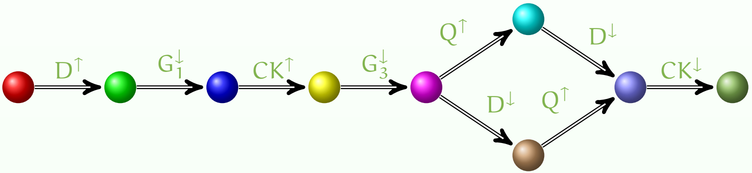
Outline

Summary

- Inverse Method: Algorithm IM
  - Original method for the synthesis of timing parameters
  - Gives a criterion of robustness to the system
  - Implementation: IMITATOR II
    - Application to an industrial case study: optimization of timing delays in the SPSMALL memory (ST-Microelectronics)
- Behavioral cartography: Algorithm BC
  - Solves the good parameters problem
- Extension of IM and BC to probabilistic systems
  - Synthesizes a set of tiles, with uniform min/max reachability probabilities within each tile
  - Allows the rescaling of timing constants
  - Application to several randomized protocols

Future Work

- Extend the inverse method to hybrid automata
  - Allow to consider continuous variables driven by differential equations
- Consider a weaker property than equality of trace sets
  - Reference trace with partial orders



- Application to other formalisms
  - Priced / Weighted Timed Automata
  - Timed extensions of Petri Nets

References I

Alur, R. and Dill, D. L. (1994).
A theory of timed automata.
TCS, 126(2):183–235.

Alur, R., Henzinger, T. A., and Vardi, M. Y. (1993).
Parametric real-time reasoning.
In *STOC '93*, pages 592–601. ACM.

André, É. (2010).
IMITATOR II: A tool for solving the good parameters problem in timed automata.
In Chen, Y.-F. and Rezine, A., editors, *INFINITY'10*, volume 39 of *Electronic Proceedings in Theoretical Computer Science*, pages 91–99.

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

André, É., Fribourg, L., and Sproston, J. (2009).
An extension of the inverse method to probabilistic timed automata.
In *AVoCS'09*, volume 23 of *Electronic Communications of the EASST*.

References II

Clarisó, R. and Cortadella, J. (2007).
The octahedron abstract domain.
Sci. Comput. Program., 64(1):115–139.

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.

Gregersen, H. and Jensen, H. E. (1995).
Formal design of reliable real time systems.
Master's thesis, Department of Mathematics and Computer Science, Aalborg University.

Henzinger, T. A. and Wong-Toi, H. (1996).
Using HyTech to synthesize control parameters for a steam boiler.
In *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control, LNCS 1165*. Springer-Verlag.

Hinton, A., Kwiatkowska, M., Norman, G., and Parker, D. (2006).
PRISM: A tool for automatic verification of probabilistic systems.
In *TACAS'06*, volume 3920 of *LNCS*, pages 441–444. Springer.

References III

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

Kwiatkowska, M., Norman, G., Segala, R., and Sproston, J. (2002a).
Automatic verification of real-time systems with discrete probability distributions.
Theoretical Computer Science, 282:101–150.

Kwiatkowska, M., Norman, G., and Sproston, J. (2002b).
Probabilistic model checking of the IEEE 802.11 wireless local area network protocol.
In *Proc. PAPM/PROBMIV'02*, volume 2399 of *LNCS*, pages 169–187. Springer.

Correctness of IM

Theorem (Correctness)

Suppose that $\mathrm{IM}(\mathcal{A}, \pi_0)$ *terminates with output* K_0. *We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Correctness of IM

Theorem (Correctness)

Suppose that $\mathrm{IM}(\mathcal{A}, \pi_0)$ *terminates with output* K_0. *We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Idea of the proof

$$\mathcal{A}[\pi] \quad \boxed{(q_0, w_0)} \xRightarrow{a_0} \boxed{(q_1, w_1)} \xRightarrow{a_1} \cdots \xRightarrow{a_{n-1}} \boxed{(q_n, w_n)}$$

$$\mathcal{A}(K_0) \quad \boxed{(q_0, C_0)} \xRightarrow{a_0} \boxed{(q_1, C_1)} \xRightarrow{a_1} \cdots \xRightarrow{a_{n-1}} \boxed{(q_n, C_n)}$$

Correctness of IM

Theorem (Correctness)

