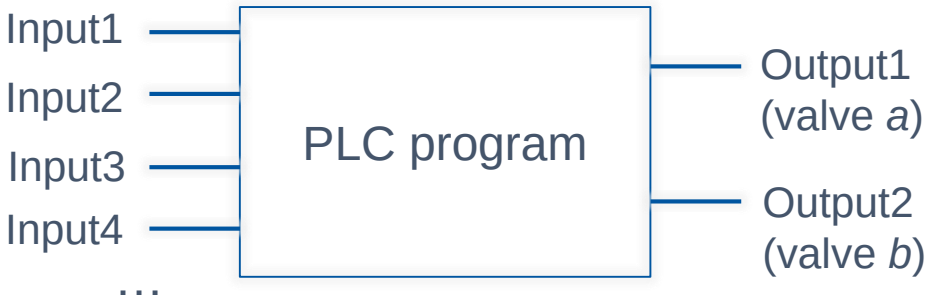
Applying Model Checking to PLC programs (PLCverif)

Borja Fernández Adiego

Context

Our goal is to be sure that the **PLC program is compliant with the specifications** (requirements)



<u>Functional Requirement</u> (Safety)
"When Output1 is true, Output2 should never be false"
*If **Output1** is TRUE*
*then **Output2** is TRUE*

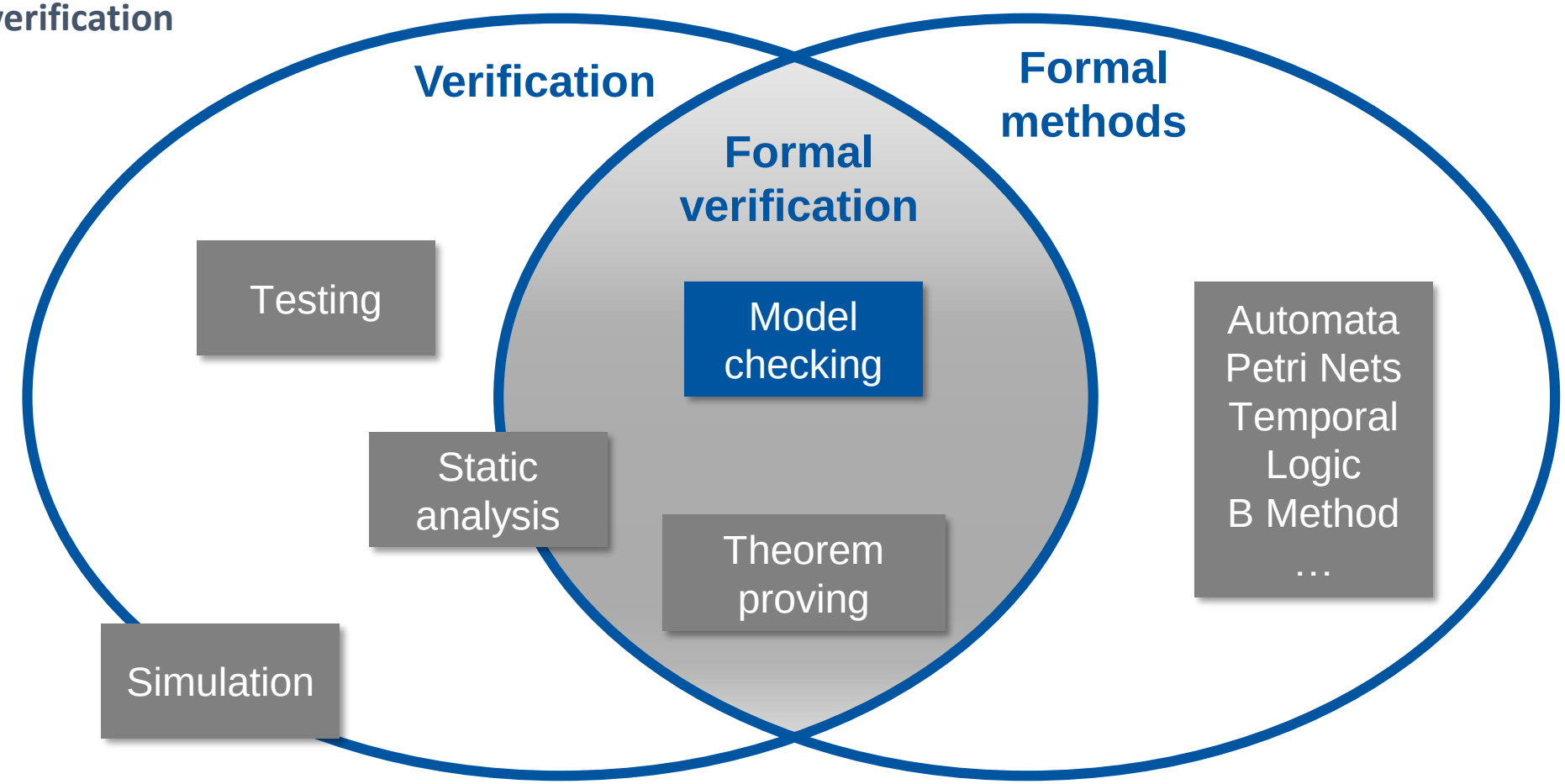
- If "*Input1*", "*Input2*", "*Input3*" and "*Input4*" are **BOOL**, then we need to check **2^4 = 16 combinations**
- If they are **INT** (16-bit), then **$2^{16*4} \approx 1.8*10^{19}$ combinations**

for large systems (many variables), such requirements **cannot** (practically) **be checked by using testing techniques**

Formal methods, formal verification and model checking

Formal methods are techniques based on **mathematics** and **formal logic** (e.g. Petri Nets, Temporal Logic, Automata, etc.)

