Bringing Automated Formal Verification to PLC Program Development

PhD Thesis
Submitted by Borja Fernández Adiego

Supervised by
Dr. Víctor Manuel González Suárez (University of Oviedo)
Dr. Enrique Blanco Viñuela (CERN)

19/12/2014
Context
Context

- Collaboration between the University of Oviedo and CERN

- **University of Oviedo**
  - Department of Electrical Engineering, Electronics, Computers, and Systems (DIEECS)

- **CERN** (European Organization for Nuclear Research)
  - ICE (Industrial Controls Engineering) group
Context
Context

CERN
Context

CERN

- Particle accelerators
Context

CERN

- Particle accelerators
Context

CERN

- Particle accelerators
- Particle detectors
Context

**CERN**

- Particle accelerators
- Particle detectors
Context

CERN

- Particle accelerators
- Particle detectors
- Facilities
  - Vacuum
Context

CERN

- Particle accelerators
- Particle detectors
- Facilities
  - Vacuum
  - Cryogenics
Context

CERN

- Particle accelerators
- Particle detectors
- Facilities
  - Vacuum
  - Cryogenics
  - C&V
  - Etc.
Motivation
Motivation

- In the process industry the most popular control device is the **PLC** (Programmable Logic Controller)
Motivation

- In the process industry the most popular control device is the **PLC** (Programmable Logic Controller)

- More 1000 PLC applications are developed and maintained at CERN
Motivation

- In the process industry the most popular control device is the **PLC** *(Programmable Logic Controller)*

- More **1000 PLC applications** are developed and maintained **at CERN**

  **Goal:**

  Increasing the **reliability** of these control systems
Motivation

- In the process industry the most popular control device is the **PLC** (Programmable Logic Controller)

- More **1000 PLC applications** are developed and maintained **at CERN**

  **Goal:**
  
  Increasing the **reliability** of these control systems

- To assess the reliability it is required to analyze:
  
  - Hardware (Markov chain analysis, probabilistic calculations, IEC 61508 provides some guidelines, etc.)
  
  - **Software (Testing, formal verification)**
Motivation

- In the process industry the most popular control device is the PLC (Programmable Logic Controller)

- More 1000 PLC applications are developed and maintained at CERN
  Goal: Increasing the reliability of these control systems

- To assess the reliability it is required to analyze:
  - Hardware (Markov chain analysis, probabilistic calculations, IEC 61508 provides some guidelines, etc.)
  - Software (Testing, formal verification)

- This thesis focuses on the software reliability
Guarantee that the PLC programs are compliant with the specifications
Motivation

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

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

- **Testing**: it checks of certain properties or test cases in the *real* system
Motivation

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

- **Testing**: it checks of certain properties or test cases in the **real system**

- **Formal verification**: it uses formal methods to check a formal property on a **model of the system**
Motivation

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

- **Testing**: it checks of certain properties or test cases in the real system

- **Formal verification**: it uses formal methods to check a formal property on a model of the system

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

- However, testing techniques have some limitations:
However, testing techniques have some **limitations**:

- Checking some **safety properties** (e.g. “if the valve \(a\) is closed, then the valve \(b\) **can never be closed** at the same time”)
However, testing techniques have some limitations:

- Checking some safety properties (e.g. “if the valve $a$ is closed, then the valve $b$ can never be closed at the same time”)

The IEC 61508 recommends the use of formal methods for Safety Instrumented Systems.
## Background

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

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.

---

PhD defense: Borja Fernández Adiego

Introduction 7
## Background

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

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.
Motivation
Motivation

- Formal methods in other industries
  - Space (*Brat et al. 2004*)
  - Aircraft (*Meenakshi et al. 2007*)
  - Subway (*James et al. 2010*)
  - Etc.

- Why formal methods (and formal verification) are not extensively applied to **industrial control systems**? (particularly to **PLC programs**)

PhD defense: Borja Fernández Adiego
Motivation

- Formal methods in other industries
  - Space (*Brat et al. 2004*)
  - Aircraft (*Meenakshi et al. 2007*)
  - Subway (*James et al. 2010*)
  - Etc.

- Why formal methods (and formal verification) are not extensively applied to industrial control systems? (particularly to PLC programs)
  - Difficulty of building formal models of real-life PLC programs
Motivation

- Formal methods in other industries
  - Space (Brat et al. 2004)
  - Aircraft (Meenakshi et al. 2007)
  - Subway (James et al. 2010)
  - Etc.

- Why formal methods (and formal verification) are not extensively applied to industrial control systems? (particularly to PLC programs)
  - Difficulty of building formal models of real-life PLC programs
  - Difficulty of writing the formal specification using the required formalism for formal verification purposes
Motivation

- Formal methods in other industries
  - Space \textit{(Brat et al. 2004)}
  - Aircraft \textit{(Meenakshi et al. 2007)}
  - Subway \textit{(James et al. 2010)}
  - Etc.

- Why formal methods (and formal verification) are not extensively applied to \textit{industrial control systems}? (particularly to \textit{PLC programs})
  - Difficulty of \textbf{building formal models} of real-life PLC programs
  - Difficulty of \textbf{writing the formal specification} using the required formalism for formal verification purposes
  - \textbf{Computational limitations} (e.g. state explosion problem in model checking)
Motivation

- Formal methods in other industries
  - Space (*Brat et al. 2004*)
  - Aircraft (*Meenakshi et al. 2007*)
  - Subway (*James et al. 2010*)
  - Etc.

- Why formal methods (and formal verification) are not extensively applied to **industrial control systems**? (particularly to **PLC programs**)?
  - Difficulty of **building formal models** of real-life PLC programs
  - Difficulty of **writing the formal specification** using the required formalism for formal verification purposes
  - **Computational limitations** (e.g. state explosion problem in model checking)
  - In critical systems, the programming languages have many **restrictions** trying to reduce the number of potential bugs on the software
Goals

Bringing together the industrial automation and formal methods communities
Goals

Bringing together the industrial automation and formal methods communities

more specifically

Provide a methodology to apply automated formal verification techniques to PLC programs
Goals

Bringing together the industrial automation and formal methods communities

more specifically

Provide a methodology to apply automated formal verification techniques to PLC programs

Requirements:

- **Hide any complexity** related to formal methods from control engineers
- **Verification of new and existing PLC programs**
- The methodology shall be **compatible with any development process of PLC programs**
Outline
Outline

- Background
- Approach (contributions)
- Case studies
- Summary and Conclusions
Background
Background

- Industrial automation community
  (PLC – based control systems)
- Formal methods community
  (Model checking)
Background

- Industrial automation community
  (PLC – based control systems)
- Formal methods community
  (Model checking)
PLC execution platform
PLC execution platform

Sensors

PI

CPU

main program

Actuators

OIM

PhD defense: Borja Fernández Adiego
Scan Cycle
Scan Cycle

1. **Reading** the actual values from periphery to the Input Image Memory
Scan Cycle

1. **Reading** the actual values from periphery to the Input Image Memory
2. **Interpreting and executing** the PLC program
### Scan Cycle

1. **Reading** the actual values from periphery to the Input Image Memory
2. **Interpreting and executing** the PLC program
3. **Writing** the computed values from the Output Image Memory to the periphery
PLC software
PLC software

- IEC 61131-3 standard defines the 5 PLC programming languages:
  - **ST** (Structured Text), **IL** (Instruction List), **FBD** (Function Block Diagram), Ladder and **SFC** (Sequential Function Chart)
PLC software

- IEC 61131-3 standard defines the **5 PLC programming languages:**
  - **ST** (Structured Text), **IL** (Instruction List), FBD (Function Block Diagram), Ladder and **SFC** (Sequential Function Chart)
  - Different PLC vendors have different implementations
PLC software

