Enhancing the Inverse Method with State Merging

É. André* L. Fribourg** R. Soulat**

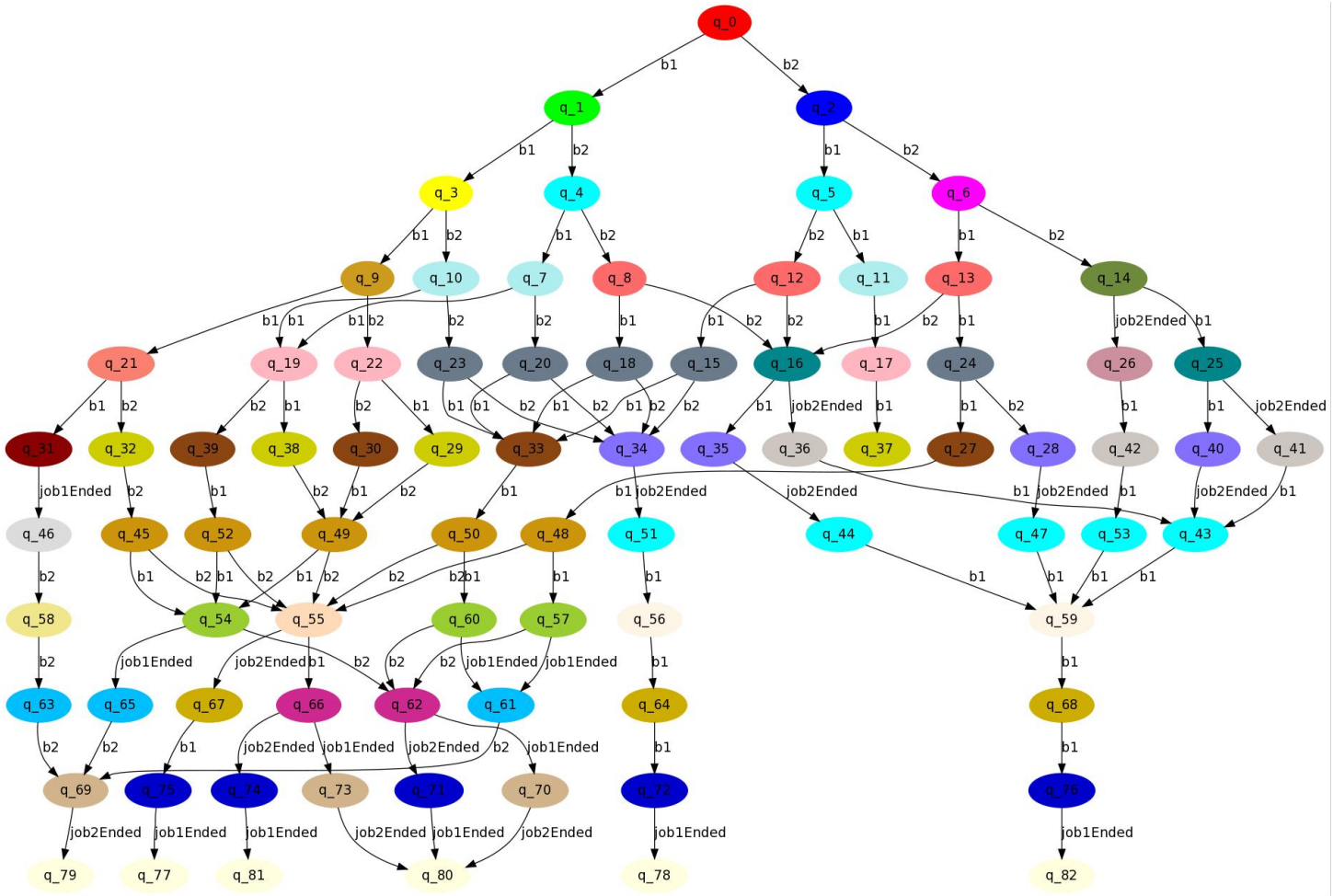
* Laboratoire d'Informatique de Paris Nord, Université Paris 13, France
** Laboratoire Spécification et Vérification, ENS Cachan, France

NFM 2012, Norfolk, VA

April, 5, 2012

Motivations

- Reduce the state explosion during Inverse Method (*IM*) analysis



- *IM* to generate a larger constraint

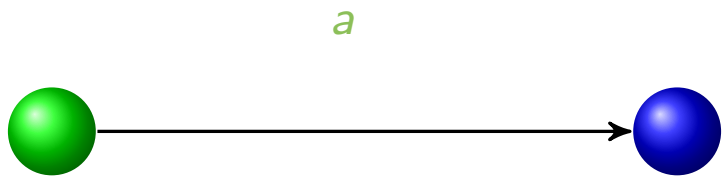
Outline

Timed Automaton

- Finite state automaton (sets of locations)

