Use of **formal methods** to **verify PLC code** and its **applicability to safety systems**

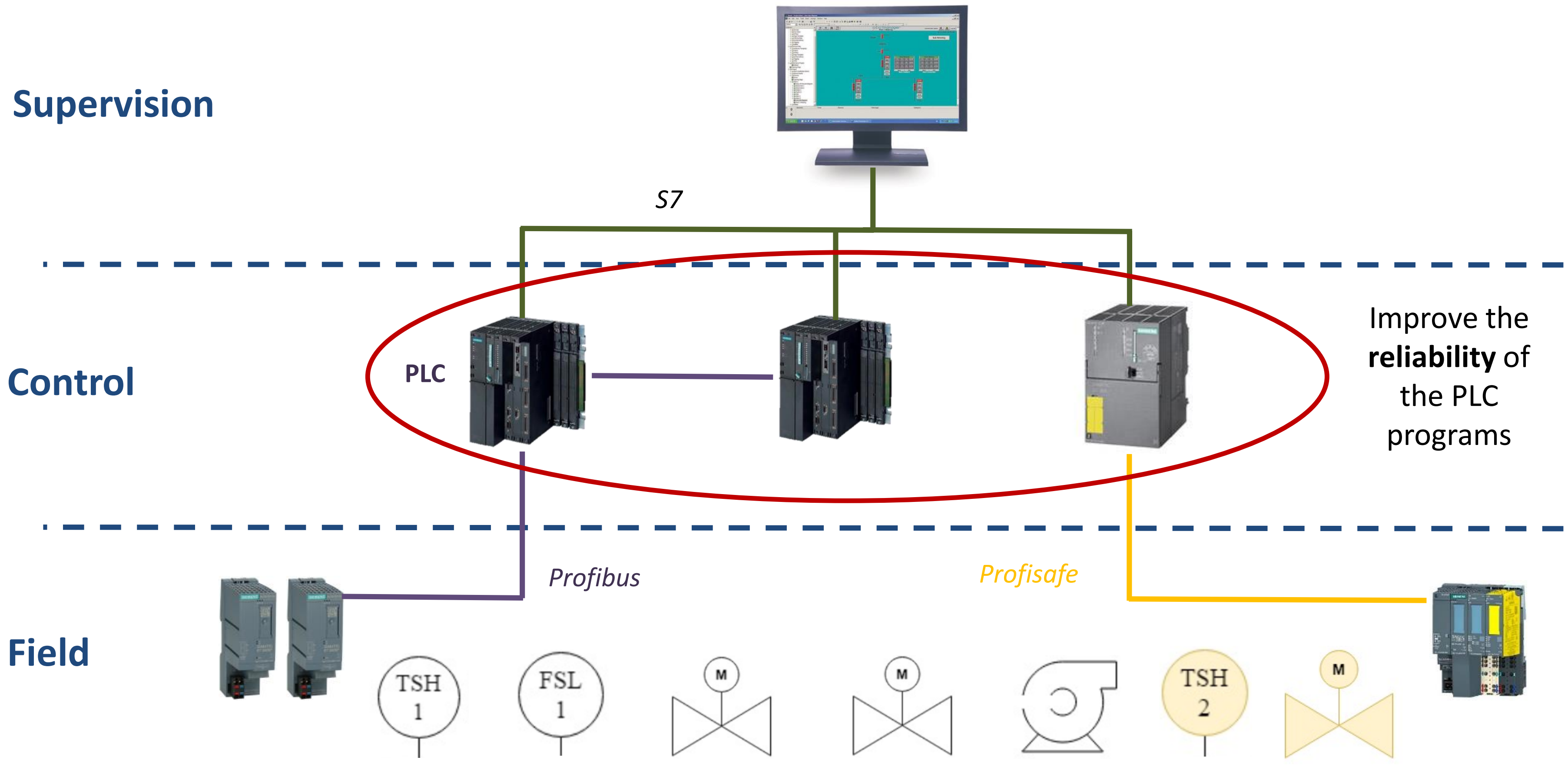
Borja Fernández Adiego **(CERN)**

Contains Joint work of

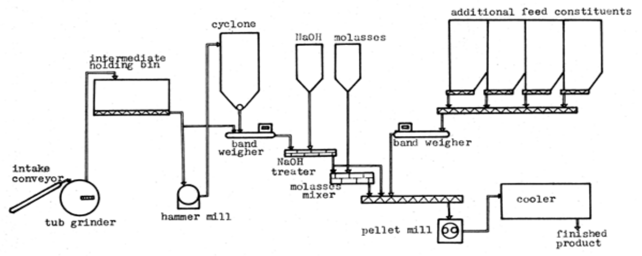
Enrique Blanco, Jean-Charles Tournier,

Daniel Darvas and Gyula Sallai

Industrial controls Safety Systems



Improve the **reliability** of the PLC programs



