4th Workshop on Reachability Problems

Brno

28th August 2010

Behavioral Cartography of Timed Automata

Étienne André, Laurent Fribourg

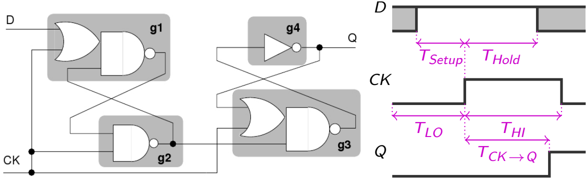
Laboratoire Spécification et Vérification
LSV, ENS de Cachan & CNRS, France

The Good Parameters Problem

- Context: Verification of Timed Systems
- Good parameters problem
  - Synthesize a set of values of the timing parameters guaranteeing that the system behaves well (e.g., avoids any bad state)
- Classical approaches
  - Computation of all the reachable states, and intersection with the set of bad states [Alur et al., 1995]
  - Approach based on CEGAR (Counter-Example Guided Abstraction Refinement [Clarke et al., 2000, Frehse et al., 2008])
- New approach: method of behavioral cartography

An Example: Flip-Flop Circuit (1/2)

- Schematics [Clarisó and Cortadella, 2007]



  - 4 elements: G_1, G_2, G_3, G_4 with internal signals g_1 to g_4
  - 2 input signals (D and CK), 1 output signal (Q)
- Timing parameters
  - Traversal delays of the gates by the electric current
    - ★ Parametric interval; example for G_1: $[\delta_1^-, \delta_1^+]$
  - Durations of low (T_{LO}) and high (T_{HI}) levels of CK
  - Stabilization time of D: T_{Setup}, T_{Hold}

An Example: Flip-Flop Circuit (2/2)

- We suppose given a valuation π_0 of the parameters (called point)

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

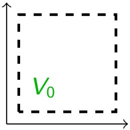
  - This point guarantees a good behavior:
    - $Q^\uparrow$ occurs before $CK^\downarrow$

- We are looking for a set of points (containing π_0) for which the system behaves well

General Problems

We consider a system modeled by a parametric timed automaton.

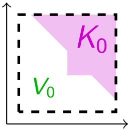
- The good parameters problem:
  - "Given a bounded parameter domain V_0, find a set of points of good behavior in V_0 (ideally the largest one)"



General Problems

We consider a system modeled by a parametric timed automaton.

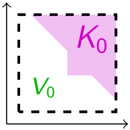
- The good parameters problem:
  - "Given a bounded parameter domain V_0, find a set of points of good behavior in V_0 (ideally the largest one)"



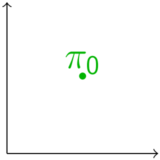
General Problems

We consider a system modeled by a parametric timed automaton.

- The good parameters problem:
  - "Given a bounded parameter domain V_0, find a set of points of good behavior in V_0 (ideally the largest one)"



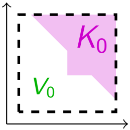
- This problem reduces to the inverse problem:
  - "Given a reference point π_0, find other points around π_0 of same behavior"



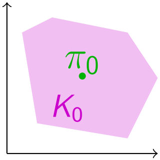
General Problems

We consider a system modeled by a parametric timed automaton.

- The good parameters problem:
  - "Given a bounded parameter domain V_0, find a set of points of good behavior in V_0 (ideally the largest one)"



- This problem reduces to the inverse problem:
  - "Given a reference point π_0, find other points around π_0 of same behavior"



Outline

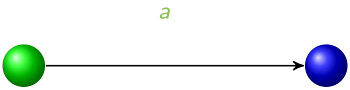
Outline

Timed Automaton

- Finite state automaton (sets of locations)

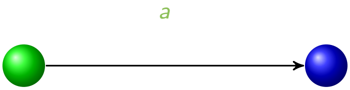
Timed Automaton

- Finite state automaton (sets of locations and actions)



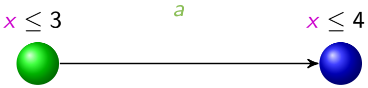
Timed Automaton

- 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)



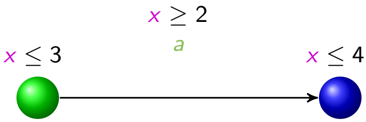
Timed Automaton

- 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)
- Features
  - Location invariant: property to be verified by the clocks to stay at a location



Timed Automaton

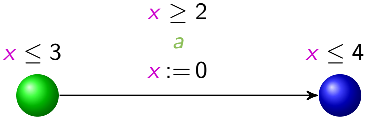
- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate)
- Features
  - Location invariant: property to be verified by the clocks to stay at a location
  - Transition guard: property to be verified by the clocks to enable a transition



Timed Automaton

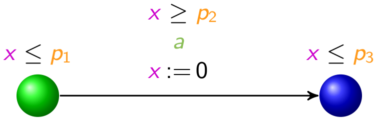
- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate)

- Features
  - Location invariant: property to be verified by the clocks to stay at a location
  - Transition guard: property to be verified by the clocks to enable a transition
  - Clock reset: 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)
  - A set P of M parameters (i.e., unknown constants), used in guards and invariants
- Features
  - Location invariant: property to be verified by the clocks and the parameters to stay at a location
  - Transition guard: property to be verified by the clocks and the parameters to enable a transition
  - Clock reset: clocks can be set to 0 at each transition



States and Traces

- 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

States and Traces

- 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
- (Parametric) state of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over the parameters

States and Traces

- 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
- (Parametric) state of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over the parameters
- Trace over a PTA: finite alternating sequence of locations and actions

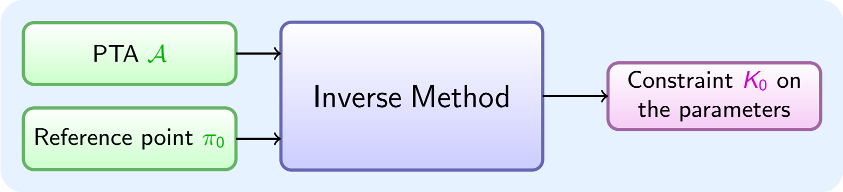
Good and Bad Traces w.r.t. a Given Property

- A trace is said to be a good trace 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

Outline

The Inverse Problem (1/2)

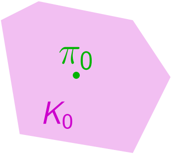


The Inverse Problem (2/2)

- Input
  - A PTA $\mathcal{A}$
  - A reference valuation π_0 of all the parameters of $\mathcal{A}$
    - ★ Exemplifying a good behavior
      (all traces of $\mathcal{A}[\pi_0]$ correspond to good behaviors)

The Inverse Problem (2/2)

