CERN

Borja Fernández

Improving the quality of PLC programs

PBCS workshop - ICALEPCS 2017
06/10/2017

Outline

- Model checking
- Static analysis
- Conclusions

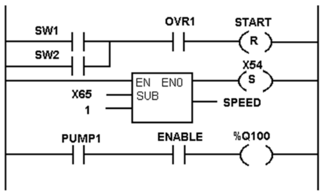
Context

- BE-ICS group: Industrial Control and Safety.
- **PLC** control systems:
  - **Standard and Safety** Instrumented Systems.

- **How can we guarantee that the PLC code is compliant with the specifications?**

IEC 61131-3

PLC program

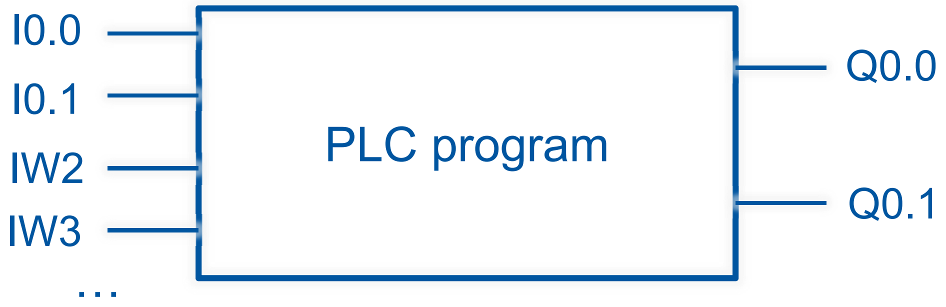


Specifications

- Before: manual and automated **testing**
  - Useful, but not efficient for every type of requirements.
  - Difficult to test **safety** requirements:

    “if out1 is true, out2 should be false”

Testing vs. model checking



Safety Requirement

*If **Q0.0** is TRUE, then **Q0.1** is FALSE*

It's **a "quite" complicated task for Testing**.

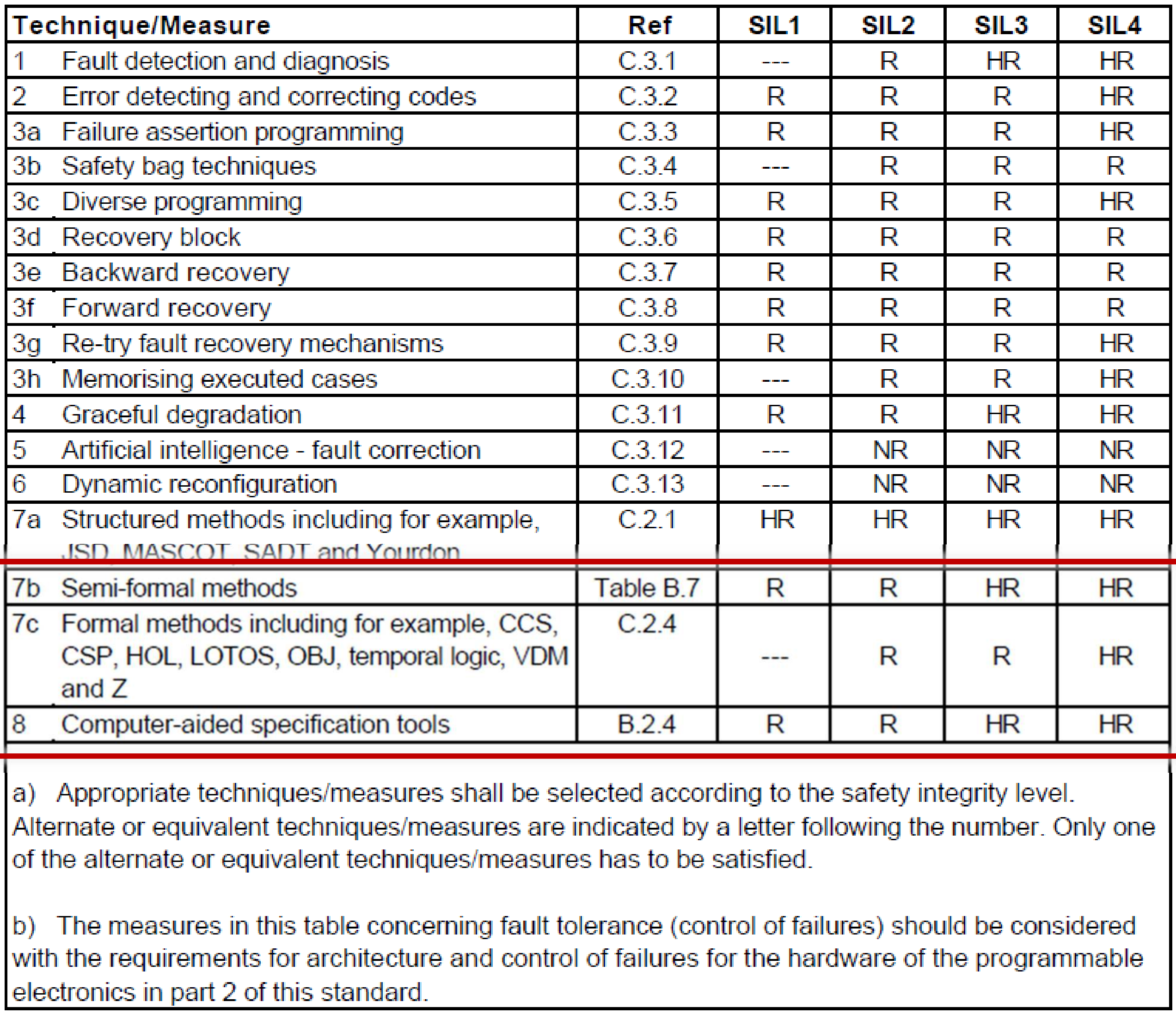
Model checking will **explore all input combinations** and will verify the safety property