Introduction to model checking

Two main techniques can be applied to increase the software reliability:

- ❑ **Testing**: it checks if certain properties or test cases in the **real system** (the code is executed)

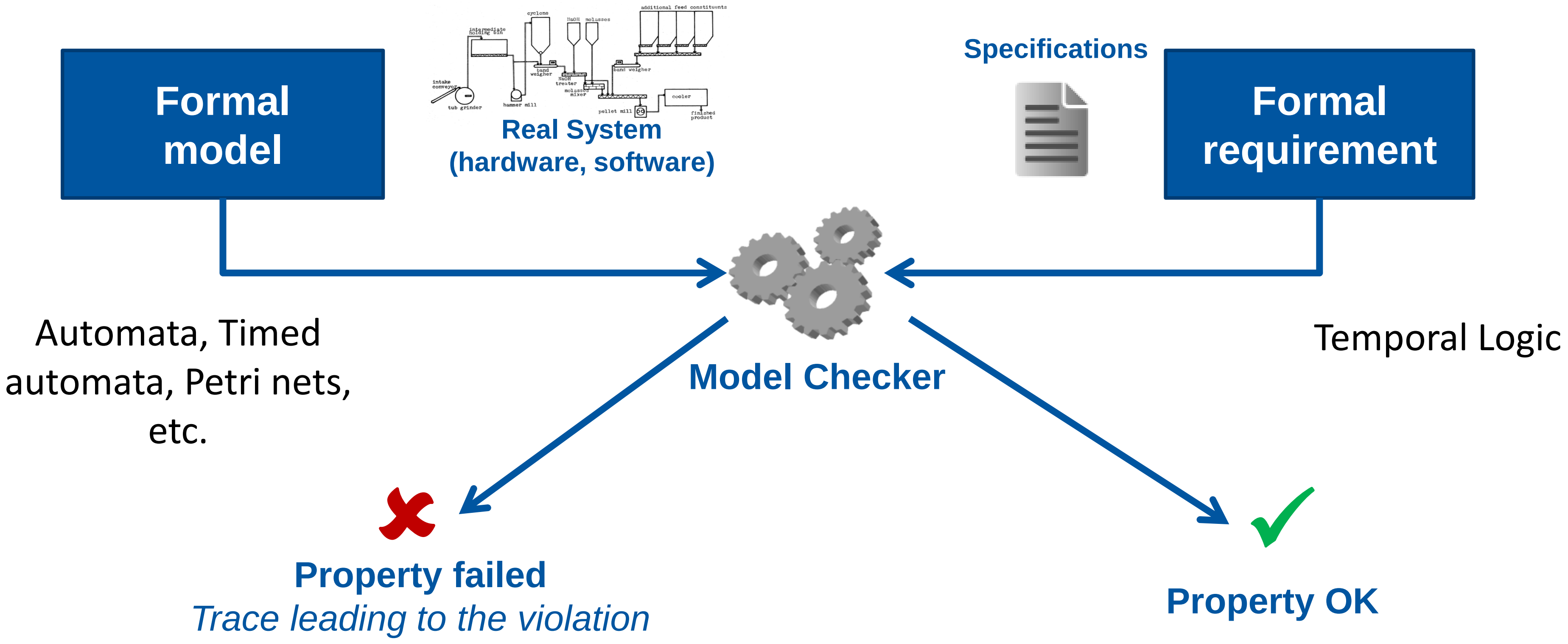
- ❑ **Formal verification:** it uses formal methods to check a formal property on a **model of the system (**e.g. **model checking)**