- Input
  - A PTA $\mathcal{A}$
  - A reference valuation π_0 of all the parameters of $\mathcal{A}$
    - Exemplifying a good behavior
      (all traces of $\mathcal{A}[\pi_0]$ correspond to good behaviors)
- Output: tile K_0
  - Convex constraint on the parameters such that
    - $\pi_0 \models K_0$
    - For all point $\pi \models K_0$, $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ have the same trace sets



The Inverse Method: General Idea [André et al., 2009a]

Start with $K_0 = True$

REPEAT

1. Compute the set S of reachable parametric states under K_0
2. 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 (1/2)

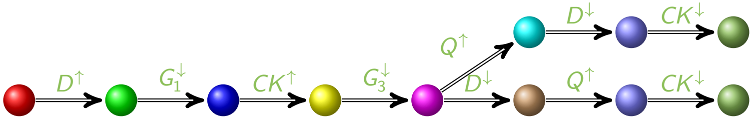
- Input: π_0

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

- Output: K_0

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

  - Corresponding trace set

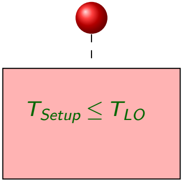


Application to the Flip-Flop Circuit (2/2)

π_0 :

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

$K_0 = True$

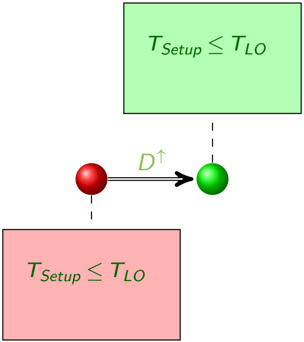


Application to the Flip-Flop Circuit (2/2)

π_0 :

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

$K_0 = True$

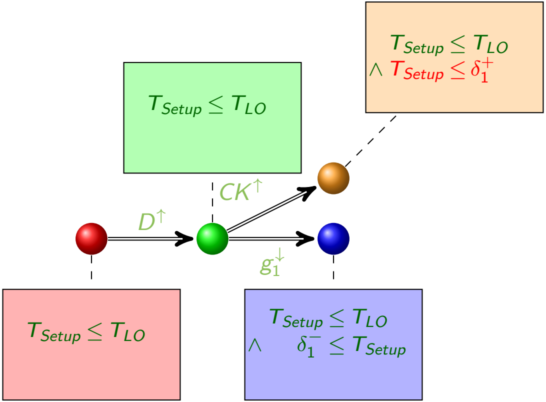


Application to the Flip-Flop Circuit (2/2)

π_0 :

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

$K_0 = True$



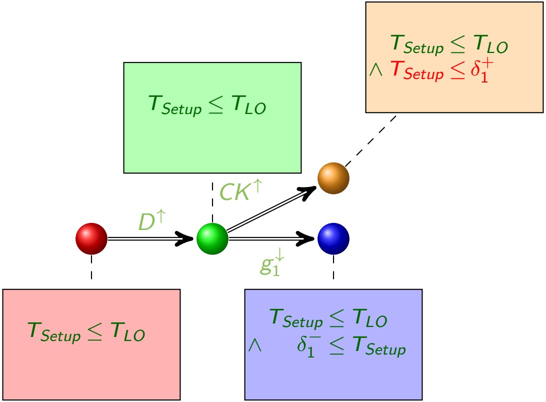
Application to the Flip-Flop Circuit (2/2)

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}	$= 17$

$K_0 =$

$T_{Setup} > \delta_1^+$



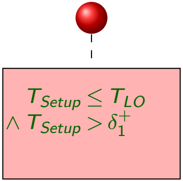
Application to the Flip-Flop Circuit (2/2)

π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$



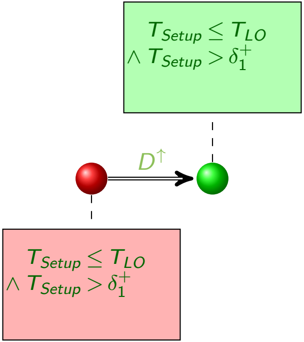
Application to the Flip-Flop Circuit (2/2)

π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$



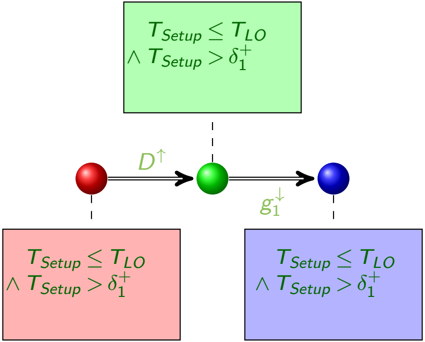
Application to the Flip-Flop Circuit (2/2)

π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$



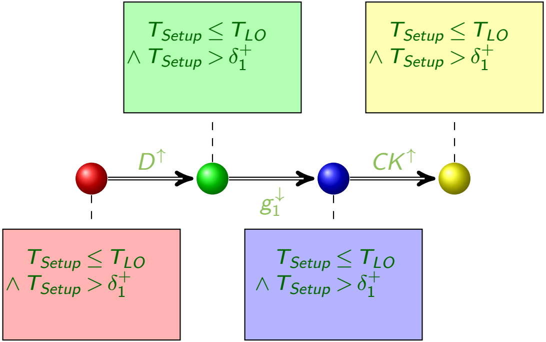
Application to the Flip-Flop Circuit (2/2)

π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$



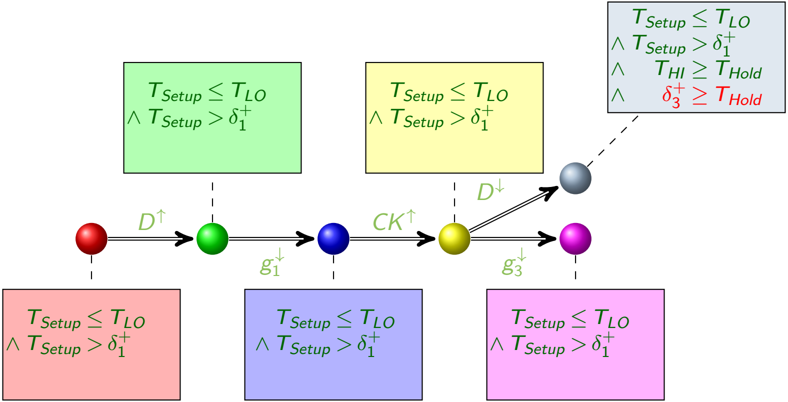
Application to the Flip-Flop Circuit (2/2)

π_0 :

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

$K_0 =$

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



Application to the Flip-Flop Circuit (2/2)

