5th Workshop on Reachability Problems

Genova

28th September 2011

Synthesis of Timing Parameters Satisfying Safety Properties

Étienne André and Romain Soulat

Laboratoire Spécification et Vérification

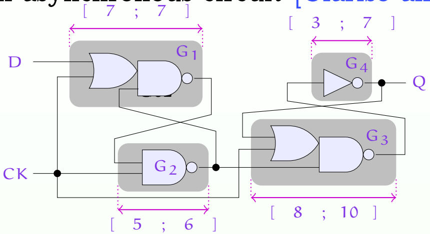
LSV, ENS de Cachan & CNRS, France

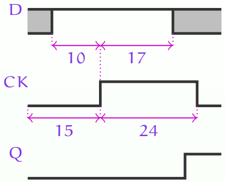
The Good Parameters Problem

- 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 [Clarke et al., 2000, Frehse et al., 2008]
- Our approach: inverse method

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

- An asynchronous circuit [Clarisó and Cortadella, 2007]

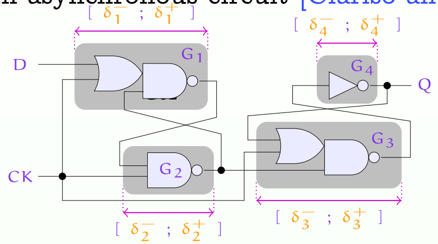


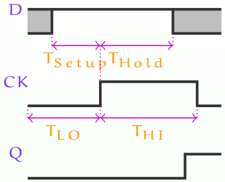


  - Concurrent behavior
    - 4 elements: G_1, G_2, G_3, G_4
    - 2 input signals (D and CK), 1 output signal (Q)

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

- An asynchronous circuit [Clarisó and Cortadella, 2007]

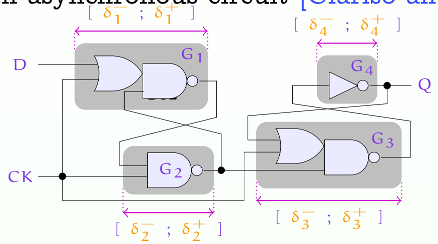


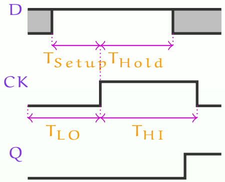


  - Concurrent behavior
    - 4 elements: G_1, G_2, G_3, G_4
    - 2 input signals (D and CK), 1 output signal (Q)
  - Timing parameters
    - Traversal delays of the gates: one interval per gate
    - 4 environment parameters: T_{LO}, T_{HI}, T_{Setup} and T_{Hold}

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

- 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 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?

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

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

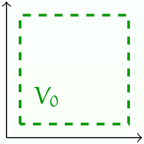
$$\begin{array}{llll} T_{HI} = 24 & T_{LO} = 15 & T_{Setup} = 10 & T_{Hold} = 17 \\ \delta_1^- = 7 & \delta_1^+ = 7 & \delta_2^- = 5 & \delta_2^+ = 6 \\ \delta_3^- = 8 & \delta_3^+ = 10 & \delta_4^- = 3 & \delta_4^+ = 7 \end{array}$$

  - 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

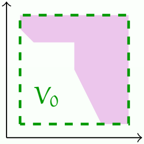
Problems

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



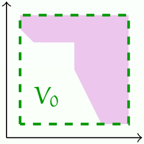
Problems

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

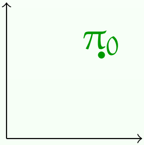


Problems

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

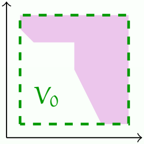


- The inverse problem
  - "Given a reference parameter valuation π_0, find other valuations around π_0 of same behavior"

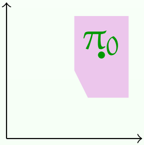


Problems

- 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

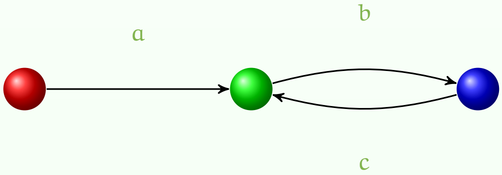
Outline

Timed Automaton

- Finite state automaton (sets of locations)

