Formal Requirement Elicitation with FRET

Anastasia Mavridou
KBR at NASA Ames Research Center

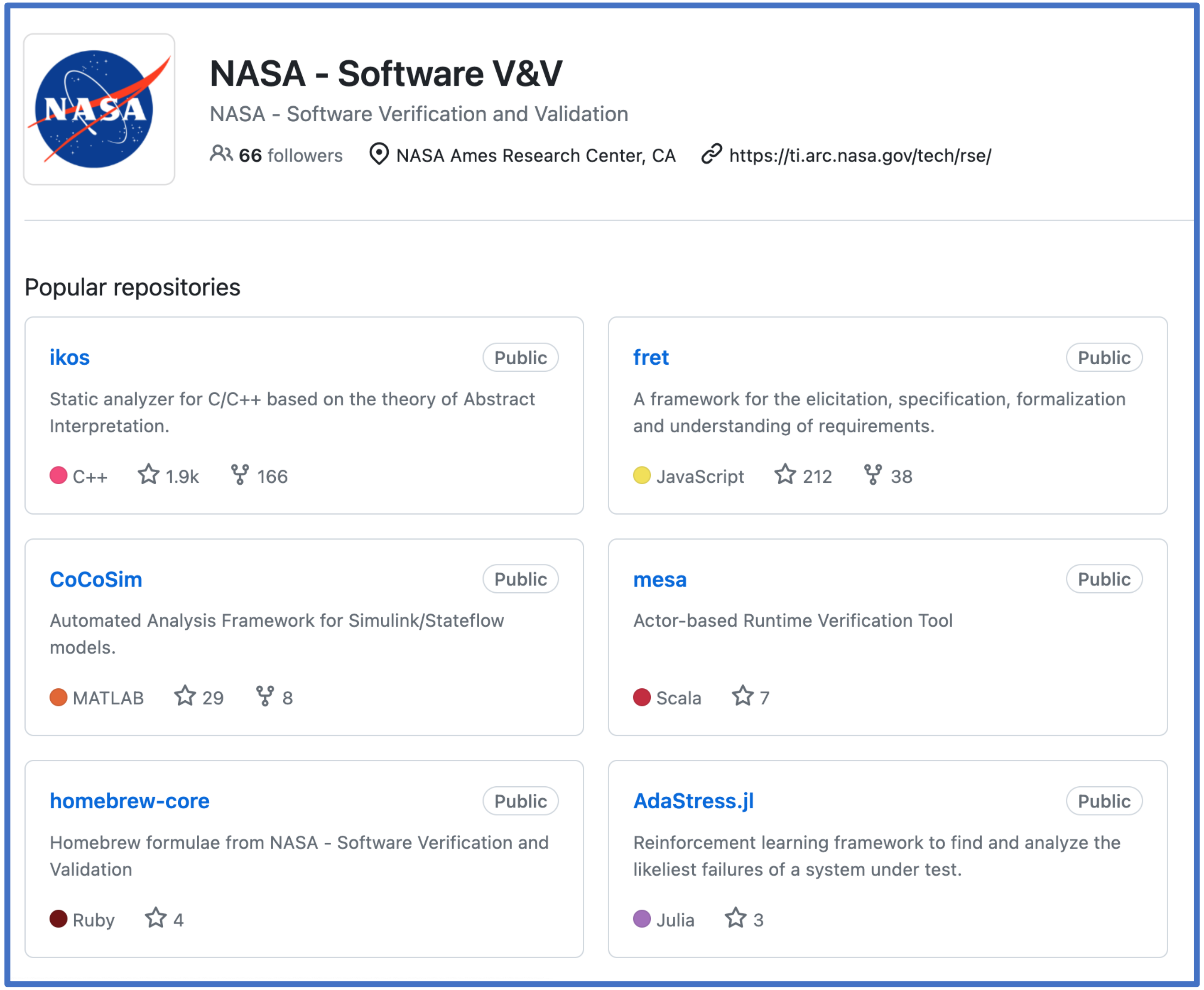
anastasia.mavridou@nasa.gov

Invited talk at DTU

NASA’s center in Silicon Valley

Aerial image of NASA Ames Research Center
Credits: NASA

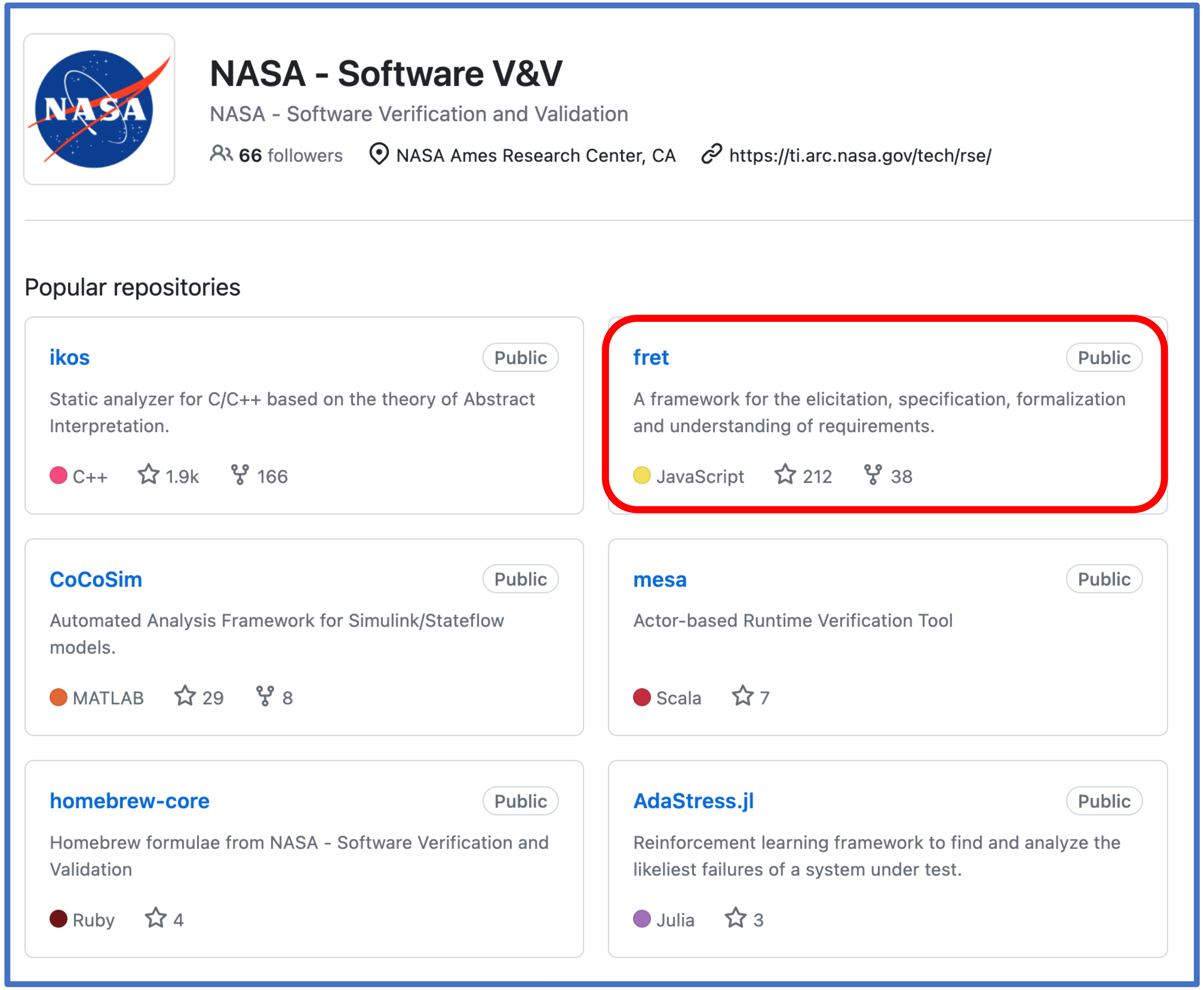
robust software engineering technical area



github.com/NASA-SW-VnV

www.nasa.gov/isd-robust-software-engineering

robust software engineering technical area



github.com/NASA-SW-VnV

www.nasa.gov/isd-robust-software-engineering

how developers write requirements

10 Lockheed Martin Cyber-Physical System Challenge, component FSM:

- Exceeding sensor limits shall latch an autopilot pullup when the pilot is not in control (not standby) and the system is supported without failures (not apfail).
- The autopilot shall change states from TRANSITION to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from TRANSITION to NOMINAL when the system is supported and sensor data is good.
- The autopilot shall change states from NOMINAL to MANEUVER when the sensor data is not good.
- The autopilot shall change states from NOMINAL to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from MANEUVER to STANDBY when the pilot is in control (standby) and sensor data is good.

how developers write requirements

10 Lockheed Martin Cyber-Physical System Challenge, component FSM:

Every time these conditions hold or only when they become true?