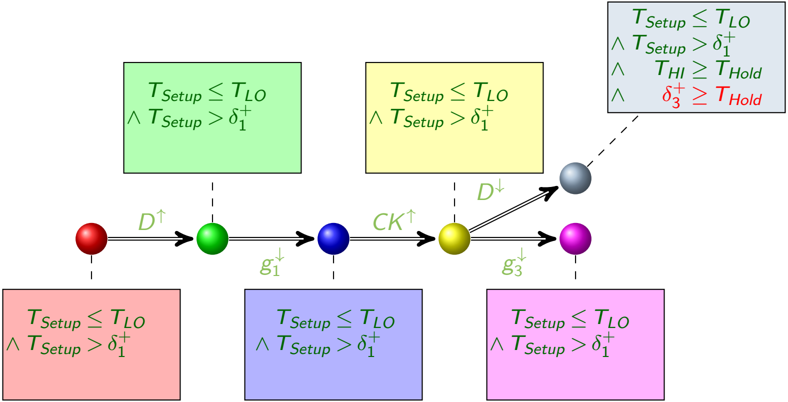
π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$

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

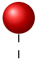


Application to the Flip-Flop Circuit (2/2)

π_0 :

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

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



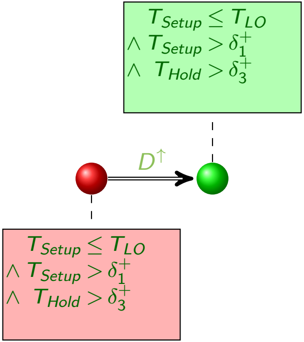
$$T_{Setup} \leq T_{LO} \wedge T_{Setup} > \delta_1^+ \wedge T_{Hold} > \delta_3^+$$

Application to the Flip-Flop Circuit (2/2)

π_0 :

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

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

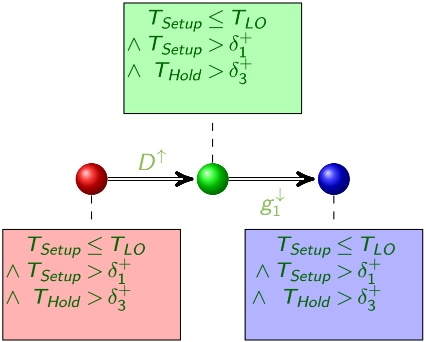


Application to the Flip-Flop Circuit (2/2)

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 (2/2)

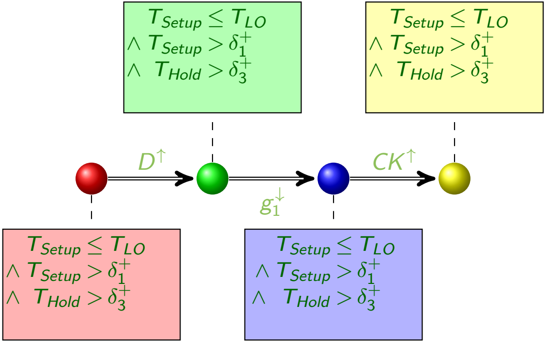
π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$

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



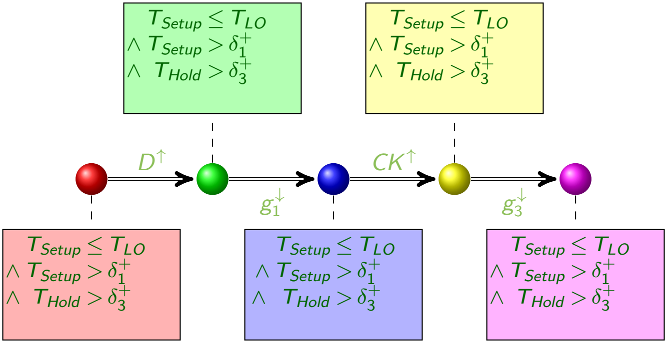
Application to the Flip-Flop Circuit (2/2)

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}	$= 17$

$K_0 =$

$$T_{Setup} > \delta_1^+ \land T_{Hold} > \delta_3^+$$

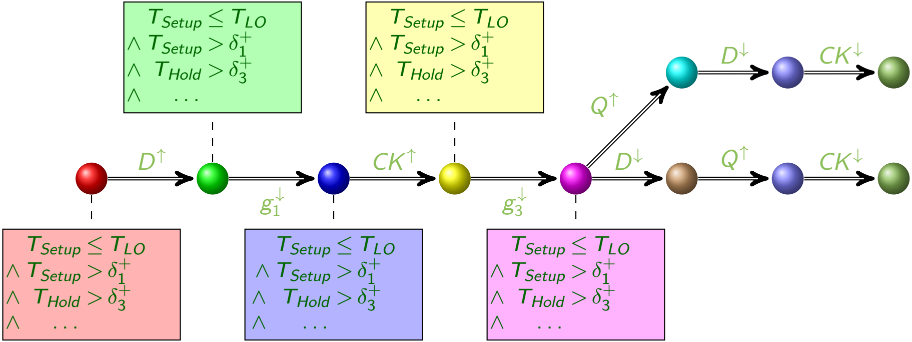


Application to the Flip-Flop Circuit (2/2)

π_0 :

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

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



Advantages and Drawbacks of the Inverse Method

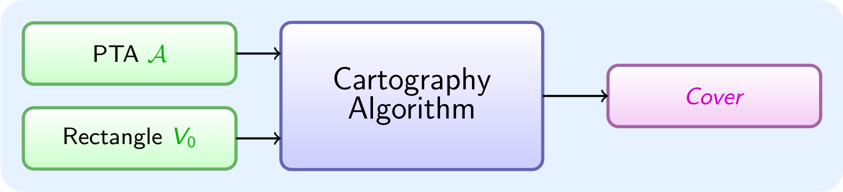
- Advantages
  - Useful to optimize timing bounds of systems
  - Terminates often in practice (unlike a brute reachability analysis, e.g., using HyTech)
  - Allows to handle dozens of parameters
- Drawbacks
  - The generated constraint K_0 is not maximal: there are points $\pi \notin K_0$ which give the same trace sets as π_0
  - The criterion of equality of trace sets may be too restrictive: 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 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