Context

IEC 61508: Software design and dev. (table A.2)

IEC 61511 gives guidelines for the “application software”

Even for SIL1 it is recommended to use [Semi]-formal methods



Technique/Measure		Ref	SIL1	SIL2	SIL3	SIL4
1	Fault detection and diagnosis	C.3.1	---	R	HR	HR
2	Error detecting and correcting codes	C.3.2	R	R	R	HR
3a	Failure assertion programming	C.3.3	R	R	R	HR
3b	Safety bag techniques	C.3.4	---	R	R	R
3c	Diverse programming	C.3.5	R	R	R	HR
3d	Recovery block	C.3.6	R	R	R	R
3e	Backward recovery	C.3.7	R	R	R	R
3f	Forward recovery	C.3.8	R	R	R	R
3g	Re-try fault recovery mechanisms	C.3.9	R	R	R	HR
3h	Memorising executed cases	C.3.10	---	R	R	HR
4	Graceful degradation	C.3.11	R	R	HR	HR
5	Artificial intelligence - fault correction	C.3.12	---	NR	NR	NR
6	Dynamic reconfiguration	C.3.13	---	NR	NR	NR
7a	Structured methods including for example, JSD, MASCOT, SADT and Yourdon	C.2.1	HR	HR	HR	HR
7b	Semi-formal methods	Table B.7	R	R	HR	HR
7c	Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	---	R	R	HR
8	Computer-aided specification tools	B.2.4	R	R	HR	HR

a) Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

b) The measures in this table concerning fault tolerance (control of failures) should be considered with the requirements for architecture and control of failures for the hardware of the programmable electronics in part 2 of this standard.