- Exceeding sensor limits shall latch an autopilot pullup when the pilot is not in control (not standby) and the system is supported without failures (not apfail).
- The autopilot shall change states from TRANSITION to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from TRANSITION to NOMINAL when the system is supported and sensor data is good.
- The autopilot shall change states from NOMINAL to MANEUVER when the sensor data is not good.
- The autopilot shall change states from NOMINAL to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from MANEUVER to STANDBY when the pilot is in control (standby) and sensor data is good.

how developers write requirements

10 Lockheed Martin Cyber-Physical System Challenge, component FSM:

- Exceeding sensor limits shall latch an autopilot pullup when the system is supported without failures (not apfail).
- The autopilot shall change states from TRANSITION to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from TRANSITION to NOMINAL when the system is supported and sensor data is good.
- The autopilot shall change states from NOMINAL to MANEUVER when the sensor data is not good.
- The autopilot shall change states from NOMINAL to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from MANEUVER to STANDBY when the pilot is in control (standby) and sensor data is good.
- …

Instantly, or within a time limit?

what analysis tools understand

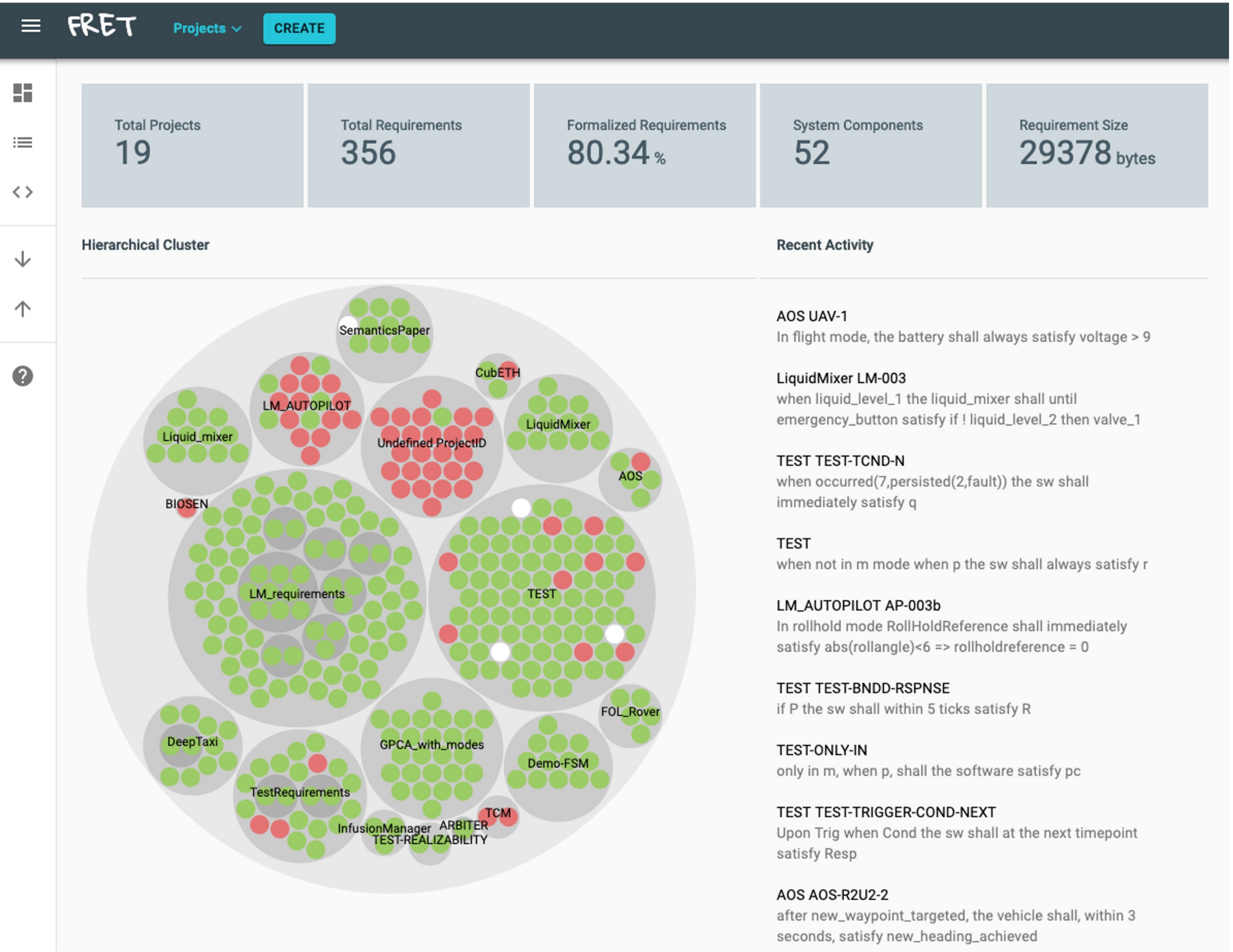
10 Lockheed Martin Cyber-Physical System Challenge, component FSM:

```
var autopilot: bool = (not standby) and supported and (not
    apfail);
var pre_autopilot: bool = false -> pre autopilot;
var pre_limits: bool = = false -> pre limits;
guarantee "FSM-001v2" S((((((autopilot and pre_autopilot and
    pre_limits) and (pre ( not (autopilot and pre_autopilot and
    pre_limits)))) or ((autopilot and pre_autopilot and
    pre_limits) and FTP)) => (pullup)) and FTP), ((((autopilot
    and pre_autopilot and pre_limits) and (pre ( not (autopilot
    and pre_autopilot and pre_limits)))) or ((autopilot and
    pre_autopilot and pre_limits) and FTP)) => (pullup)));
```