The Behavioral Cartography Algorithm



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

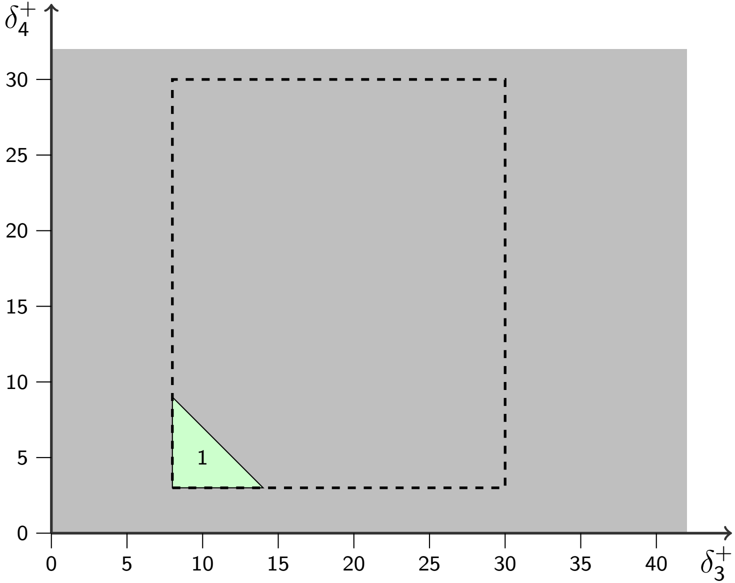
Partition into Good and Bad Tiles

- 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

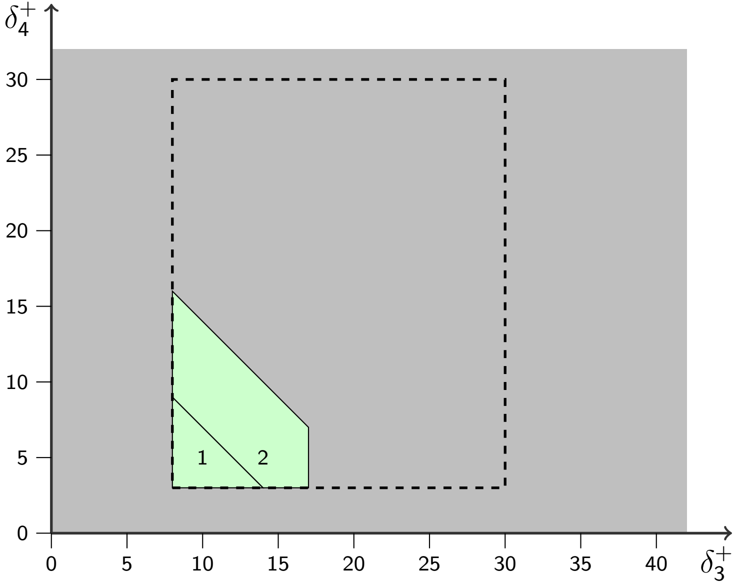
Application to our Flip-Flop Example

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

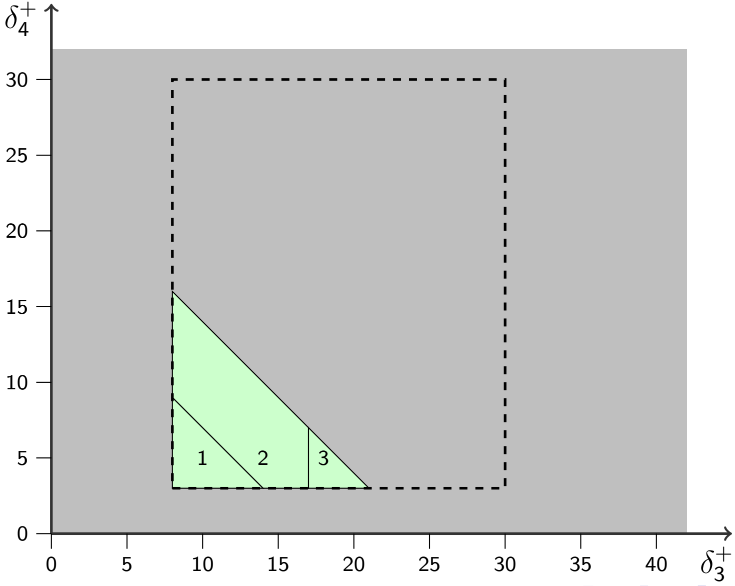
Behavioral Cartography of the Flip-Flop (δ_3^+ and δ_4^+)



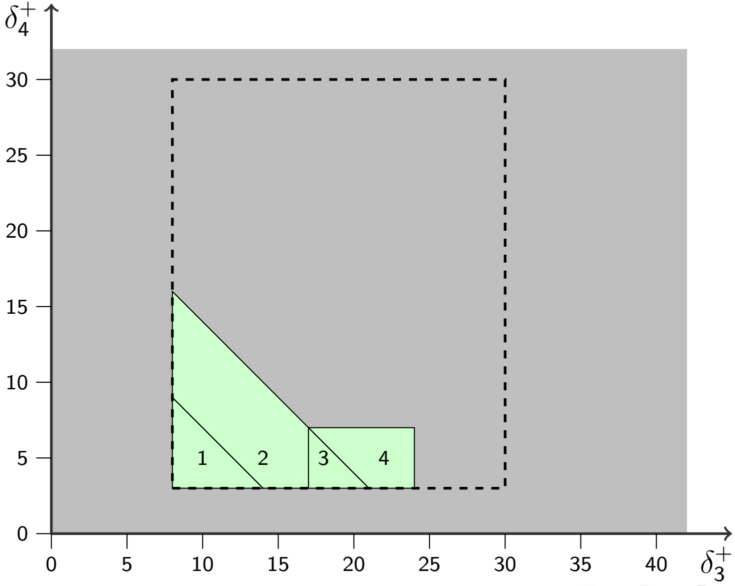
Behavioral Cartography of the Flip-Flop (δ_3^+ and δ_4^+)



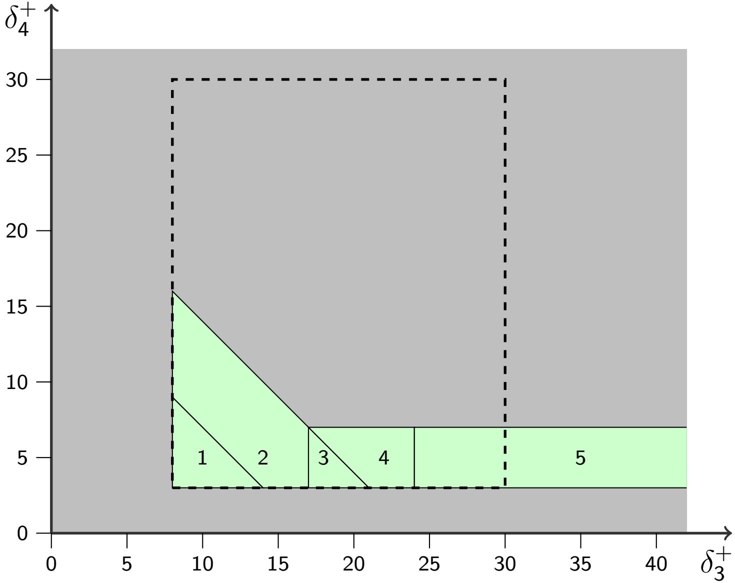
Behavioral Cartography of the Flip-Flop (δ_3^+ and δ_4^+)



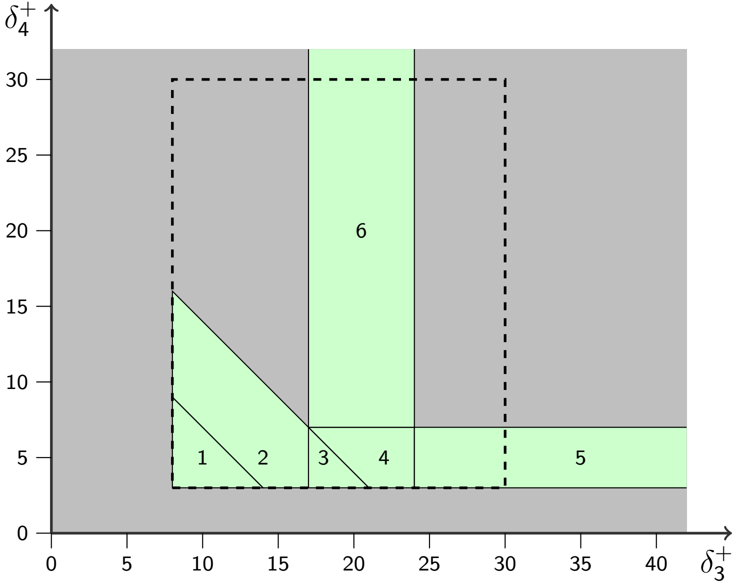
Behavioral Cartography of the Flip-Flop (δ_3^+ and δ_4^+)



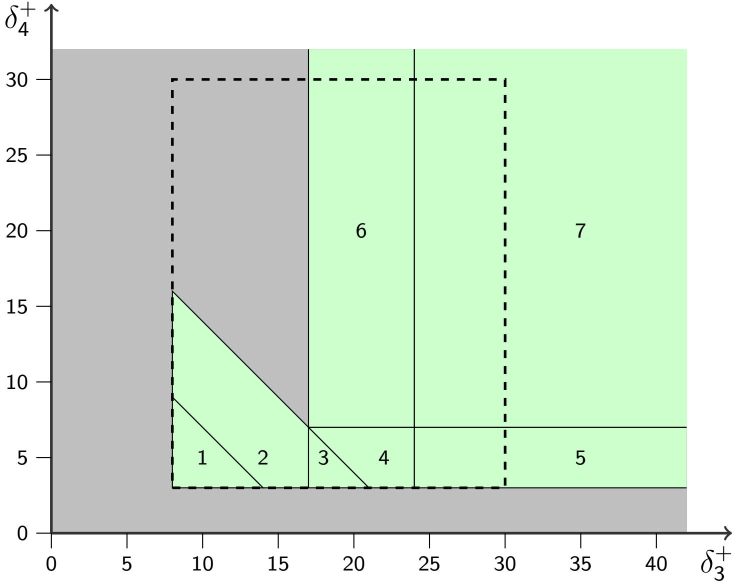
Behavioral Cartography of the Flip-Flop (δ_3^+ and δ_4^+)



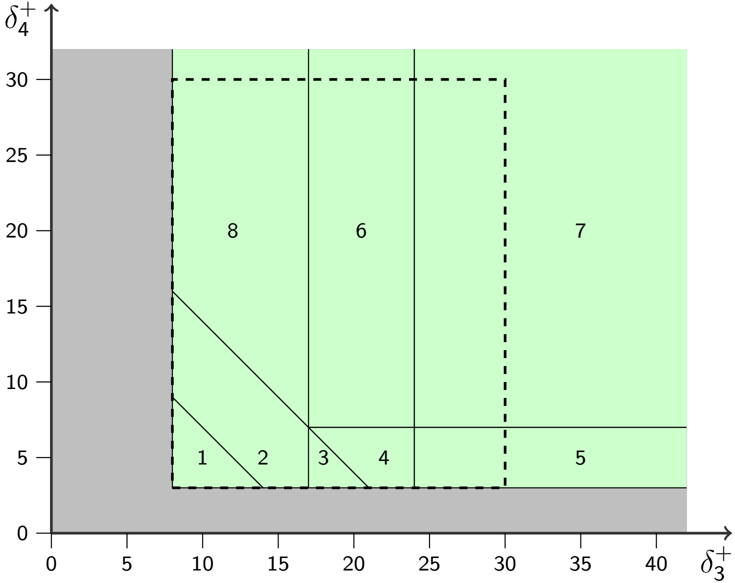
Behavioral Cartography of the Flip-Flop (δ_3^+ and δ_4^+)



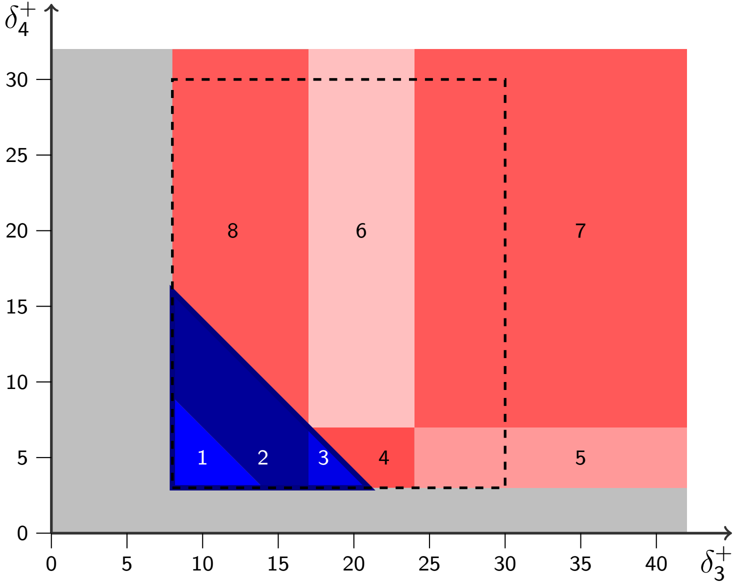
Behavioral Cartography of the Flip-Flop (δ_3^+ and δ_4^+)



Behavioral Cartography of the Flip-Flop (δ_3^+ and δ_4^+)

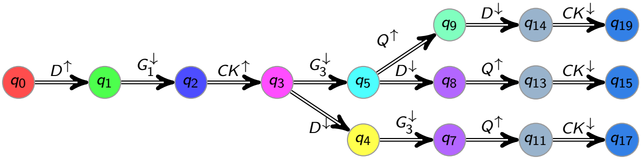


Behavioral Cartography of the Flip-Flop (δ_3^+ and δ_4^+)