In industry (including CERN) **manual, automated testing or simulation** techniques are the most popular approaches

Introduction to model checking

Given a **global model** of the system and a **formal property**, the **model checking algorithm checks exhaustively** that the model meets the property

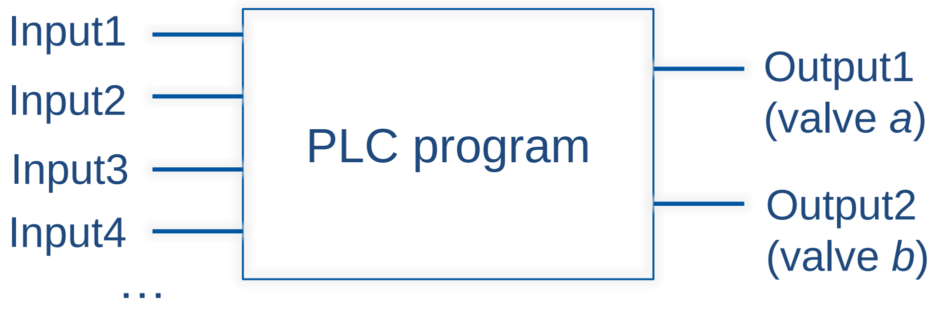
Clarke and Emerson (1982) and Queille and Sifakis (1982)



Formal model
Real System (hardware, software)
Specifications
Formal requirement
Automata, Timed automata, Petri nets, etc.
Temporal Logic
Model Checker
Property failed
Trace leading to the violation
Property OK

Model checking vs Testing

Model checking tools: nuXmv, UPPAAL, CBMC, SPIN, KRONOS, etc.



Requirement 1 (Functionality)
*If **Input1** is FALSE*
*then **Output2** is FALSE*

Requirement 2 (Safety)
*If **Output1** is FALSE*
*then **Output2** is TRUE*

4 Boolean input variables -> **2^4 = 16 combinations**
4 Word (16-bit) input variables -> **$2^{16*4} \approx 1.8*10^{19}$ combinations**

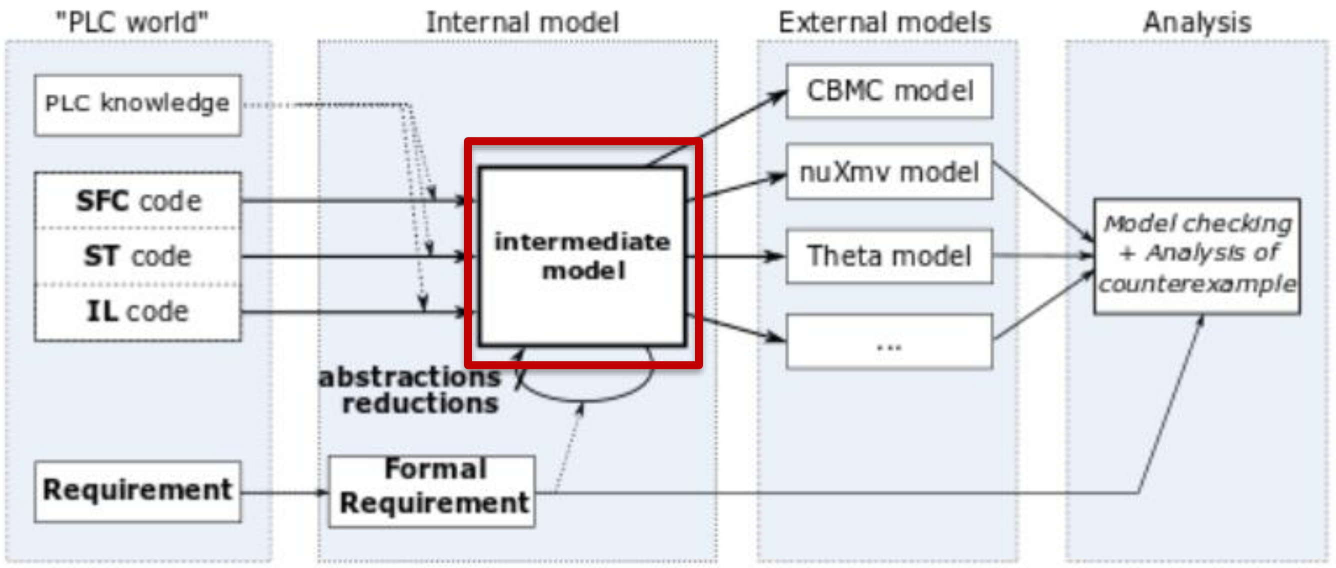
for large systems (many input var.), Requirement 2 cannot (**practically**) be solved **by using testing techniques**

