



# Verifying PLC Programs via Monitors: Extending the Integration of FRET and PLCverif

---

\*Xaver Fink<sup>1</sup>, Anastasia Mavridou<sup>2</sup>, \*Andreas Katis<sup>2</sup>, Borja Fernández Adiego<sup>1</sup>

June 6, 2024

<sup>1</sup>CERN Industrial Control Systems group (BE-ICS)

<sup>2</sup>KBR at NASA Ames Research Center

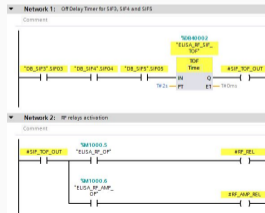
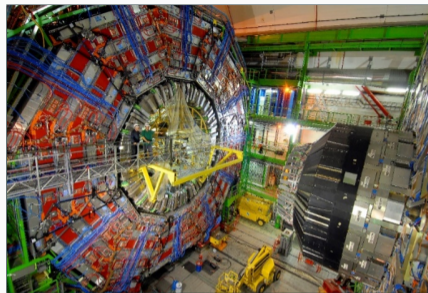
# Motivation: Critical PLC Installations at CERN

**CERN:** European Organization for Nuclear Research

- Biggest particle accelerator complex in the world: Large **H**adron **C**ollider (LHC)
- Hundreds of **control systems** with thousands ( $\approx 3050$  in 2023) of **Programmable Logic Controllers (PLC)**
  - Critical installations: Cryogenics, ventilation, vacuum systems.



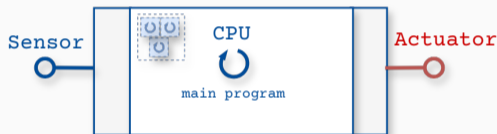
**Figure 1:** Siemens S7-1500 F PLC



**Figure 2:** PLC Ladder diagram program 1/16

# Motivation: Critical PLC Installations at CERN

## Programmable Logic Controller



```
(* Interlocks / FailSafe *)
IF #PoutOff THEN
  IF #InterlockR THEN
    IF #FPulse AND NOT #FFsPosOn2 THEN
      IF #FFsPosOn THEN
        #OutOnOVSt := #PulseOn;
        #OutOffOVSt := FALSE;
        #OutOVSt_aux := TRUE;
      ELSE
        #OutOnOVSt := FALSE;
        #OutOffOVSt := #PulseOff;
        #OutOVSt_aux := FALSE;
      END_IF;
    ELSE
      #OutOnOVSt := (#FFsPosOn AND NOT #FFsPosOn2) OR (#FFsPosOn AND #FFsPosOn2);
      #OutOffOVSt := (NOT #FFsPosOn AND NOT #FFsPosOn2) OR (#FFsPosOn AND #FFsPosOn2);
      #OutOVSt_aux := #FFsPosOn;
    END_IF;
  END_IF;
ELSE
  IF #InterlockR THEN
    #OutOnOVSt := #FFsPosOn;
    #OutOVSt_aux := #FFsPosOn;
  END_IF;
END_IF;
```

Figure 3: PLC Structured Text program

## PLC Scan Cycle simplified behaviour:

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

# Motivation: Critical PLC Installations at CERN

**Critical applications:** Cryogenics, ventilation, vacuum systems.

**PLC software failures have significant negative consequences**



# Motivation: Critical PLC Installations at CERN

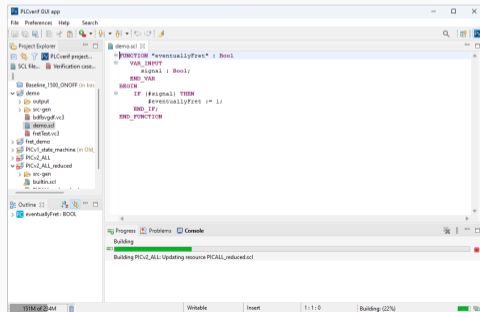
**Critical applications:** Cryogenics, ventilation, vacuum systems.