FRET bridges the gap

- Captures requirements in structured natural language with unambiguous semantics.
- Explains formal semantics in various forms.
- Formalizes requirements in a compositional (hence extensible) manner.
- Checks realizability of requirements compositionally.
- Connects with analysis tools:
  - Exports formalizations in SMV language.
  - Exports Lustre code.
  - Exports specifications for runtime monitoring.

welcome to FRET



github.com/NASA-SW-VnV/fret

Team: Andreas Katis, Anastasia Mavridou, Tom Pressburger, Johann Schumann, Khanh Trinh

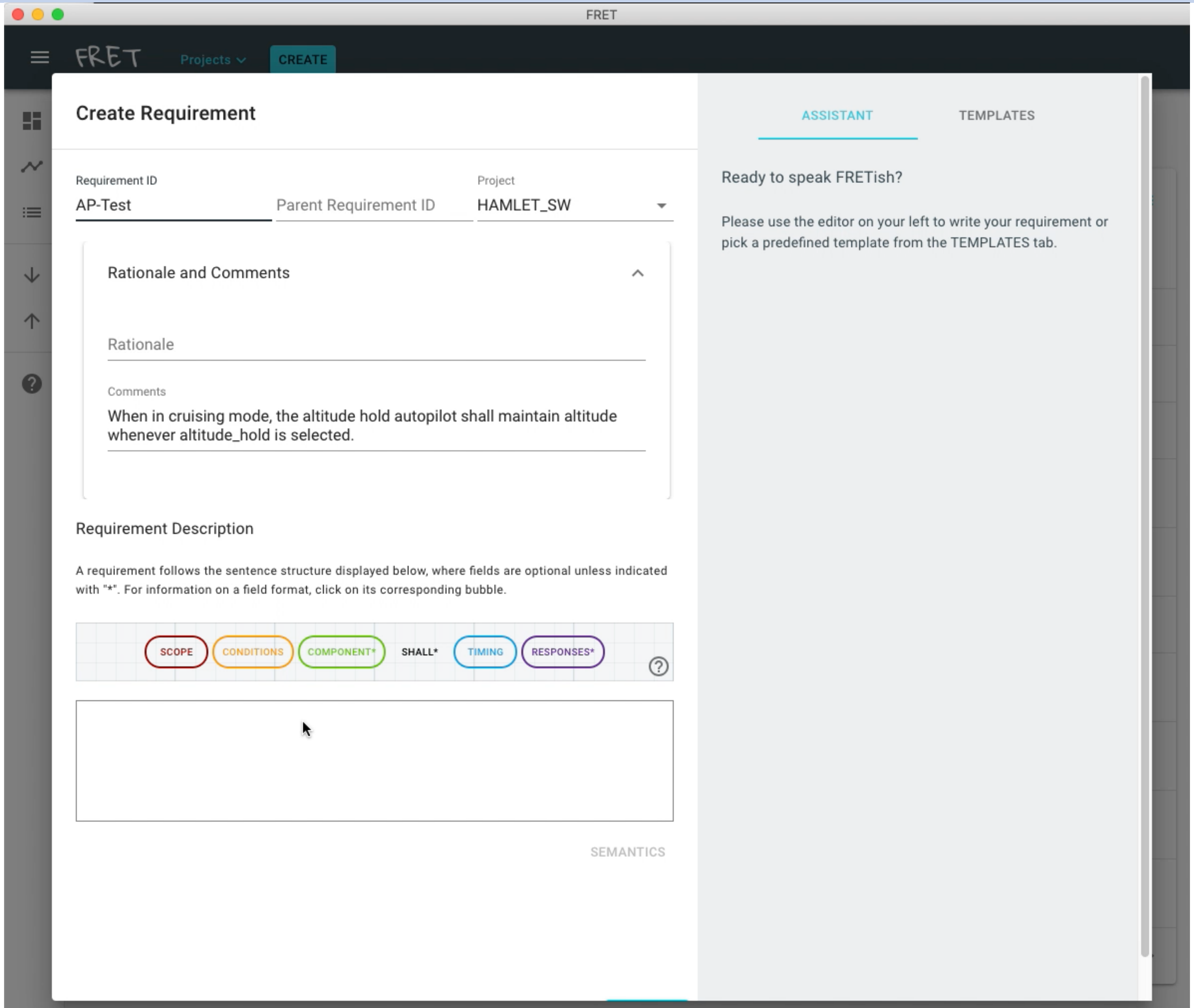
Alumni: David Bushnell, Dimitra Giannakopoulou, Nija Shi