FUNCTION_BLOCK FB100
  VAR_INPUT
    a : BOOL;
  END_VAR
  VAR_TEMP
    b : BOOL;
  END_VAR
  VAR
    c : BOOL;
  END_VAR
  BEGIN
    b := NOT a;
    c := b;
  END_FUNCTION_BLOCK

DATA_BLOCK DB1 FB100
  BEGIN
  END_DATA_BLOCK

ORGANIZATION_BLOCK OB1
  VAR_TEMP
    info : ARRAY[0..19] OF BYTE;
  END_VAR
  BEGIN
    FB100.DB1(a := FALSE);
    Q1.0 := DB1.c;
  END_ORGANIZATION_BLOCK
Background

- Industrial automation community
  (PLC – based control systems)
- Formal methods community
  (Model checking)
Background

- Industrial automation community
  (PLC – based control systems)
- Formal methods community
  (Model checking)
Formal verification
Formal verification

- It is the act of checking the **correctness** of a system with respect to a **formal property or specification** by using **formal methods** (**techniques from logic and discrete mathematics**)
Formal verification

- It is the act of checking the **correctness** of a system with respect to a **formal property or specification** by using **formal methods** (**techniques from logic and discrete mathematics**)

- Two main groups:
  - **Axiomatic** verification: based on the use of rules formalizing the semantics of the language. **Theorem proving**
    - Driven by skilled researchers and engineers
Formal verification

- It is the act of checking the **correctness** of a system with respect to a **formal property or specification** by using **formal methods** (*techniques from logic and discrete mathematics*)

- Two main groups:
  - **Axiomatic** verification: based on the use of rules formalizing the semantics of the language. **Theorem proving**
    - Driven by skilled researchers and engineers
  - **Algorithmic** verification: semi-algorithms to check that a global model, which represents the system, meets the given formal requirements. **Model checking**
Formal verification

- It is the act of checking the **correctness** of a system with respect to a **formal property or specification** by using **formal methods** (*techniques from logic and discrete mathematics*).

- Two main groups:
  - **Axiomatic** verification: based on the use of rules formalizing the semantics of the language. **Theorem proving**
    - Driven by skilled researchers and engineers
  - **Algorithmic** verification: semi-algorithms to check that a global model, which represents the system, meets the given formal requirements. **Model checking**

**Remark:**

*Safety*: Safety Instrumented Systems vs Safety property $AG(\alpha \to \beta)$
Model checking
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)
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)
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)
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)
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)
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)

PhD defense: Borja Fernández Adiego
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)
State of the art
State of the art

- Axiomatic verification to PLC programs (e.g. Blech and Ould Biha 2011, Mader et al. 2010, Sadolewski 2011)
State of the art

- Axiomatic verification to PLC programs (e.g. Blech and Ould Biha 2011, Mader et al. 2010, Sadolewski 2011)
- Algorithmic verification
State of the art

- Axiomatic verification to PLC programs (e.g. Blech and Ould Biha 2011, Mader et al. 2010, Sadolewski 2011)
- Algorithmic verification

<table>
<thead>
<tr>
<th>Reference</th>
<th>Input lang.</th>
<th>Verifier</th>
<th>Req. language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bauer et al. (2004)</td>
<td>SFC</td>
<td>Cadence SMV</td>
<td>CTL</td>
</tr>
<tr>
<td>Sarmento et al. (2008)</td>
<td>SFC</td>
<td>UPPAAL</td>
<td>CTL subset</td>
</tr>
<tr>
<td>Bauer et al. (2004)</td>
<td>timed SFC</td>
<td>UPPAAL</td>
<td>CTL subset</td>
</tr>
<tr>
<td>Blech et al. (2011)</td>
<td>SFC, FBD</td>
<td>BIP</td>
<td>—</td>
</tr>
<tr>
<td>Bartha et al. (2012)</td>
<td>FBD</td>
<td>PetriDotNet</td>
<td>CTL</td>
</tr>
<tr>
<td>Canet et al. (2000b)</td>
<td>IL</td>
<td>Cadence SMV</td>
<td>LTL</td>
</tr>
<tr>
<td>Mader and Wupper (1999)</td>
<td>IL</td>
<td>UPPAAL</td>
<td>CTL subset</td>
</tr>
<tr>
<td>Gourcuff et al. (2006)</td>
<td>ST</td>
<td>NuSMV</td>
<td>—</td>
</tr>
</tbody>
</table>

PLC languages and constrains
State of the art

- Axiomatic verification to PLC programs (e.g. Blech and Ould Biha 2011, Mader et al. 2010, Sadolewski 2011)
- Algorithmic verification

<table>
<thead>
<tr>
<th>Reference</th>
<th>Abstraction techniques</th>
<th>PLC language</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gourcuff et al. (2008)</td>
<td>data and interpretation abstraction</td>
<td>IL</td>
<td>only Boolean var.</td>
</tr>
<tr>
<td>Lange et al. (2013)</td>
<td>constant folding, slicing, and forward expression propagation</td>
<td>IL</td>
<td>only Safety properties</td>
</tr>
<tr>
<td>Biallas et al. (2010)</td>
<td>CEGAR</td>
<td>IL</td>
<td>only Safety properties</td>
</tr>
<tr>
<td>Biallas et al. (2013)</td>
<td>Predicate abstraction</td>
<td>ST</td>
<td>only Safety properties</td>
</tr>
</tbody>
</table>
State of the art

- Axiomatic verification to PLC programs (e.g. Blech and Ould Biha 2011, Mader et al. 2010, Sadolewski 2011)
- Algorithmic verification

<table>
<thead>
<tr>
<th>Reference</th>
<th>Formalism</th>
<th>Ver. tool</th>
<th>Time model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perin and Faure (2013)</td>
<td>Timed automaton</td>
<td>–</td>
<td>monotonic representation</td>
</tr>
<tr>
<td>Mokadem et al. (2010)</td>
<td>Timed automaton</td>
<td>UPPAAL</td>
<td>monotonic representation</td>
</tr>
<tr>
<td>Wang et al. (2013)</td>
<td>BIP language</td>
<td>–</td>
<td>monotonic representation</td>
</tr>
</tbody>
</table>

Time modeling
Proposed approach

- Methodology overview
- Methodology steps (contributions)
Proposed approach

- Methodology overview
- Methodology steps (contributions)
Methodology overview
Methodology overview

- It is meant to be **integrated in any development process** of PLC programs
Methodology overview

- It is meant to be **integrated in any development process** of PLC programs.
- Creates automatically formal models **out of the PLC code**, as the PLC code is the common element of any PLC program development.
Methodology overview
Methodology overview

"PLC world"

PLC knowledge

SFC code

ST code

IL code

Internal model

Intermediate model

Formal Requirement

abstractions / reductions

External models

BIP model

nuXmv model

UPPAAL model

Model checking + Analysis of counterexample

Analysis


PhD defense: Borja Fernández Adiego
Methodology overview
Methodology overview

- General
  - Multiple PLC languages
  - Multiple verification tools
Methodology overview

- **General**
  - Multiple PLC languages
  - Multiple verification tools

- **Fully automated**
  - Automatic model transformations and abstraction techniques
  - Counterexample analysis

PhD defense: Borja Fernández Adiego
Intermediate model
Intermediate model

IM definition and example
Intermediate model

IM definition and example
Intermediate model

IM definition and example

\[ N = (A, \hat{I}) \]
Intermediate model

IM definition and example

\[ N = (A, \dot{I}) \]
\[ a = (L, T, l_0, V_a, Val_0) \in A \]
\[ L = \{l_0, l_1, \ldots\} \]
\[ V_a = \{v_1, \ldots, v_m\} \]
\[ Val_0 = (Val_{1,0}, \ldots, Val_{m,0}) \]
\[ t = (l, g, amt, i, l') \in T \]
Intermediate model