Why model checking is not widely used?

Pros	Cons
Checks exhaustively all combinations	*We have to create the **model** of the system*
	*We have to use **temporal logic** (requirements)*
	State space explosion

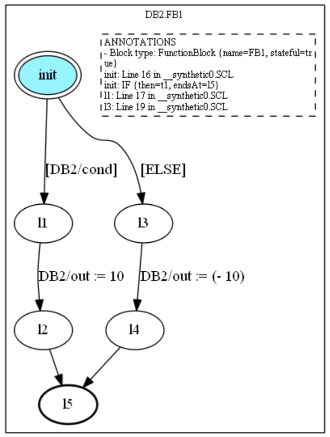
- **Modelling**: find the appropriate formalism and the right level of abstraction
- **Temporal logic**: hard to use
- **State space explosion**: there is a limitation on the number of combinations to check

Methodology overview



- **General methodology**
  - Multiple PLC languages
  - Multiple verification tools
  - Counterexample analysis

Control Flow Automata


DB2.FB1
ANNOTATIONS
- Block type: FunctionBlock {name=FB1, stateful=true}
init: Line 16 in __synthetic0.SCL
init: IF {then=l1, endsAt=l5}
l1: Line 17 in __synthetic0.SCL
l3: Line 19 in __synthetic0.SCL
init
[DB2/cond]
[ELSE]
l1
l3
DB2/out := 10
DB2/out := (- 10)
l2
l4
l5

More details: *MOBPP01 presentation - PLCverif Re-engineered: An Open Platform for the Formal Analysis of PLC Programs*

Is it worth to use Model Checking to PLCs?

- Not without tool support
- It is (very) hard to **create models** out of PLC programs
- It is (very) hard to **formalize the requirements** in temporal logic
- At CERN, we have developed the tool **PLCverif** **http://cern.ch/plcverif**
  - **Hide the complexity of using formal methods** from the user
  - The methodology shall be compatible with any development process of PLC programs
  - Technologies: Java, Xtext, EMF, …

PLCverif demo

Formal methods applied to Safety critical PLC programs (SM18 ClusterG)

- Test benches for superconducting magnets (**SM18**, FAIR, B311)

- **Risks** to personnel and equipment
  - Cryogenics
  - Vacuum
  - Power converters
  - Cooling & ventilation

- Need for **Safety Instrumented Systems**

  (IEC 61511 standard)

Running PLCverif

- The fail-safe PLC program **(STL version)** was imported to **PLCverif**
  - 2000 lines of STL exported code
  - 240 input Boolean variables