Idea – Problems – State of the art

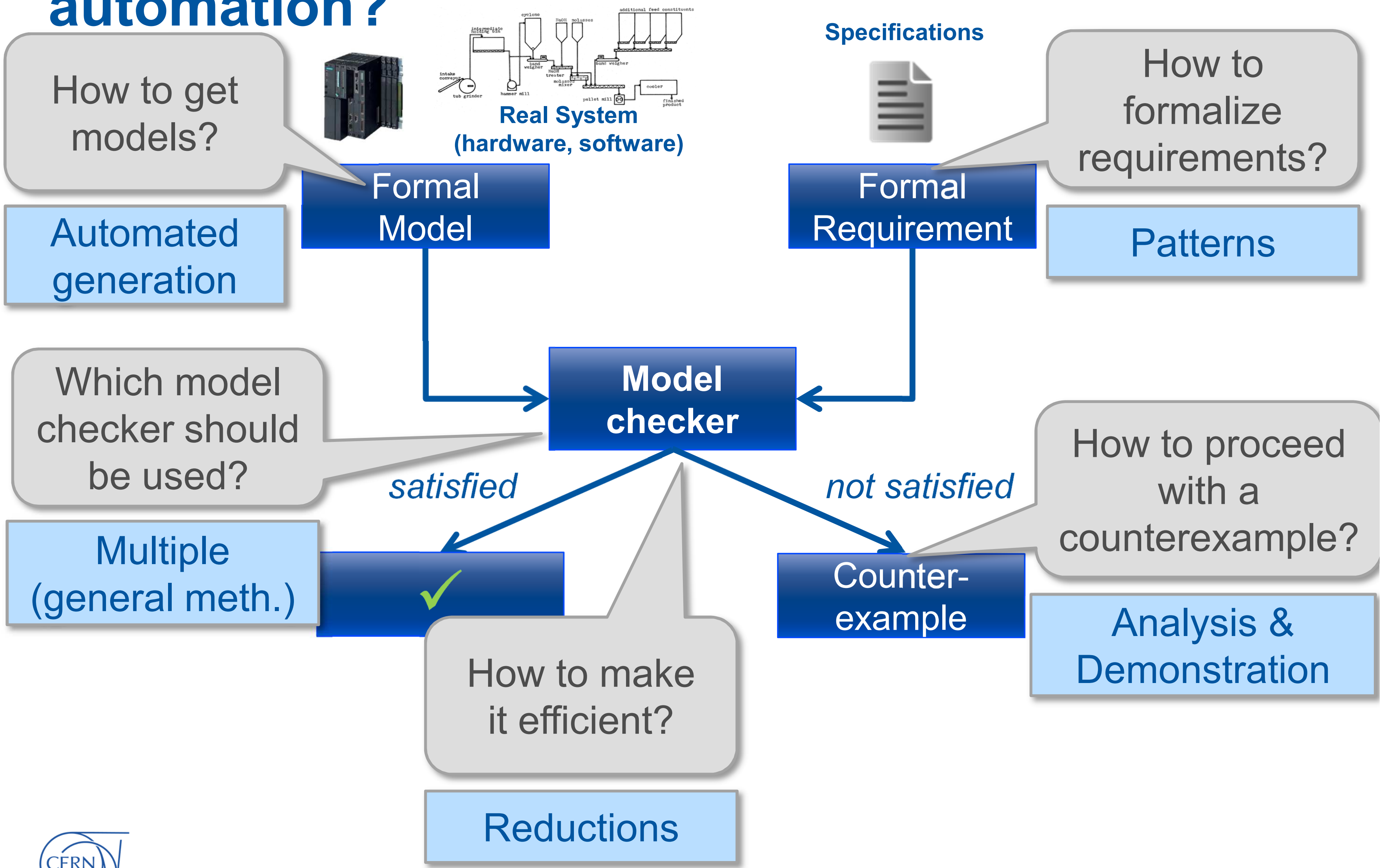
Applying formal verification to PLC programs (new developments and existing systems independently of the purpose)

- But…
  - Why formal verification **is not widely used** in industry yet?
  - How can we fill the gap between the **automation** and **formal verification** worlds?

- Other industries using formal methods:
  - **NASA**: Remote Agent spacecraft control system (Deep Space 1 mission).
  - **Aircraft** industry: Airbus A340 flight control , etc.
  - **Train** systems: **Subway** in Paris, Line 14.
  - **Communication** protocols: **IPv6** protocol.
  - Etc.

- What about **formal verification in PLC based control systems**?
  - Industry: ESTEREL (SCADE), Siemens and ABB doing research.
  - Academia: RWTH Aachen University, TU Dortmund, ENS Cachan, …

Why model checking is not widely used in automation?