**PLC software failures have significant negative consequences**

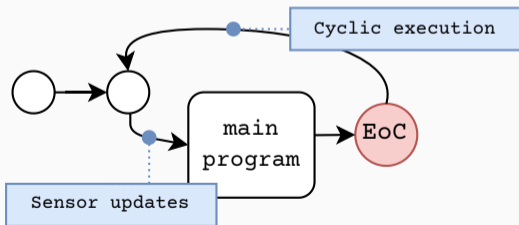
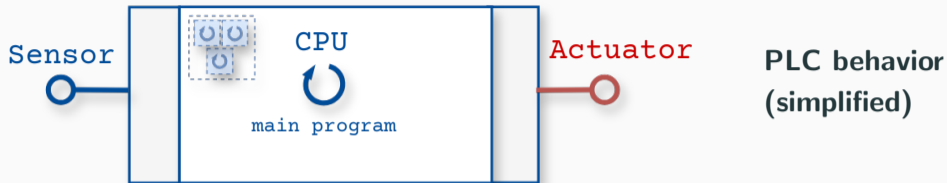


## CERN PLCverif

- Formal verification tool for PLC programs actively developed and used at CERN
- Complements traditional testing for systems of higher criticality



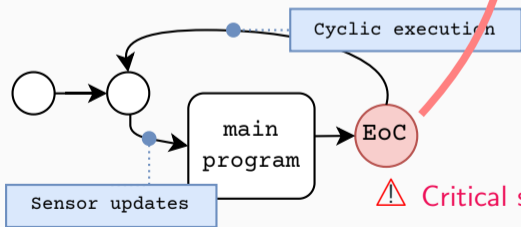
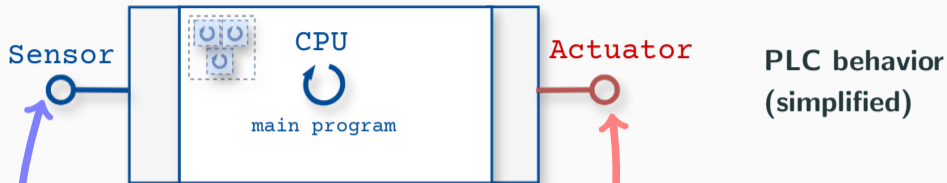
# Motivation: PLC model for verification



## PLCverif abstraction

- Infinite program loop
- Sensor signals modelled as non-deterministic variables
- Critical system state at actuator updates