Suppose that $\mathrm{IM}(\mathcal{A}, \pi_0)$ *terminates with output* K_0*. We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Idea of the proof

$\mathcal{A}[\pi]$ $\quad (q_0, w_0) \xrightarrow{a_0} (q_1, w_1) \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} (q_n, w_n)$

$\mathcal{A}(K_0)$ $\quad (q_0, C_0) \xrightarrow{a_0} (q_1, C_1) \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} (q_n, C_n)$

Correctness of IM

Theorem (Correctness)

Suppose that $IM(\mathcal{A}, \pi_0)$ *terminates with output* K_0*. We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Idea of the proof

$$\mathcal{A}[\pi] \qquad (q_0, w_0) \xrightarrow{a_0} (q_1, w_1) \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} (q_n, w_n)$$

$$\Uparrow$$

$$\mathcal{A}(K_0) \qquad (q_0, C_0) \xrightarrow{a_0} (q_1, C_1) \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} (q_n, C_n)$$

Correctness of IM

Theorem (Correctness)

Suppose that $\mathrm{IM}(\mathcal{A}, \pi_0)$ *terminates with output* K_0. *We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Idea of the proof

$$\begin{array}{ll} \mathcal{A}[\pi] & (q_0, w_0) \xrightarrow{a_0} (q_1, w_1) \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} (q_n, w_n) \\ & \hphantom{(q_0, w_0) \xrightarrow{a_0} (q_1, w_1) \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}}} \Uparrow \\ \mathcal{A}(K_0) & (q_0, C_0) \xrightarrow{a_0} (q_1, C_1) \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} (q_n, C_n) \end{array}$$

Correctness of IM

Theorem (Correctness)

Suppose that $\mathrm{IM}(\mathcal{A}, \pi_0)$ *terminates with output* K_0. *We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Idea of the proof

$$
\begin{tikzcd}
\mathcal{A}[\pi] & (q_0, w_0) \arrow[r, "a_0"] & (q_1, w_1) \arrow[r, "a_1"] & \cdots \arrow[r, "a_{n-1}"] & (q_n, w_n) \\
\mathcal{A}(K_0) & (q_0, C_0) \arrow[r, "a_0"] & (q_1, C_1) \arrow[r, "a_1"] & \cdots \arrow[r, "a_{n-1}"] & (q_n, C_n) \arrow[u, Rightarrow]
\end{tikzcd}
$$

Correctness of IM

Theorem (Correctness)

Suppose that $\mathrm{IM}(\mathcal{A}, \pi_0)$ *terminates with output* K_0. *We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Idea of the proof

$$
\begin{tikzcd}
\mathcal{A}[\pi] & (q_0, w_0) \arrow[r, Rightarrow, "a_0"] & (q_1, w_1) \arrow[r, Rightarrow, "a_1"] & \cdots \arrow[r, Rightarrow, "a_{n-1}"] & (q_n, w_n) \\
\mathcal{A}(K_0) & (q_0, C_0) \arrow[r, Rightarrow, "a_0"] & (q_1, C_1) \arrow[r, Rightarrow, "a_1"] & \cdots \arrow[r, Rightarrow, "a_{n-1}"] & (q_n, C_n) \arrow[u, Rightarrow]
\end{tikzcd}
$$

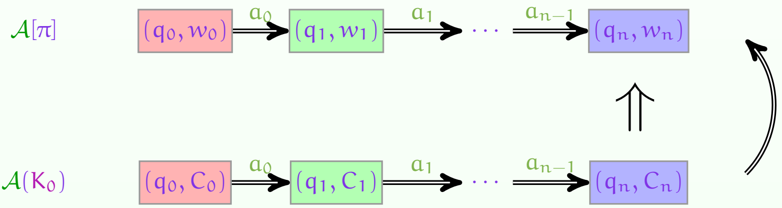
Correctness of IM

Theorem (Correctness)

Suppose that $\mathrm{IM}(\mathcal{A}, \pi_0)$ *terminates with output* K_0. *We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Idea of the proof