IM definition and example

\[ N = (A, I) \]
\[ a = (L, T, l_0, V_a, Val_0) \in A \]
\[ L = \{l_0, l_1, \ldots \} \]
\[ V_a = \{v_1, \ldots, v_m\} \]
\[ Val_0 = (Val_{1,0}, \ldots, Val_{m,0}) \]
\[ t = (l, g, amt, i, l') \in T \]
\[ i = (t, t') \in I \]
Proposed approach
Proposed approach

- Methodology overview
- Methodology steps (contributions)
1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis

- Methodology overview
- Methodology steps
  (contributions)
Proposed approach

1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis

- Methodology overview
- Methodology steps
  (contributions)
Requirements formalization
Requirements formalization

- Solution based on **patterns (natural language and well-defined semantics)**
Requirements formalization

- Solution based on patterns (natural language and well-defined semantics)
- Safety and liveness properties (Involving one or several PLC cycles and “time-related”):
Requirements formalization

- Solution based on patterns (natural language and well-defined semantics)
- Safety and liveness properties (Involving one or several PLC cycles and “time-related”):
  - Pattern TL1. **General truth under condition:** If ____[2] is true (at the end of the PLC cycle), ____[1] is always true (at the end of the PLC cycle).

\[ AG\left((EoC \land [2]) \rightarrow [1]\right) \]
1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis

- Methodology overview
- Methodology steps (contributions)
1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis

Approach

- Methodology overview
- Methodology steps (contributions)
PLC execution platform modeling
PLC execution platform modeling

"PLC world"
- PLC knowledge
  - SFC code
  - ST code
  - IL code
- Requirement

Internal model
- Intermediate model
  - Formal Requirement

External models
- BIP model
- nuXmv model
- UPPAAL model
- ...

Analysis
- Model checking
- Analysis of counterexample
PLC execution platform modeling
Assumption 1. Currently, only centralized PLC control systems consisting in one single PLC are considered.

Assumption 2. Other hardware devices, such as input and output cards, communication interfaces, field buses or any kind of communication with the SCADA, are not modelled.
PLC execution platform modeling
PLC execution platform modeling

- PLC execution platform:

```
parameters

init
  initialization of inputs
  $l_1$

  $[\neg ia > 0]$
  $xa := F$

  $[ia > 0]$
  $xa := T$

  $l_2$

  $[ib > 0]$
  $xb := T$

  $[\neg ib > 0]$
  $xb := F$

  $l_3$

  $c := c + 1$

end
```
PLC execution platform modeling

- PLC execution platform:
  - **Cyclic execution of the IM**, representing the PLC scan cycle
PLC execution platform modeling

- PLC execution platform:
  - Cyclic execution of the IM, representing the PLC scan cycle
  - Initialization of the variables of the IM in the first location
    - Parameters
PLC execution platform modeling

- PLC execution platform:
  - **Cyclic execution of the IM**, representing the PLC scan cycle
  - **Initialization of the variables** of the IM in the first location
    - Parameters
  - **Non-deterministic values are assigned to the PLC input variables in the first transition**
PLC execution platform modeling

- PLC execution platform:
  - **Cyclic execution of the IM**, representing the PLC scan cycle
  - **Initialization of the variables** of the IM in the first location
    - Parameters
  - **Non-deterministic values are assigned to the PLC input variables in the first transition**
  - The rest of the model represents the execution of the PLC program
PLC execution platform modeling

- PLC execution platform:
  - **Cyclic execution of the IM**, representing the PLC scan cycle
  - **Initialization of the variables** of the IM in the first location
    - Parameters
  - **Non-deterministic values are assigned to the PLC input variables in the first transition**
  - The rest of the model represents the execution of the PLC program
  - The location **end** represents the moment when the values are written from the OIM to the output periphery
Approach

1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis

- Methodology overview
- Methodology steps
  (contributions)
1. Requirements formalization
2. PLC execution platform modeling
3. **PLC code – IM transformation**
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis

- Methodology overview
- Methodology steps (contributions)
PLC code – IM transformations

Approach
PLC code – IM transformations

The PLC code corresponds to the Siemens SCL (ST from the IEC 61131-3) and Graph (SFC from the IEC 61131-3) implementation.
PLC – IM transformations
PLC – IM transformations

- Multiple concurrent code blocks
- Functions (FCs)
- Function Block (FBs) instances
- Variables
- Fc or FB call
- ST statement
- Variable assignment
- Conditional statement
- Building blocks (Timers)
PLC – IM transformations

- Multiple concurrent code blocks
- Functions (FCs)
- Function Block (FBs) instances
- Variables
- Fc or FB call
- ST statement
- Variable assignment
- Conditional statement
- Building blocks (Timers)
PLC – IM transformations

<table>
<thead>
<tr>
<th>Rule</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC or FB call</td>
<td>All the input variables are assigned in the caller, and all the output variables are assigned in the callee</td>
</tr>
</tbody>
</table>
### PLC – IM transformations

<table>
<thead>
<tr>
<th>Rule</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC or FB call</td>
<td>All the input variables are assigned in the caller, and all the output variables are assigned in the callee</td>
</tr>
</tbody>
</table>

**Diagram:**
- **Caller Automaton**
  - States: $l_1, l_2, l_3$
  - Transitions: $t_1, t_2$
  - Output: $i_1!$
  - Input: $i_2?$

- **Called Automaton**
  - States: init, end
  - Transitions: $i_1?$, $i_2!$
PLC – IM transformations

<table>
<thead>
<tr>
<th>Rule</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional Statement</td>
<td>--</td>
</tr>
</tbody>
</table>
PLC – IM transformations

<table>
<thead>
<tr>
<th>Rule</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional Statement</td>
<td>--</td>
</tr>
</tbody>
</table>

IF c THEN
sl1;
ELSE
sl2;
END_IF;

Diagram:
- Node $l_1$ with transitions $t_1$ and $t_2$.
- Transition $t_1$ with condition $[c]$.
- Transition $t_2$ with condition $[-c]$.
PLC – IM transformations
PLC – IM transformations

**PLC example**

ORGANIZATION_BLOCK OB1
  
  VAR_TEMP
  a2 : BOOL;
  a3 : BOOL;

  END_VAR

  BEGIN
  IF a2 THEN
      FB_B.DB1(b1:=NOT a2);
  END_IF;
  a2 := NOT I0.0;
  a3 := DB1.b2;

  END_FUNCTION_BLOCK

FUNCTION_BLOCK FB_B
  
  VAR_INPUT
  b1 : BOOL;

  END_VAR

  VAR_OUTPUT
  b2 : BOOL;

  END_VAR

  BEGIN
  b2 := NOT b1;

  END_FUNCTION_BLOCK

DATA_BLOCK DB1 FB_B
  
  BEGIN

  END_DATA_BLOCK
PLC – IM transformations

**PLC example**

```plaintext
ORGANIZATION_BLOCK OB1
  VAR_TEMP
    a2 : BOOL;
    a3 : BOOL;
  END_VAR
BEGIN
  IF a2 THEN
    FB_B.DB1(b1:=NOT a2);
  END_IF;
  a2 := NOT I0.0;
  a3 := DB1.b2;
END_FUNCTION_BLOCK

FUNCTION_BLOCK FB_B
  VAR_INPUT
    b1 : BOOL;
  END_VAR
  VAR_OUTPUT
    b2 : BOOL;
  END_VAR
BEGIN
  b2 := NOT b1;
END_FUNCTION_BLOCK

DATA_BLOCK DB1 FB_B
BEGIN
END_DATA_BLOCK
```

**Corresponding IM Model**

![IM Model Diagram](image)