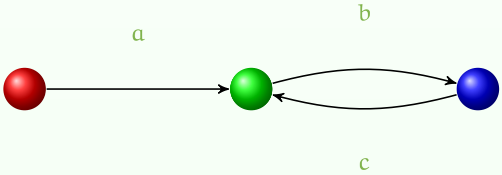
Timed Automaton

- Finite state automaton (sets of locations and actions)



Timed Automaton

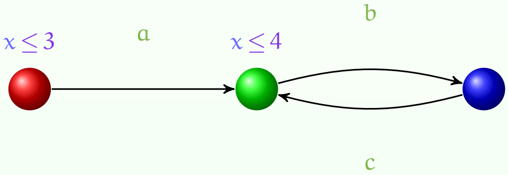
- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate [Alur and Dill, 1994])



Timed Automaton

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

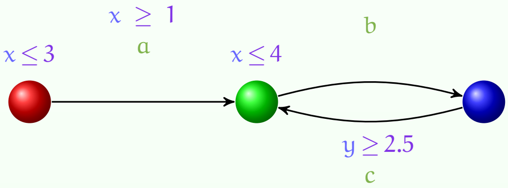
- Features
  - Location invariant: property to be verified to stay at a location



Timed Automaton

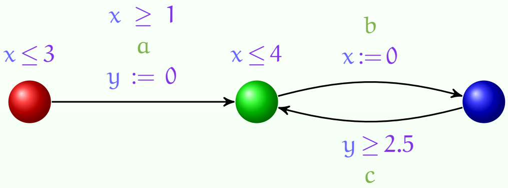
- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate [Alur and Dill, 1994])

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



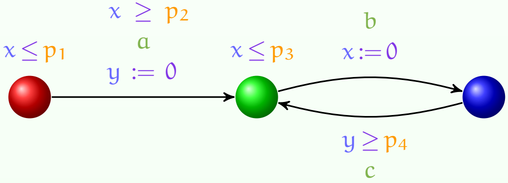
Timed Automaton

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



Parametric Timed Automaton (PTA)

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



Semantics of Parametric Timed Automata

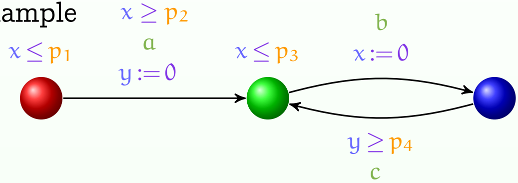
- State of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over X and P

Semantics of Parametric Timed Automata

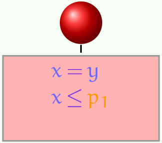
- State of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over X and P
- Run: alternating sequence of states and actions

