Dániel Darvas (BE-ICS-PCS)

Formal verification of industrial control systems

3th Workshop on PLC/COTS-based Interlock and Protection Systems
02/02/2016, CERN

Formal verification?
Yes, boring stuff
Academic
Just survive until the next pres :)
Source: http://www.iphonetextgenerator.com/
Context – CERN

- PLCs for controlling **vacuum**, **cryogenics**, **CV**, etc. systems + **safety** systems

- Failures might have *negative impact*

- **Increasing complexity** without **decreasing quality**?
Context – PLCs at CERN

– Programmable Logic Controllers
  *robust industrial computers*

– Small computing capacity,
  special programming languages

– **1000+ PLCs** at CERN
Goal

- To **improve the quality** by eliminating bugs
  - Complementing automated and manual testing

- **Model checking** to find “high quality” bugs

- **Integrating** formal verification to the development process
What is formal verification?

- **Formal verification**: mathematically sound methods to check properties of specifications / implementations / …

- **Model checking**
  - Automated formal verification method
  - Checks all possible executions (contrarily to testing)
  - Goal: prove correctness OR find hidden/rare problems
Testing vs. model checking

**Testing**

- Inputs are known, outputs are checked
- \( \text{add}(5, 3) = 8 \) ?

**Model checking**

- E.g. the possibility of an **output combination** is checked.
- \( \text{add}(\ldots, \ldots) < 0 \) ?
- Temporal expressions are possible
Usage of formal verification

- Used both in **industry** and **academia**
  - Typically when the *cost of failure is high*

- Formal verification for **PLCs**
  - Mostly in academic environment
  - Not widely spread yet in industry – **too difficult!**

*The presented logos are trademarks of the corresponding companies and their usage is nominative fair use. None of the companies endorse or support the presented work.*
Main challenges of model checking

- **Formalization**
  - ... of source code
  - ... of requirements

- **Performance**

- Making it **accessible to the developers**

*(No general solution to date.)*
Model checking (extended workflow for PLCs)

PLC code (ST, SFC, IL) → Formal model

Requirement patterns → Formal requirement

Model checker

Reductions

Satisfied

Not satisfied

Verification report

Counter-example
Model checking (extended workflow for PLCs)

- PLC code (ST, SFC, IL)
- Requirement patterns
- Verification report
The PLCverif tool

Eclipse-based editor for PLC programs
The PLCverif tool

Defining verification cases (requirement, fine-tuning, etc.)
No model checker-related things or temporal logic logic expressions
The PLCverif tool

Requirement patterns
The PLCverif tool

PLCverif — Verification report

Generated at Mon Jul 07 15:19:22 CEST 2014 | PLCverif v2.0.1 | (C) CERN EN-ICE-PLC | Show/hide expert details

<table>
<thead>
<tr>
<th>ID:</th>
<th>Demo001</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>If A is false, C cannot be true.</td>
</tr>
<tr>
<td>Description:</td>
<td>If A is false, C cannot be true. As this function block models an AND-gate, if any of the inputs (A or B) is false, the output should be false too. The requirement is based on the documentation of the function block and the following Jira case: <a href="https://icecontrols.its.cern.ch/jira/browse/UCPC-1111">https://icecontrols.its.cern.ch/jira/browse/UCPC-1111</a></td>
</tr>
<tr>
<td>Source file:</td>
<td>DemoSource.scl</td>
</tr>
<tr>
<td>Requirement:</td>
<td>3. A = false &amp; C = true is impossible at the end of the PLC cycle.</td>
</tr>
<tr>
<td>Result:</td>
<td>Not satisfied</td>
</tr>
</tbody>
</table>

Tool: nusmv

Total runtime (until getting the verification results): 212 ms
Total runtime (incl. visualization): 361 ms

Counterexample

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>End of Cycle 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input a</td>
<td></td>
<td>FALSE</td>
</tr>
<tr>
<td>Input b</td>
<td></td>
<td>TRUE</td>
</tr>
<tr>
<td>Output c</td>
<td></td>
<td>TRUE</td>
</tr>
</tbody>
</table>