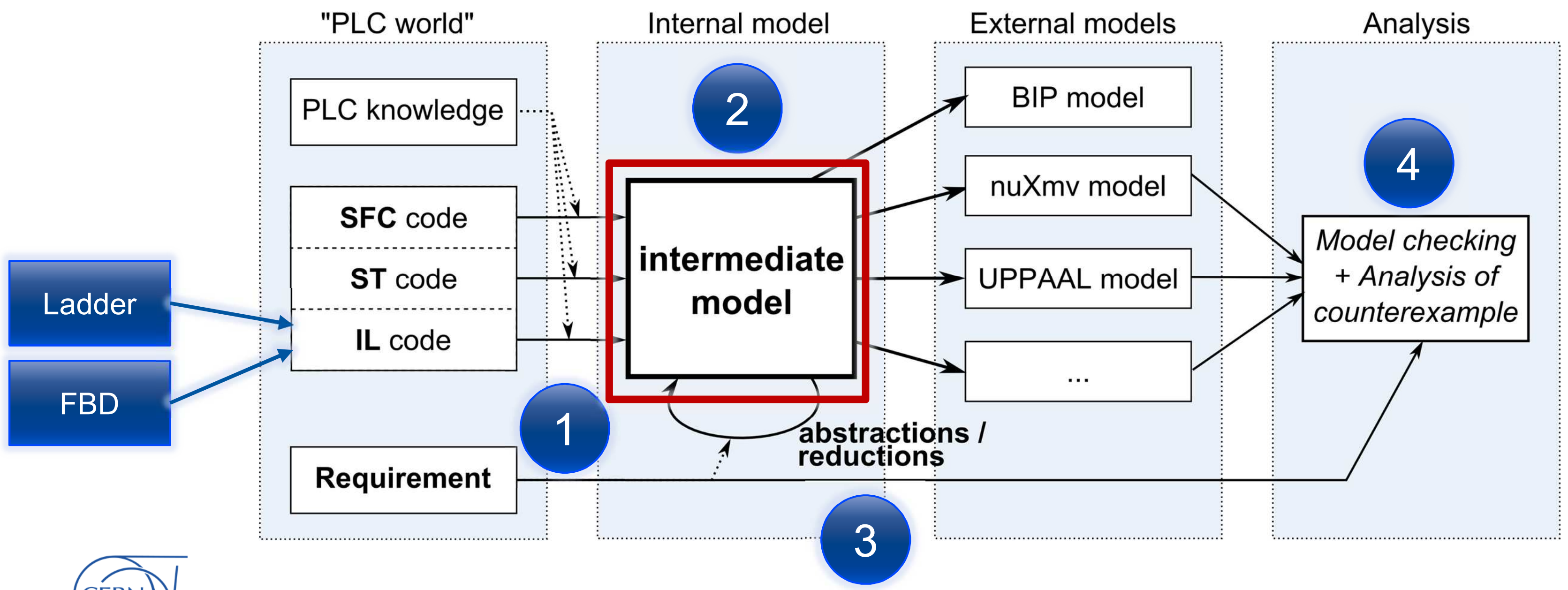


Model checking vs. Testing

- MC checks the specifications against a **model** instead of the real system.
- Allows to **check properties** that are almost **impossible** to test (e.g. liveness properties).
- Checks all **possible combinations.**
- Gives a **counterexample** when a discrepancy is found.
- Possible to **automatize** (can be used by non-formal method experts).
- State space **explosion.**

Our approach: methodology overview

- ❑ **General** method for applying formal verification:
  - ❑ **Generate** formal models **automatically** out of PLC code.
  - ❑ Includes **several input PLC languages** (IEC 61131-3: SFC, ST, IL, Ladder, FBD).
  - ❑ Easy integration of **different formal verification** tools.



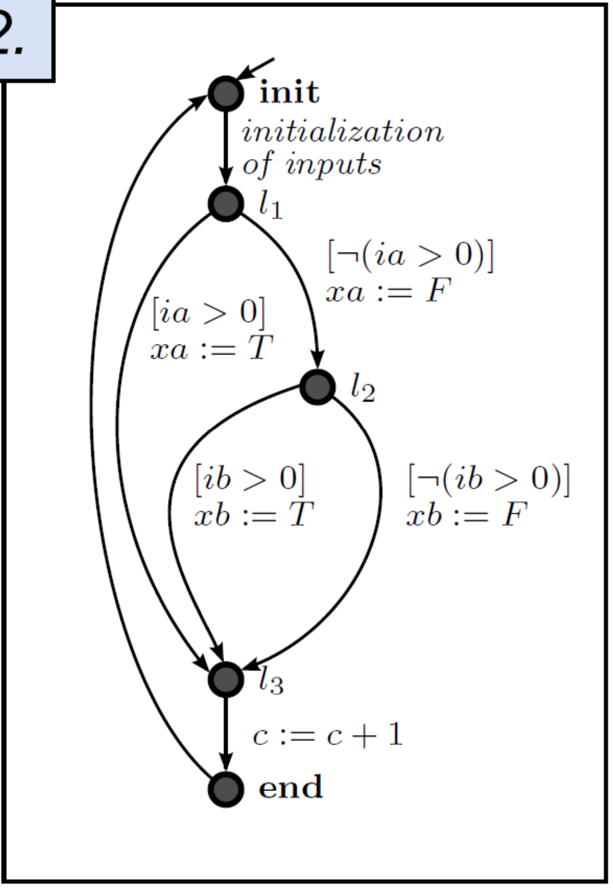
PLC formal verification at CERN: http://cern.ch/project-plc-formalmethods

Model example

1.

```
IF ia > 0 THEN
    xa := TRUE;
ELSE
    xa := FALSE;
    IF ib > 0 THEN
        xb := TRUE;
    ELSE
        xb := FALSE;
    END_IF;
END_IF;
c := c + 1;
```

2.



3.

