CSP-based inference of function block finite-state models from execution traces

Daniil Chivilikhin, Vladimir Ulyantsev, Anatoly Shalyto, Valeriy Vyatkin

INDIN 2017, Emden, Germany
25 July 2017
Program synthesis

- Derive implementation from examples/specification
  - From seminal work [A. Church, 1963]

- Motivation
  - Fundamental in computer science
  - Automation of software engineering
    - Reverse engineering
Reverse engineering of software

- Rights limitations
- Changing standards
- ....

- Understanding
- Optimization
- Verification
- ....
Black-box approach

What’s in the box?
Target language: IEC 61499 function blocks
Test-based reverse engineering

Preparation

Simulation

Tests gen → Tests

Tests

Execution traces

Model inference

Model
Scenarios

Input Event

Output Event

Output vars values

$S_1 = \langle A[x_1 = 1, x_2 = 0], B[z_1 = 1, z_2 = 0]\rangle$;

$\langle A[x_1 = 1, x_2 = 1], B[z_1 = 1, z_2 = 1]\rangle$;

$\langle A[x_1 = 0, x_2 = 0], B[z_1 = 0, z_2 = 1]\rangle$. 

Input vars values
Basic function block model

Boolean input/output vars

\[
\begin{align*}
  z_1 &: 0 \rightarrow 1, 1 \rightarrow 0 \\
  z_2 &: 0 \rightarrow 0, 1 \rightarrow 1
\end{align*}
\]

\[
\begin{align*}
  z_1 &\equiv B \\
  z_2 &\equiv B
\end{align*}
\]

\[
\begin{align*}
  A [x_1 &\& \neg x_2] &\rightarrow z_1 \\
  A [\neg x_1 &\& x_2] &\rightarrow z_2
\end{align*}
\]

\[
\begin{align*}
  A [\neg x_1 &\& \neg x_2] &\rightarrow A [x_1 &\& x_2] \\
  A [\neg x_1 &\& x_2] &\rightarrow A [x_1 &\& x_2]
\end{align*}
\]

\[
\begin{align*}
  \text{IF NOT } z_1 &\text{ THEN} \\
  z_1 &\text{ := TRUE} \\
  \text{ELSE} \\
  z_1 &\text{ := FALSE} \\
  \text{END_IF;}
\end{align*}
\]
Previous/proposed approaches

1. Metaheuristic: [Chivilikhin et al / INDIN’15]
   • Slow
   • Approximate

2. We propose **CSP-translation** approach
   • Could be faster in practice
   • Exact
Proposed approach: translation to Constraint Satisfaction Problem

Data → Propositional encoding → CSP-solver → Solution reconstruction → Solution

https://srlabs.de/bites/minisat-intro/
Proposed approach scheme

- Traces $S_{\langle\ldots\rangle,\ldots,\langle\ldots\rangle}$
- Number of states $N$
- Translation function $f$
- Traces tree construction
- Constraints $\mathbb{C}$ on variables $\mathbb{V}$ with domains $\mathbb{D}$
- CSP $\langle\mathbb{V},\mathbb{D},\mathbb{C}\rangle$
- CSP solving
- No solution (UNSAT)
- Values of variables $\mathbb{V}$
- Automaton
$S_1 = \langle A[x_1 = 1, x_2 = 0], B[z_1 = 1, z_2 = 0]\rangle$;
\[\langle A[x_1 = 1, x_2 = 1], B[z_1 = 1, z_2 = 1]\rangle;\]
\[\langle A[x_1 = 0, x_2 = 0], B[z_1 = 0, z_2 = 1]\rangle.\]

$S_2 = \langle A[x_1 = 0, x_2 = 1], B[z_1 = 0, z_2 = 1]\rangle$;
\[\langle A[x_1 = 0, x_2 = 0], B[z_1 = 1, z_2 = 1]\rangle;\]
\[\langle A[x_1 = 1, x_2 = 1], B[z_1 = 1, z_2 = 0]\rangle.\]
Variables

- \( c_v \in [1..N] \) for \( v \in V \) – color of traces tree vertex \( v \);
- \( t_{n,e,x} \in [1..N] \) for \( n \in [1..N] \), \( e \in [1..|E^I|] \), \( x \in [1..|\hat{X}|] \) – target state of the transition from state \( n \) labeled with input event \( e \) and guard condition \( x \);
- \( o_n \in [1..|E^O| + 1] \) for \( n \in [1..N] \) – index of the output event in state \( n \) (\(|E^O| + 1\) corresponds to \( \varepsilon \));
- \( d_{n,i}^0 \in \{0, 1\} \) for \( n \in [1..N] \), \( i \in [1..|Z|] \) – value of the \( i \)-th output variable in state \( n \) if its previous value equals zero;
- \( d_{n,i}^1 \in \{0, 1\} \) for \( n \in [1..N] \), \( i \in [1..|Z|] \) – value of the \( i \)-th output variable in state \( n \) if its previous value equals one.
Main constraints (1)

\[ \land \land (t_{c_u}, e^{\text{in}}_{uv}, x_{uv} = c_v) \]

\[ u,v \in V \land uv \in E \]

Tree

Automaton
Main constraints (2)

\[ \bigwedge_{u,v \in V} \bigwedge_{1 \leq i \leq |Z|} (z_{u,i} = 0 \implies d_{c_v,i}^0 = z_{v,i}) \]
Case study: pick-and-place manipulator
Trace generation

All tests with length = 1, 2, 3
#1:  1
#2:  2
...
#12: 1, 1, 1
...
#39: 3, 3, 3
Experimental setup

• Methods
  ✓ MuACO
   • metaheuristic [Chivilikhin et al (2015)]
  ✓ fbCSP
   • Proposed approach
  ✓ fbCSP+BFS
   • fbCSP + BFS-based symm breaking

• State-of-the-art CSP-solvers
  ✓ Opturion CPX (Minizinc Challenge 2015 winner)
  ✓ HaifaCSP (Minizinc Challenge 2016 winner)
Fixed number of states: $N = 10$
Minimal model generation

• Most general pattern in the given data
• Occam’s razor (law of parsimony):
  “Among competing hypotheses, the one with the fewest assumptions should be selected”
Minimal model generation

\begin{align*}
N &:= N + 1 \\
Solve &\quad \text{N} = 1 \\
\text{Satisfiable} &
\end{align*}

\begin{align*}
\text{UNSAT} &
\end{align*}

\begin{align*}
\text{Return solution}
\end{align*}
Minimal model generation: results
Generated model example
Conclusion and Future work

- Proposed fast **exact algorithm** for inferring **minimal-sized** models of basic FBs for logic control
- Available: https://github.com/chivdan/cspgen
- Future/ongoing work
  - Integer/real variables
  - Timers
  - Composite FBs
  - CEGAR for LTL/CTL based inference
Thank you for your attention!

Daniil Chivilikhin, chivdan@rain.ifmo.ru