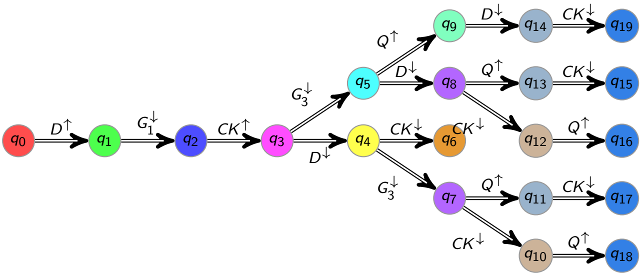


Example of good and bad tiles

- Good tile 3



- Bad tile 7



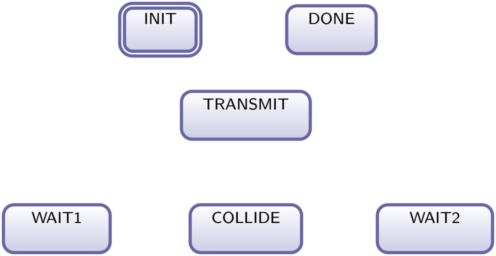
Behavioral Cartography of the Flip-flop: Remarks

- 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^+
  - $\delta_3^+ + \delta_4^+ \geq 24 \quad \wedge \quad \delta_3^+ \geq 8 \quad \wedge \quad \delta_4^+ \geq 3$

Outline

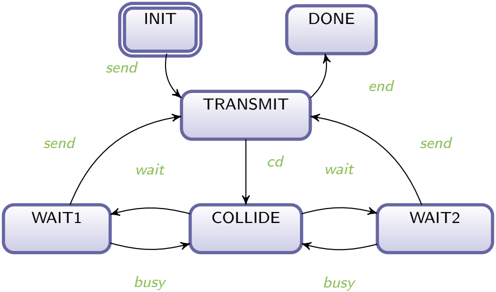
Probabilistic Timed Automaton

- Parametric Timed Automaton
  - Set of locations



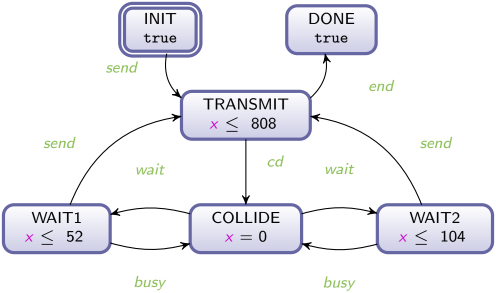
Probabilistic Timed Automaton

- Parametric Timed Automaton
  - Set of locations, set of actions



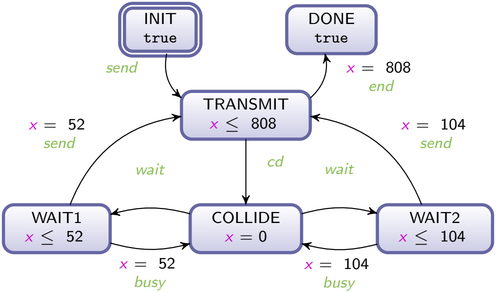
Probabilistic Timed Automaton

- Parametric Timed Automaton
  - Set of locations, set of actions
  - Set of clocks (real-valued variables increasing at the same linear rate)
    - Features: Location invariant



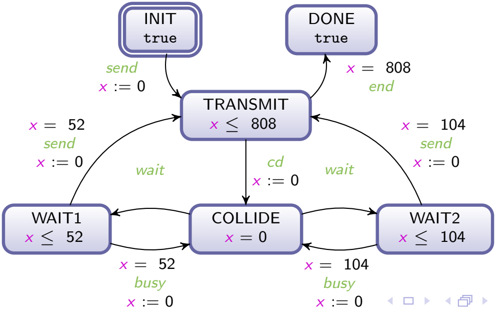
Probabilistic Timed Automaton

- Parametric Timed Automaton
  - Set of locations, set of actions
  - Set of clocks (real-valued variables increasing at the same linear rate)
    - Features: Location invariant, transition guard



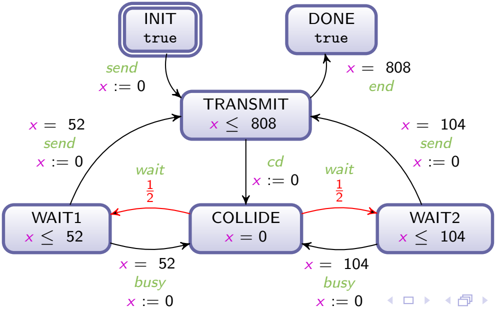
Probabilistic Timed Automaton

- Parametric Timed Automaton
  - Set of locations, set of actions
  - Set of clocks (real-valued variables increasing at the same linear rate)
    - ★ Features: Location invariant, transition guard, clock reset



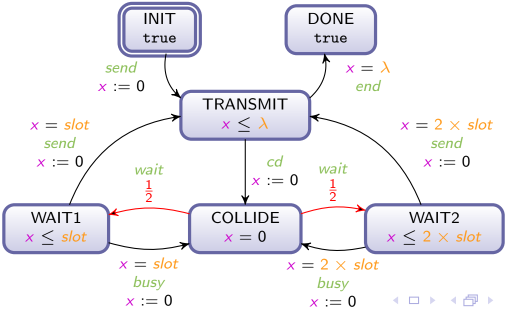
Probabilistic Timed Automaton

- Parametric Timed Automaton
  - Set of locations, set of actions
  - Set of clocks (real-valued variables increasing at the same linear rate)
    - Features: Location invariant, transition guard, clock reset
- Augmented with probabilities [Kwiatkowska et al., 2002]
  - The sum of the probabilities leaving a given location through a given action is equal to 1



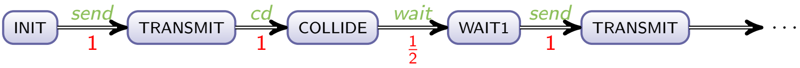
Parametric Probabilistic Timed Automaton (PPTA)

- Parametric Timed Automaton
  - Set of locations, set of actions, set of parameters (unknown constants) [André et al., 2009b]
  - Set of clocks (real-valued variables increasing at the same linear rate)
    - ★ Features: Location invariant, transition guard, clock reset
- Augmented with probabilities [Kwiatkowska et al., 2002]
  - The sum of the probabilities leaving a given location through a given action is equal to 1



Semantics

- Semantics for timed automata
  - Time elapsing in a location, and
  - Discrete actions: instantaneous transition from a location to another one
- Semantics for probabilistic timed automata
  - Time elapsing in a location, and
  - Discrete actions: instantaneous transition from a location to a distribution of locations
- Probabilistic traces
  - Finite alternating sequence of locations and actions with probabilities



Minimum and Maximum Probabilities of Reaching a State

- A scheduler s associates to every state one output distribution
  - Denoted by $\mathcal{A}^s$