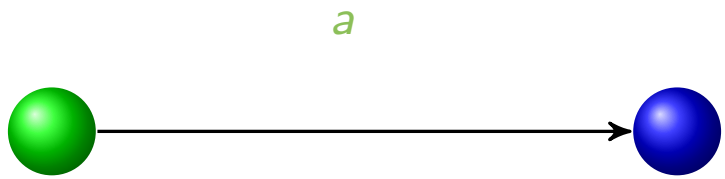
Timed Automaton

- 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

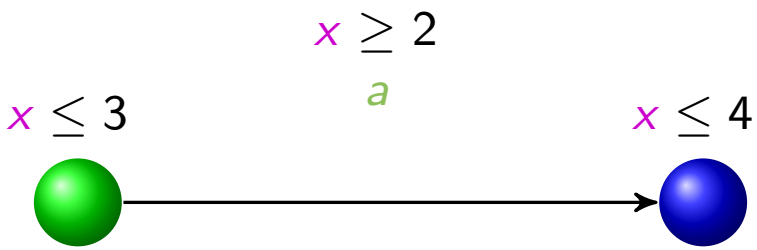
- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate)

- 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)

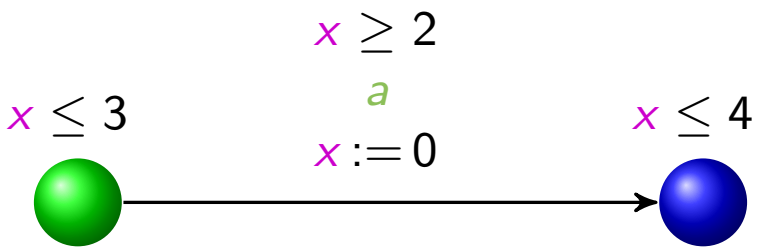
- Features
  - Location invariant : property to be verified by the clocks to stay at a location
  - Transition guard : property to be verified by the clocks to enable a transition



Timed Automaton

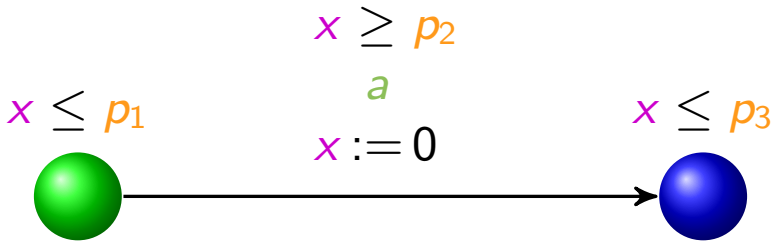
- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate)

- 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)

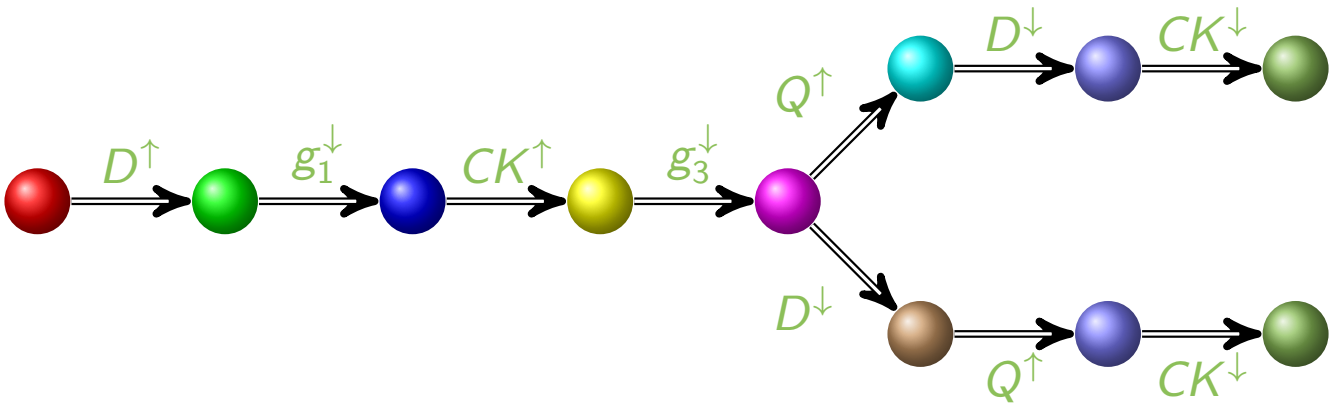
- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate)
  - A set P of parameters (i.e., unknown constants), used in guards and invariants
- Features
  - Location invariant : property to be verified 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

