Reconstruction of Function Block Controllers Based on Test Scenarios and Verification

Daniil Chivilikhin
PhD
ITMO University

Ilya Ivanov
Undergrad student
ITMO University

Anatoly Shalyto
Dr. Sci., professor
ITMO University

Valeriy Vyatkin
Dr. Eng., professor,
Aalto University,
Luleå University of Technology

INDIN ’16, Futuroscope-Poitiers, France, July 19, 2016
IEC 61499 function blocks

Function block interface

Execution Control Chart (ECC)
Motivation

Source code → Legacy application → New source code
Previous work: Test-based FB reconstruction

Reconstruct Function Block Logic without using code

Chivilikhin D. et al. Reconstruction of Function Block Logic using Metaheuristic Algorithm: Initial Explorations / In Proceedings of INDIN'15
Test-based FB reconstruction

1. Human designer
   - Refactoring
   - Instrumented function block
   - Test cases
   - Simulation
   - Execution scenarios

2. Execution scenarios
   - Search-based Inference
Execution scenario

<table>
<thead>
<tr>
<th>$S_0$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ</td>
<td>01011010</td>
<td>101010</td>
<td>CNF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input event | Input variables | Output variables | Output event
REQ       | 01011010       | 101010            | CNF

FB inference with testing and verification
Recording execution scenarios

Original function block

Automated refactoring
Solution representation

\[
\begin{array}{c|cc|c}
 & x_0 & x_1 & x_2 \\
\hline
m_0 & 1 & 0 & 0 \\
\hline
x_0 & 0 & -1 \\
\hline
\Phi_0 & 1 & 1 \\
\hline
\end{array}
\quad
\begin{array}{c|cc|c}
 & x_0 & x_1 & x_2 \\
\hline
m_1 & 0 & 1 & 1 \\
\hline
x_1 & 0 & 0 & -1 \\
\hline
\Phi_1 & 0 & 1 & -1 \\
\hline
1 & 1 & 0 & -1 \\
\hline
1 & 0 & -1 \\
\hline
\end{array}
\quad
\begin{array}{c|c}
 & x_1 \& x_2 \\
\hline
0 & x_0 \\
\hline
0 & \overline{x_1} \& \overline{x_2} \\
\hline
1 & 1 \\
\hline
\end{array}
\]
Inference algorithm (1)

- Parallel MuACO algorithm [Chivilikhin et al, 2014]
Inference algorithm (2)

1. Start with **random** solution
2. **Build** new solutions with **mutation operators**
3. **Evaluate** new solutions with **fitness function**
Issue with previous approach

Black-box inference

How do we ensure sufficient coverage?
Proposal

• Use **Temporal Logic formulas** as input
• We assume that these temporal properties cover the **most important** functionality of the FB
Essence of the approach

Candidate solution

Fitness function

Testing

Verification

Fitness value
Which temporal logic to use?

- Linear temporal logic
- NuSMV is used for formula verification
Closed-loop verification

**Issues**

- We need the model of the plant
- Verification will take a lot of time
  - e.g., verification of PnP properties takes several hundred seconds
Closed-loop verification with surrogate plant model

- Solution – create small surrogate model
- Use the model for FB synthesis
  - “+”: fast verification
  - “-”: it may be nontrivial to create the model
Algorithmic ideas

How can we use this information for FB synthesis?

1. Ratio of satisfied formulas
2. Longest counterexample length
3. Verification-aware mutation operator
1. Ratio of satisfied formulas

\[
\frac{\text{# of satisfied formulas}}{\text{# of formulas}}
\]

Issue – this variable has too few possible values!
2. Length of the longest counterexample

- Solutions with long counterexamples are probably better than solutions with short ones

\[ F_{\text{smv}}^{\text{ce}} = \begin{cases} 
1, & \text{if } l_{\text{max}} = 0; \\
1 - \frac{1}{(1 + \frac{1}{10}l_{\text{max}})^\frac{1}{10}}, & \text{otherwise.}
\end{cases} \]
3. Verification-aware mutation operator

LTL formula $f$ → NuSMV → Counterexample

Increase probability of changing visited transitions
Efficiency issues

• Closed-loop with surrogate model is fast, but not fast enough
  • Verification takes ~ 0.5-1 seconds

Solution

• Calculate verification-based fitness only for $p$ % of solutions
• Definitely calculate for good solutions
Example: Pick-and-Place manipulator
## Considered LTL properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G(\neg (c_1 \text{Extend} &amp; c_1 \text{Retract}))$</td>
<td>Cylinder I must never be given commands to extend and retract simultaneously</td>
</tr>
<tr>
<td>$G(\neg (c_2 \text{Extend} &amp; c_2 \text{Retract}))$</td>
<td>Analogous safety property about cylinder II</td>
</tr>
<tr>
<td>$G(pp_1 \rightarrow \neg F(vp_1))$</td>
<td>If a work piece appears on the first input track, it will eventually be picked up by the manipulator</td>
</tr>
</tbody>
</table>
Experiments

- Machine with 64-core AMD OpteronTM6378 @ 2.4 GHz processor, 32 Gb of RAM
- Used 16 cores

<table>
<thead>
<tr>
<th>#</th>
<th>Configuration</th>
<th>Time, s</th>
<th>Satisfied all LTL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>min</td>
<td>mean</td>
</tr>
<tr>
<td>1</td>
<td>Scenario</td>
<td>32</td>
<td>85</td>
</tr>
<tr>
<td>2</td>
<td>Scenario + LTL (no $F_{svnL}^{ce}$)</td>
<td>222</td>
<td>752</td>
</tr>
<tr>
<td>3</td>
<td>Scenario + LTL</td>
<td>164</td>
<td>563</td>
</tr>
</tbody>
</table>
Conclusion

• Developed method of FB inference from tests and LTL properties
• Demonstrated viability on the PnP example
• Still a long way to go…
Ongoing work: CSP-based inference

Advantages

• Typically – very fast
• Possibility to find all solutions
• Symmetry breaking
• Indirect solution of "tests + LTL" synthesis problem
Acknowledgements

- Financially supported by Government of Russian Federation, Grant 074-U01.
- Partially funded by RFBR according to the research project No. 16-37-00205 mol_a.
Thank you for your attention!

rain.ifmo.ru/~chivdan