Projet VALMEM

Mardi 3 novembre 2009

IMITATOR 1, IMITATOR 2 :

État des lieux

Étienne ANDRÉ

Laboratoire Spécification et Vérification

LSV, ENS de Cachan & CNRS, France

Outline

Outline

Inputs and Outputs (1/2)

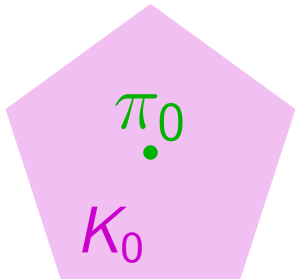
Inputs and Outputs (2/2)

- 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)

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)

The Algorithm

ALGORITHM *InverseMethod*($\mathcal{A}$, π_0)

Inputs $\mathcal{A}$: PTA of initial state s_0
π_0 : Reference valuation of the parameters
Output K_0 : Constraint on the parameters

$i := 0$; $K := True$; $S := \{s_0\}$
DO
  DO UNTIL there are no π_0-incompatible states in S
    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. $\pi_0 \not\models J$)
    $K := K \wedge \neg J$; $S := \bigcup_{j=0}^{i} Post^j_{\mathcal{A}(K)}(\{s_0\})$
  OD
  IF $Post_{\mathcal{A}(K)}(S) = \emptyset$ **THEN RETURN** $K_0 := \bigcap_{(q,C)\in S}(\exists X : C)$
  FI
  $i := i + 1$; $S := S \cup Post_{\mathcal{A}(K)}(S)$
OD

Implementation

- IMITATOR
  - IMITATOR: "Inverse Method for Inferring Time AbstracT BehaviOR"
  - 1500 lines of code
  - 4 man-months of work
  - Program written in Python
  - Calls the parametric model checker HYTECH
    - ★ Tool written in C
    - ★ Used by IMITATOR for the computation of the *Post* operation

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

Case Studies (1/2)

- Abstraction of SPSMALL (memory circuit by ST-Microelectronics)
  - Modelisation with 10 automata, 10 clocks, 22 parameters
  - Initial value (π_0) for $T_{setupwen}$ and T_{setupd}:

$$\begin{aligned} T_{setupwen} &= 48 \\ T_{setupd} &= 108 \end{aligned}$$

- Application of IMITATOR
  - 31 iterations of IMITATOR, 31 reachable states
  - Generated constraint K_0: 21 inequalities
  - After instantiation of all the parameters in K_0 with their value in π_0, except $T_{setupwen}$ and T_{setupd}:

$$\begin{aligned} & 46 < T_{setupwen} < 54 \\ \wedge \quad & 99 < T_{setupd} < 110 \\ \wedge \quad & T_{setupd} < T_{setupwen} + 61 \end{aligned}$$

  - Which allows us to minimize:
    - ★ $T_{setupwen} = 47$
    - ★ $T_{setupd} = 100$ (optimization of 7,4 %)

Case Studies (2/2)

- More case studies
  - SIMOP : model of manufacturing system with sensors and controllers communicating through a network
    - ★ Allow to define zones of good behavior
  - Various protocols of communications

- Computation times of various case studies
  - Experiences conducted on an Intel Quad Core 3 Ghz with 3.2 Gb

Example	# of PTAs	loc. per PTA	# of clocks	# of param.	# of iter.	$\lvert Post^* \rvert$	$\lvert K_0 \rvert$	CPU time
Flip-flop	5	[4, 16]	5	12	8	11	7	2 s
CSMA/CD	3	[3, 8]	3	3	17	218	3	44 s
RCP	5	[6, 11]	6	5	18	154	2	70 s
SPSMALL	10	[3, 8]	10	22	31	31	23	78 min
SIMOP	5	[6, 16]	9	16	51	848	7	419 min