# Motivation: PLC model for verification

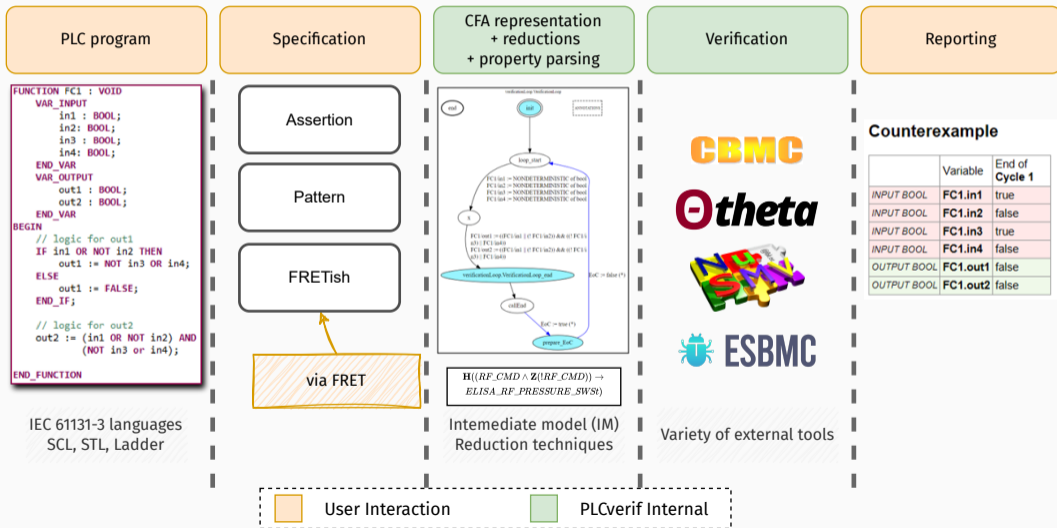


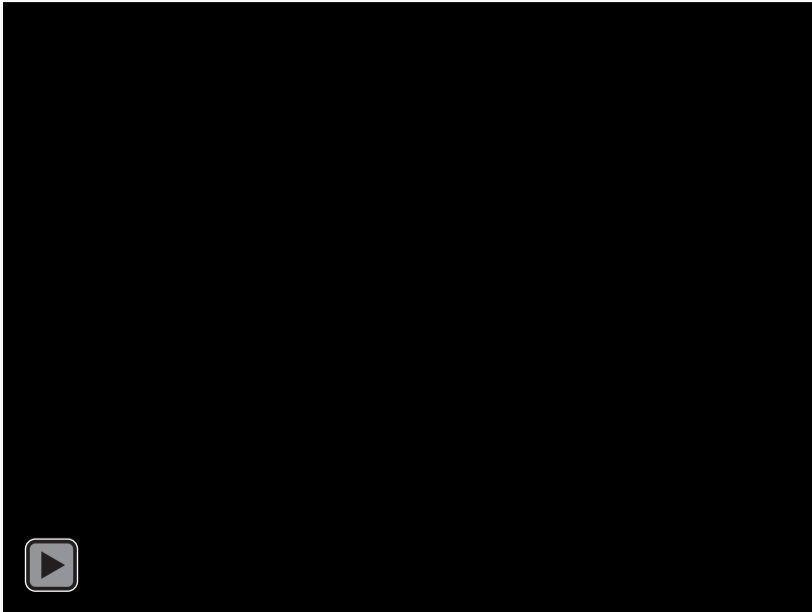
## PLCverif abstraction

- Infinite program loop
- Sensor signals modelled as non-deterministic variables
- Critical system state at actuator updates