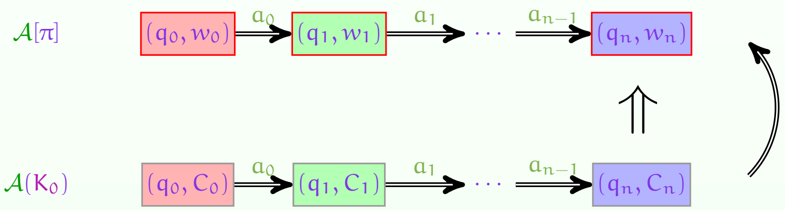
Correctness of IM

Theorem (Correctness)

Suppose that $\mathrm{IM}(\mathcal{A}, \pi_0)$ *terminates with output* K_0*. We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Idea of the proof



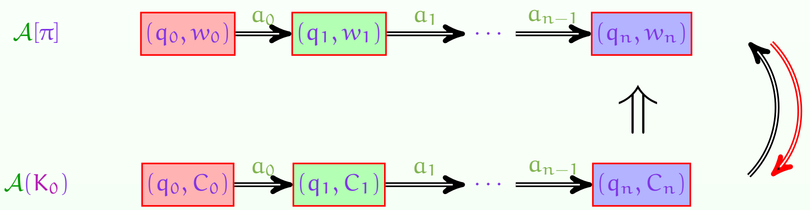
Correctness of IM

Theorem (Correctness)

Suppose that $\mathrm{IM}(\mathcal{A}, \pi_0)$ *terminates with output* K_0*. We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Idea of the proof



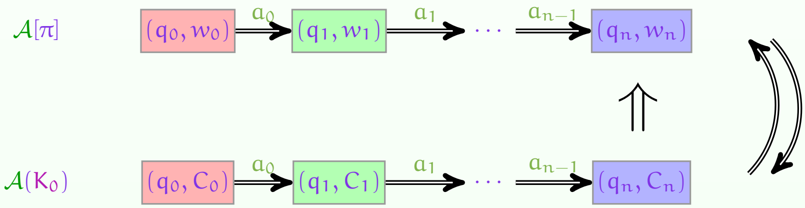
Correctness of IM

Theorem (Correctness)

Suppose that $IM(\mathcal{A}, \pi_0)$ *terminates with output* K_0. *We have:*

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, $\mathcal{A}[\pi_0]$ *and* $\mathcal{A}[\pi]$ *have the same trace sets.*

Idea of the proof



Preservation of LTL Formulae

Corollary of the correctness of IM

Proposition (LTL-preservation)

Let $K_0 = IM(\mathcal{A}, \pi_0)$, $\pi \models K_0$ *and* φ *an LTL formula verifiable on finite traces.*
Then φ *holds for* $\mathcal{A}[\pi]$ *iff* φ *holds for* $\mathcal{A}[\pi_0]$.

Idea of the Proof of Correctness for PPTAs

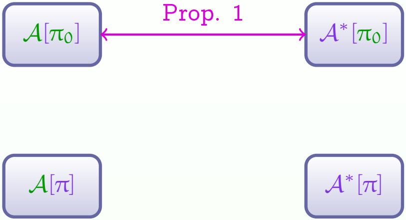
- Given $\pi \models IM(\mathcal{A}^*, \pi_0)$:

$\boxed{\mathcal{A}[\pi_0]}$ $\boxed{\mathcal{A}^*[\pi_0]}$

$\boxed{\mathcal{A}[\pi]}$ $\boxed{\mathcal{A}^*[\pi]}$

Idea of the Proof of Correctness for PPTAs

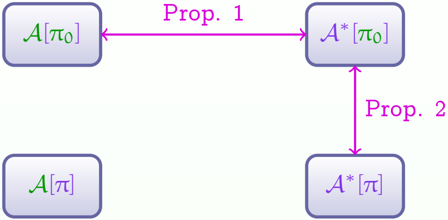
- Given $\pi \models \mathrm{IM}(\mathcal{A}^*, \pi_0)$:



- Justification
  - Prop. 1: The set of derandomized traces of $\mathcal{A}[\pi_0]$ is equal to the set of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$
    [A., Fribourg, Sproston, AVoCS'09]

Idea of the Proof of Correctness for PPTAs

- Given $\pi \models IM(\mathcal{A}^*, \pi_0)$:



- Justification
  - Prop. 1: The set of derandomized traces of $\mathcal{A}[\pi_0]$ is equal to the set of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ [A., Fribourg, Sproston, AVoCS'09]
  - Prop. 2: The sets of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ and $\mathcal{A}^*[\pi]$ are equal [A., Chatain, Encrenaz, Fribourg, IJFCS'09]