- Given a scheduler, one can define the probability of reaching a location

- Minimum and maximum probabilities of reaching a given location
  - Minimum and maximum for all possible schedulers

- Derandomized form $\mathcal{A}^*$ of a PPTA $\mathcal{A}$: replace distributions by non-determinism: $\mathcal{A}^*$ becomes a PTA
  - Given some π, we have:

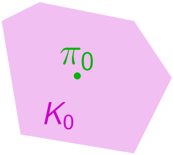
$$Traces(\mathcal{A}^*[\pi]) = \bigcup_{s \in Sched} Traces(\mathcal{A}^s[\pi])$$

The Inverse Problem for PPTAs

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

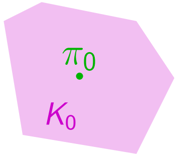
The Inverse Problem for PPTAs

- 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



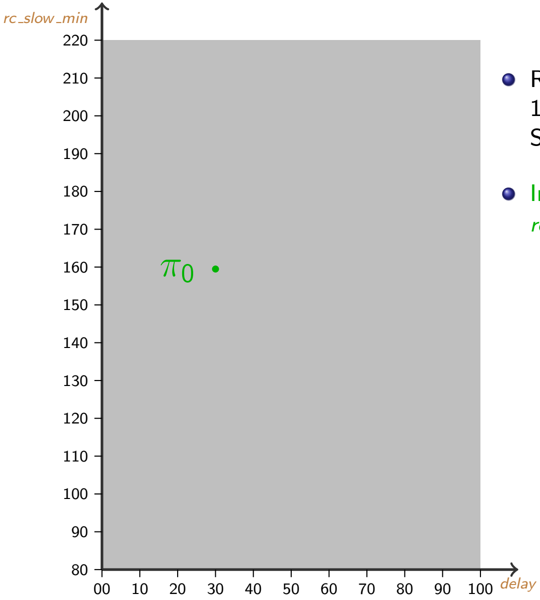
As a consequence, the minimum and maximum probabilities for reachability properties are the same in $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$

Extension of the Inverse Method to Probabilistic Systems

1. Construct a derandomized (non-probabilistic) version $\mathcal{A}^*$ of $\mathcal{A}$
2. Compute a constraint K_0 by applying the inverse method to $\mathcal{A}^*$ and π_0

Then the minimum (resp. maximum) probability of reaching a given location of $\mathcal{A}$ is the same for all $\pi \in K_0$.

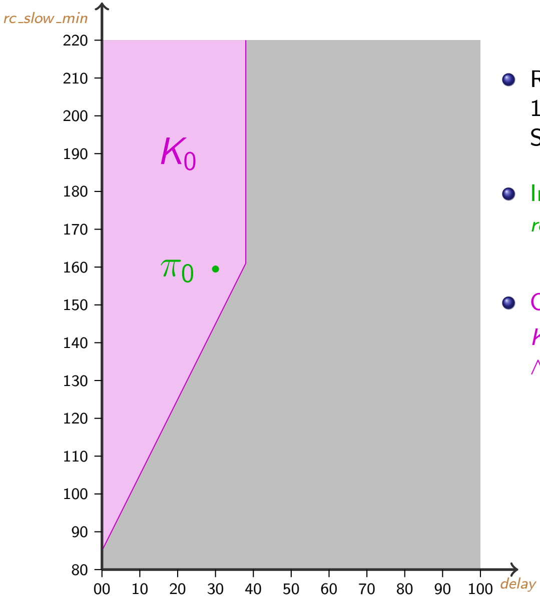
Application to The Root Contention Protocol



- 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

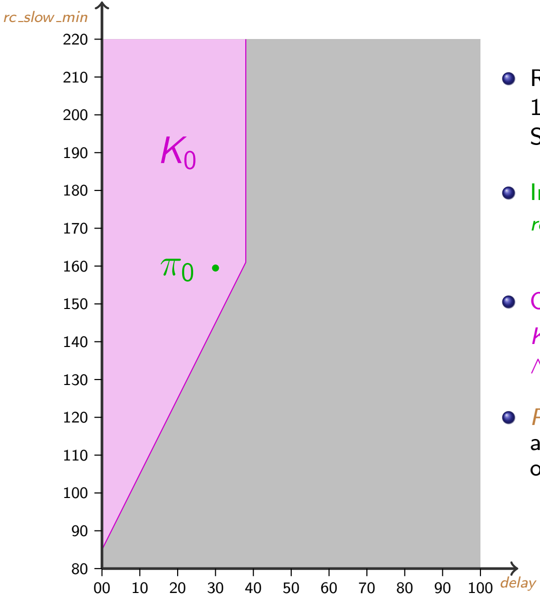
  $rc\_slow\_min = 159ns$

  $delay = 30ns$
- Output:

  $K_0 : 2delay < 76$

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

Application to The Root Contention Protocol



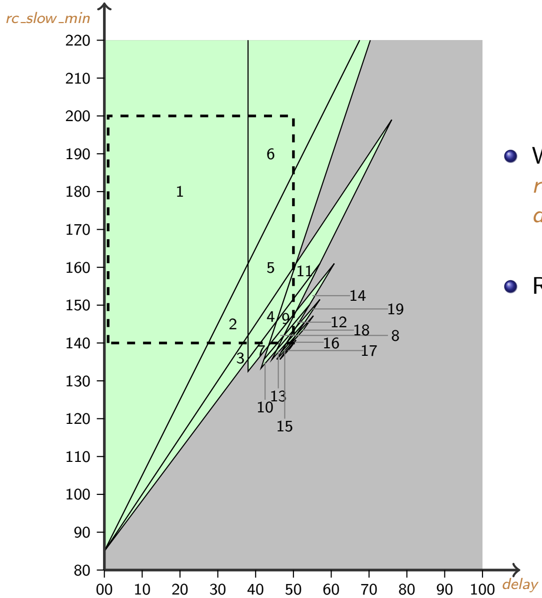
- Root contention protocol of the IEEE 1394 ("FireWire") High Performance Serial Bus [Hune et al., 2002]
- Input: IEEE reference valuation
  $rc\_slow\_min = 159ns$
  $delay = 30ns$
- Output:
  $K_0 : 2delay < 76$
  $\wedge\ 2delay + 85 < rc\_slow\_min$
- $Prop_3$: The minimum probability that a leader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

Extension of the Cartography to Probabilistic Systems

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

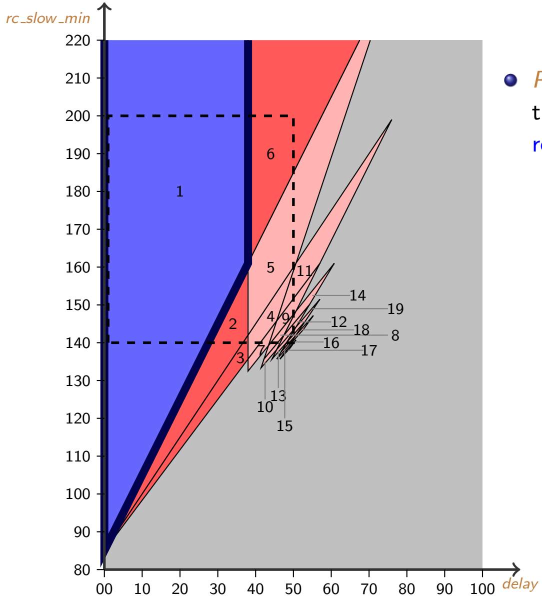
Then the minimum (resp. maximum) probability of reaching a given location of $\mathcal{A}$ is uniform within each tile of the cartography.