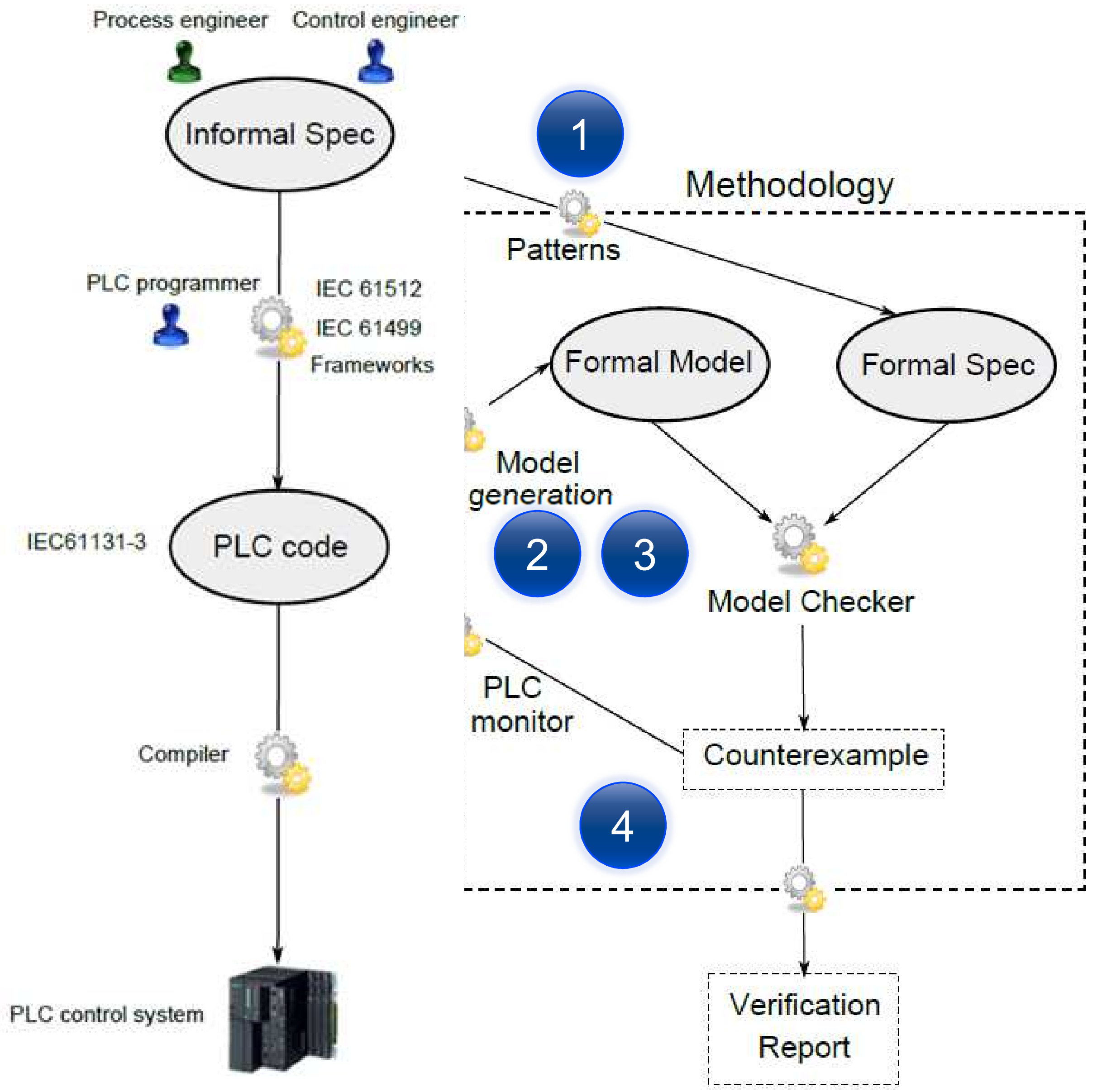
```
init(loc) := initial;
next(loc) := case
  loc = end : initial;
  loc = initial : l1;
  loc = l1 & ((IA > 0sd16_0)) : l3;
  loc = l1 & (!((IA > 0sd16_0))) : l2;
  loc = l2 & ((IB > 0sd16_0)) : l3;
  loc = l2 & (!((IB > 0sd16_0))) : l3;
  loc = l3 : end;
  TRUE: loc;
esac;
next(XA) := case
  loc = l1 & ((IA > 0sd16_0)) : TRUE;
  loc = l1 & (!((IA > 0sd16_0))) : FALSE;
  TRUE : XA;
esac;
...
```