Semantics of Parametric Timed Automata

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

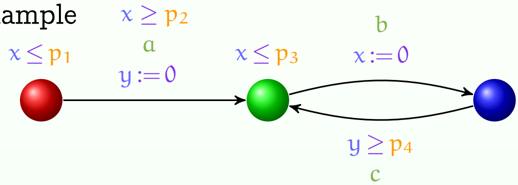


  - Possible run for this PTA

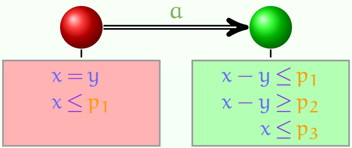


Semantics of Parametric Timed Automata

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

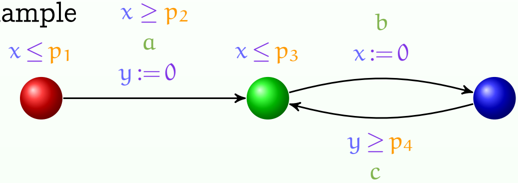


  - Possible run for this PTA

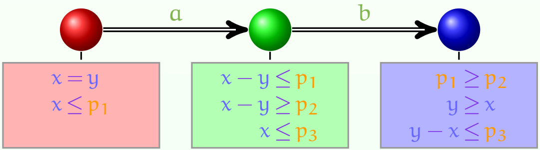


Semantics of Parametric Timed Automata

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

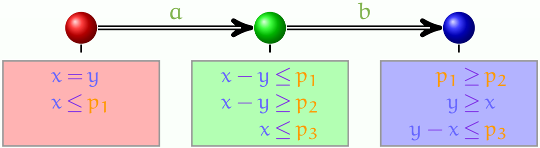


  - Possible run for this PTA



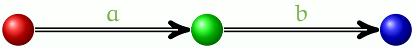
Good and Bad Traces

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



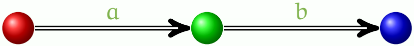
Good and Bad Traces

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

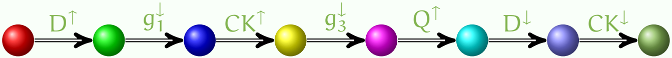


Good and Bad Traces

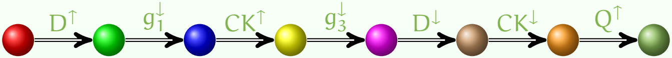
- Trace over a PTA: time-abstract run
  - Finite alternating sequence of locations and actions



- 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

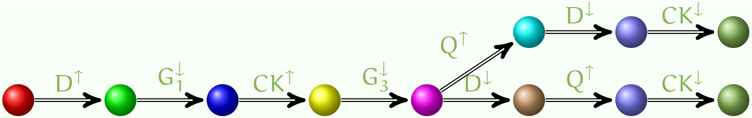


Notation

- 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

- Trace set: set of all traces of a PTA

  - Example: trace set for the flip-flop instantiated with π_0



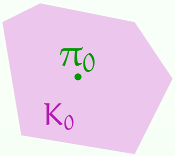
Outline

The Inverse Problem

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

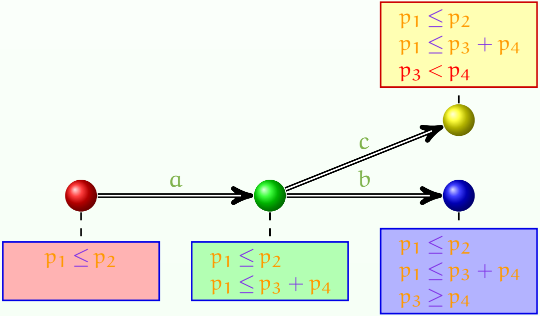
The Inverse Problem

- 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



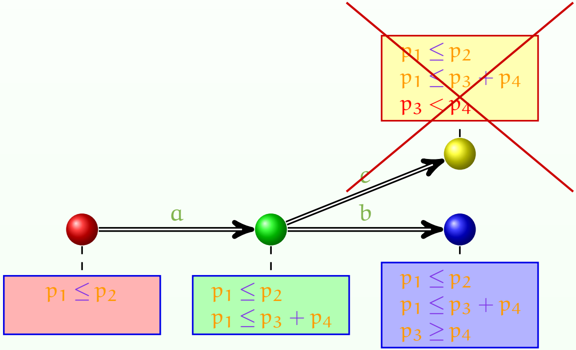
The Inverse Method *IM*: General Idea

- Our idea [André et al., 2009]
  - CEGAR-like approach
  - Instead of negating bad states, we remove π_0-incompatible states



The Inverse Method *IM*: General Idea

- Our idea [André et al., 2009]
  - CEGAR-like approach
  - Instead of negating bad states, we remove π_0-incompatible states



The Inverse Method *IM*: Simplified Algorithm

Start with $K_0 = \texttt{true}$

REPEAT

1. Compute a set S of new reachable states under K_0
2. Project the constraints onto the parameters
3. Refine K_0 by removing π_0-incompatible states 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 all states are π_0-compatible in S

UNTIL all new states computed in S are **equal** to previous states

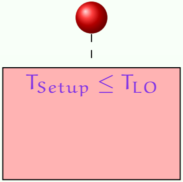
RETURN the **intersection** of the projection onto the parameters of all reachable states

Application to the Flip-Flop Circuit

π_0 :

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

$K_0 = \texttt{true}$

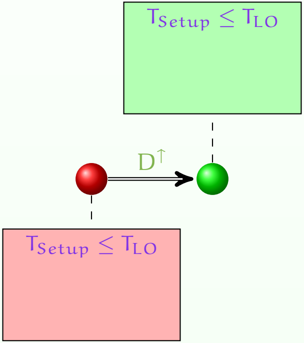


Application to the Flip-Flop Circuit

π_0 :