- `mainBlock: OB1`
- `Variables:
  - a1 : BOOL
  - a2 : BOOL
  - I0.0 : BOOL`
- `init:
  - [a2]
  - [¬a2]`
- `l2: ¬a2
  DB1.b1 := ¬a2
  iDB1!
  l2_ret`
- `l3:
  - a2 := ¬I0.0
  l4:
  - a3 := DB1.b2`
- `DB1: FB_B
  Variables:
  - b1 : BOOL
  - b2 : BOOL`
- `init:
  - iDB1?
  l1:
  - b2 := ¬b1
  end`

PhD defense: Borja Fernández Adiego
PLC – IM transformations

**PLC example**

```
ORGANIZATION_BLOCK OB1
  VAR_TEMP
  a2 : BOOL;
a3 : BOOL;
END_VAR
BEGIN
  IF a2 THEN
    FB_B.DB1(b1:=NOT a2);
  END_IF;
a2 := NOT I0.0;
a3 := DB1.b2;
END_FUNCTION_BLOCK

FUNCTION_BLOCK FB_B
  VAR_INPUT
    b1 : BOOL;
  END_VAR
  VAR_OUTPUT
    b2 : BOOL;
  END_VAR
BEGIN
  b2 := NOT b1;
END_FUNCTION_BLOCK

DATA_BLOCK DB1 FB_B
BEGIN
END_DATA_BLOCK
```

**Corresponding IM Model**

```
mainBlock: OB1

Variables
  a1 : BOOL
  a2 : BOOL
  I0.0 : BOOL

init
  I0.0 := random

l2
  [a2]
  DB1.b1 := ¬a2
  iDB1!
  l2-ret
  iDB1_ret?

l3
  a2 := ¬I0.0

l4
  a3 := DB1.b2
  end

DB1: FB_B

Variables
  b1 : BOOL
  b2 : BOOL

init
  iDB1?
l1
  b2 := ¬b1
  end
```
PLC – IM transformations

**PLC example**

**Corresponding IM Model**

**ORGANIZATION_BLOCK OB1**

VAR_TEMP

- a2 : BOOL;
- a3 : BOOL;

END_VAR

BEGIN
  IF a2 THEN
    FB_B.DB1(b1:=NOT a2);
  END_IF;
  a2 := NOT I0.0;
  a3 := DB1.b2;
END_FUNCTION_BLOCK

**FUNCTION_BLOCK FB_B**

VAR_INPUT

- b1 : BOOL;

END_VAR

VAR_OUTPUT

- b2 : BOOL;

END_VAR

BEGIN
  b2 := NOT b1;
END_FUNCTION_BLOCK

**DATA_BLOCK DB1 FB_B**

BEGIN

END_DATA_BLOCK

PhD defense: Borja Fernández Adiego
PLC – IM transformations

**PLC example**

```
ORGANIZATION_BLOCK OB1
  VAR_TEMP
  a2 : BOOL;
  a3 : BOOL;
  END_VAR
  BEGIN
    IF a2 THEN
      FB_B.DB1(b1:=NOT a2);
    END_IF;
    a2 := NOT I0.0;
    a3 := DB1.b2;
  END_FUNCTION_BLOCK

FUNCTION_BLOCK FB_B
  VAR_INPUT
    b1 : BOOL;
  END_VAR
  VAR_OUTPUT
    b2 : BOOL;
  END_VAR
  BEGIN
    b2 := NOT b1;
  END_FUNCTION_BLOCK

DATA_BLOCK DB1 FB_B
  BEGIN
  END_DATA_BLOCK
```

**Corresponding IM Model**

```
mainBlock: OB1

Variables
a1 : BOOL
a2 : BOOL
I0.0 : BOOL

[i2]

[a2]  [-a2]

DB1.b1 := -a2
iDB1!

iDB1_ret!

iDB1_ret?

l2_ret

l3

a2 := -I0.0
l4

a3 := DB1.b2
end

DB1: FB_B

Variables
b1 : BOOL
b2 : BOOL
```

PhD defense: Borja Fernández Adiego
PLC – IM transformations

PLC example

ORGANIZATION_BLOCK OB1
  VAR_TEMP
    a2 : BOOL;
    a3 : BOOL;
  END_VAR
  BEGIN
    IF a2 THEN
      FB_B.DB1(b1:=NOT a2)
    END_IF;
    a2 := NOT I0.0;
    a3 := DB1.b2;
  END_FUNCTION_BLOCK

FUNCTION_BLOCK FB_B
  VAR_INPUT
    b1 : BOOL;
  END_VAR
  VAR_OUTPUT
    b2 : BOOL;
  END_VAR
  BEGIN
    b2 := NOT b1;
  END_FUNCTION_BLOCK

DATA_BLOCK DB1 FB_B
  BEGIN
  END_DATA_BLOCK

PhD defense: Borja Fernández Adiego
PLC – IM transformations

PLC software timers
PLC – IM transformations

*PLC software timers*

- Two modeling approaches:
PLC – IM transformations

**PLC software timers**

- Two modeling approaches:
  - **Realistic approach**: to verify time–related properties with explicit time in it (e.g. “if $C_1$ is true, after 3 seconds $C_2$ will be true”)

---

PhD defense: Borja Fernández Adiego
PLC – IM transformations

**PLC software timers**

- Two modeling approaches:
  - **Realistic approach**: to verify time-related properties with explicit time in it (e.g. “if C1 is true, after 3 seconds C2 will be true”)
    - Time is modeled as a finite variable
    - Time is incremented by adding the cycle time at the end of the PLC cycle
    - Timer Potential State Space is $5.91 \times 10^{15}$
Two modeling approaches:

- **Realistic approach**: to verify time-related properties with explicit time in it (e.g. “if $C_1$ is true, after 3 seconds $C_2$ will be true”)
  - Time is modeled as a finite variable
  - Time is incremented by adding the cycle time at the end of the PLC cycle
  - Timer Potential State Space is $5.91 \times 10^{15}$

- **Abstract approach**: to verify time properties without explicit time in it (e.g. “if $C_1$ is sometime true, eventually $C_2$ will be true”) or any “non-time-related” property with variables affected by a timer
Two modeling approaches:

- **Realistic approach**: to verify time-related properties with explicit time in it (e.g. “if $C_1$ is true, after 3 seconds $C_2$ will be true”)
  - Time is modeled as a finite variable
  - Time is incremented by adding the cycle time at the end of the PLC cycle
  - Timer Potential State Space is $5.91 \times 10^{15}$

- **Abstract approach**: to verify time properties without explicit time in it (e.g. “if $C_1$ is sometime true, eventually $C_2$ will be true”) or any “non-time-related” property with variables affected by a timer
  - Time is **not** modeled
  - Potential State Space is 6
Approach

- Methodology overview
- Methodology steps (contributions)

1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis
1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. **Reduction techniques**
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis

- Methodology overview
- **Methodology steps**
  (contributions)
Reduction and abstraction techniques

PhD defense: Borja Fernández Adiego
Reduction and abstraction techniques

- Real-life PLC program models have a huge state space
Real-life PLC program models have a **huge state space**.