Interns: Milan Bhandari, Tanja DeJong, Kelly Ho, George Karamanolis, David Kooi, Jessica Phelan, Julian Rhein, Daniel Riley, Gricel Vazquez

And many other collaborators..

capturing, explaining, and formalizing requirements

let’s speak FRETish



FRETish fields

In cruising mode, the autopilot shall always satisfy if altitude_hold then maintain_altitude

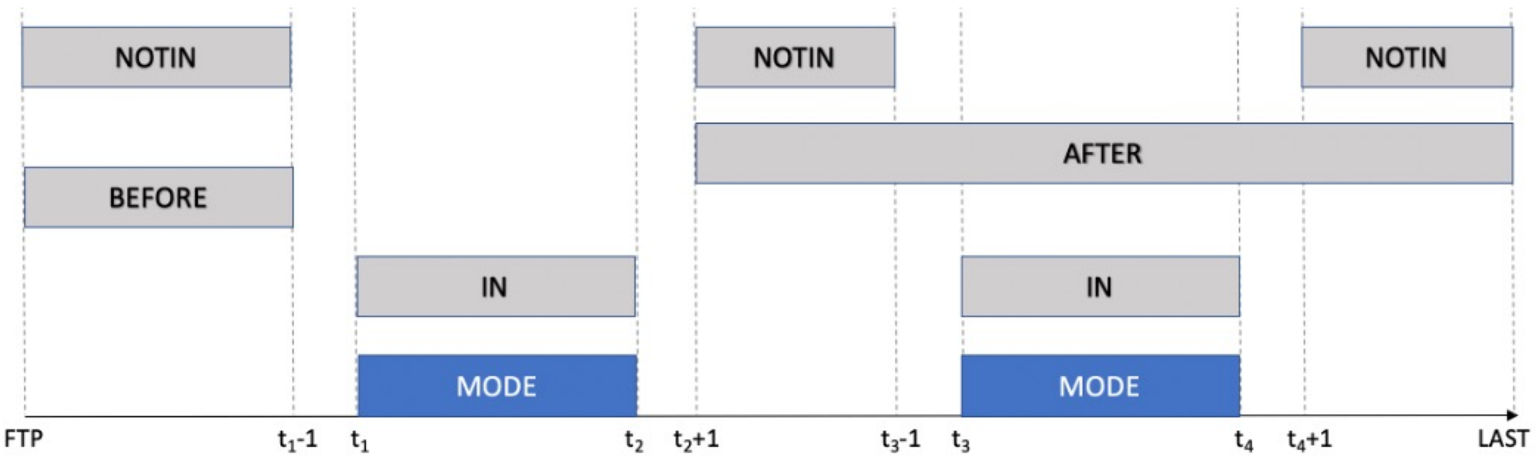
SCOPE	in, before, after, notin, onlyIn, onlyBefore, onlyAfter, null (global)
CONDITION	null, regular
TIMING	always, never, eventually, immediately, for, within, after, until, before
RESPONSE	satisfaction

compositional generation of LTL formulas

In cruising mode, the autopilot shall always satisfy if altitude_hold then maintain_altitude

scope in: [LEFT, RIGHT) → [FiM, LiM)

```
FiM = MODE and (FTP or previous (not MODE))
LiM = not MODE and previous MODE
```



compositional generation of LTL formulas

In cruising mode, the autopilot shall always satisfy if altitude_hold then maintain_altitude

scope in: [LEFT, RIGHT) → [FiM, LiM)

```
FiM = MODE and (FTP or previous (not MODE))
LiM = not MODE and previous MODE
```

timing always: BASEFORM → RES

```
historically (RIGHT implies previous (BASEFORM since inclusive required LEFT))
```