Our approach: methodology overview

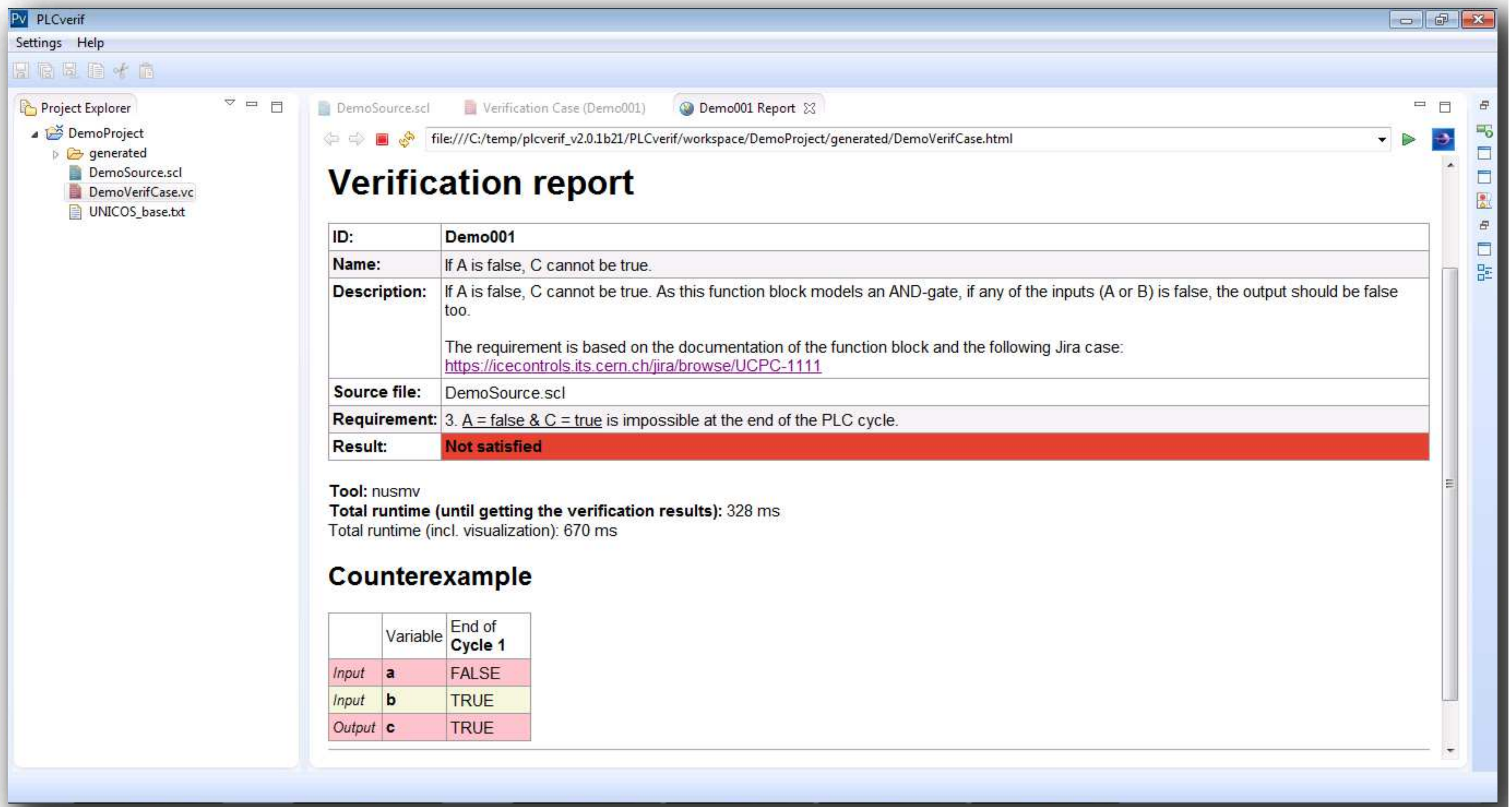
Traditional PLC program development

How can we be sure that a **bug** found by this methodology **is real**?

We can **use the counterexample in the real system** and prove it



Project status: CASE tool prototype



Verification report

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): 328 ms
Total runtime (incl. visualization): 670 ms

Counterexample

	Variable	End of **Cycle 1**
Input	**a**	FALSE
Input	**b**	TRUE
Output	**c**	TRUE

PLC formal verification at CERN: http://cern.ch/project-plc-formalmethods

Project status & Results

- The methodology **has been applied** to PLC programs at **CERN**:
  - UNICOS library: **Bugs/discrepancies have been found** in previously tested PLC programs
  - Full UNICOS applications: Cryogenic control system (QSDN)
  - **Safety PLC programs**: SM18, FAIR, AWAKE (*THCPA01 paper*), ITER HIOC protocol (*THPHA161 paper*), etc.