Idea of the Proof of Correctness for PPTAs

- Given $\pi \models IM(\mathcal{A}^*, \pi_0)$:

$$\begin{tikzcd}
\mathcal{A}[\pi_0] \arrow[rr, leftrightarrow, "\text{Prop. 1}"] & & \mathcal{A}^*[\pi_0] \arrow[d, leftrightarrow, "\text{Prop. 2}"] \\
\mathcal{A}[\pi] \arrow[rr, leftrightarrow, "\text{Prop. 1}"] & & \mathcal{A}^*[\pi]
\end{tikzcd}$$

- Justification
  - Prop. 1: The set of derandomized traces of $\mathcal{A}[\pi_0]$ is equal to the set of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ [A., Fribourg, Sproston, AVoCS'09]
  - Prop. 2: The sets of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ and $\mathcal{A}^*[\pi]$ are equal [A., Chatain, Encrenaz, Fribourg, IJFCS'09]

Idea of the Proof of Correctness for PPTAs

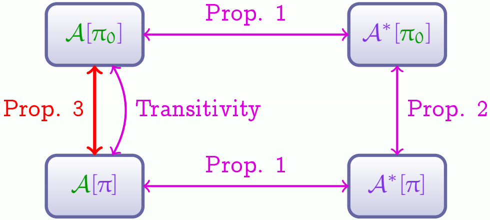
- Given $\pi \models IM(\mathcal{A}^*, \pi_0)$:

$$\begin{tikzcd}
\mathcal{A}[\pi_0] \arrow[rrr, leftrightarrow, "\text{Prop. 1}"] \arrow[d, leftrightarrow, bend left, "\text{Transitivity}"] & & & \mathcal{A}^*[\pi_0] \arrow[d, leftrightarrow, "\text{Prop. 2}"] \\
\mathcal{A}[\pi] \arrow[rrr, leftrightarrow, "\text{Prop. 1}"] & & & \mathcal{A}^*[\pi]
\end{tikzcd}$$

- Justification
  - Prop. 1: The set of derandomized traces of $\mathcal{A}[\pi_0]$ is equal to the set of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ [A., Fribourg, Sproston, AVoCS'09]
  - Prop. 2: The sets of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ and $\mathcal{A}^*[\pi]$ are equal [A., Chatain, Encrenaz, Fribourg, IJFCS'09]

Idea of the Proof of Correctness for PPTAs

- Given $\pi \models IM(\mathcal{A}^*, \pi_0)$:



- Justification
  - Prop. 1: The set of derandomized traces of $\mathcal{A}[\pi_0]$ is equal to the set of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ [A., Fribourg, Sproston, AVoCS'09]
  - Prop. 2: The sets of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ and $\mathcal{A}^*[\pi]$ are equal [A., Chatain, Encrenaz, Fribourg, IJFCS'09]
  - Prop. 3: If the sets of derandomized traces of $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are equal, then the sets of probabilistic traces of $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are equal [kns02]

Summary of Experiments: IM

- Computation times of various case studies
  - Experiments conducted on an Intel Core2 Duo 2.4 GHz with 2 Gb

Example	PTAs	loc./PTA	$\|X\|$	$\|P\|$	iter.	$\|K_0\|$	states	trans.	Time
SR-latch	3	[3, 8]	3	3	5	2	4	3	0.007
Flip-flop	5	[4, 16]	5	12	9	6	11	10	0.122
And–Or	3	[4, 8]	4	12	14	4	13	13	0.15
Latch circuit	7	[2, 5]	8	13	12	6	18	17	0.345
CSMA/CD	3	[3, 8]	3	3	19	2	219	342	1.01
RCP	5	[6, 11]	6	5	20	2	327	518	2.3
BRP	6	[2, 6]	7	6	30	7	429	474	34
SIMOP	5	[5, 16]	8	7	53	9	1108	1404	67
SPSMALL	28	[2, 11]	28	62	94	45	129	173	461

Summary of Experiments: BC

- 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
RCP	5	$[6, 11]$	6	3	186050	19	5688	9312	7018
SPSMALL	28	$[2, 11]$	28	3	784	213	145	196	31641