Analyse de SPSMALL avec IMITATOR 2
(suite)

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

1 Imitator II
   - Principle
   - Features
   - Implementation

2 Analysis of the SPSMALL Memory

3 Future Works
Outline

1. Imitator II
   - Principle
   - Features
   - Implementation

2. Analysis of the SPSMALL Memory

3. Future Works
Inputs and Outputs

- PTA $\mathcal{A}$
- Reference instantiation $\pi_0$
- Constraint $K_0$ on the parameters
The General Idea of Our Method

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 $\pi_0$-incompatible state from $S$
   - Select a $\pi_0$-incompatible state $(q, C)$ within $S$ (i.e., $\pi_0 \not| C$)
   - Select a $\pi_0$-incompatible inequality $J$ within $C$ (i.e., $\pi_0 \not| J$)
   - Add $\neg J$ to $K_0$

UNTIL no more $\pi_0$-incompatible state in $S$
Features

- **Improved Features**
  - **Optimization** of the *InverseMethod* algorithm
    - Do not start from the beginning at each iteration, but simply update the reachable states
    - Increase speed
  - **Dynamic computation** of the reachable states
    - Allow to treat more automata in parallel
    - Increase speed

- **New Features**
  - Computation of the *traces* in both instantiated and parametric analysis
  - Implementation of a *cartography algorithm* (work in progress)
Implementation

- Standalone tool
  - About 8000 lines of code
  - Use of a standard library for polyhedra

- Language: OCaml
  - Safety
  - Various facilities to build compilers
  - Interface with external libraries (Apron, PPL)

- New improvements
  - Use of PPL instead of Apron
  - Various optimizations
Outline

1. IMITATOR II
   - Principle
   - Features
   - Implementation

2. Analysis of the SPSMALL Memory

3. Future Works
Abstract Model

- **Model considered in the *Blueberry* project**
  - Model built manually
  - File spsmall_blueb_lsv

- Abstraction of the memory for the write operation
  - 10 automata, 10 clocks, 26 parameters, 450 lines of code

- Constraint generated by *IMITATOR II* in 1 second (31 states, 30 transitions)
  - To be compared with 1 hour and 20 minutes using *IMITATOR*

- After projection onto $T_{\text{setup}}^D$ and $T_{\text{setup}}^{Wen}$:

  $110 \geq T_{\text{setup}}^D$
  $T_{\text{setup}}^{Wen} + 61 > T_{\text{setup}}^D$
  $54 > T_{\text{setup}}^{Wen}$
  $T_{\text{setup}} > 46$
  $T_{\text{setup}}^D > 99$
Generated Model

- **Generated model**
  - File `lsv`
  - Automatically generated by LIP6
  - 28 automata, 28 clocks, 62 parameters, 32 discrete variables, 1500 lines of code

- **Constraint generated for some parameters**
  - Instantiation of all parameters except 6, 8, 10 or 12 (setup, latch delays, high and low clock cycles)

| $|P|$ | Iter. | $|K_0|$ | States | Trans. | Time  |
|-----|-------|--------|--------|--------|-------|
| 6   | 158   | 11     | 213    | 294    | 1008  |
| 8   | 158   | 15     | 213    | 294    | 1091  |
| 10  | 158   | 19     | 213    | 294    | 1146  |
| 12  | 158   | 20     | 213    | 294    | 1228  |

- **With 62 parameters: fails after 110 iterations (out of memory)**
  - Experimental technique to reach iteration 118 (by starting again with the constraint output at iteration 110)
Full SPSMALL 1*2

- **Full SPSMALL memory 1*2**
  - File `sp_1x2_md_no`
  - Automatically generated by LIP6
  - 101 automata, 101 clocks, 200 parameters, 130 discrete variables, more than 6000 lines of code

- **To do!**
Future Works

- Improve the generated constraint
  - Use an extension of Imitator II allowing to get a maximal constraint

- Improve Imitator II
  - Experimental techniques used by Romain Soulat (to be implemented)

- In the VALMEM project
  - Analyze bigger parts of the SPSMALL memory
  - Fully automated analysis from the transistor level to the constraint $K_0$