- Future development & research:
  - **Production-ready CASE tool**
  - More abstraction and reduction techniques
  - Control system specifications

Static Analysis

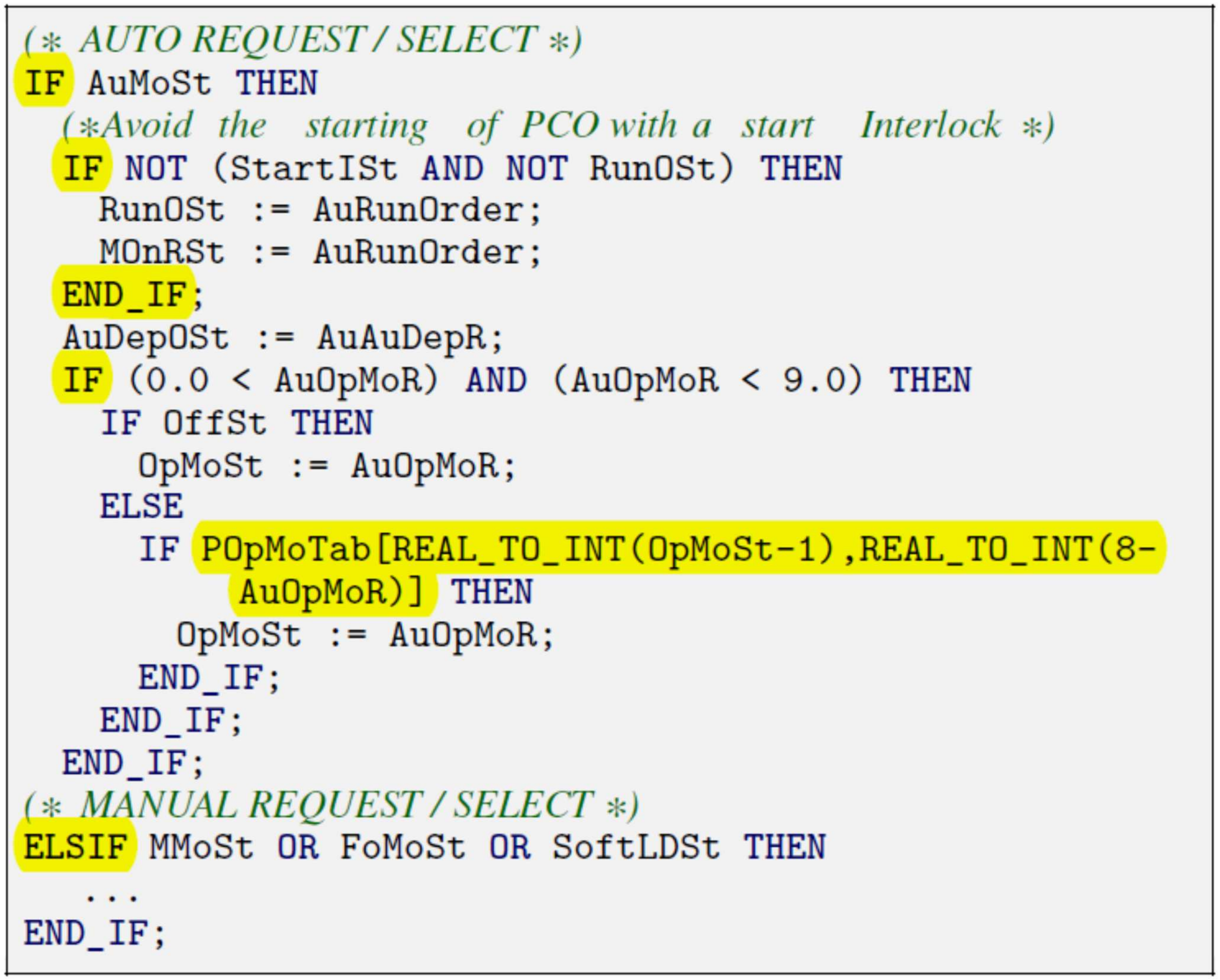
- **What is it?**
  - Technique that **examines a program without executing it**
  - Similar to **code review** or code comprehension performed by automated tools
  - Good complement to testing and formal verification

- **Which method?**
  - Rule-based **AST** (Abstract Syntax Tree) **analysis**, **control-flow analysis**, data-flow analysis, call graph analysis, etc.

