Dániel Darvas (CERN / TU Budapest), István Majzik, Enrique Blanco

PLC code generation based on a formal specification language

INDIN2016 Conference
19-21/07/2016, Poitiers, France
Motivation

Critical (PLC-based) industrial control systems ⇒ Verify them!

- Testing
  - Traditional, but not enough

- Formal methods (model checking)
  - Without involving formal methods experts!

We focus on the software now
What is the **source of requirements**?

**Completeness of verification?**
PLCspecif
Formal specification for PLC modules
Goals and requirements

Goals:
- Provide **unambiguous, consistent** requirements
- Help the **understanding** and **formal verification**

Requirements:
- **Lightweight** method: easy to introduce, adapted to the available **knowledge**
- **Domain-specific**
### Example Module

**Assigned inputs:**
- ValueReq : INT16
- EnableReq_fromLogic : BOOL
- EnableReq_fromScada : BOOL
- EnableReq_fromField : BOOL
- DisableReq : BOOL
- PMin : INT16 param
- PMax : INT16 param

**Assigned outputs:**
- Value : INT16
- Status : BOOL

**Input definitions:** (none)

**Event definitions:**
- @disable ⇐ rising_edge(DisableReq) (pri=1)
- @enable ⇐ EnableReq_fromLogic OR EnableReq_fromScada OR EnableReq_fromField (pri=2)

**Core logic (state machine)**

```
@disable
```

```
<table>
<thead>
<tr>
<th></th>
<th>ValueReq &lt; PMin</th>
<th>ValueReq &gt; PMax</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>PMin</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>PMax</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>ValueReq</td>
</tr>
</tbody>
</table>

```

**Output definitions:**
- Value = in_state(Enabled) | result
- Status = in_state(Enabled)

**Invariant properties:**
- ALWAYS PMin ≤ Value ≤ PMax ASSUMING PMin ≤ PMax
Formal semantics of PLCspecif

- Based on finite (timed) automaton
  - Defined PLCspecif $\rightarrow$ TA construction

- Supports different use cases
  - Constructed TA is similar to intermediate model (of PLCverif)
    $\rightarrow$ Model and conformance checking
  - Constructed TA is similar to CFG
    $\rightarrow$ Code generation
Code generation
Goals of code generation

High-level goals:
- To show that PLCspecif is implementable
- To foster the acceptance of generated code

Objectives:
- Correctness (equivalent behaviour)
- Readability, maintainability
Currently: **concrete syntax of SCL code is generated from the abstract syntax of PLCspecif**
Idea of translation

IF _E_reset AND s_On THEN // transition tReset
  s_On := FALSE; s_Off := TRUE;
END_IF;

IF _E_set AND s_Off THEN // transition tSet
  s_Off := FALSE; s_On := TRUE;
END_IF;

We do similarly for behaviours specific to PLCspecific.
Invariant and general properties

- Not straightforward to see invariant properties of e.g. a SM

    - **Invariant properties:**
      - ALWAYS \( PMin \leq Value \leq PMax \) ASSUMING \( PMin \leq PMax \)

- Model checking invariant properties *(directly on specification)*

<table>
<thead>
<tr>
<th>PLCspecif</th>
<th>PLCverif</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formal semantics</td>
<td>( \rightarrow ) Intermediate model</td>
</tr>
<tr>
<td>Invariant property</td>
<td>( \approx ) Requirement pattern</td>
</tr>
</tbody>
</table>

- SAT solver (Z3) for **static analysis**
  - Conflicting transitions
  - Infinite firing runs
Maintainability

The stand-by service finds a problem in the implementation of the cryogenics control system. What to do?

a) Modify specification, regenerate code, stop the plant (!), reload PLC
b) Modify the [generated] code on-line without stop

- Modification on site → discrepancy between impl. and spec.
- **Conformance checking** to re-establish the consistency
  (After manual update of the specification.)
Readability

- Generated code **must not be a black box**
  - Structure of the code = structure of the specification

- **Configurable** code generator
  - How to **represent a state machine**?
  - How to **represent enumerations**?
    - Native enumeration type / Many Booleans / One integer / Special cases
  - What are the **naming conventions**?
  - What to **extract as a function block**? What should be inline?
  - …
Current state

- **Code generation**: used in experimental settings (currently)

- **Formal specification + conf. checking**: used in real projects
  - But code generation is not possible for fail-safe PLCs

- **UNICOS baseline re-engineering**: 
  - Ongoing project
Summary: Quality improvements using FM

- **Formal methods for ICS at CERN**

<table>
<thead>
<tr>
<th>Informal specification</th>
<th>Implementation</th>
<th>Formal specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>+</td>
<td>+ Model checking</td>
</tr>
<tr>
<td>#2</td>
<td>+</td>
<td>+ Conformance checking</td>
</tr>
<tr>
<td>#3</td>
<td>+</td>
<td>+ Code generation</td>
</tr>
</tbody>
</table>

- **Usage**
  - For **modules** of our **framework** (UNICOS)
  - For **real projects** (e.g. superconducting test facility)

**Case studies:**
- [WODES’14]
- [iFM’16]

For more information…

- Project website (with publication list)
  http://cern.ch/project-plc-formalmethods/
- PLCverif tool website  http://cern.ch/plcverif
- PLCspecif website     http://cern.ch/plcspecif
- CERN website         http://home.cern

- Contact me
  daniel.darvas@cern.ch
  http://cern.ch/ddarvas
Model checking at CERN


- D. Darvas et al. **PLCverif: A tool to verify PLC programs based on model checking techniques.** Proc. 15th Int. Conf. on Accelerator & Large Experimental Physics Control Systems, pp. 911-914, JaCoW, 2015. [http://dx.doi.org/10.18429/JACoW-ICALEPCS2015-WEPGF092](http://dx.doi.org/10.18429/JACoW-ICALEPCS2015-WEPGF092)

- B. Fernández et al. **Applying model checking to industrial-sized PLC programs.** IEEE Trans. on Industrial Informatics, 11(6):1400-1410, 2015. [http://dx.doi.org/10.1109/TII.2015.2489184](http://dx.doi.org/10.1109/TII.2015.2489184)
Formal specification at CERN


- D. Darvas et al. **Formal verification of safety PLC based control software.** Integrated Formal Methods (LNCS 9681), pp. 508-522. Springer, 2016. [http://dx.doi.org/10.1007/978-3-319-33693-0_32](http://dx.doi.org/10.1007/978-3-319-33693-0_32)