scope: in, condition: null, timing: always, response: satisfaction

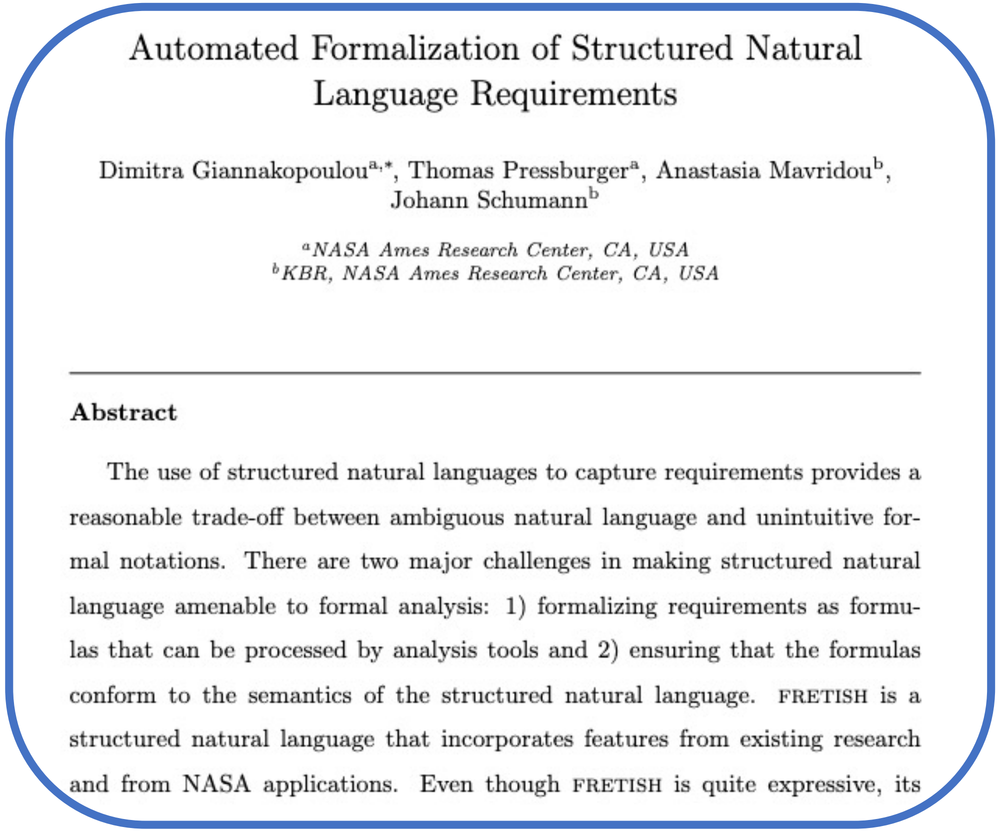
```
historically (LiM implies previous (RES since inclusive required FiM))
```

optimize `historically (MODE implies RES)`

translate to SMV `(H (MODE → RES))`

instantiate `(H (cruising → (altitude_hold → maintain_altitude)))`

related papers



Automated Formalization of Structured Natural Language Requirements

Dimitra Giannakopoulou[a,*], Thomas Pressburger[a], Anastasia Mavridou[b], Johann Schumann[b]

[a]*NASA Ames Research Center, CA, USA*
[b]*KBR, NASA Ames Research Center, CA, USA*

Abstract

The use of structured natural languages to capture requirements provides a reasonable trade-off between ambiguous natural language and unintuitive formal notations. There are two major challenges in making structured natural language amenable to formal analysis: 1) formalizing requirements as formulas that can be processed by analysis tools and 2) ensuring that the formulas conform to the semantics of the structured natural language. FRETISH is a structured natural language that incorporates features from existing research and from NASA applications. Even though FRETISH is quite expressive, its

Dimitra Giannakopoulou, Thomas Pressburger, Anastasia Mavridou, Johann Schumann (2021). Automated formalization of structured natural language requirements, Information and Software Technology (IST) Journal, 137, 106590, Special Section on REFSQ'20, 2021.

Dimitra Giannakopoulou, Thomas Pressburger, Anastasia Mavridou, Johann Schumann. Generation of Formal Requirements from Structured Natural Language, REFSQ 2020.

checking realizability of requirements

even simple requirements can be conflicting

10 Lockheed Martin Cyber-Physical System Challenge, component FSM:

- The autopilot shall change states from TRANSITION to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from TRANSITION to NOMINAL when the system is supported and sensor data is good.

even simple requirements can be conflicting

10 Lockheed Martin Cyber-Physical System Challenge, component FSM:

- The autopilot shall change states from TRANSITION to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from TRANSITION to NOMINAL when the system is supported and sensor data is good.

- Input state: TRANSITION

even simple requirements can be conflicting

10 Lockheed Martin Cyber-Physical System Challenge, component FSM:

- The autopilot shall change states from TRANSITION to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from TRANSITION to NOMINAL when the system is supported and sensor data is good.

- Input state: TRANSITION
- Condition 1: standby
- Condition 2: supported & good_sensor_data

even simple requirements can be conflicting

10 Lockheed Martin Cyber-Physical System Challenge, component FSM:

- The autopilot shall change states from TRANSITION to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from TRANSITION to NOMINAL when the system is supported and sensor data is good.