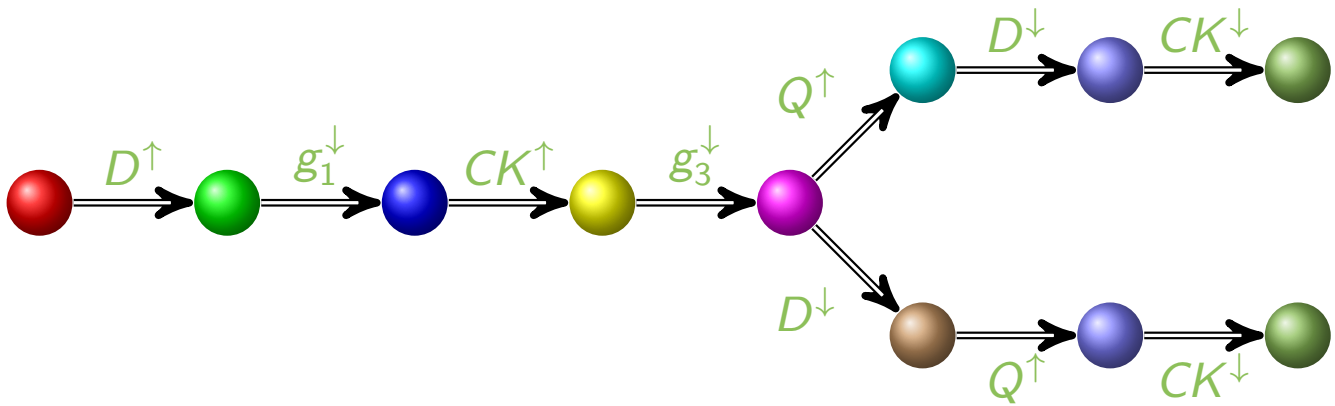
- Symbolic state of a PTA : couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over the parameters and clocks

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



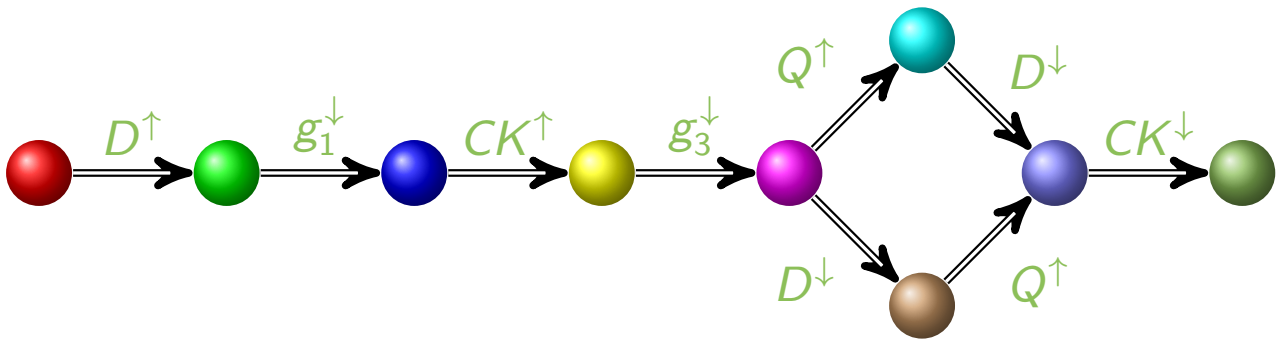
Mergeable

- Mergeable Let $s = (q, C)$ and $s' = (q', C')$ be two states. s and s' are mergeable iff :
  - $q = q'$
  - $C \cup C'$ is convex



