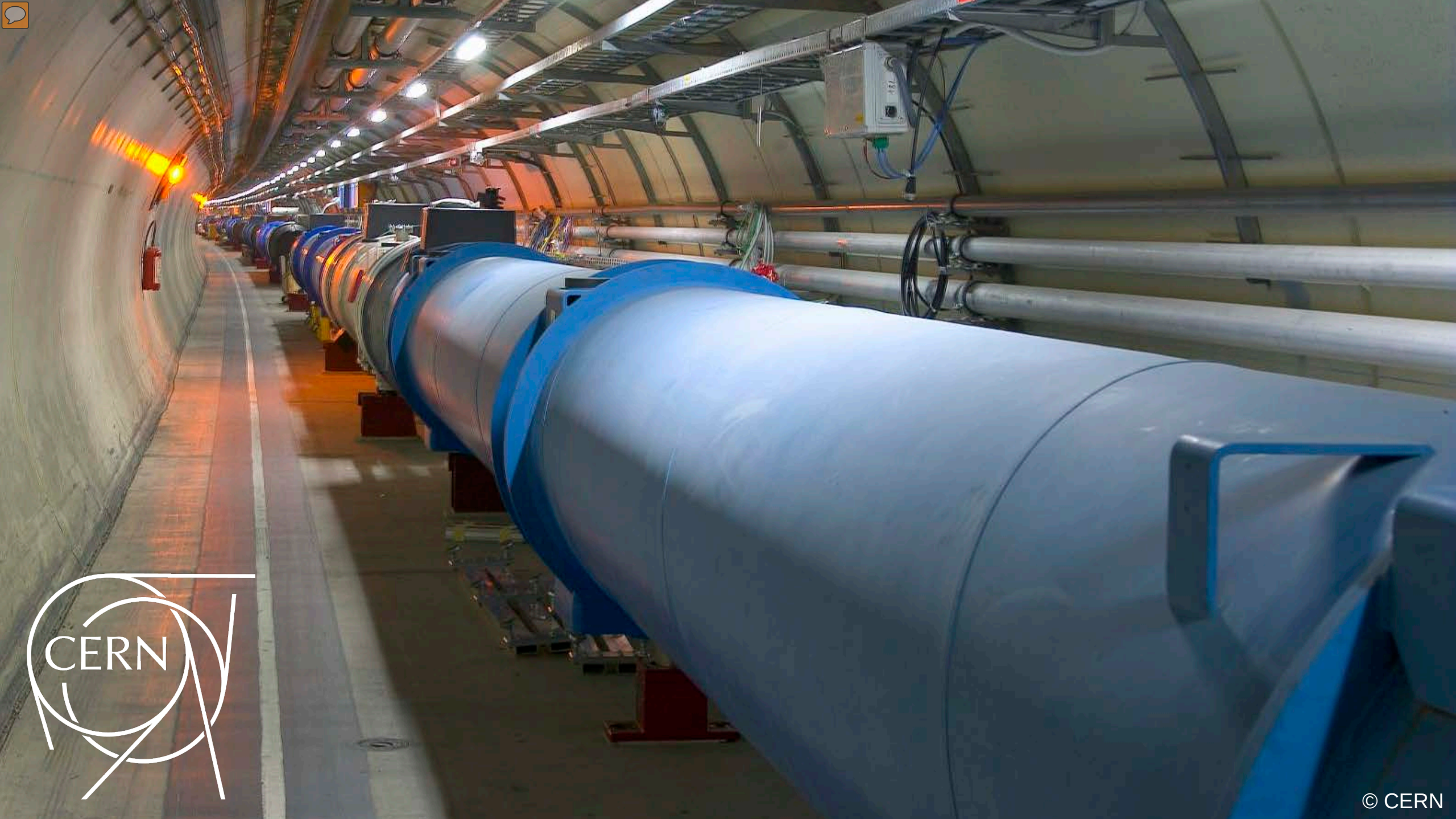

CERN

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

http://go.cern.ch/8wR8



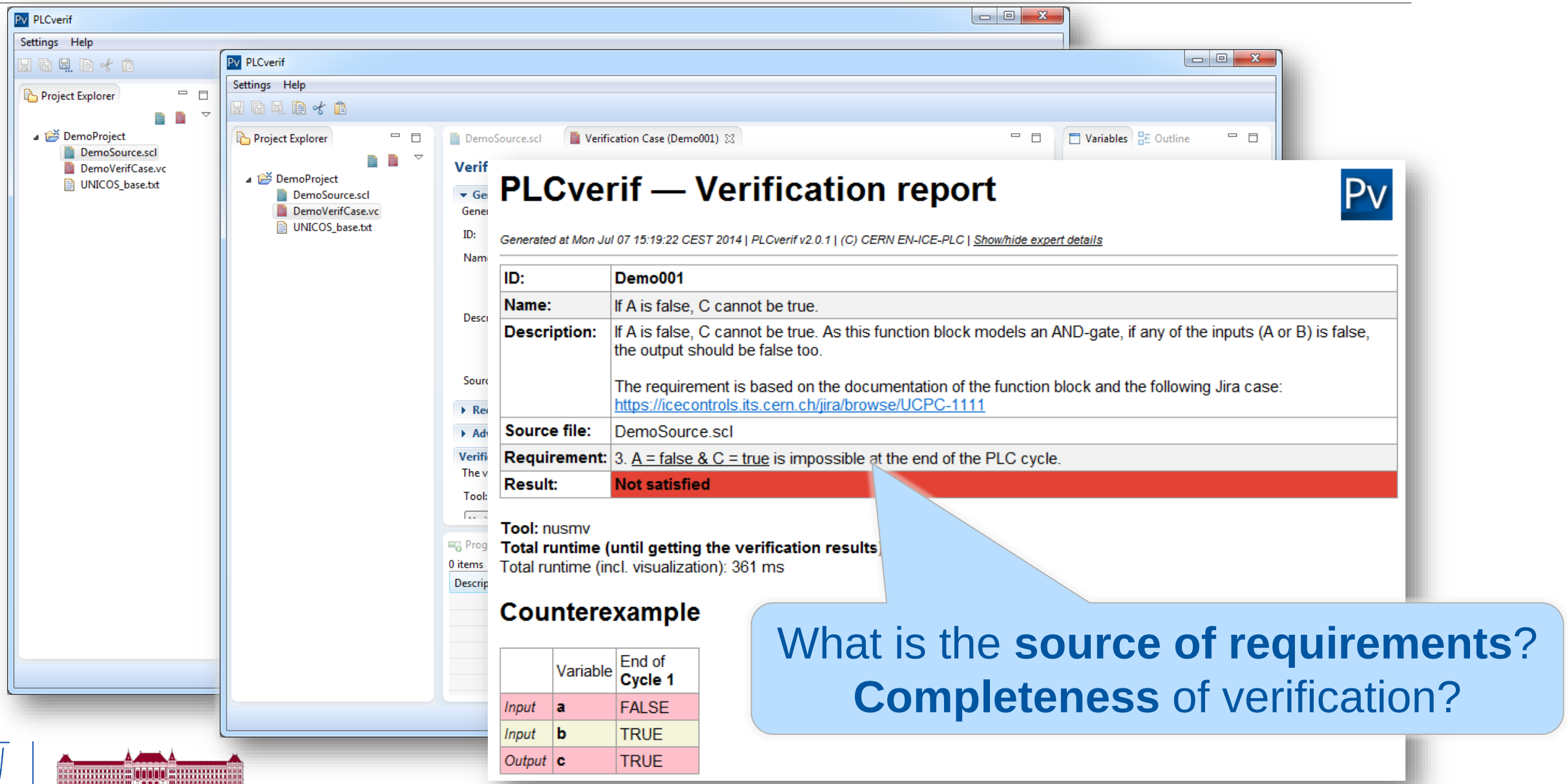
Motivation

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

We focus on the **software** now

- **Testing**
  - Traditional, but not enough
- **Formal methods** (model checking)
  - **Without involving formal methods experts!**

PLCverif


PLCverif
Settings Help
Project Explorer
DemoProject
DemoSource.scl
DemoVerifCase.vc
UNICOS_base.txt
Verification Case (Demo001)
Variables
Outline
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
ID: Demo001
Name: If A is false, C cannot be true.
Description: 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: https://icecontrols.its.cern.ch/jira/browse/UCPC-1111
Source file: DemoSource.scl
Requirement: 3. A = false & C = true is impossible at the end of the PLC cycle.
Result: Not satisfied
Tool: nusmv
Total runtime (until getting the verification results)
Total runtime (incl. visualization): 361 ms
Counterexample
Variable
End of Cycle 1
Input a FALSE
Input b TRUE
Output c TRUE
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**