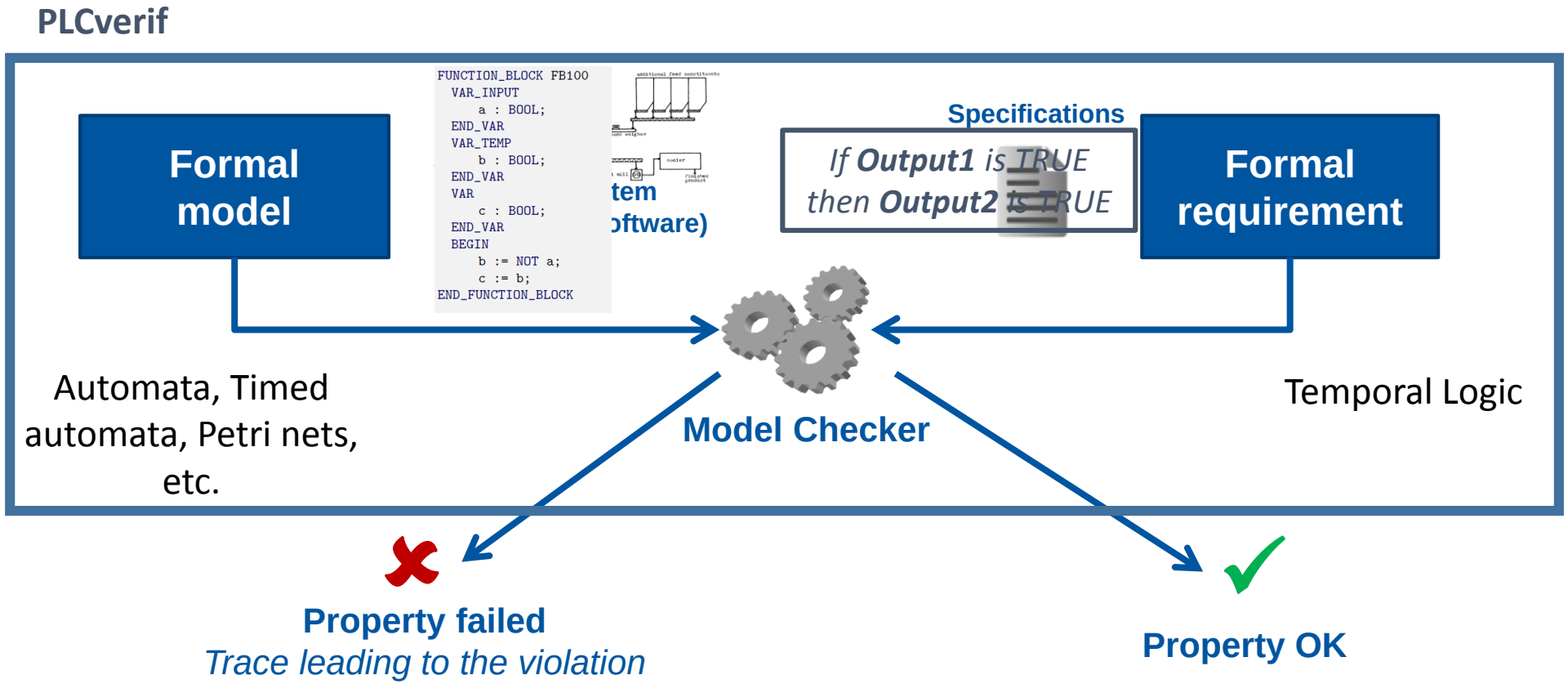
Formal verification



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 methods and the functional safety standards

IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems

Table A.1 – Software safety requirements specification

(See 7.2)

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1a	Semi-formal methods	Table B.7	R	R	HR	HR
1b	Formal methods	B.2.2, C.2.4	---	R	R	HR
2	Forward traceability between the system safety requirements and the software safety requirements	C.2.11	R	R	HR	HR
3	Backward traceability between the safety requirements and the perceived safety needs	C.2.11	R	R	HR	HR
4	Computer-aided specification tools to support appropriate techniques/measures above	B.2.4	R	R	HR	HR

Table A.5 – Software design and development – software module testing and integration

(See 7.4.7 and 7.4.8)

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Probabilistic testing	C.5.1	---	R	R	R
2	Dynamic analysis and testing	B.6.5 Table B.2	R	HR	HR	HR
3	Data recording and analysis	C.5.2	HR	HR	HR	HR
4	Functional and black box testing	B.5.1 B.5.2 Table B.3	HR	HR	HR	HR
5	Performance testing	Table B.6	R	R	HR	HR
6	Model based testing	C.5.27	R	R	HR	HR
7	Interface testing	C.5.3	R	R	HR	HR
8	Test management and automation tools	C.4.7	R	HR	HR	HR
9	Forward traceability between the software design specification and the module and integration test specifications	C.2.11	R	R	HR	HR
10	Formal verification	C.5.12	---	---	R	R

IEC 61511: Functional safety – Safety instrumented systems for the process industry sector

- several references to model checking. For example from IEC 61511-2:2016 Annex B:

*"… specification should be implemented in the graphical language of the **model checking** workbench environment…"*

PLCverif Demo

more details: http://cern.ch/plcverif

source code: https://gitlab.com/plcverif-oss

Conclusions

Partially hidden by PLCverif

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
- **Requirements formalization (e.g. temporal logic)**: hard to use
- **State space explosion**: there is a limitation on the number of combinations to check