Summary of IMITATOR

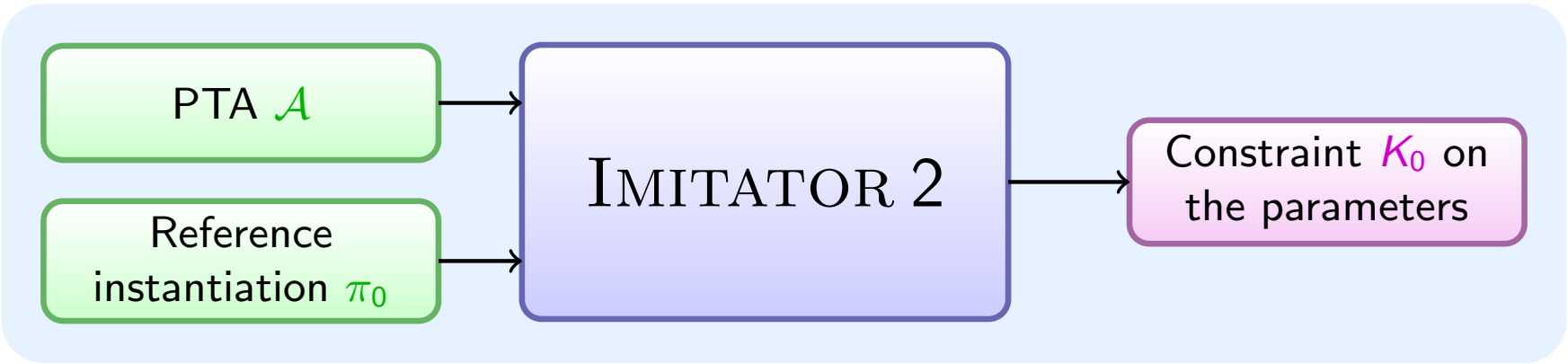
- Inverse method implemented in IMITATOR
  - Modeling of a system with parametric timed automata
  - Starting with an instantiation point π_0 of the system, IMITATOR generates a constraint K_0 on the parameters guaranteeing the same set of traces
- Advantages
  - Sufficient termination conditions
  - Useful to optimize timing bounds of systems
  - Powerful even on fully parameterized big systems
    - ⋆ Can handle dozens of parameters
- Drawbacks
  - The zone (set of points) generated by the constraint is rather small compared to exhaustive point by point methods
  - The generated constraint is not minimal: it is possible to find valuations $\pi \not\models K_0$ s.t. the set of traces under π and π_0 are the same

Future Works

- Improve the constraint K_0
  - Goal: generate a minimal constraint K_0
    - ⋆ The minimal K_0 may exist under a disjunctive form only
  - Use IMITATOR in an incremental way
    - ⋆ Given a constraint K generated by IMITATOR, run IMITATOR again on a new point $\pi \not\models K$
- Increase the scalability
  - Write an *ad hoc* tool instead of using HYTECH
  - Use a library of polyhedra

Outline

Inputs and Outputs



Features

- Must-Features
  - Computation of the traces in both instantiated and parametric analysis
  - Implementation of the current *InverseMethod* algorithm
  - Dynamic computation of the reachable states
    - ★ Allow to treat more automata in parallel
    - ★ Increase speed

- May-Features
  - Optimization of the *InverseMethod* algorithm
    - ★ Do not start from the beginning at each iteration, but simply updates the reachable states
    - ★ Increase speed
  - Generation of a minimal constraint
    - ★ Generation of a constraint in disjunctive form
  - Allow to consider intervals for π_0 instead of a single point

Implementation

- New standalone tool
  - No call to HyTech
  - Use of a standard library for polyhedra (PPL ?)
- Language: OCaml
  - Safety
  - Various facilities to build compilers
  - Interface with external libraries (PPL)
- Goal
  - Analyze bigger parts of the SPSMALL memory
  - Fully automated analysis from the transistor level to the constraint K_0
- Coming soon
  - Work in progress!
  - Release early 2010?