- **What can we detect?**
  - Naming conventions violations, bad code smells (e.g. dead or duplicated code), overcomplicated expressions, multitasking problems, etc.

Static Analysis in PLC

- UNICOS code guidelines?



```
(* AUTO REQUEST / SELECT *)
IF AuMoSt THEN
  (*Avoid the starting of PCO with a start Interlock *)
  IF NOT (StartISt AND NOT RunOSt) THEN
    RunOSt := AuRunOrder;
    MOnRSt := AuRunOrder;
  END_IF;
  AuDepOSt := AuAuDepR;
  IF (0.0 < AuOpMoR) AND (AuOpMoR < 9.0) THEN
    IF OffSt THEN
      OpMoSt := AuOpMoR;
    ELSE
      IF POpMoTab[REAL_TO_INT(OpMoSt-1),REAL_TO_INT(8-
          AuOpMoR)] THEN
        OpMoSt := AuOpMoR;
      END_IF;
    END_IF;
  END_IF;
(* MANUAL REQUEST / SELECT *)
ELSIF MMoSt OR FoMoSt OR SoftLDSt THEN
   ...
END_IF;
```

Static Analysis for PLC programs

- **Lack of Static PLC Code Analysis tools** comparing with general purpose programing languages

- Several researchers and companies are working to bring static analysis to PLC programs but **still far from being a common practice in this industry**

- **UNICOS specific code guidelines** implies specific static analysis rules for our programs

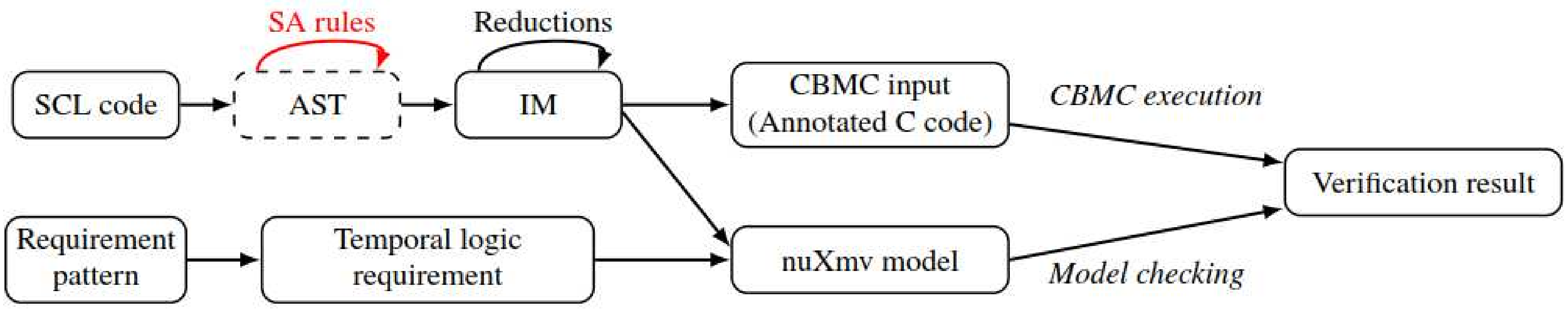
Static Analysis in PLC

- Some relevant tools

Tool	PLC language	SA method	Violations
PLC Checker (Itris Automation)	Siemens Step7,PLCopen XML, CoDeSys, Scheinder-Electric Unity and Rockwell Automation RsLogix5000 etc.	AST and control flow analysis (?)	naming rules, commenting rules, writing rules, structure rules, information utilities and options
JKU (Johannes Kepler University & Hagenberg and the ENGEL Austria GmbH)	Kemro language developed by KEBA AG	AST analysis, control flow and data flow	code metrics, naming conventions, program complexity and possible performance problems, bad code smells, architectural issues, incompatible configuration settings, multitasking problems and dynamic statement dependencies
ARCADE.PLC (Aachen University)	IEC 61131 ST IEC 61131 IL Siemens S7 AWL(STL)	Abstract interpretation	loss of precision in expressions, assignments required on specific cast, unused variables and output variables that were assigned more than once in the source code

Our solution: Extending PLCverif

- ICALEPCS 2017: **THPHA160 paper**
- Modular architecture of PLCverif
- Integration of AST rule-based analysis



Project status & Results

- Static analysis integrated in PLCverif

- Very **early stage** of the project
  - **AST-based analysis**
  - Naming convention and bad code smells rules (some of them UNICOS specific rules)

- Future work
  - More AST-based rules
  - Extend to other methods (e.g. control flow graph, call graph analysis, etc.)

CERN

www.cern.ch