Click-button verification, verification report with the analysed counterexample
Example – SMTP safety system
Sc Magnet Test Plant safety system

**Goal:** ensuring **safety** by allowing/forbidding tests

**Core:**

- selected test
- switch statuses
- current voltages
- cryo conditions

---

SM18 PLCSE
safety logic

---

Safety-critical, can be dangerous

---

test allowed
Model checking workflow for this case

SM18 PLCSE Safety Logic

PLC code / Formal model

Requirement

Model checker

Satisfied

Counterexample

Not satisfied

E.g. Test starts only if cryo conditions are OK

Example inputs to violate the requirement

Black magic inside

SM18 PLCSE Safety Logic

E.g. Test starts only if cryo conditions are OK

Example inputs to violate the requirement

Black magic inside
SM18 PLCSE safety logic

- selected test
- switch statuses
- current voltages
- cryo conditions

test allowed
Excerpt from the work of R. Speroni
Problems found *(before putting in production!)*

**Requirement misunderstanding**
- Recognised while specifying requirements

**Functionality problems**
- “The [magnet] test should start, but it doesn’t.”

**Safety problems**
- “The [magnet] test **should NOT start**, but it does.”
Problems found

In total **14 issues** found

- **4** requirement misunderstanding
- **6** problems could not be found using our testing
Continuous verification

HTMLSummary.html

Some pictures have been blocked to help prevent the sender from identifying your computer. Open this item to view the pictures.

DO_FLASHBOX_ADJ_ON_req1

4 DO_FLASHBOX_ADJ_ON = (((SEL_1_TEST_TYPE >= 0ud8_5 AND SEL_1_TEST_TYPE <= 0ud8_9 AND SEL_ACTIVE_BENCH = 0ud8_1 AND (PCO1_OnSt OR PCO2_OnSt)) OR (SEL_2_TEST_TYPE >= 0ud8_5 AND SEL_2_TEST_TYPE <= 0ud8_9 AND SEL_ACTIVE_BENCH = 0ud8_2 AND (PCO1_OnSt OR PCO2_OnSt)))) is always true at the end of the PLC cycle.

Satisfied

Total: 19082 ms* (MChk: 250 ms) [Open the Verification Report]

DO_FLASHBOX_ADJ_ON_req2

2 If DO_FLASHBOX_ADJ_ON is true at the end of the PLC cycle, then (_SEL_FLASHBOX_ADJ_POWER = 0ud8_1) should always be true at the end of the same cycle.

Satisfied

Total: 10489 ms* (MChk: 258 ms) [Open the Verification Report]

DO_INTERCC_req2

4 DO_INTERCC = (_SEL_SWITCH_INTERCON = 0ud8_3) is always true at the end of the PLC cycle.

Satisfied

Total: 8491 ms* (MChk: 98 ms) [Open the Verification Report]

DO_INTERCC_req3 (safety)

2 If DO_INTERCC is true at the end of the PLC cycle, then SEL_SWITCH_INTERCON = 0ud8_3 and _SEL_SWITCH_INTERCON = 0ud8_3 should always be true at the end of the same cycle.

Satisfied

Total: 5704 ms* (MChk: 149 ms) [Open the Verification Report]

See more about Jenkins PLCvent.
Alternative method *(side note)*

Formal specification + Behaviour **equivalence checking**

- Formal specification is needed
- Computationally difficult
- Complete
- No need for requirement extraction
Summary

− “Formal verification is not relevant to industry.”  **FALSE**!

− First steps to **apply formal verification** to PLCs
  - **Interesting bugs** found *(with joint effort)*
  - **Critical parts** can be checked
  - **Complementary** to testing

− Still long way to go
  - Improving the **performance**
  - **Formal specification**

http://cern.ch/plcverif
Formal verification?

Yes, boring stuff

Academic

Just survive until the next pres :)

Formal verification is great!

Well... At least now it's over.