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

$K_0 = \texttt{true}$

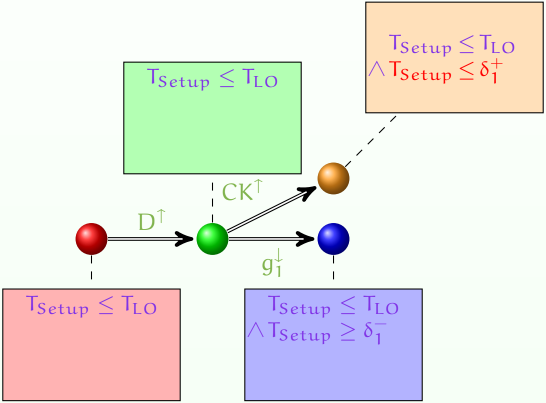


Application to the Flip-Flop Circuit

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}	$= 17$

$K_0 = \mathtt{true}$



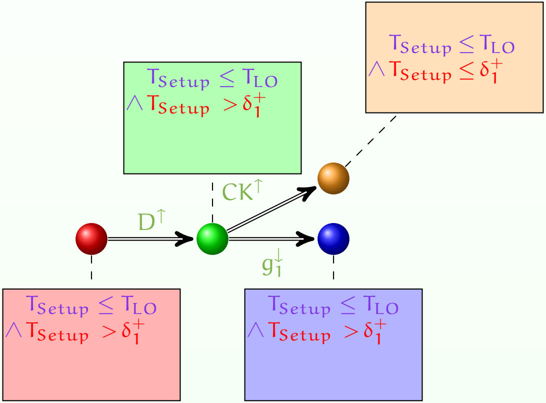
Application to the Flip-Flop Circuit

π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$



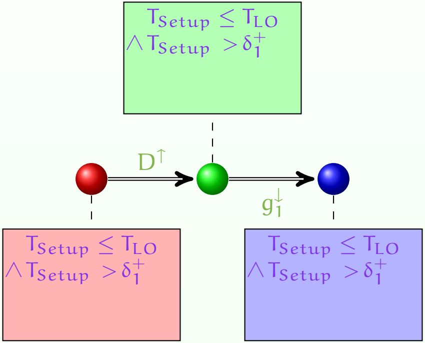
Application to the Flip-Flop Circuit

π_0 :

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}	$= 17$

$K_0 =$

$T_{Setup} > \delta_1^+$



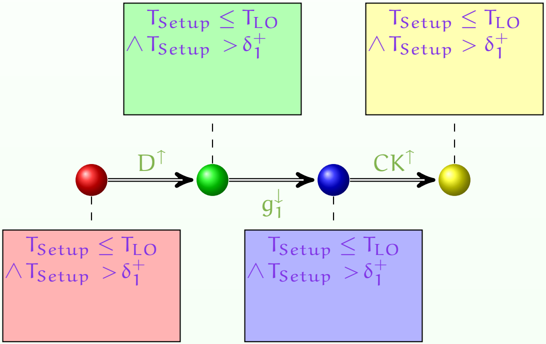
Application to the Flip-Flop Circuit

π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$



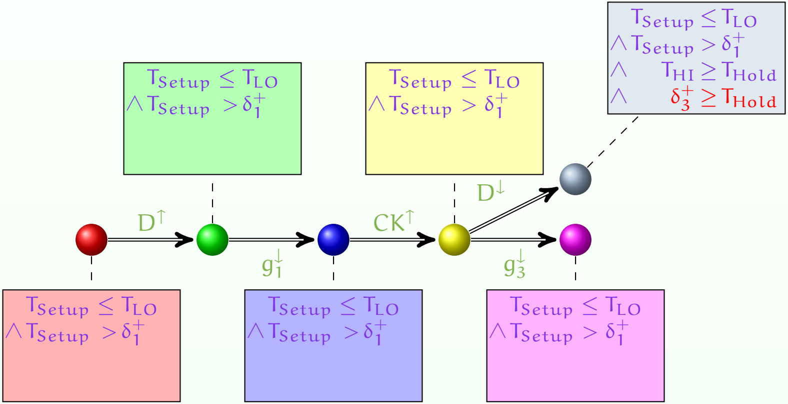
Application to the Flip-Flop Circuit

π_0 :