Reduction and abstraction techniques **applied to the IM** because:

- All the verification tools included in the methodology **can benefit from them**
- **It is more effective.** A higher level model contains useful information
Reduction and abstraction techniques

- Real-life PLC program models have a **huge state space**
- Reduction and abstraction techniques **applied to the IM** because:
  - All the verification tools included in the methodology **can benefit** from them
  - It is more effective. A higher level model contains useful information
- Two families of reductions are included:
  - Property preserving reduction techniques
  - Variable abstraction

PhD defense: Borja Fernández Adiego
Property preserving reduction techniques
Property preserving reduction techniques

- For a given property, the original and the reduced models are equivalent (variable values at the end of PLC cycle)
Property preserving reduction techniques

- For a given property, the original and the reduced models are equivalent (variable values at the end of PLC cycle)

- Even for complex properties (several temporal operators)
Property preserving reduction techniques

- For a given property, the original and the reduced models are equivalent (variable values at the end of PLC cycle)

- Even for complex properties (several temporal operators)

- Two reduction techniques are included:
  - Customized Cone of Influence (COI)
  - General rule-based reductions
Cone of Influence
Cone of Influence

- It removes all variables from the model, that do not have any effect on the property to verify.
Cone of Influence

- It removes all variables from the model, that do not have any effect on the property to verify
- Some verifications tools (i.e. NuSMV) implement COI but it is not effective in our models
Cone of Influence

- It removes all variables from the model, that do not have any effect on the property to verify
- Some verifications tools (i.e. NuSMV) implement COI but it is not effective in our models
- But applying COI to the IM, we are more effective (contribution)
Cone of Influence

- It removes all variables from the model, that do not have any effect on the property to verify
- Some verifications tools (i.e. NuSMV) implement COI but it is not effective in our models
- But applying COI to the IM, we are more effective (contribution)

Property EF c > 10
Cone of Influence

- It removes all variables from the model, that do not have any effect on the property to verify
- Some verifications tools (i.e. NuSMV) implement COI but it is not effective in our models
- But applying COI to the IM, we are more effective (contribution)

Property $\text{EF } c > 10$
Cone of Influence

- It removes all variables from the model, that do not have any effect on the property to verify
- Some verifications tools (i.e. NuSMV) implement COI but it is not effective in our models
- But applying COI to the IM, we are more effective (contribution)

Property EF $c > 10$
General rule-based reductions
General rule-based reductions

<table>
<thead>
<tr>
<th>Rule</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model simplifications</td>
<td><img src="image1" alt="Model simplification diagram" /></td>
</tr>
<tr>
<td>Model reductions</td>
<td><img src="image2" alt="Model reduction diagram" /></td>
</tr>
<tr>
<td>Domain-specific reductions</td>
<td><img src="image3" alt="Domain-specific reduction diagram" /></td>
</tr>
</tbody>
</table>
Approach

- Methodology overview
- Methodology steps
  (contributions)

1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis
1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. **Reduction techniques**
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis

- Methodology overview
- Methodology steps (contributions)
Approach

- Property preserving reduction techniques
- Variable abstraction

Methodology overview

Methodology steps (contributions)
Approach

- Property preserving reduction techniques
- Variable abstraction

1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis
(Iterative) Variable abstraction
(Iterative) Variable abstraction

- It was designed to verify models where the property preserving reduction techniques are not effective
(Iterative) Variable abstraction

- It was designed to verify models where the property preserving reduction techniques are not effective

- Main idea: verify the property in an abstract model (over-approximation)
(Iterative) Variable abstraction

- It was designed to verify models where the property preserving reduction techniques are not effective

- Main idea: verify the property in an abstract model (over-approximation)

- It creates abstract models using the variable dependency graph of the variables involved in the property
(Iterative) Variable abstraction

- It was designed to verify models where the property preserving reduction techniques are not effective.

- Main idea: verify the property in an abstract model (over-approximation).

- It creates abstract models using the variable dependency graph of the variables involved in the property.

- It focuses on simple safety properties $AG (\alpha \rightarrow \beta)$: More aggressive abstraction method.
Iterative Variable abstraction

- Some concepts: Abstract models
Iterative Variable abstraction

- Some concepts: Abstract models

```
ORGANIZATION_BLOCK OB1
VAR_TEMP
  info : ARRAY[0..19] OF BYTE;
END_VAR
BEGIN
  FB100.DB1(a := FALSE);
  Q1.0 := DB1.c;
END_ORGANIZATION_BLOCK
```
Iterative Variable abstraction

- Some concepts: Abstract models

```
ORGANIZATION_BLOCK OB1
VAR_TEMP
    info : ARRAY[0..19] OF BYTE;
END_VAR
BEGIN
    FB100.DB1(a := FALSE);
    Q1.0 := DB1.c;
END_ORGANIZATION_BLOCK
```

PLC – IM transformation

OM
Iterative Variable abstraction

- Some concepts: Abstract models

```
ORGANIZATION_BLOCK OB1
  VAR_TEMP
    info : ARRAY[0..19] OF BYTE;
  END_VAR
BEGIN
  FB100.DB1(a := FALSE);
  Q1.0 := DB1.c;
END_ORGANIZATION_BLOCK
```

PLC – IM transformation

Prop. preserving reductions

$OM$ $OM'$
Iterative Variable abstraction

- Some concepts: Abstract models
Iterative Variable abstraction

- Some concepts: Abstract models

- AM' (Abstract model) contains less variables
Iterative Variable abstraction

- Some concepts: Abstract models

- AM' (Abstract model) contains less variables
- AM' has a smaller PSS (Potential State Space)
Iterative Variable abstraction

- Some concepts: Abstract models

- \( AM'_n \) (Abstract model) contains less variables
- \( AM' \) has a smaller PSS (Potential State Space)
- \( AM' \) represents a **bigger** range of possible behaviors of the system (over-approximation)
Iterative Variable abstraction

- Some concepts: Abstract models

- $AM'_n$ (Abstract model) contains less variables
- $AM'$ has a smaller PSS (Potential State Space)
- $AM'$ represents a **bigger** range of possible behaviors of the system (over-approximation)
- If $AG(\alpha \rightarrow \beta)$ is true on $AM'$ then it is true on $OM'$ (and OM)
Iterative Variable abstraction

- Some concepts: Abstract models

- AM' (Abstract model) contains less variables
- AM' has a smaller PSS (Potential State Space)
- AM' represents a bigger range of possible behaviors of the system (over-approximation)
- If \( \text{AG}(\alpha \rightarrow \beta) \) is true on AM' then it is true on OM' (and OM)
- If \( \text{AG}(\alpha \rightarrow \beta) \) is false on AM' then we need extra information
Iterative Variable abstraction
Iterative Variable abstraction

- How to create the abstract models
Iterative Variable abstraction

- How to create the abstract models
  - Distance in the variable dependency graph ($\delta$)
  - Replacing the selected variables by non-deterministic values
Iterative Variable abstraction

- How to create the abstract models
  - Distance in the variable dependency graph ($\delta$)
  - Replacing the selected variables by non-deterministic values

$$AG\left( (EoC \land b) \rightarrow a \right)$$
Iterative Variable abstraction

- How to create the abstract models
  - Distance in the variable dependency graph ($\delta$)
  - Replacing the selected variables by non-deterministic values

$$AG\left( (EoC \land b) \rightarrow a \right)$$

\[\delta = 1\]
Iterative Variable abstraction

- High level description
Iterative Variable abstraction

- High level description

\[ AG(\alpha \rightarrow \beta) \]

\[ OM' \]

\[ TO = 30 \text{ s} \]
\[ m = 10 \]
\[ \delta' = 0 \]
\[ \delta'' = 0 \]

PhD defense: Borja Fernández Adiego
Iterative Variable abstraction

High level description

\[ AG(\alpha \rightarrow \beta) \]

\[ OM' \]

\[ OM' \]

\[ \delta' = \delta' + 1 \]

\[ TO = 30 \text{ s} \]

\[ m = 10 \]

\[ \delta' = 0 \]

\[ \delta'' = 0 \]

PhD defense: Borja Fernández Adiego

Approach
Iterative Variable abstraction

- High level description

If $p$ is true for $AM'_n$ then $p$ is true for $OM'$

$\delta' = \delta' + 1$

TO = 30 s
$m = 10$
$\delta' = 0$
$\delta'' = 0$

True

PhD defense: Borja Fernández Adiego
Iterative Variable abstraction

- High level description

\[ AG(\alpha \rightarrow \beta) \]

\[ OM' \]

\[ \delta' = \delta' + 1 \]

- If \( p \) is true for \( AM'_n \), then \( p \) is true for \( OM' \)

- If \( p \) is false for \( AM'_n \), then we need extra info

Approach

- TO = 30 s
- \( m = 10 \)
- \( \delta' = 0 \)
- \( \delta'' = 0 \)

PhD defense: Borja Fernández Adiego
Iterative Variable abstraction

- High level description

\[ \text{AG}(\alpha \rightarrow \beta) \]

\[ OM' \]

\[ OM' \]

\[ TO = 30 \text{ s} \]
\[ m = 10 \]
\[ \delta' = 0 \]
\[ \delta'' = 0 \]

If \( p \) is true for \( AM'_n \) then \( p \) is true for \( OM' \)

If \( p \) is false for \( AM'_n \) then we need extra info

\[ \delta' = \delta' + 1 \]
Iterative Variable abstraction

High level description

If $p$ is true for $AM'_n$ then $p$ is true for $OM'$

If $p$ is false for $AM'_n$ then we need extra info

Counterexample $(\Upsilon, \theta)$

$\delta' = \delta' + 1$

$\text{TO} = 30 \text{ s}$

$\delta' = 0$

$m = 10$

$\delta'' = 0$

PhD defense: Borja Fernández Adiego
Iterative Variable abstraction

- High level description

\[ \text{AG}(\alpha \rightarrow \beta) \]

\[ \text{OM}' \]

TO = 30 s
m = 10
\( \delta' = 0 \)
\( \delta'' = 0 \)

If \( p \) is true for \( AM'_n \) then
\( p \) is true for \( OM' \)

If \( p \) is false for \( AM'_n \) then
Counterexample \((\Upsilon, \theta)\)
we need extra info

\[ \text{AG}(\alpha \rightarrow \neg \beta) \]

\[ \text{OM}' \]

\[ \text{OM}' \]

\[ \delta' = \delta' + 1 \]
Iterative Variable abstraction

- High level description

From counterexample to Reachability property

\[ \text{AG}(\alpha \rightarrow \beta) \]

\[ \text{OM'} \rightarrow \text{OM}'' \]

\[ \delta'' = \delta'' + 1 \]

\[ \text{EF}(\gamma \& \theta) \]

\[ \text{OM}'' \]

\[ \delta'' = \delta'' + 1 \]

\[ \text{AG}(\alpha \rightarrow \neg \beta) \]

\[ \text{OM}' \]

[False or TO]

If \( p \) is true for \( AM'_n \) then \( p \) is true for \( OM' \)

If \( p \) is false for \( AM'_n \) then we need extra info

Counterexample \((\Upsilon, \theta)\)

\[ \text{TO} = 30 \text{ s} \]

\[ m = 10 \]

\[ \delta' = 0 \]

\[ \delta'' = 0 \]

[True]

[False]

PhD defense: Borja Fernández Adiego
Iterative Variable abstraction

- High level description

\[ AG(\alpha \rightarrow \beta) \]

- If \( p \) is true for \( AM'_n \) then \( p \) is true for \( OM' \)

- If \( p \) is false for \( AM'_n \) then we need extra info

From counterexample to Reachability property

\[ \delta'' = \delta'' + 1 \]

\[ TO = 30 \text{ s} \]
\[ m = 10 \]
\[ \delta' = 0 \]
\[ \delta'' = 0 \]

PhD defense: Borja Fernández Adiego
Iterative Variable abstraction

- High level description

AG(α → β)

OM’

δ’ = δ’ + 1

INVAR if m ≤ 10

[True]

If p is true for AM’₀ then p is true for OM’

[True]

AG(α → ¬β)

δ'' = δ'' + 1

[True]

If p is false for AM’₀ then we need extra info

Counterexample (ϒ, θ)

[False]

From counterexample to Reachability property

TO = 30 s
m = 10
δ’ = 0
δ'' = 0

[False or TO]
Iterative Variable abstraction

- High level description

- If $p$ is true for $AM'_n$ then $p$ is true for $OM'$
- If $p$ is false for $AM'_n$ then we need extra info

From counterexample to Reachability property

- $TO = 30\ s$
- $m = 10$
- $\delta' = 0$
- $\delta'' = 0$

- $\delta' = \delta' + 1$
- $\delta'' = \delta'' + 1$

- $\text{INVAR if } m \leq 10$
- $\text{EF}(\gamma \& \theta)$
- $\text{AG}(\alpha \rightarrow \beta)$
Iterative Variable abstraction

- High level description

\[ \text{AG}(\alpha \rightarrow \beta) \]

\[ OM' \]

TO = 30 s  
\[ m = 10 \]
\[ \delta' = 0 \]
\[ \delta'' = 0 \]

From counterexample to Reachability property

\[ \text{EF}(\gamma \& \theta) \]

\[ AM''_n \]

\[ \delta'' = \delta'' + 1 \]

\[ \text{INVAR if } m \leq 10 \]

\[ \text{AG}(\alpha \rightarrow \beta) \]

\[ OM' \]

\[ \delta' = \delta' + 1 \]

If \( p \) is true for \( AM'_n \), then \( p \) is true for \( OM' \)

If \( p \) is false for \( AM'_n \), then we need extra info

Counterexample \((\Upsilon, \theta)\) then we need extra info

\[ \text{TO} \]

[True]

[False or TO]

[False or TO]
1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis
Approach

1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis

- Methodology overview
- Methodology steps (contributions)
IM – verification tools transformation
**IM – verification tools transformation**

<table>
<thead>
<tr>
<th>IM</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Synchronization i</strong></td>
<td>nuXmv</td>
</tr>
<tr>
<td><img src="image-url" alt="Diagram" /></td>
<td>UPPAAL</td>
</tr>
<tr>
<td><img src="image-url" alt="Diagram" /></td>
<td>BIP</td>
</tr>
</tbody>
</table>

---

**IM Representation**

- **Synchronization i**
  - **IM**
    - **a**
      - $l_1$ and $g_1$
    - **b**
      - $l_1$ and $g_2$

**PhD defense: Borja Fernández Adiego**
IM – verification tools transformation

**IM**

- **Synchronization $i$**
  - $a$
    - $l_1$
    - $[g_1]$ $i_1!$
    - $l_2$
  - $b$
    - $l_1$
    - $[g_2]$ $i_1?$
    - $l_2$

**Representation**

- **nuXmv**
  - `INVAR (interaction = i1 <-> (inst_a.loc = l1 & inst_b.loc = l1 & g1 & g2));`

- **UPPAAL**
  - `<label kind="synchronization">i1!</label>`
  - `<label kind="synchronization">i1?</label>`

- **BIP**
  - `port type voidPort()`
  - `on t1 from l1 to l2`
  - `connector VoidPortConnector interaction1(a.p1, b.p2)`
IM – verification tools transformation

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. init

3. init(loc) := initial;
next(loc) := case
  loc = end : initial;
  loc = initial : 11;
  loc = 11 & ((IA > 0sd16_0)) : 13;
  loc = 11 & (!(IA > 0sd16_0)) : 12;
  loc = 12 & ((IB > 0sd16_0)) : 13;
  loc = 12 & (!(IB > 0sd16_0)) : 13;
  loc = 13 : end;
TRUE : loc;
esac;
next(XA) := case
  loc = 11 & ((IA > 0sd16_0)) : TRUE;
  loc = 11 & (!(IA > 0sd16_0)) : FALSE;
  TRUE : XA;
esac;
...
Approach

- Requirements formalization
- PLC execution platform modeling
- PLC code – IM transformation
- Reduction techniques
- IM – verification tools transformation
- Process modeling
- Counterexample analysis