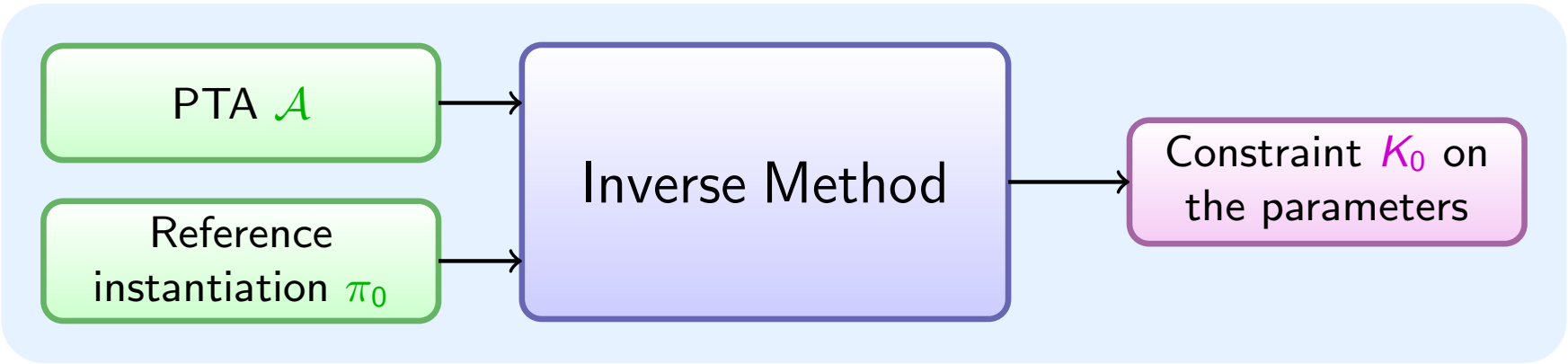
Mergeable

- Mergeable Let $s = (q, C)$ and $s' = (q', C')$ be two states. s and s' are mergeable iff :
  - $q = q'$
  - $C \cup C'$ is convex



- Merged State $s'' = (q, C \cup C')$

Inputs and Outputs (1/2)



Inputs and Outputs (2/2)

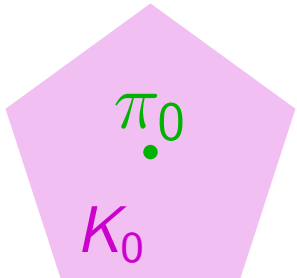
- Input
  - A PTA $\mathcal{A}$
  - A reference instantiation π_0 of all the parameters of $\mathcal{A}$
    - ⋆ Exemplifying a good behavior
      (all traces under π_0 correspond to good behaviors)

π_0

Inputs and Outputs (2/2)

- Input
  - A PTA $\mathcal{A}$
  - A reference instantiation π_0 of all the parameters of $\mathcal{A}$
    - ⋆ Exemplifying a good behavior
      (all traces under π_0 correspond to good behaviors)

- Output : generalization
  - A constraint K_0 on the parameters such that
    - ⋆ $\pi_0 \models K_0$
    - ⋆ For all instantiation $\pi \models K_0$, the set of traces under π is the same as the set of traces under π_0



The General Idea of Our Method

Start with $K_0 = True$

1. Compute the set S of reachable symbolic 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
3. Go to (1)

Until fix point (no more π_0-incompatible states in S)

Theoretical results :

- Smaller set of symbolic states
- Output constraint is larger or equal
  - Let s, s' be two mergeable states
  - s π_0-compatible, s' π_0-incompatible
  - $s \cup s'$ is π_0-compatible
- Let K_0 be the constraint output by IM_{merge}
  - $\pi_0 \models K_0$
  - For every $\pi \models K_0$, the set of traces of $\mathcal{A}$ under π is the same as the set of traces of $\mathcal{A}$ under π_0

PTA	X	P	IM			IM_{merge}			$K_0 \subseteq K_{0\,merge}$
			t	States	M	t	States	M	
AndOr	4	12	0.112	16	1,262	0.101	13	1,187	$=$
Flip-Flop	5	12	0.183	14	1,692	0.227	14	1,762	$=$
Latch	8	13	1.18	18	3,686	0.621	12	2,662	$\subsetneq$
BRP	7	6	4.29	428	25,483	7.015	426	25,845	$=$
WLAN	2	8	220.157	7,038	733,044	286.141	6,020	1,408,702	$=$
$SPSMALL_1$	10	26	1.578	31	5,098	1.642	31	5,442	$=$
$SPSMALL_2$	28	62	-	-	overflow	593	397	180,888	-
SIMOP	8	7	18.959	1,108	43,333	5.179	239	14,371	$\subsetneq$
CSMA/CD	3	3	0.801	240	6,580	0.947	240	7,049	$=$
Jobshop	3	8	1.865	253	10,658	1.147	118	5,221	$\subsetneq$
Mutex 3	3	2	0.802	307	14,598	0.671	241	11,934	$=$
Mutex 4	4	2	22.373	4,769	373,900	22.03	3,287	260,962	$=$

TABLE: Comparison between IM and IM_{merge}. Results obtained with IMITATOR 2

- Small overcost when no or little merging is found
- Allows to perform case studies that were too big for IM
- Can find larger constraints, hence better robustness results

Conclusion and Future Work

- Conclusion
  - Larger constraint generated
  - Small overcost in worst case studies
  - Improvement in both memory usage and computational time for real life case studies
- Ongoing and future work
  - Ongoing : Application to a real industrial case studies with industrial partners
  - Future work : Improve the merging condition