The Root Contention Protocol: Cartography (1/2)



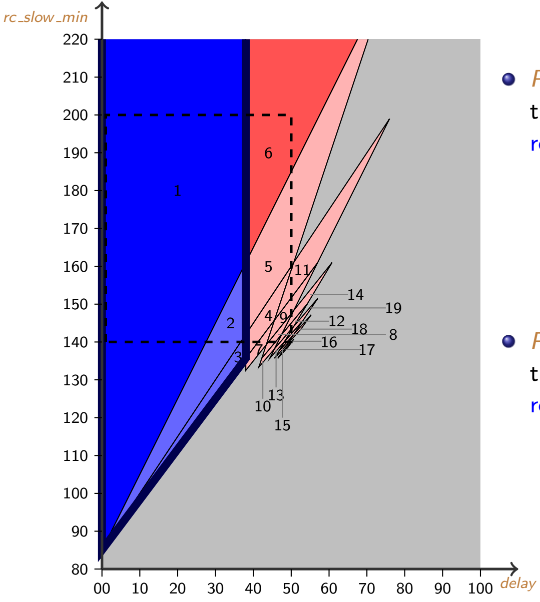
- We consider the following V_0 : $rc\_slow\_min \in [140; 200]$, and $delay \in [1; 50]$
- Remarks
  - Tiles 1 and 6 are infinite towards one dimension
  - The cartography does not cover the whole real-valued space within V_0 (holes in the lower right corner of V_0)

The Root Contention Protocol: Cartography (2/2)



- $Prop_3$: "The minimum probability that a leader is elected after three rounds or less is equal to p"
  - Tile 1: $p = 0.75$
  - Tiles 2, 3, 6: $p = 0.625$
  - Other tiles: : $p = 0.5$
  - Good tile if $p \geq 0.75$

The Root Contention Protocol: Cartography (2/2)



- *Prop*$_3$: "The minimum probability that a leader is elected after three rounds or less is equal to p"
  - Tile 1: $p = 0.75$
  - Tiles 2, 3, 6: $p = 0.625$
  - Other tiles: : $p = 0.5$
  - Good tile if $p \geq 0.75$
- *Prop*$_5$: "The minimum probability that a leader is elected after five rounds or less is equal to p"
  - Tile 1: $p = 0.94$
  - Tiles 2 and 3: $p = 0.79$
  - Tile 6: $p = 0.66$
  - Other tiles: : $p = 0.5$
  - Good tile if $p \geq 0.75$

Advantages of the Probabilistic Cartography

- Quantitative refinement of the good parameters problem
  - Instead of a partition with a binary criterion (good / bad), we have a partition according to various probabilities
- The cartography is independent from the probabilistic property
  - Only the probability associated to each tile depends on the property
  - No need to compute a cartography for each property

Implementation

- Tool IMITATOR II [André, 2010]
  - IMITATOR: "Inverse Method for Inferring Time AbstracT BehaviOR"
  - 8000 lines of code
  - 6 man-months of work
  - Program written in OCaml
  - Makes use of the PPL library

- IMITATOR II is available on its Web page
  - http://www.lsv.ens-cachan.fr/~andre/IMITATOR2

Case Studies

- Implementation in IMITATOR II
  - outputs a list of tiles with their corresponding trace set under a graphical form
  - outputs the cartography under a graphical form (for 2 parameter dimensions)
- Computation times of various case studies
  - Experiments conducted on an Intel Core2 Duo 2.4 GHz with 2 Gb

Example	PTAs	loc./PTA	$\|X\|$	$\|P\|$	$\|V_0\|$	tiles	states	trans.	Time
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 (1/2)

- Inverse Method: Algorithm *IM*
  - Modeling of a system with parametric timed automata
  - Starting with a valuation π_0 of the system, we generate a constraint K_0 with the same trace set as π_0
- Behavioral cartography: Algorithm *BC*
  - Solves the good parameters problem: synthesizes the largest set of points within a rectangle V_0 corresponding to a given good behavior
  - Under certain conditions, covers the whole real-valued parametric space

Summary (2/2)

- Extension to probabilistic systems
  - Synthesizes a set of tiles, with uniform min/max reachability probabilities within each tile
  - Useful to compute probabilities (e.g., using Prism) for systems with large constants (notion of rescaling)
  - Avoid the repeated computation of probabilities for many different values of the parameters

Future Work

- Extend the behavioral cartography to hybrid automata
  - Allow to consider different clock rates
- 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. (2010).
IMITATOR II: A tool for solving the good parameters problem in timed automata.
In *INFINITY'10*.
To appear.

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

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

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

References II

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.

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.

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

The Inverse Method: Algorithm

Algorithm 1: $IM(\mathcal{A}, \pi_0)$

input : A PTA $\mathcal{A}$ of initial state s_0
input : Reference point π_0 of the parameters
output: Constraint K_0 on the parameters

$i \leftarrow 0$; $K_0 \leftarrow True$; $S \leftarrow \{s_0\}$
while $True$ **do**
  while *there are* π_0*-incompatible states in* S **do**
    Select a π_0-incompatible state (q, C) of S (i.e., s.t. $\pi_0 \not\models C$) ;
    Select a π_0-incompatible J in C (i.e., s.t. $\pi \not\models J$) ;
    $K_0 \leftarrow K_0 \wedge \neg J$;
    $S \leftarrow \bigcup_{j=0}^{i} Post^{j}_{\mathcal{A}(K_0)}(\{s_0\})$;
  if $Post_{\mathcal{A}(K_0)}(S) = \emptyset$ **then return** $K_0 \leftarrow \bigcap_{(q,C) \in S} (\exists X : C)$
  $i \leftarrow i + 1$;
  $S \leftarrow S \cup Post_{\mathcal{A}(K_0)}(S)$; // $S = \bigcup_{j=0}^{i} Post^{j}_{\mathcal{A}(K_0)}(\{s_0\})$