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

$K_0 =$

$T_{Setup} > \delta_1^+$



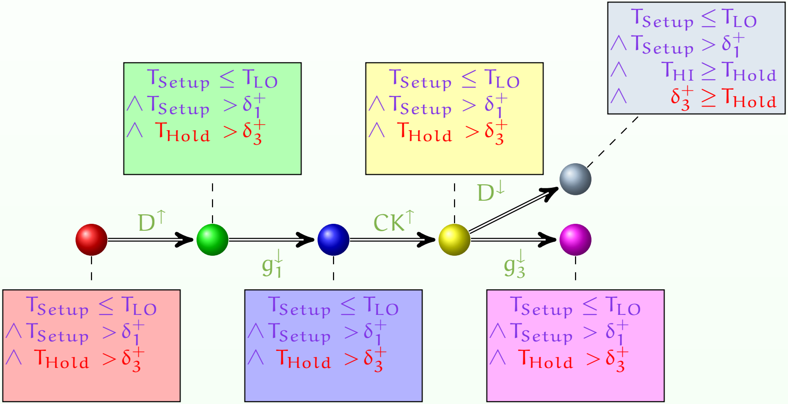
Application to the Flip-Flop Circuit

π_0 :

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

$K_0 =$

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

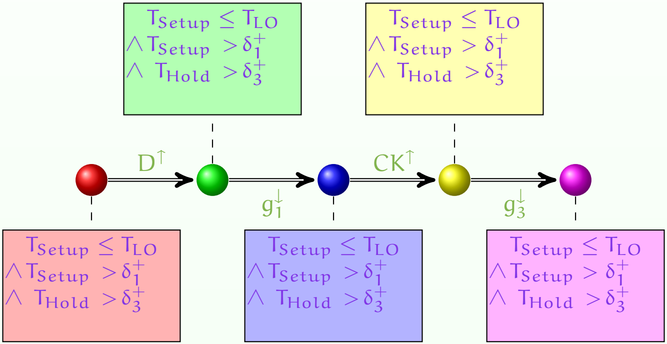


Application to the Flip-Flop Circuit

π_0 :

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

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

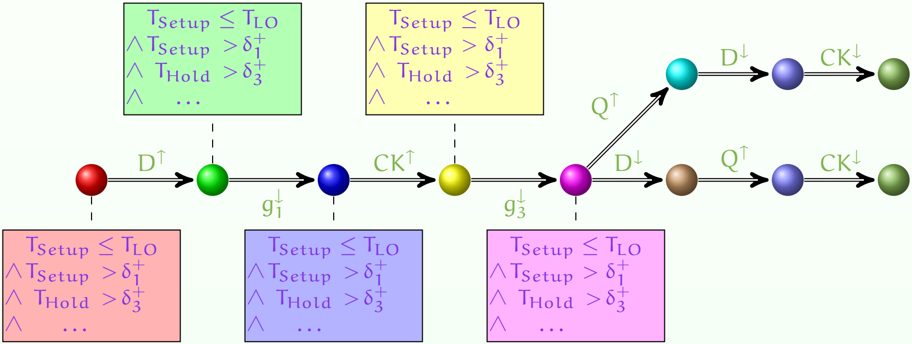


Application to the Flip-Flop Circuit

π_0 :

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

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



Summary of *IM* (1/2)

- Advantages
  - Useful to optimize timing delays in concurrent systems
  - Guarantees the preservation of LTL properties
  - Gives a criterion of robustness to the system
  - Independent of the property one wants to check
  - Efficient: allows to handle dozens of parameters

Summary of *IM* (2/2)

- Termination
  - Parameter synthesis undecidable in general for PTAs
  - Sufficient condition for the termination of *IM* for subclasses of PTA
  - Does not terminate in the general case

- 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
  - There are good points which correspond to a different behavior from π_0
    - For a given property φ, there may be different trace sets satisfying φ