- Input state: TRANSITION
- Condition 1: standby ✓
- Condition 2: supported & good_sensor_data ✓

even simple requirements can be conflicting

10 Lockheed Martin Cyber-Physical System Challenge, component FSM:

- The autopilot shall change states from TRANSITION to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from TRANSITION to NOMINAL when the system is supported and sensor data is good.

- Input state: TRANSITION
- Condition 1: standby ✓
- Condition 2: supported & good_sensor_data ✓
- Output state 1: STANDBY
- Output state 2: NOMINAL

why realizability?

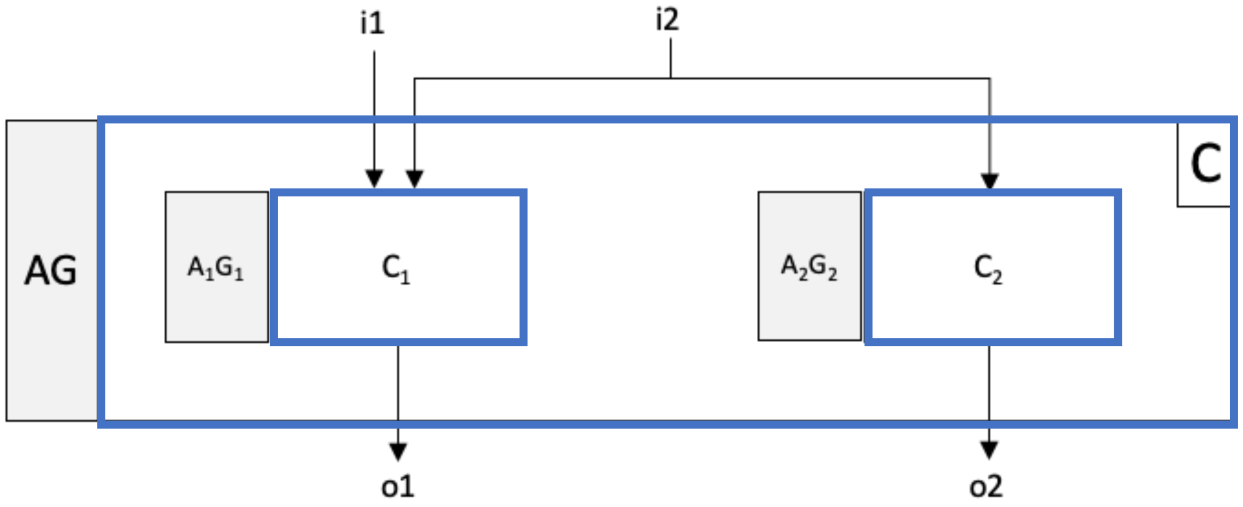
- Defining requirements is a challenging, error prone task
- Realizability checking >> consistency checking
- We want to ensure requirement consistency for all inputs
- And we want to do it efficiently

An AG contract is realizable if there exists a system implementation that satisfies the contract guarantees for all assumption-complying stimuli provided by the environment.