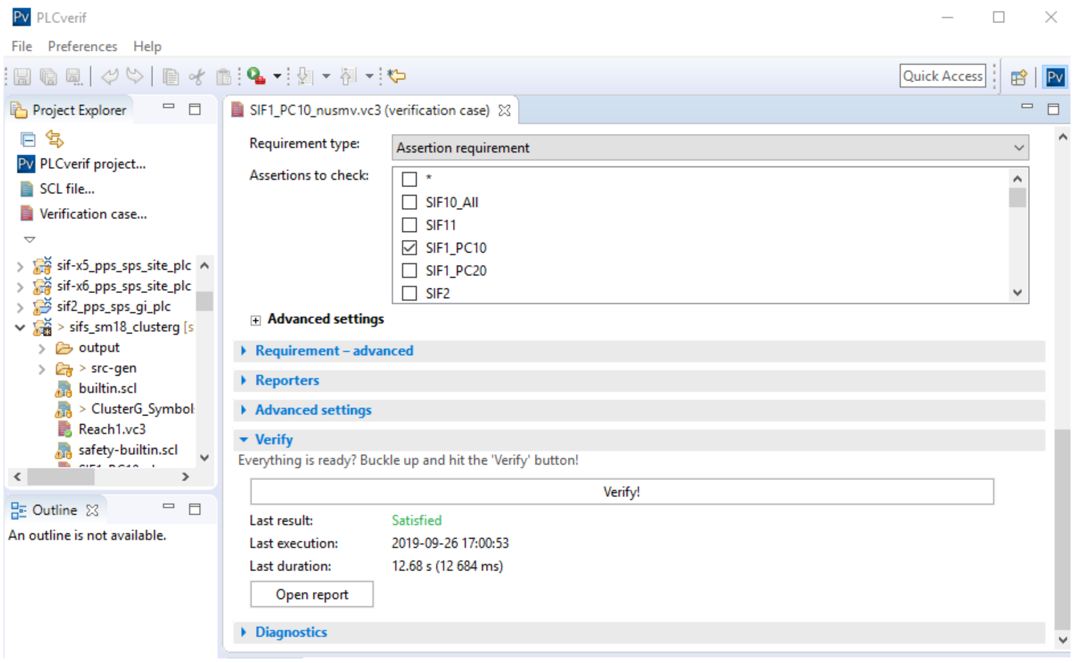
- We formalized around 30 verification cases (SIFs)

Reference	**SIF1_PC10**
Related risk	Risk analysis references 1 and 2
Functionality	Shutdown the PC (10KA) if the corresponding temperature of the cable is high (thermoswitch signal = 0) or the water flow is low (switch signal = 0)
Formalized functionality	**If** (**NOT** MTBG_WCCF_HFM_BIS_P **OR** **NOT** MTBG_WCCF_HFM_BIS_M **OR** **NOT** MTBG_WCCT_PC10_EE10 **OR** **NOT** MTBG_WCCT_EE10_INV **OR** **NOT** MTBG_WCCT_INV_HFM) **Then** (MTBG_PC10_PERMIT_Q = FALSE)
Safety level	SIL1
Safety mode	Low demand

```
//#ASSERT(
    (
        NOT MTBG_WCCF_HFM_BIS_P OR
        NOT MTBG_WCCF_HFM_BIS_M OR
        NOT MTBG_WCCT_PC10_EE10 OR
        NOT MTBG_WCCT_EE10_INV OR
        NOT MTBG_WCCT_INV_HFM
    )
    --> (MTBG_PC10_PERMIT_Q = FALSE)
): SIF1_PC10;
```

PLCverif results

- Verification average time of 1 – 2 minutes per assertion
- **Several discrepancies between the specification and the PLC program** were found:
  - Problems in the specification
  - Bugs in the PLC programs



Requirement type: Assertion requirement

Assertions to check:
- [ ] *
- [ ] SIF10_All
- [ ] SIF11
- [✓] SIF1_PC10
- [ ] SIF1_PC20
- [ ] SIF2

Advanced settings

Everything is ready? Buckle up and hit the 'Verify' button!

Last result:	Satisfied
Last execution:	2019-09-26 17:00:53
Last duration:	12.68 s (12 684 ms)

PLCverif demo

Conclusions

- *It is worth to use formal methods* ***for critical systems****, but it comes with a (big) cost …*
- *They can be applied to* ***specification****, code verification, simulation, etc.*
- *We have found several (many?)* ***PLC programs bugs or specification problems*** *by using PLCverif (model checking)*
  - *… even in well-tested production systems*
- *If you want to apply* ***model checking to PLC programs****, consider* ***PLCverif***