Outline

Beyond the Inverse Method

- *IM* guarantees the equality of trace sets
  - Can be seen as too strong in practice
  - One is often interested in the (non-)reachability of certain states only

- Key points of the algorithm
  - Iterative negation of π_0-incompatible inequalities: prevents behaviors absent from $\mathcal{A}[\pi_0]$
  - State equality in the fixpoint condition: guarantees the same size for all traces
  - Final intersection of the constraints associated to all reachable states: guarantees that all behaviors in $\mathcal{A}[\pi_0]$ are available in $\mathcal{A}[\pi']$, for $\pi' \models IM(\mathcal{A}, \pi_0)$

Beyond the Inverse Method

- *IM* guarantees the equality of trace sets
  - Can be seen as too strong in practice
  - One is often interested in the (non-)reachability of certain states only

- Key points of the algorithm
  - Iterative negation of π_0-incompatible inequalities: prevents behaviors absent from $\mathcal{A}[\pi_0]$
    - Essential for safety
  - State equality in the fixpoint condition: guarantees the same size for all traces

  - Final intersection of the constraints associated to all reachable states: guarantees that all behaviors in $\mathcal{A}[\pi_0]$ are available in $\mathcal{A}[\pi']$, for $\pi' \models IM(\mathcal{A}, \pi_0)$

Beyond the Inverse Method

- *IM* guarantees the equality of trace sets
  - Can be seen as too strong in practice
  - One is often interested in the (non-)reachability of certain states only

- Key points of the algorithm
  - Iterative negation of π_0-incompatible inequalities: prevents behaviors absent from $\mathcal{A}[\pi_0]$
    - Essential for safety
  - State equality in the fixpoint condition: guarantees the same size for all traces
    - Non-essential for safety
  - Final intersection of the constraints associated to all reachable states: guarantees that all behaviors in $\mathcal{A}[\pi_0]$ are available in $\mathcal{A}[\pi']$, for $\pi' \models IM(\mathcal{A}, \pi_0)$

Beyond the Inverse Method

- *IM* guarantees the equality of trace sets
  - Can be seen as too strong in practice
  - One is often interested in the (non-)reachability of certain states only

- Key points of the algorithm
  - Iterative negation of π_0-incompatible inequalities: prevents behaviors absent from $\mathcal{A}[\pi_0]$
    - Essential for safety
  - State equality in the fixpoint condition: guarantees the same size for all traces
    - Non-essential for safety
  - Final intersection of the constraints associated to all reachable states: guarantees that all behaviors in $\mathcal{A}[\pi_0]$ are available in $\mathcal{A}[\pi']$, for $\pi' \models IM(\mathcal{A}, \pi_0)$
    - Non-essential for safety

Variant with State Inclusion in the Fixpoint (1/2)

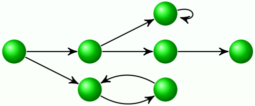
- Fixpoint condition of the standard inverse method IM
  - Termination when each new state is equal to a state encountered before
  - Exact cyclicity of the system
- Variant of the fixpoint: algorithm $IM_{\subseteq}$
  - Termination when each new state is included into a state encountered before
    - State inclusion: equality of locations, inclusion of constraints
  - Non-diverging loops

Variant with State Inclusion in the Fixpoint (2/2)

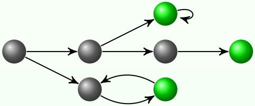
- States are merged more often than *IM*
  - Termination earlier and more often than *IM*
  - State space smaller than *IM*
- Properties
  - Equality of trace sets not preserved
  - Property: the trace sets are equal up to depth n, where n is the number of iterations of $IM_{\subseteq}(\mathcal{A}, \pi_0)$
  - More interested property: non-reachability preserved
    - If a location is not reachable in $\mathcal{A}[\pi_0]$, then it is also not reachable in $\mathcal{A}[\pi]$, for $\pi \models IM_{\subseteq}(\mathcal{A}, \pi_0)$
- Comparison of the constraint
  - Weaker constraint than *IM* (i.e., a larger set of parameter valuations)

Variant with Union of Constraints (1/2)

- Constraint returned by *IM*
  - Return the intersection of the constraints on the parameters associated to all the reachable states