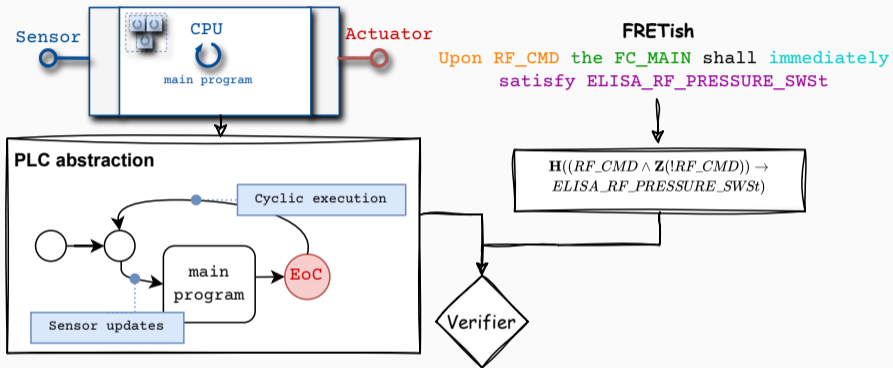
# Motivation: PLCverif pipeline

Main structure: Pipeline from PLC source code → Verification report

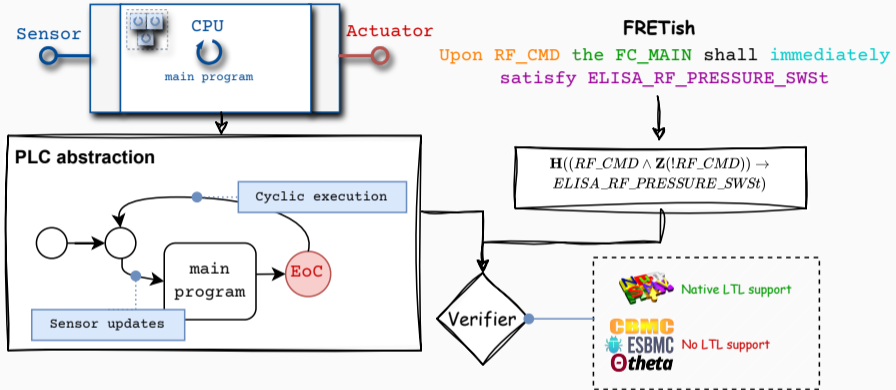




# Problem Setup



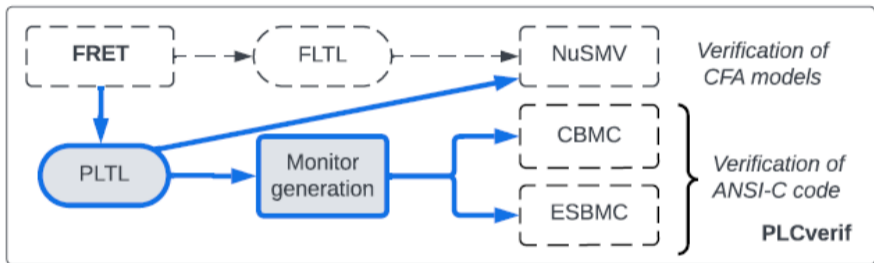
# Problem Setup



**Problem:** not all verification backends support Linear Temporal Logic (LTL) specifications ⚠

A prior work integrated the tool FRET into PLCverif<sup>1</sup>

→ ⚠ LTL property verification was only possible via NuSMV

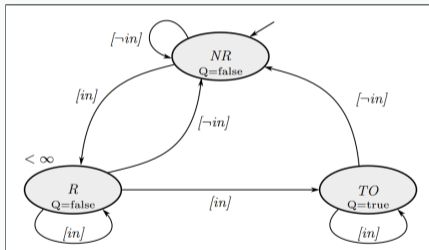


## Our contribution:

1. Extend PLCverif with an algorithm for monitor-based verification of LTL properties via Bounded Model Checking (BMC) assertion-verification
2. Validate the algorithm on two CERN case studies

**Goal:** LTL-property verification via PLCverif with arbitrary verification backends

→ via problem instance modification, **without** verification algorithm modification



**Figure 4:** “ON delay timer” property modeled as state machine<sup>1</sup>

## Prior manual approach:

1. Create state machine for property
2. Extend PLC program with the state machine
3. Verify assertion over the state machine

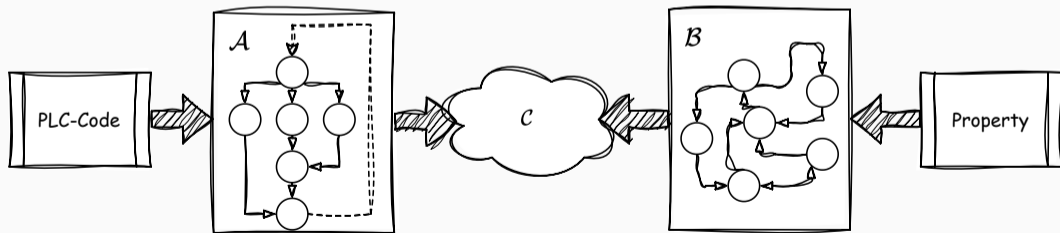
**How can we generalize?**

# Problem Definition: Generalized Algorithm

**Goal:** LTL-property verification via PLCverif with arbitrary verification backends

**Planned generalized pipeline:**

1. Transform PLC code to CFA  $\mathcal{A}$  + LTL to monitor automaton  $\mathcal{B}$
2. Merge  $\mathcal{A}$  and  $\mathcal{B}$  to combined CFA  $\mathcal{C}$
3. Verify merged automaton  $\mathcal{C}$

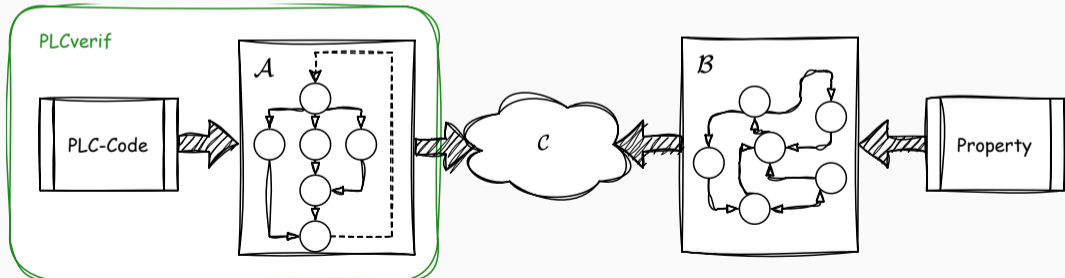


# Problem Definition: Generalized Algorithm

**Goal:** LTL-property verification via PLCverif with arbitrary verification backends

**Planned generalized pipeline:**

1. Transform PLC code to CFA  $\mathcal{A}$  + LTL to monitor automaton  $\mathcal{B}$
2. Merge  $\mathcal{A}$  and  $\mathcal{B}$  to combined CFA  $\mathcal{C}$
3. Verify merged automaton  $\mathcal{C}$



