Combining Requirement Mining, Software Model Checking and Simulation-Based Verification for Industrial Automotive Systems

2016/10/06

Tomoya Yamaguchi and Tomoyuki Kaga
TOYOTA MOTOR CORPORATION
Alexandre Donzé and Sanjit A. Seshia
University of California, Berkeley
TOYOTA V&V Perspective

Automobile system becomes more complex and larger in scale.

Purpose: Establish prevention process with advanced V&V

The num. of NHTSA recalls related to control software increasing.

Automotive system is really
- Production
- Cyber Physical System
- Closed loop controller

Num

Increasing
Applying model checking to our CPS

An issue occurred when we were developing.

Under a rare operating pattern

In a software module of linear sensor calibration

Linear actuator

An issue happened when we were developing.

I applied model checking to this issue and analyzed.

Applying model checking to our CPS

Total Work hour: 560 min

- Making model: 30 min
- Making property: 40 min
- Revising model: 70 min
- Executing model checking: 70 min
- Revising property: 10 min
- Mapping counterexample: 15 min

Making/revising property: 110 min
Mapping counterexample: 280 min for just 1 module
The problem of applying model checking

Problem 1: Mapping system level requirement to module

Problem 2: Mapping counterexamples found at the module level to system-level counterexamples
V&V object: Injected issue on actual Engine SILS

- Pedal
- Brake
- WaterTemp
- AirTemp

Controller has 300,000 line of C code

Production
CPS
Closed loop
**V&V object: Injected issue on actual Engine SILS**

- **Controller**
  - Has 300,000 line of C code
  - Module
    - Module
      - Module
        - Module
          - Controller

- **Plant**
  - Actuator output

- **Corner case issue injected**
  - Other modules must be in specific combination of states
  - Property: actuator output < 150

- **Checkmarks**
  - Production
  - CPS
  - Closed loop
Overview of our methodology

1. Pre-condition (range) mining
   Pre-condition for software module

2. Software model checking
   Module level counterexample
   SLDV/CBMC

3. Simulation-Based Verification
   System level counterexample

Diagram:
- Pedal
- Brake
- WaterTemp
- AirTemp
- Sensor
- Module
- Controller
- Plant
- Target
Problem 1 Mapping system level requirement to module

We have system level requirement

Hard to map system level requirement to module level
Counter measure for problem 1: Requirement Mining

Apply requirement mining to mine pre-condition (range)

\[ x = (x_1, \ldots, x_n) \]
\[ \pi_{\text{min}} = (\pi_{\text{min}} 1, \ldots, \pi_{\text{min}} n) \]
\[ \pi_{\text{max}} = (\pi_{\text{max}} 1, \ldots, \pi_{\text{max}} n) \]
\[ \Box \left( \bigwedge_{i=1}^{n} ((\pi_{\text{min}} i \leq x_i) \land (x_i \leq \pi_{\text{max}} i)) \right) \]
\[ \Box (\neg 100 < x_1) \land (x_1 < 100) \ldots \]
Result of using module level requirement

Counterexample comes from model checking

<table>
<thead>
<tr>
<th>Input variable</th>
<th>No range</th>
<th>With range mining</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>counterexample</td>
<td>range</td>
</tr>
<tr>
<td>waterTemp [°C]</td>
<td>89.4</td>
<td>[-30.0, 100.0]</td>
</tr>
<tr>
<td>atmosphericPressure [bar]</td>
<td>3.5</td>
<td>[0.0, 1.0]</td>
</tr>
<tr>
<td>gear</td>
<td>5</td>
<td>[0, 6]</td>
</tr>
<tr>
<td>gearHoldFlag</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>idlFlag</td>
<td>0</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>catalystTempHIGHflag</td>
<td>1</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>fuelCutFlag</td>
<td>0</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>engRpm [rpm]</td>
<td>2600.0</td>
<td>[0.0, 5310.9]</td>
</tr>
</tbody>
</table>

false positive case is avoided by using range mining
Problem 2 Mapping counterexamples found at the module level to system-level counterexamples.

Pre-condition of module is extracted by requirement mining. Now model checking is more accurate!

Pedal

Controller

Plant

Module

Module

Module

Module

Actuator

Output

Problem 2 Mapping counterexamples found at the module level to system-level counterexamples.
Problem 2 Mapping counterexamples found at the module level to system-level counterexamples

Is this module level counterexample from model checking false positive or true positive?

Generally, it needs much work-hour, HI-level V&V skill and system knowledge
Problem 2 Mapping counterexamples found at the module level to system-level counterexamples

Hypothesis: Module level CE is a true positive, when system level CE containing module level CE is found.
Simulation-Based Verification with cost function

Drive system to module level CE using Simulation-Based Verification

Want to falsify property: (minimize distance to CE) \[ \varphi(x) = \square \left( \sum_{i=1}^{n} (x_i(t) - \hat{x}_i)^2 \right) \geq \varepsilon \]
Found system level corner case issue

- Pedal
- Brake
- WaterTemp
- AirTemp

Found false case

Violation area of post-condition

Find system level violation
actuator output < 150
Comparison with just Simulation-based Falsification

- significantly more effective than using just software Model checking or just Search-based falsification

Graph showing comparison between different falsification methods:
- Post-condition only falsification
- Combined methodology

Legend:
- Pre-condition mining
- Software model checking
- Simulation-based falsification

Reduce over 20min
Conclusion

- We propose combined methodology (= Requirement Mining + Model Checking + Simulation-based verification)
- New methodology is applied to production closed loop CPS
- Our combined methodology can be significantly more effective than using just software Model checking or just Simulation-based verification

Special thanks

**Breach:** Breach is provided by U.C. Berkeley, Prof. Sanjit Seshia and Dr. Alexandre Donzé. Breach has flexible extendibility for the requirement mining and the simulated-based verification.

**CBMC:** CBMC is provided by Univ. Oxford, Prof. Daniel Kroening and Dr. Martin Brain. CBMC is a sophisticated tool and was greatly helpful for our case study.

**SMiL:** Toyota in-house engine SILS. Fujitsu-ten provides and also supports us well.