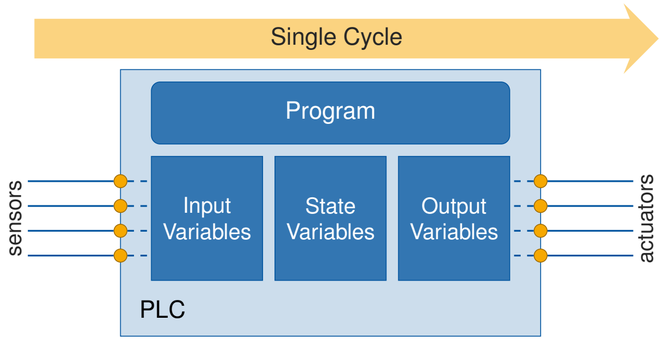
Verification of Reactive Programs from Industrial Automation

Dimitri Bohlender, Stefan Kowalewski

ARCADE.PLC

Programmable Logic Controllers (PLCs)

- Tailored to the domain of industrial automation
- Realise reactive systems, repeatedly executing the same task

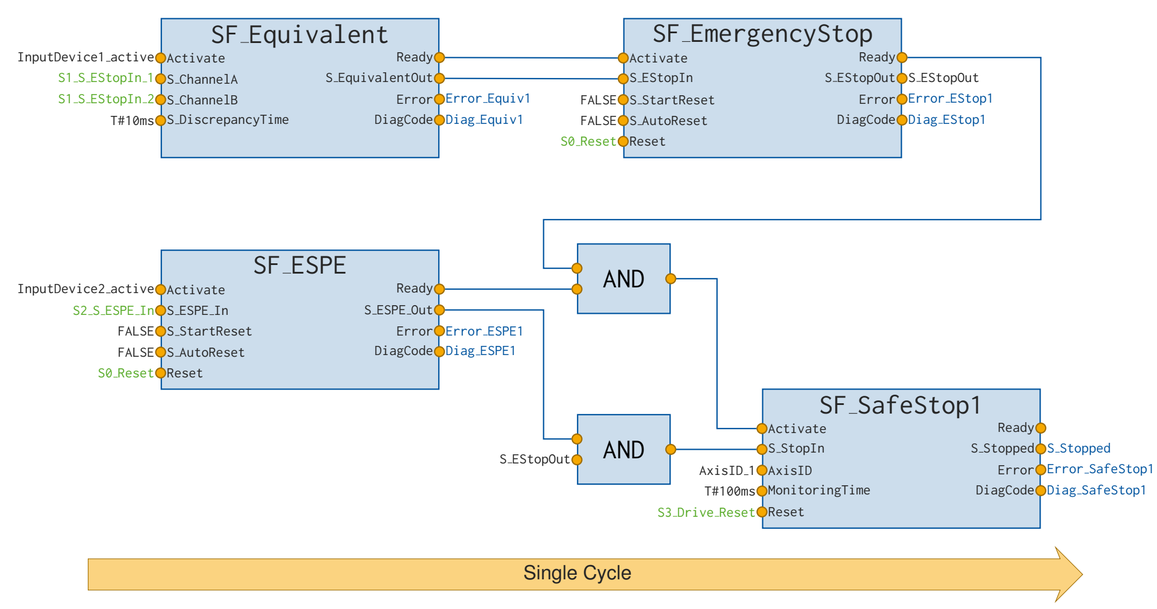


PLC Software

- Programming languages standardised (IEC 61131-3)
- Combination of several languages typical
- Typically graphical on higher level but textual on lower level

Observations

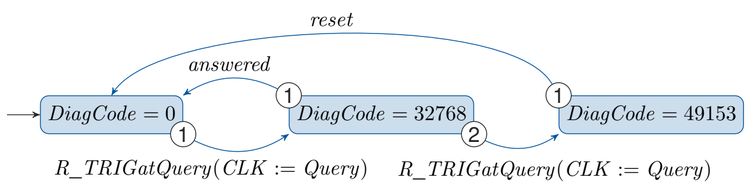
- Specifications refer to observable state at cycle-end
- Function blocks exhibit mode-semantics