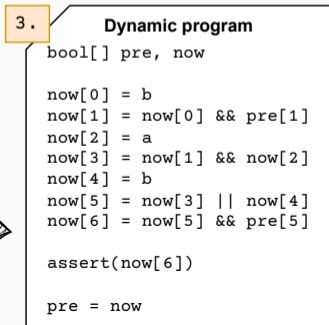
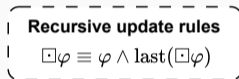
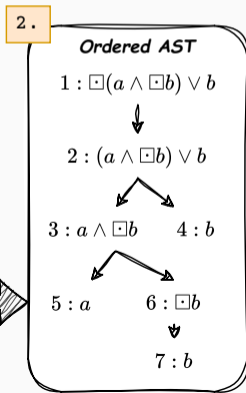
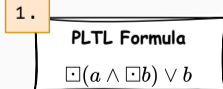
## 1. Transform PLC code to CFA $\mathcal{A}$ + **LTL to monitor automaton $\mathcal{B}$**

Monitoring algorithm by Havelund & Rosu <sup>1</sup> based on dynamic programming

**Key idea:** PLTL evaluation of  $w, i \models \varphi$  only depends on current and last state

Restricts to:

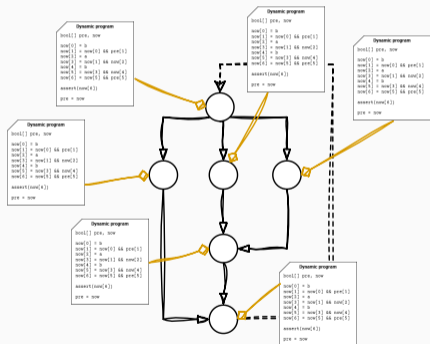
- Safety properties
- PLTL



# Verification Pipeline: 2. Merging

## 2. Merge $\mathcal{A}$ and $\mathcal{B}$ to combined CFA $\mathcal{C}$

Naive approach: Insert dynamic program computation after each assignment



**Problem: Efficiency** ⚠

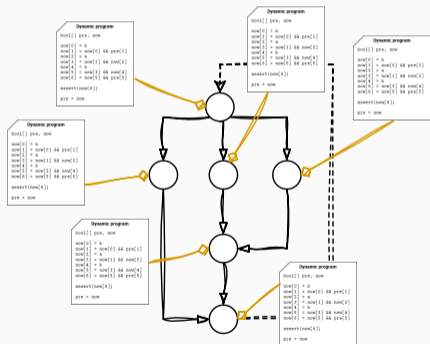
We tackle this problem via:

- Sparse updates based on COI
- State-space reduction on  $\mathcal{A}$

# Verification Pipeline: 2. Merging

## 2. Merge $\mathcal{A}$ and $\mathcal{B}$ to combined CFA $\mathcal{C}$

Naive approach: Insert dynamic program computation after each assignment



**Problem: Efficiency** ⚠

We tackle this problem via:

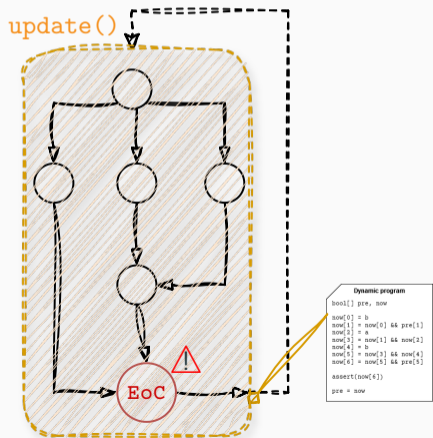
- Sparse updates based on COI
- State-space reduction on  $\mathcal{A}$