D. Darvas, E. Blanco, I. Majzik. Requirements towards a formal specification language for PLCs. In *Proc. 22nd PhD Mini-Symposium*, pp. 18-21. BUTE, 2015. DOI: 10.5281/zenodo.14907

ExampleModule

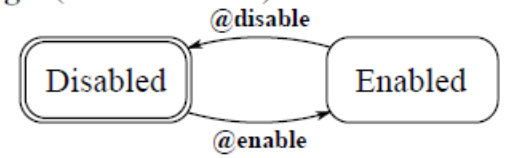
Assigned inputs:
- ValueReq : INT16
- EnableReq_fromLogic : BOOL
- EnableReq_fromScada : BOOL
- EnableReq_fromField : BOOL
- DisableReq : BOOL
- PMin : INT16 param
- PMax : INT16 param

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)



Output definitions:
- _Value =

ValueReq < PMin	ValueReq > PMax	result
T	·	PMin
F	T	PMax
F	F	ValueReq

- Value =

in_state(Enabled)	result
T	_Value
F	0

- Status = `in_state`(Enabled)

Invariant properties:
- ALWAYS *PMin* ≤ *Value* ≤ *PMax* ASSUMING *PMin* ≤ *PMax*

Detailed behaviour specification

Structured, **hierarchical**

Separation of concerns

Domain-specific semantics

Unifies different semi-formal formalisms

Supports **verification**

Formal semantics of PLCspecif

- Based on **finite (timed) automaton**
  - Defined PLCspecif → TA **construction**

- Supports different use cases
  - Constructed TA is similar to **intermediate model** (of PLCverif)
    → **Model** and **conformance checking**
  - Constructed TA is similar to **CFG**
    → **Code generation**

D. Darvas, E. Blanco, I. Majzik. Syntax and semantics of PLCspecif.
Report EDMS 1523877, CERN, 2015. https://edms.cern.ch/document/1523877

Code generation

Goals of code generation

High-level goals:

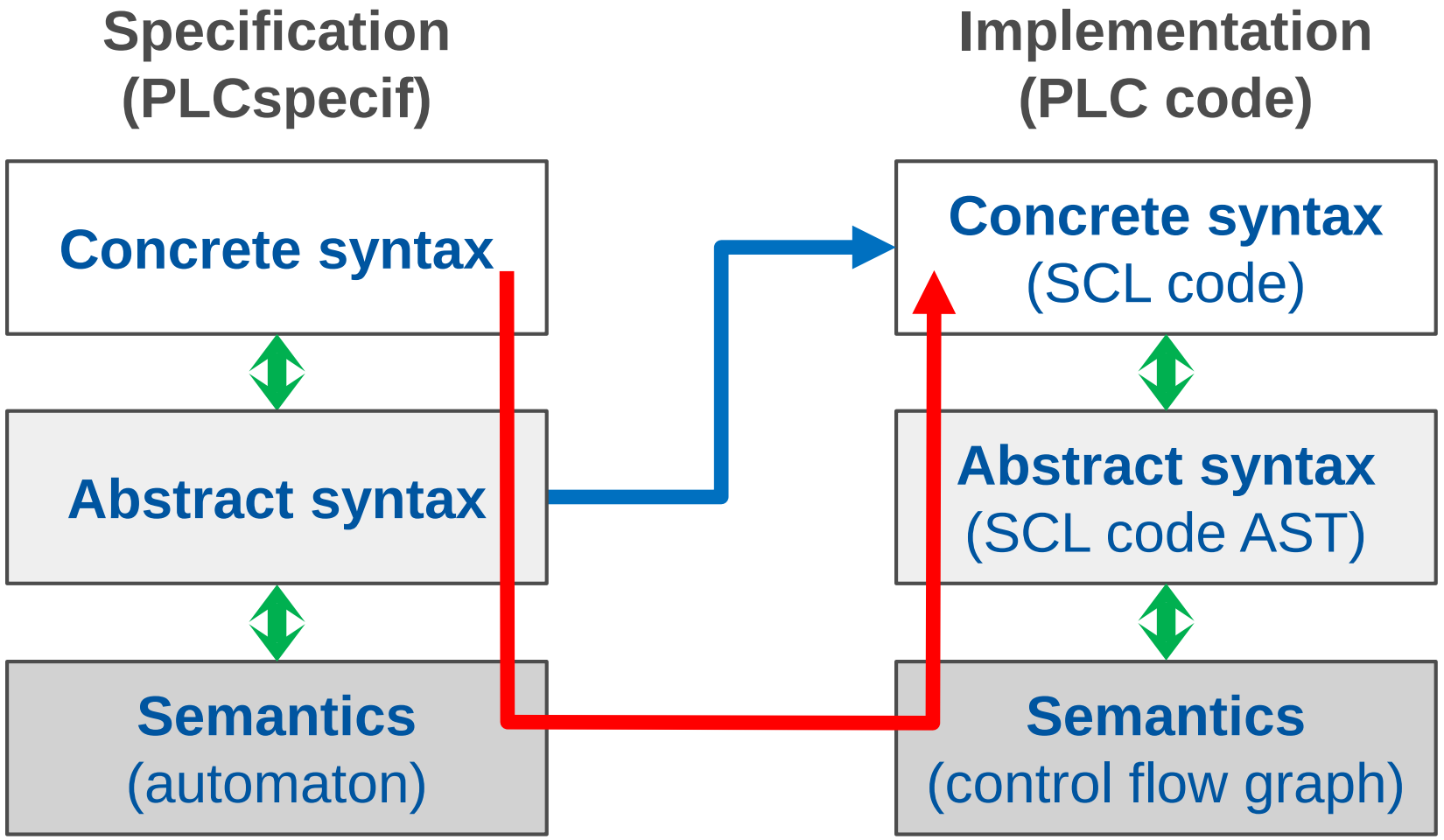
- To show that PLCspecif is implementable
- To foster the acceptance of generated code

