A Generalisation Method for Parametric Timed Automata

Étienne André
Thomas Chatain
Emmanuelle Encrenaz
Laurent Fribourg

Laboratoire Spécification et Vérification
LSV, CNRS & ENS de Cachan

Context : Real-Time Concurrent Systems

- Verification of safety property : ensure the absence of any bad behaviour (reachability property)
- A well-known method : CEGAR (Counter-Example Guided Abstraction Refinement [Clarke & Lu 2000])
  - They use (repeatedly) an example of bad behaviour in order to refine the model of the system
- We present here a generalisation method
  - We use a given example of good behaviour in order to generalise the model of the system

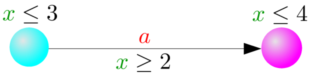
Outline

Outline

Timed Automata

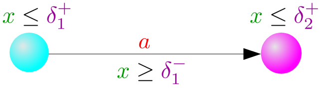
- Finite state automata (sets of locations and labelled transitions) augmented with
  - A set X of clocks evolving linearly at the same rate

- Operations
  - Transition guard : property to be verified by the clocks to enable a transition
  - Location invariant : property to be verified by the clocks to stay at a location
  - Clock reset : clocks can be set to 0 at each transition



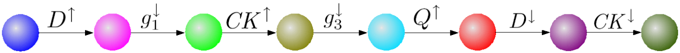
Parametric Timed Automata (PTA)

- Finite state automata (sets of locations and labelled transitions) augmented with
  - A set X of clocks evolving linearly at the same rate
  - A set P of parameters used in guards and invariants
- Operations
  - Transition guard : property to be verified by the clocks and the parameters to enable a transition
  - Location invariant : property to be verified by the clocks and the parameters to stay at a location
  - Clock reset : clocks can be set to 0 at each transition



States and Traces

- State
  - Timed automaton :
    - a concrete state is a pair (q, v), where q is a location and v a clock valuation
    - a symbolic state is a pair (q, C), where q is a location and C a constraint (conjunction of inequalities) on clocks
  - PTA : a parametric symbolic state is a triple (q, C, D), where :
    - q a location,
    - C a constraint over clocks and parameters,
    - D a constraint over parameters
- Trace (or run) over a PTA : finite alternating sequence of locations and transitions

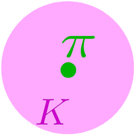


Outline

Our Method

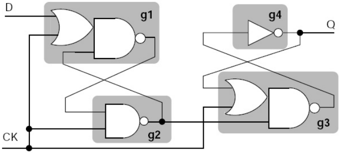
- Input
  - A PTA $\mathcal{A}$ with initial state s_{init}
  - An instantiation π of all the parameters of $\mathcal{A}$
    - ⋆ Exemplifying a good behaviour

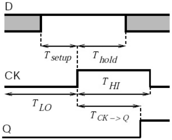
- Output : generalisation
  - A constraint K on the parameters such that
    - ⋆ $\pi \models K$
    - ⋆ For all instantiation $\pi' \models K$, the set of traces under π' is the same as the set of traces under π



An Example of Circuit (1/2)

- Memory circuit [Clarisó & Cortadella 04]





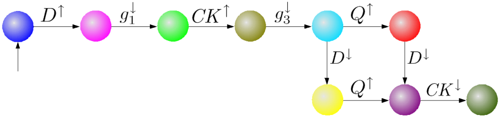
  - 4 elements : g_1, g_2, g_3, g_4
  - 2 input signals (D and CK), 1 output signal (Q)
  - 4 internal signals : g_1, g_2, g_3, g_4 (output of each element)
- Timed parameters of the system
  - Traversal delays of the gates by the electric current
    - ★ Parametric interval ; example for element g_1 : $[\delta_1^-, \delta_1^+]$
  - Stabilisation time of input signal D
    - ★ T_{Setup}, T_{Hold}
  - CK low and high durations
    - ★ T_{LO}, T_{HI}

An Example of Circuit (2/2)

- We are given an instantiation of the parameters

$$T_{HI} = 20 \quad T_{LO} = 15 \quad T_{Setup} = 10 \quad T_{Hold} = 15$$
$$\delta_1^+ = 1 \quad \delta_1^- = 1 \quad \delta_3^+ = 6 \quad \delta_3^- = 5$$
$$\delta_2^+ = 10 \quad \delta_2^- = 8 \quad \delta_4^+ = 5 \quad \delta_4^- = 3$$