# Verification Pipeline: 2. Merging

## 2. Merge $\mathcal{A}$ and $\mathcal{B}$ to combined CFA $\mathcal{C}$ – State-space reduction by abstraction

**Key observation:** PLC programs do not return intermediate cycle values

*! domain-specific assumption*



**Idea:** PLC-update as blackbox `update()` function  
→ Implicitly transforms  $\varphi$  into *cycle-end* →  $\varphi$

Property structure restriction  
vs  
Runtime efficiency

# Verification Pipeline: 3. Verification

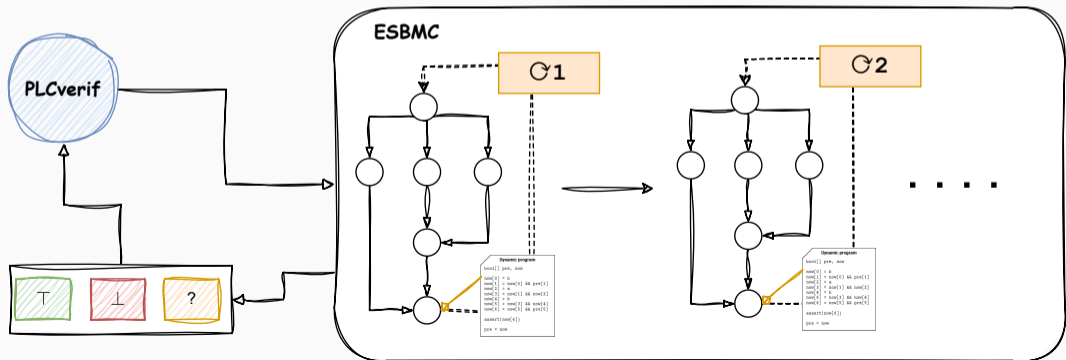
## 3. Verify merged automaton $\mathcal{C}$

**Idea:** Hand-over to verifiers and verify assertion over the monitor

**Observation:** Only PLC-cycle loop is unbounded

*! domain-specific*

→ (i) fully unroll inner loops and (ii) perform BMC over PLC-cycle



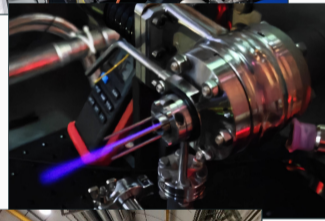
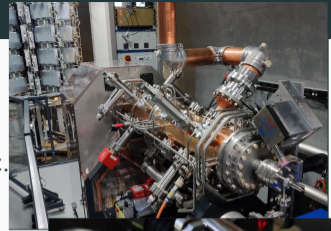
# Case Studies: ELISA & UNICOS OnOff Library

## ELISA (Experimental Linear accelerator for Surface Analysis)

- PLC running a safety program: protection against voltage, etc.

## UNICOS OnOff Library (industrial control system framework)

- Deployed in hundreds of industrial installations at CERN



Tool		EL2		OnOff2	
		time	result	time	result
<i>new!</i> ESBMC	k-Ind	1.42	T	4.57	⊥
	BMC	TO(810)	T	3.21	⊥
NuSMV	BDD	0.91	T	TO	—
	BMC	TO(155)	T	42.83	⊥

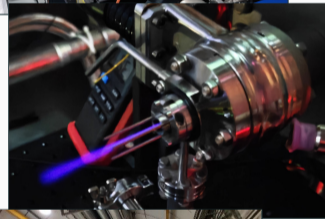
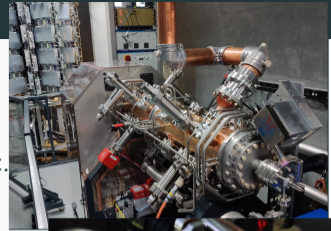
# Case Studies: ELISA & UNICOS OnOff Library

## ELISA (Experimental Linear accelerator for Surface Analysis)

- PLC running a safety program: protection against voltage, etc.

## UNICOS OnOff Library (industrial control system framework)

- Deployed in hundreds of industrial installations at CERN



<i>Tool</i>		EL2		OnOff2	
		time	result	time	result
<i>new!</i> ESBMC	k-Ind	1.42	T	4.57	⊥
	BMC	TO(810)	T	<b>3.21</b>	⊥
NuSMV	BDD	0.91	T	TO	—
	BMC	TO(155)	T	<b>42.83</b>	⊥

# Conclusion & Future Work

## Conclusions

- Extending PLCverif-FRET integration:
  - now **any safety PLTL formula with any verification backend**
- **(PLC) Domain-specific information was used** to improve verification performance on two critical CERN case studies
- The experiments show **up to one order of magnitude** improved verification times;
  - additionally **verifying previously unverifiable properties**

## Future Work

- Experiment with other monitor types, e.g., NBA-based monitors
- Further evaluate an extension of the monitoring algorithm to metric PLTL for PLC timing modeling

**Learn more:**

**PLCverif**

[https://gitlab.com/  
plcverif-oss](https://gitlab.com/plcverif-oss)

**FRET**

[https://github.com/  
NASA-SW-VnV/fret](https://github.com/NASA-SW-VnV/fret)