Analyse de SPSMALL avec IMITATOR 2

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

IMITATOR II
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., $\pi_0 \not\models C$)
   - Select a $\pi_0$-incompatible inequality $J$ within $C$ (i.e., $\pi_0 \not\models J$)
   - Add $\neg J$ to $K_0$

UNTIL no more $\pi_0$-incompatible state in $S$
The Algorithm

**Algorithm 1: InverseMethod(\(A, \pi_0\))**

- **input**: A PTA \(A\) of initial state \(s_0\)
- **input**: Reference point \(\pi_0\) of the parameters
- **output**: Constraint \(K_0\) on the parameters

1. \(i \leftarrow 0\); \(K_0 \leftarrow True\); \(S \leftarrow \{s_0\}\)
2. **while** **True** **do**
   3. **while** there are \(\pi_0\)-incompatible states in \(S\) **do**
      4. Select a \(\pi_0\)-incompatible state \((q, C)\) of \(S\) (i.e., s.t. \(\pi_0 \not\models C\));
      5. Select a \(\pi_0\)-incompatible \(J\) in \(C\) (i.e., s.t. \(\pi \not\models J\));
      6. \(K_0 \leftarrow K_0 \land \neg J\);
      7. \(S \leftarrow \bigcup_{j=0}^{i} \text{Post}_{A(K_0)}^{j}(\{s_0\})\);
3. **if** \(\text{Post}_{A(K_0)}(S) = \emptyset\) **then return** \(K_0 \leftarrow \bigcap_{(q, C) \in S} (\exists X : C)\)
4. \(i \leftarrow i + 1\);
5. \(S \leftarrow S \cup \text{Post}_{A(K_0)}(S)\);  // \(S = \bigcup_{j=0}^{i} \text{Post}_{A(K_0)}^{j}(\{s_0\})\)
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

- New standalone tool
  - About 8000 lines of code
  - No call to HyTech
  - Use of a standard library for polyhedra (Apron)

- Language: OCaml
  - Safety
  - Various facilities to build compilers
  - Interface with external libraries (Apron, PPL)
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 $T^D_{\text{setup}}$ and $T^{\text{Wen}}_{\text{setup}}$:
  - $110 \geq T^D_{\text{setup}}$
  - $T^{\text{Wen}}_{\text{setup}} + 61 > T^D_{\text{setup}}$
  - $54 > T^{\text{Wen}}_{\text{setup}}$
  - $T^{\text{Wen}}_{\text{setup}} > 46$
  - $T^D_{\text{setup}} > 99$
Generated Model

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

- File successfully parsed and treated by Imitator II

- Inverse method fails after about 20 iterations (out of memory)
  - Too many states?
  - Bad representation of the constraints?
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

- File successfully parsed by Imitator II

- Conversion to the abstract structure fails
  - Most probably a bad representation of the constraints (matrices)
  - Solution: use polyhedra instead of matrices
Future Works

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

- Improve Imitator II
  - More efficient representation of the polyhedra (PPL?)

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