- Variant of the returned constraint: algorithm $IM^{\cup}$
  - Return the union of the constraints on the parameters associated to some of the reachable states
    - Last state of each run

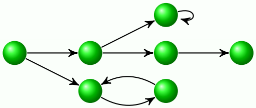


Variant with Union of Constraints (2/2)

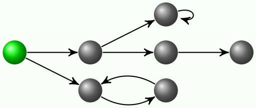
- Same termination and memory consumption than IM
- Properties
  - Equality of trace sets not preserved
  - The trace set of $\mathcal{A}[\pi]$ is included into the trace set of $\mathcal{A}[\pi_0]$, for $\pi \models IM_{\subseteq}(\mathcal{A}, \pi_0)$
  - Corollary: non-reachability preserved
  - Furthermore: At least one trace of $\mathcal{A}[\pi_0]$ is present in $\mathcal{A}[\pi]$
- Comparison of the constraint
  - Weaker than IM
  - Incomparable with $IM_{\subseteq}$

Variant with Simple Return (1/2)

- Constraint returned by *IM*
  - Return the intersection of the constraints on the parameters associated to all the reachable states



- Variant of the returned constraint: algorithm IM^{K}
  - Return the constraint associated to the first state only



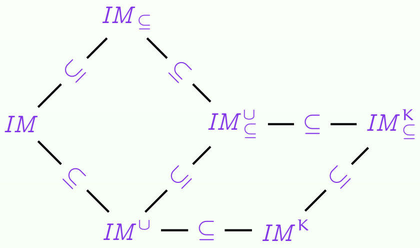
Variant with Simple Return (2/2)

- Same termination and memory consumption than IM
- Properties
  - Equality of trace sets not preserved
  - Only non-reachability is preserved
- Comparison of the constraint
  - Weaker than IM and $IM^{\cup}$
  - Incomparable with $IM_{\subseteq}$

Comparison of the Constraints

- Combined variants
  - One can combine the fixpoint variant ($IM_{\subseteq}$) with the two return variants ($IM^{\cup}$ and IM^{K})
    - $\rightsquigarrow$ $IM^{\cup}_{\subseteq}$ and $IM^{K}_{\subseteq}$ respectively

- Comparison of the constraints output



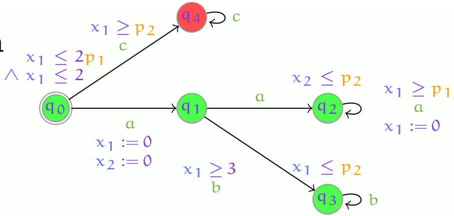
  - All variants improve the size of the set of parameter valuations

Comparison of the Constraints: Example

- A toy PTA for comparison

π_0

$p_1 = 1$
$\wedge\ p_2 = 4$

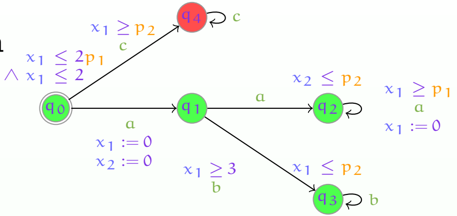


Comparison of the Constraints: Example

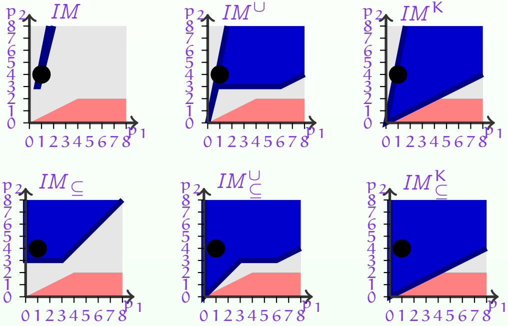
- A toy PTA for comparison

π_0

$$p_1 = 1 \wedge p_2 = 4$$



- Comparison of the constraints output



Comparison of the Properties

Property	IM	$IM_{\subseteq}$	$IM^{\cup}$	IM^{K}	$IM^{\cup}_{\subseteq}$	$IM^{K}_{\subseteq}$
Equality of trace sets	√	×	×	×	×	×
Equality of trace sets up to n	√	√	×	×	×	×
Inclusion into the trace set of $\mathcal{A}[\pi_0]$	√	×	√	√	×	×
Preservation of at least one trace	√	×	√	×	×	×
Equality of location sets	√	√	×	×	×	×
Convex output	√	√	×	√	×	√
Preservation of non-reachability	√	√	√	√	√	√