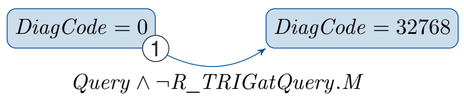
PDR-based Model Checking

PLCopen Automaton

- Specifies safe observable behaviour of a block



- Compliance w.r.t. a transition can often be checked locally
- Encode program in terms of CHCs for a single cycle
- Consider the following (pre-processed) transition



- Check reachability of unsafe behaviour via PDR, where

$$\varphi_{bad} := \varphi_{source} \wedge b_1 \wedge \neg\varphi_{target_1} \wedge \varphi_{atExit}$$

$$\begin{aligned} \varphi_{source} &:= DiagCode_{cpy} = 0 \\ b_1 &:= Query_{cpy} \wedge \neg R\_TRIGatQuery.M_{cpy} \\ \varphi_{target_1} &:= DiagCode = 32768 \\ \varphi_{atExit} &:= state(l_{exit}, \mathbf{x}, \mathbf{x}_{cpy}) \end{aligned}$$

- Local check may yield spurious counterexamples

```
1 R_TRIGatQuery(CLK:=Query);
2 IF (R_TRIGatQuery.Q) THEN
3     DiagCode:=0x8000;
4 END_IF;
5 // rest omitted                ✓
```

```
1 IF (b) THEN
2     R_TRIGatQuery(CLK:=Query);
3     IF (R_TRIGatQuery.Q) THEN
4         DiagCode:=0x8000;
5 // rest omitted                ✗
```

- If so, check with closed cycle

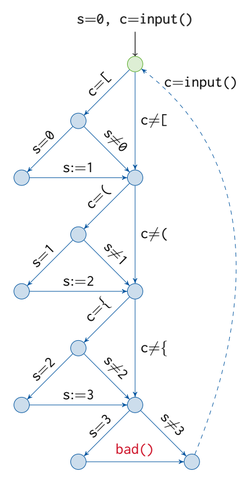
Future Work

- Analysis of restart behaviour:
  Variables may retain their value after restart/power cut. Starting from these new states no new behaviour shall be observable.
- Mode-oriented PDR:
  Software-oriented PDR variants partition the transition relation by program locations. An analogous partitioning by modes may help with invariants disjunctive over modes.

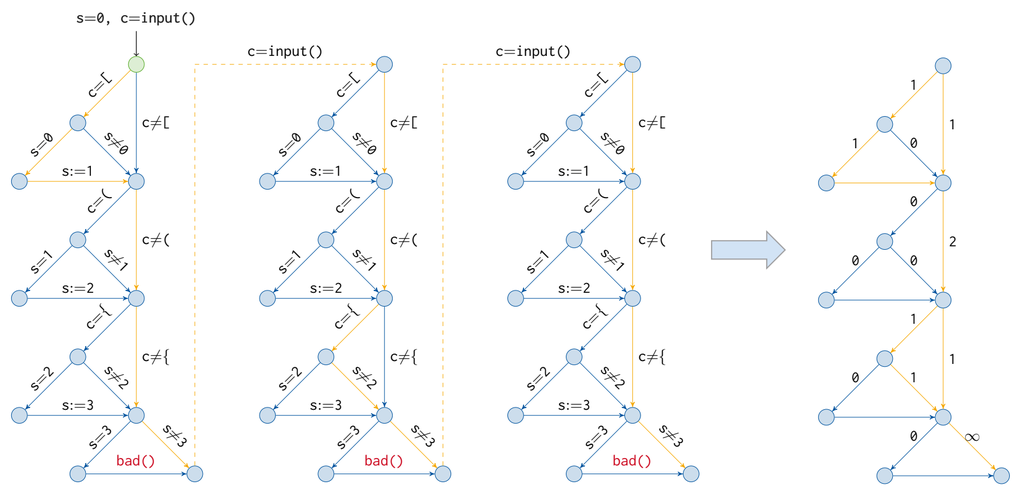
Symbolic Execution

Guided by Mode-Space

- Consider the right-hand program
- Implicit state-machine (state s)
- Fails on input sequence “[({“
- Bad choices hard to identify (cyclicity)



- CFG-based guidance is local, needs bound and degenerates into random search:



- Mode change cannot be enforced arbitrarily

- Also, some branches are exclusive to certain modes
- Better estimation with mode-space & slicing: