Motivation

Improve the quality of UNICOS PLC programs

UNICOS PLC programs

- Large PLC programs based on Library of objects
- Function Blocks (FBs)
- Programs generated by the UNICOS Generation tool
- Most common PLC languages
  - ST (Schneider) and SCL (Siemens)

Problems

- Complex expressions, specific naming conventions,
  dead code, code repetition, potential concurrency
  problems (PLC interrupts), unused variables,
  multiple assignment of output variables, etc.
- Lack of formal and complete specification

Challenges

- Lack of Static 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

What is it?

- Technique that examines a program without executing it
- Similar to code review or code comprehension performed by automated tools
- Leads to the early detection of bugs
- 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.

Approach

Goal

Our goal is to be one step closer to ensure the reliability of our PLC programs by applying static analysis, complementing the existing model checking methodology

Our solution: extending PLCverif

Static Analysis Rules

- Pattern-based requirement
- Formal requirement (temporal logic)
- CBMC input (C code for verification)
- nuxmv model
- Verification result

PLCverif

PLCverif methodology

- Designed at CERN to apply model checking to PLC programs
- Possibility to be extended to apply rule-based AST static analysis (modular approach)

PLCverif AST

- SCL programs are represented in the AST as PlcCode objects which contain:
  - block declarations (FCs, FBs and OBs), data block declarations and user-defined type declarations
  - Static analysis rules can be applied directly to the AST using predefined methods
    (which simplify the logic of the rules)

Static analysis in the PLCverif enviroment

- Currently nineteen basic static analysis rules were developed in JAVA
- The rules mainly concern: UNICOS naming conventions and potential code smells
  (nested ifs, dead code, etc.)

Limitations

- Not everything can be resolved with the AST method
- Concurrency problems may require the use of different static analysis methods (e.g.
  call graph techniques)

Outcome

- First prototype of AST-based basic static analysis rules
- Applied to UNICOS object Function Blocks (SCL language)
- Detection of naming convention violations in our code
- Detection of problematic structure that could lead to maintainability and readability problems

Future work

- Add new and more complex AST-based rules
- Integrate more static analysis methods in PLCverif

Static Analysis in PLCverif

- AST (Abstract Syntax Tree)
- Rule-based AST (Abstract Syntax Tree) analysis
- Control-flow analysis
- Data-flow analysis
- Call graph analysis

EXPERIENCE WITH STATIC PLC CODE ANALYSIS AT CERN

C. Tsiplaki, B. Fernández, E. Blanco, CERN, Geneva, Switzerland