Objectives:

- Correctness (equivalent behaviour)
- Readability, maintainability

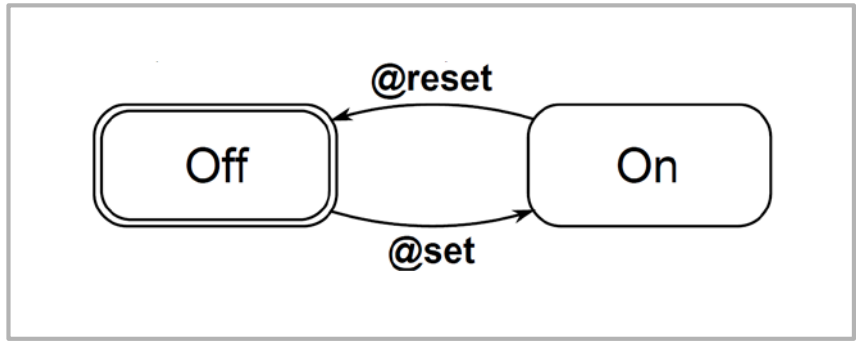
D. Darvas, E. Blanco, I. Majzik. Requirements towards a formal specification language for PLCs. In *Proc. 22nd PhD Mini-Symposium*, pp. 18-21. BUTE, 2015. DOI: 10.5281/zenodo.14907

From specification to code



Currently: ***concrete syntax of SCL*** *code is generated from the* ***abstract syntax of PLCspecif***