- Methodology overview
- Methodology steps (contributions)
Approach

- Requirements formalization
- PLC execution platform modeling
- PLC code – IM transformation
- Reduction techniques
- IM – verification tools transformation
- **Process modeling**
- Counterexample analysis

- Methodology overview
- **Methodology steps**
  (contributions)
Process Modeling
Process Modeling
Process Modeling

- A model of the process is not provided by the methodology
Process Modeling

- A model of the process is not provided by the methodology
- However, the process information may be fundamental to verify a property
Process Modeling

- A model of the process is not provided by the methodology
- However, the process information may be fundamental to verify a property
Process Modeling

- A model of the process is not provided by the methodology
- However, the process information may be fundamental to verify a property

\[
\text{AG}\left( (EoC \land \text{Open} \land \text{Opened}) \rightarrow \text{Status} \right)
\]
Process Modeling

- A model of the process is not provided by the methodology
- However, the process information may be fundamental to verify a property

\[ AG\left( (E_{oC} \land \text{Open} \land \text{Opened}) \rightarrow \text{Status} \right) \]

<table>
<thead>
<tr>
<th>Variable</th>
<th>End of Cycle 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opened</td>
<td>FALSE</td>
</tr>
<tr>
<td>Closed</td>
<td>TRUE</td>
</tr>
<tr>
<td>Close</td>
<td>FALSE</td>
</tr>
<tr>
<td>Open</td>
<td>TRUE</td>
</tr>
<tr>
<td>Status</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
Process Modeling

- A model of the process is not provided by the methodology
- However, the process information may be fundamental to verify a property

\[
\text{AG}\left((EoC \land \text{Open} \land \text{Opened}) \rightarrow \text{Status}\right)
\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>End of Cycle 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opened</td>
<td>TRUE</td>
</tr>
<tr>
<td>Closed</td>
<td>TRUE</td>
</tr>
<tr>
<td>Close</td>
<td>FALSE</td>
</tr>
<tr>
<td>Open</td>
<td>TRUE</td>
</tr>
<tr>
<td>Status</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
Process Modeling

- A model of the process is not provided by the methodology
- However, the process information may be fundamental to verify a property
- Invariants can be added, containing the information of the process

\[ \text{AG} \left( (EoC \land \text{Open} \land \text{Opened}) \rightarrow \text{Status} \right) \]

<table>
<thead>
<tr>
<th>Variable</th>
<th>End of Cycle 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opened</td>
<td>TRUE</td>
</tr>
<tr>
<td>Closed</td>
<td>TRUE</td>
</tr>
<tr>
<td>Close</td>
<td>FALSE</td>
</tr>
<tr>
<td>Open</td>
<td>TRUE</td>
</tr>
<tr>
<td>Status</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
Process Modeling

- A model of the process is not provided by the methodology
- However, the process information may be fundamental to verify a property
- Invariants can be added, containing the information of the process

\[ \text{AG} \left( (EoC \land \text{Open} \land \text{Opened}) \rightarrow \text{Status} \right) \]

**Invariant:** \( \neg (\text{Opened} \land \text{Closed}) \)

**False:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>End of Cycle1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opened</td>
<td>TRUE</td>
</tr>
<tr>
<td>Closed</td>
<td>TRUE</td>
</tr>
<tr>
<td>Close</td>
<td>FALSE</td>
</tr>
<tr>
<td>Open</td>
<td>TRUE</td>
</tr>
<tr>
<td>Status</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

PhD defense: Borja Fernández Adiego
Process Modeling

- A model of the process is not provided by the methodology
- **However, the process information may be fundamental to verify a property**
- Invariants can be added, containing the information of the process

\[ AG\left( (EoC \land \text{Open} \land \text{Opened}) \rightarrow \text{Status} \right) \]

\[ \text{INVAR: } \neg (\text{Opened} \land \text{Closed}) \]

True

PhD defense: Borja Fernández Adiego
Approach

1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis

- Methodology overview
- Methodology steps (contributions)
Approach

1. Requirements formalization
2. PLC execution platform modeling
3. PLC code – IM transformation
4. Reduction techniques
5. IM – verification tools transformation
6. Process modeling
7. Counterexample analysis

- Methodology overview
- Methodology steps
  (contributions)
Counterexample analysis
Counterexample analysis

Counterexamples are not necessarily simple (e.g. 1.4 MB, 34,000 lines)
Counterexample analysis

- Counterexamples are not necessarily simple (e.g. 1.4 MB, 34,000 lines)
  - The relevant information can be extracted automatically and a verification report can be created
Counterexample analysis

- Counterexamples are not necessarily simple (e.g. 1.4 MB, 34,000 lines)
  - The relevant information can be extracted automatically and a verification report can be created

- Is it a real bug or just due to the modelling?
Counterexample analysis

- Counterexamples are not necessarily simple (e.g. 1.4 MB, 34,000 lines)
  - The relevant information can be extracted automatically and a verification report can be created

- Is it a real bug or just due to the modelling?

---

**Generated counterexample**

-> State: 1.17 <-
- input1 := TRUE
- input2 := FALSE
- input3 := FALSE
- input4 := FALSE

---

**PLC Demonstrator**

IF cycle = 4 THEN
  input1 := TRUE
  input2 := FALSE
  input3 := FALSE
  input4 := FALSE
...
  FB1.DB1();
  Failure := Output1;
END_IF;
Methodology CASE tool
Methodology CASE tool

- Implemented using EMF and Xtext
Methodology CASE tool

- Implemented using EMF and Xtext
Methodology CASE tool

- Implemented using EMF and Xtext

![Requirement Diagram]

The requirement to be checked should be defined in this section.

Requirement pattern: 3. \{1\} is impossible at the end of the PLC cycle.

Pattern params: [1] A = false & C = true

3. \( A = \text{false} \land C = \text{true} \) is impossible at the end of the PLC cycle.
Methodology CASE tool

- Implemented using EMF and Xtext
Case studies

Real-life PLC programs at CERN
- Function block PLC program
- Complete PLC program
Funtion block PLC program
This Function block represents a physical equipment driven by digital signals, e.g. an on-off valve, heater or motor (OnOff object from the UNICOS framework developed at CERN)
Funtion block PLC program

- This Function block represents a physical equipment driven by digital signals, e.g. an on-off valve, heater or motor (OnOff object from the UNICOS framework developed at CERN)

<table>
<thead>
<tr>
<th>Metric</th>
<th>OnOff PLC code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of code</td>
<td>≈ 820</td>
</tr>
<tr>
<td>Program blocks</td>
<td>1 main FB, 2 timers and 3 FCs</td>
</tr>
<tr>
<td>Function calls</td>
<td>21</td>
</tr>
<tr>
<td>Input variables</td>
<td>29</td>
</tr>
<tr>
<td>Output variables</td>
<td>31</td>
</tr>
<tr>
<td>Internal variables</td>
<td>82</td>
</tr>
<tr>
<td>PLC data types</td>
<td>BOOL, INT, REAL, WORD, TIME, STRUCT and ARRAY</td>
</tr>
<tr>
<td>Timers</td>
<td>3 instances</td>
</tr>
</tbody>
</table>
Complex properties have been verified on this object

\[
G \left( (EoC \land (\neg Au \land Au \land Mo \land Ro \land (Au \land Mo \land St \lor MM \land Mo \land St \lor Fo \land Mo \land St)) \land (\neg MA \land Au \land Mo \land Ro \land \neg MM \land Mo \land Ro \land \neg HLD \land \neg M So \land ft \land LDR \land \neg MFo \land Mo \land Ro)) \land X (\neg EoC \lor (EoC \land Au \land Au \land Mo \land Ro \land (Au \land Mo \land St \lor MM \land Mo \land St \lor Fo \land Mo \land St)) \land (\neg MA \land Au \land Mo \land Ro \land \neg MM \land Mo \land Ro \land \neg HLD \land \neg M So \land ft \land LDR \land \neg MFo \land Mo \land Ro)\right) \land X (\neg EoC \lor (EoC \land Au \land Mo \land St))
\]
Funtion block PLC program