- Most interesting variants
  - IM for the equality of trace sets
  - $IM^{\cup}$ for the preservation of at least one maximal trace
  - $IM^{K}_{\subseteq}$ for the sole preservation of non-reachability

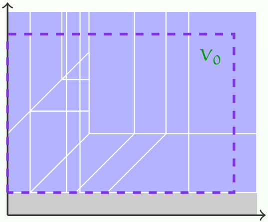
Outline

Implementation

- IMITATOR II [André, 2010]
  - IMITATOR: "Inverse Method for Inferring Time AbstracT BehaviOR"
  - 10 000 lines of code
  - Written in OCaml, using the PPL library
- Available on the Web
  - http://www.lsv.ens-cachan.fr/Software/imitator/

Experiments: Method

- In order to evaluate the size of the constraints, we use the behavioral cartography algorithm [André and Fribourg, 2010]
  - Coverage of a rectangular parameter domain V_0 with tiles
    - Tile: constraint output by *IM*
  - Full coverage of V_0 under certain conditions



- The less tiles for a given V_0, the larger the constraints are

Experiments: Comparison

Example			Tiles						Time (s)					
Name	$\lvert P \rvert$	$\lvert V_0 \rvert$	IM	$IM^{\cup}$	IM^{K}	$IM_{\subseteq}$	$IM^{\cup}_{\subseteq}$	$IM^{K}_{\subseteq}$	IM	$IM^{\cup}$	IM^{K}	$IM_{\subseteq}$	$IM^{\cup}_{\subseteq}$	$IM^{K}_{\subseteq}$
Toy PTA	2	72	14	10	10	7	5	5	0.101	0.079	0.073	0.036	0.028	0.026
Flip-flop	2	644	8	7	7	8	7	7	0.823	0.855	0.696	0.831	0.848	0.699
AND–OR	5	151 200	16	14	16	14	14	14	274	7154	105	199	551	68.4
Latch	4	73 062	5	3	3	5	3	3	16.2	25.2	9.2	15.9	25	9.1
CSMA/CD	3	2 000	139	57	57	139	57	57	112	276	76.0	46.7	88.0	22.6
SPSMALL	2	3 082	272	78	77	272	78	77	894	405	342	894	406	340

- Size of the constraint
  - All experiments conform to the theory
  - In particular, $IM^{K}_{\subseteq}$ outputs the largest constraints
- Computation time
  - $IM^{\cup}$ is sometime slower than IM although it implies less tiles
    - Comes from the non-efficient implementation of the disjunction
    - Subject of future work

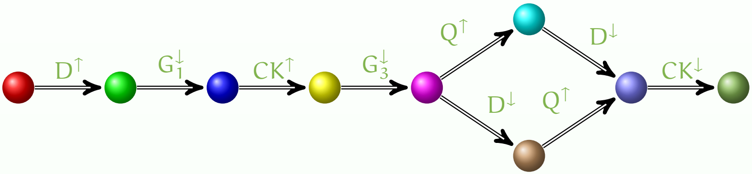
Outline

Summary

- Toolbox of algorithms based on the inverse method *IM* for the synthesis of timing parameters
  - Relaxation of the strong criterion of trace set equality
    - Preservation of non-reachability
      $\rightsquigarrow$ Preservation of safety properties expressed in LTL
  - List of properties satisfied by some algorithms
    - Preservation of at least one trace
    - Inclusion into the original trace set
    - Equality of location sets
  - Advantages over *IM*
    - Better and faster termination
    - Larger sets of parameter valuations

Future Work

- Consider partial orders
  - Consequence: state space reduction



- Extend the variants of *IM* to probablistic systems
  - Study the properties preserved by the algorithms

- Extend the inverse method to hybrid automata
  - Allow to consider continuous variables driven by differential equations

References I

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., 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.

Alur, R. and Dill, D. (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 *INFINITY'10*, volume 39 of *EPTCS*, pages 91–99.

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

References II

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

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

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.