Symbolic Execution of Programmable Logic Controller Code

Shengjian Guo  
Meng Wu  
Chao Wang

Virginia Tech  
USC University of Southern California
Programmable Logic Controller

- PLC, a specialized digital computer
Programmable Logic Controller

• Widely used
Programmable Logic Controller
• Important, but also vulnerable

Stuxnet is a malicious computer worm, first identified in 2010, that specifically targets programmable logic controllers (PLCs) and was responsible for causing substantial damage to Iran's nuclear program.

--Wikipedia
PLC Programs

- **Five Programming Languages**
  - IEC 61131-3 Standard

- **Program → Task**
  - Single- or multi-task

- **Non-terminating**
  - Periodic execution
Challenges of validating PLC Programs

- Periodic execution
  - What inputs for each periodic execution?
- Non-terminating
  - When to stop? Why?
- Priority-based execution
  - How to handle task interleavings?
Existing methods

Verification

Building models and verify

Raw PLC Programs

Handcraft test cases

Simulator testing
Real device testing

Validated PLC Programs
Existing methods

• Verifications
  • Dedicated skills to build models
  • Suitable for verifying high-level designs
• Simulators && Stress testing on real devices
  • When to stop ?
  • How to handle various inputs combination ?
  • How to handle increasing task interleavings ?
Existing methods

Better methods to fill the gap?
Our tool: SymPLC

- Raw PLC Programs
- Validation
- Building models and verify
- SymPLC: Automated test case generation, systematic testing, termination detection
- Handcraft test cases
- Simulator testing
- Real device testing
- Validated PLC Programs
Symbolic Execution

• A program analysis technique
  • Static/dynamic
  • Systematically exploring program paths

• Program inputs
  • Symbols, NOT concrete values (e.g., integers)

• Output
  • Concrete solutions of the symbolic inputs regarding path constraints
  • Leading to different program paths
The overall flow of SymPLC

PLC Programs → Modeling in C → Generate Test Harness and Constraints

Current inputs (timing, input, schedule) → KLEE-based PLC Symbolic Execution → Eliminating Redundancy (priority, period, stateful)

Generate new test inputs
Snapshot: A toy PLC example

```plaintext
FUNCTION_BLOCK FB_G4LTL
(* Variable definition *)
VAR_INPUT
  x0 : BOOL;
  x1 : BOOL;
END_VAR
VAR_OUTPUT
  y0 : BOOL;
END_VAR
VAR
  cstate15 : LINT := 0;
END_VAR
CASE cstate15 OF
  0:
    IF ((x0 = FALSE) AND ( TRUE )) OR ((x0 = TRUE) AND (x1 = TRUE)) THEN
      cstate15 := 0;
      y0 := FALSE;
    ELSIF ((x0 = TRUE) AND (x1 = FALSE)) THEN
      cstate15 := 0;
      y0 := TRUE;
    END_IF;
END_CASE;
END_FUNCTION_BLOCK

PROGRAM ProgA
VAR_IN_OUT
  x0: BOOL ;
  x1: BOOL;
  y0: BOOL ;
  FB1 :FB_G4LTL;
END_VAR
FB1(x0:=x0,x1:= x1,y0 => y0);
END_PROGRAM
```
Snapshot: The C model of the PLC code
Snapshot: The execution result