<table>
<thead>
<tr>
<th>IM metrics</th>
<th>Original Model</th>
<th>Reduced Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>259</td>
<td>33</td>
</tr>
<tr>
<td>Automata</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>locations</td>
<td>460</td>
<td>17</td>
</tr>
<tr>
<td>PSS</td>
<td>$1.61 \cdot 10^{218}$</td>
<td>$3.65 \cdot 10^{10}$</td>
</tr>
<tr>
<td>Generation time</td>
<td>0.3 s</td>
<td>12.6 s</td>
</tr>
</tbody>
</table>

- Property preserving reduction techniques were applied
Function block PLC program

<table>
<thead>
<tr>
<th>IM metrics</th>
<th>Original Model</th>
<th>Reduced Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>259</td>
<td>33</td>
</tr>
<tr>
<td>Automata</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>locations</td>
<td>460</td>
<td>17</td>
</tr>
<tr>
<td>PSS</td>
<td>$1.61 \cdot 10^{218}$</td>
<td>$3.65 \cdot 10^{10}$</td>
</tr>
<tr>
<td>Generation time</td>
<td>0.3 s</td>
<td>12.6 s</td>
</tr>
</tbody>
</table>

- Property preserving reduction techniques were applied
  - Without reductions the property was not verified (nuXmv runtime $\geq$ 1 day)
Property preserving reduction techniques were applied

- **Without reductions the property was not verified** (nuXmv runtime ≥ 1 day)
- **With reductions** the property was verified using nuXmv in less than 1 second
Complete PLC program
Complete PLC program

- PLC program of a subsystem of the LHC (Large Hadron Collider) cryogenics process
Complete PLC program

- PLC program of a subsystem of the LHC (Large Hadron Collider) cryogenics process

- PLC program
  - 17,500 lines of code
  - 110 FBs FCs
  - SCL + Graph (ST + SFC)
Complete PLC program

- PLC program of a subsystem of the LHC (Large Hadron Collider) cryogenics process

- PLC program
  - 17,500 lines of code
  - 110 FBs FCs
  - SCL + Graph (ST + SFC)

- IM model (302 automata and PSS = 10^{31985})
Complete PLC program

- PLC program of a subsystem of the LHC (Large Hadron Collider) cryogenics process

- PLC program
  - 17,500 lines of code
  - 110 FBs FCs
  - SCL + Graph (ST + SFC)

- IM model (302 automata and PSS = 10^{31985})

- Function analysis (Specification): Safety properties

\[ \text{AG}\left( (EoC \land QSDN\_4\_DN1CT\_SEQ\_DB\_Stop.x) \rightarrow QSDN\_4\_1PV408\_AuOffR \right) \]
Complete PLC program

- IM model (302 automata and PSS \(= 10^{31985} \))

- Function analysis (Specification): Safety properties

\[
\text{AG}\left((EoC \land \text{QSDN}_4\_\text{DN1CT}_\text{SEQ}_\text{DB}.\text{Stop}.x) \rightarrow \text{QSDN}_4\_1\text{PV408}.\text{AuOffR}\right)
\]
Complete PLC program

- The **variable abstraction** technique was needed

<table>
<thead>
<tr>
<th>Metric</th>
<th>Non-reduced model ($M_1$)</th>
<th>Reduced model ($M_2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSS</td>
<td>$10^{31985}$</td>
<td>$10^{5048}$</td>
</tr>
<tr>
<td>Variables</td>
<td>31,402</td>
<td>3757</td>
</tr>
<tr>
<td>Generation</td>
<td>4.2 s</td>
<td>15.3 s</td>
</tr>
</tbody>
</table>

$$AG\left((EoC \land QSDN\_4\_DN1CT\_SEQ\_DB\_Stop\_x) \rightarrow QSDN\_4\_1PV408.AuOffR\right)$$
Complete PLC program

- The variable abstraction technique was needed

<table>
<thead>
<tr>
<th>Metric</th>
<th>Non-reduced model ($M_1$)</th>
<th>Reduced model ($M_2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSS</td>
<td>$10^{31985}$</td>
<td>$10^{5048}$</td>
</tr>
<tr>
<td>Variables</td>
<td>31,402</td>
<td>3757</td>
</tr>
<tr>
<td>Generation</td>
<td>4.2 s</td>
<td>15.3 s</td>
</tr>
</tbody>
</table>

This property was verified in 45 seconds (including the model generation and the nuXmv verification runtime)

- 2 iterations of the variable abstraction
Summary and conclusions
Summary and conclusions

- Solution to verify automatically new and existing PLC programs
Summary and conclusions

- Solution to verify automatically new and existing PLC programs
Summary and conclusions

- Solution to verify automatically new and existing PLC programs

- Patterns specification
  - Simple natural language, covering a big range of properties
  - It is not the final solution for requirement specifications, but it helps to find bugs
Summary and conclusions

- Execution platform modeling
  - Skeleton for the models
  - Focus on centralized control systems
Execution platform modeling
- Skeleton for the models
- Focus on centralized control systems

Controlled process modeling
- Not a complete model of the process is provided
- Invariants to avoid false positives
PLC code into IM transformation

- Rule – based transformation + assumptions
- Currently ST and SFC
- Timers transformation
Summary and conclusions

- PLC code into IM transformation
  - Rule – based transformation + assumptions
  - Currently ST and SFC
  - Timers transformation

- IM into verification tools transformation
  - Rule – based transformation
  - Currently nuXmv (best verification performance), UPPAAL and BIP
Summary and conclusions
Summary and conclusions

- Reduction and abstraction techniques
  - Property preserving reduction techniques (COI and general rule-based reductions)
  - Variable abstraction
Conclusions
Conclusions

- The methodology can be applied to new and existing PLC programs
Conclusions

- The methodology can be applied to **new and existing PLC programs**

- It is **valid for any development process** of PLC programs, as the models are created out of the PLC code
Conclusions

- The methodology can be applied to **new and existing PLC programs**

- It is **valid for any development process** of PLC programs, as the models are created out of the PLC code

- Any **complexity related to formal methods** is **hidden from control engineers**
Conclusions

- The methodology can be applied to new and existing PLC programs

- It is valid for any development process of PLC programs, as the models are created out of the PLC code

- Any complexity related to formal methods is hidden from control engineers

- It can be applied to real-life systems (large), e.g. CERN PLC programs
Conclusions

- The methodology can be applied to **new and existing PLC programs**

- It is **valid for any development process** of PLC programs, as the models are created out of the PLC code

- Any **complexity** related to formal methods is **hidden** from control engineers

- It can be applied to real-life systems (large), e.g. CERN PLC programs

- Many **discrepancies** between the specification and the (already tested) PLC programs were found
Future work
Future work

- Abstraction techniques
  - Predicate abstraction
Future work

- Abstraction techniques
  - Predicate abstraction

Requirement specification
- Replacing the patterns by a complete, unambiguous and easy-to-use specification formalism (Correctness by construction)
Future work

- **Abstraction techniques**
  - Predicate abstraction

- **Requirement specification**
  - Replacing the patterns by a complete, unambiguous and easy-to-use specification formalism (Correctness by construction)

- Applying **different formal verification algorithms** to our models
  - Bounded model checking
  - Compositional verification (Opening the door to **distributed** control systems)
This thesis proposes an alternative to the traditional verification approaches for control software in the industrial automation community.
This thesis proposes an alternative to the traditional verification approaches for control software in the industrial automation community.

Thank you for your attention.