Idea of translation



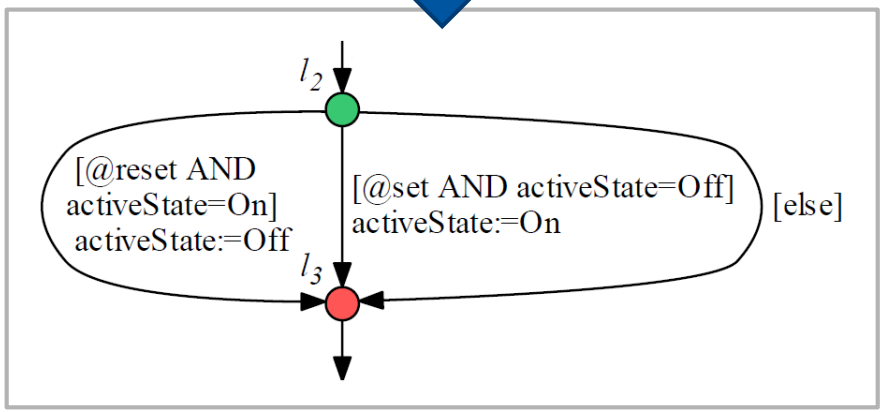
Generation

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

=

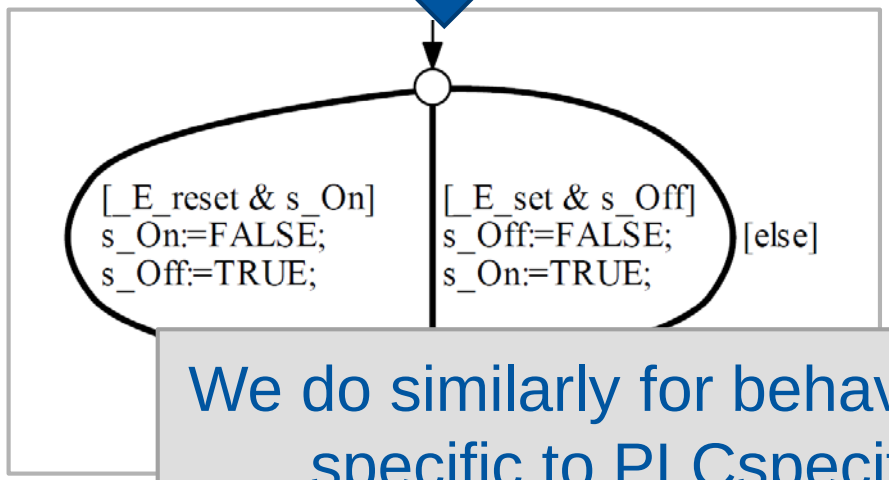
```
IF _E_reset AND s_On THEN   // transition tReset
    s_On := FALSE; s_Off := TRUE;
ELSIF _E_set AND s_Off THEN   // transition tSet
    s_Off := FALSE; s_On : = TRUE;
END_IF;
```

Semantics



Correspondence

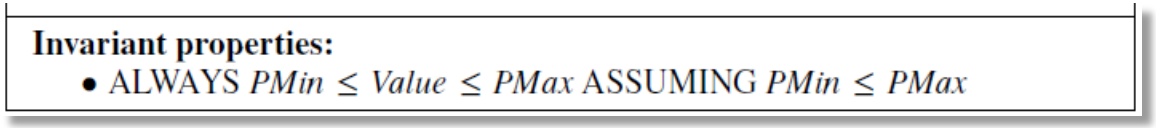
Semantics



We do similarly for behaviours specific to PLCspecif.

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)*

PLCspecif		PLCverif
Formal semantics	→	Intermediate model
Invariant property	≈	Requirement pattern

- **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

	Informal specification	Implementation	Formal specification	
#1	+	+		**Model checking**
#2	+	+	+	**Conformance checking**
#3	+		+	**Code generation**

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

Case studies:
[WODES'14]
[iFM'16]

D. Darvas. Practice-oriented formal methods for PLC programs of industrial control systems. In *Proc. PhD Symposium at iFM'16 on Formal Methods: Algorithms, Tools and Applications.* Reykjavik University, 2016. Extended abstract.

www.cern.ch

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. **Formal verification of complex properties on PLC programs**. Formal Techniques for Distributed Objects, Components, and Systems (LNCS 8461), pp. 284-299, Springer, 2014.
- B. Fernández et al. **Bringing automated model checking to PLC program development – A CERN case study**. Proc. 12th Int. Workshop on Discrete Event Systems, pp. 394-399, 2014.
- 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
- 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

Formal specification at CERN

- D. Darvas et al. **Requirements towards a formal specification language for PLCs.** Proc. 22nd PhD Mini-Symposium, pp. 18-21. BUTE DMIS, 2014. http://dx.doi.org/10.5281/zenodo.14907
- D. Darvas et al. **A formal specification method for PLC-based applications.** Proc. 15th Int. Conf. on Accelerator & Large Experimental Physics Control Systems, pp. 907-910, JaCoW, 2015. http://dx.doi.org/10.18429/JACoW-ICALEPCS2015-WEPGF091
- D. Darvas et al. **Syntax and semantics of PLCspecif.** CERN Report, EDMS 1523877. CERN, 2015. https://edms.cern.ch/document/1523877
- 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
- D. Darvas et al. **Conformance checking for programmable logic controller programs and specifications.** 11th IEEE International Symp. on Industrial Embedded Systems (SIES). IEEE, 2016.