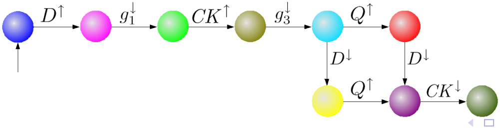
  - This instantiation point guarantees a good behaviour



- Output : a constraint K

$$\begin{array}{rrcl} & T_{Setup} & < & T_{LO} \\ \wedge & \delta_1^+ & < & T_{Setup} \\ \wedge & \delta_3^+ & < & T_{Hold} \\ \wedge & \delta_4^+ + \delta_3^+ & < & T_{HI} \end{array}$$

  - This constraint guarantees the same set of good traces



The Algorithm

Variable	Type	Description	Initially
i	Integer	Current step	$i := 1$
K	Constraint	Output	$K := \top$

DO

if $Post^i_K(s_{init}) = Post^{i+1}_K(s_{init})$ **then return** K **fi**

DO until $Post^i_K(s_{init})$ contains only π-compatible states **:**

Select:

- a π-incompatible state (q, C, D) of $Post^i_K(s_{init})$ ($\pi \not\models D$)
- an inequality J of D such that $\pi \models K \wedge \neg J$

$K := K \wedge \neg J$

OD

$i := i + 1$

OD

Termination and Complexity (Acyclic Case)

Proposition

The algorithm terminates if the set of traces under π contains no cyclic trace (trace passing twice by the same location).

In this case, $Post^* = Post^n$, where n is the number of locations.

- Complexity Analysis
  - Exponential in the number of locations of $\mathcal{A}$
  - Exponential in the number of parameters P
  - Doubly exponential in the number of clocks X

Correctness

Proposition

For all instantiation $\pi' \models K$, the set of traces under π' is the same as the set of traces under π.

The set of traces are time-abstract equivalent.

Implementation

- Script written in Python with calls to HyTech
  - 1500 lines of code
- Some real cases treated
  - SPSMALL : memory circuit (ST-Microelectronics)
  - SIMOP : model of manufacturing system with sensors and controllers communicating through a network
- Some computation times

Name	# PTA	# loc / PTA	# clocks	# param	# iterations	Time
SPSMALL	10	~ 7	11	28	32	20 mn
SIMOP	5	~ 9	5	7	51	2 h

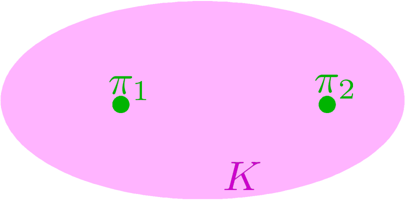
Outline

Conclusion

- The Generalisation Method
  - Modeling of a system with parametric timed automata
  - Starting with an instantiation point of the system, we give a constraint on the parameters guaranteeing the same set of traces
- Advantage
  - Powerful even on fully parameterized big systems
    - ⋆ Can handle dozens of parameters
- Drawback
  - The zone (set of points) generated by the constraint is rather small compared to exhaustive point by point methods

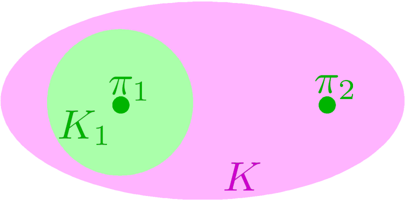
Future Work

- Termination of cyclic case
- Use more than one point as input :
  - Two different points π_1 and π_2



Future Work

- Termination of cyclic case
- Use more than one point as input : either
  - Two different points π_1 and π_2, or
  - One constraint K_1 and one point π_2



Extra Slides

References

- [Clarke & Lu 2000] Counterexample-guided abstraction refinement, CAV 2000
- [Clarisó & Cortadella 2004] Verification of timed circuits with symbolic delays, ASP–DAC 2004

Parametric Timed Automaton

Definition

A *parametric timed automaton* is $\mathcal{A}(K) = (\Sigma, Q, q_{init}, X, P, I, \rightarrow)$, where:

- Σ is a finite set of actions (or "step labels"),
- Q is a finite set of locations (or "control states"),
- q_{init} is the initial location,
- X is a finite set of clocks,
- P is a finite set of parameters partitioned as $P = P^l \biguplus P^u$,
- K is a P-constraint on the set of parameters P,
- I is the invariant, assigning to every $q \in Q$ a conjunction $I_q(X)$ of (X, P)-atoms of the form $x \leq p^u$, for some clock variable $x \in X$ and parameter $p^u \in P^u$, and
- $\rightarrow$ is a step (or "transition") relation consisting of elements of the form (q, g, a, ρ, q')