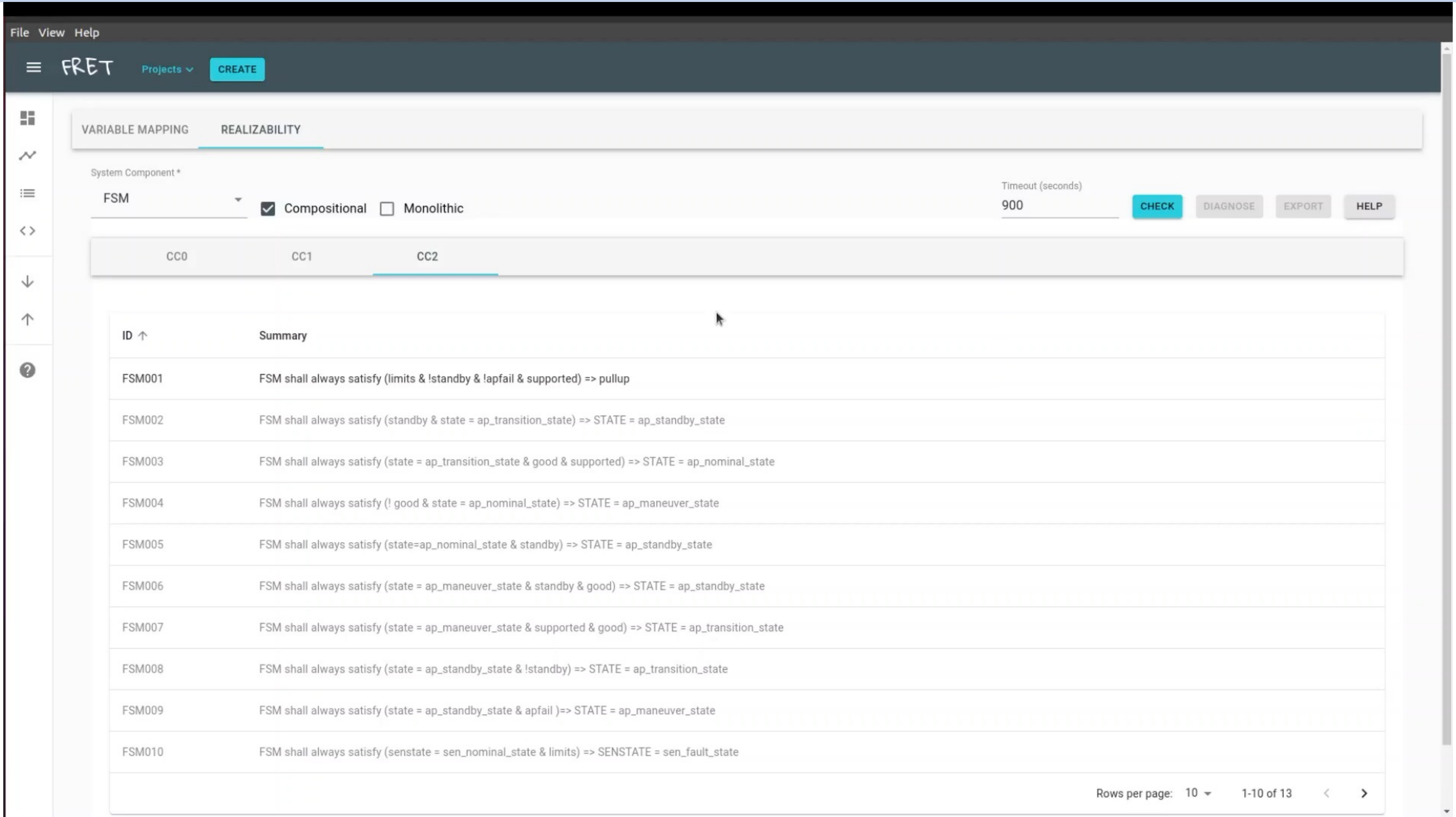
We proposed a novel approach for compositional realizability checking.

compositional realizability

Partial AG contracts:



checking realizability within FRET



related papers

From Partial to Global Assume-Guarantee Contracts: Compositional Realizability Analysis in FRET

Anastasia Mavridou[1], Andreas Katis[1], Dimitra Giannakopoulou[2], David Kooi[3], Thomas Pressburger[2], and Michael W. Whalen[4]

[1] KBR, NASA Ames Research Center, CA, USA
[2] NASA Ames Research Center, CA, USA
{anastasia.mavridou, andreas.katis, dimitra.giannakopoulou, tom.pressburger}@nasa.gov
[3] University of California, Santa Cruz, CA, USA dkooi@ucsc.edu
[4] University of Minnesota, MN, USA whalen@cs.umn.edu

Abstract. Realizability checking refers to the formal procedure that aims to determine whether an implementation exists, always complying to a set of requirements, regardless of the stimuli provided by the system's environment. Such a check is essential to ensure that the specification does not allow behavior that can force the system to violate safety constraints. In this paper, we present an approach that decomposes realizability checking into smaller, more tractable problems. More

Capture, Analyze, Diagnose: Realizability Checking of Requirements in FRET

Andreas Katis[1[0000-0001-7013-1100]], Anastasia Mavridou[1], Dimitra Giannakopoulou[2*], Thomas Pressburger[2], and Johann Schumann[1]

[1] Employed by KBR; NASA Ames Research Center, CA, USA
[2] NASA Ames Research Center, CA, USA

Abstract. Requirements formalization has become increasingly popular in industrial settings as an effort to disambiguate designs and optimize development time and costs for critical system components. Formal requirements elicitation also enables the employment of analysis tools to prove important properties, such as consistency and realizability. In this paper, we present the realizability analysis framework that we developed as part of the Formal Requirements Elicitation Tool (FRET). Our

Andreas Katis, Anastasia Mavridou, Dimitra Giannakopoulou, Thomas Pressburger, Johann Schumann. Capture, Analyze, Diagnose: Realizability Checking of Requirements in FRET, CAV 2022.

Anastasia Mavridou, Andreas Katis, Dimitra Giannakopoulou, David Kooi, Thomas Pressburger, Michael W. Whalen. From Partial to Global Assume-Guarantee Contracts: Compositional Realizability Analysis in FRET, FM 2021.

connection with analysis tools

generation of Simulink monitors

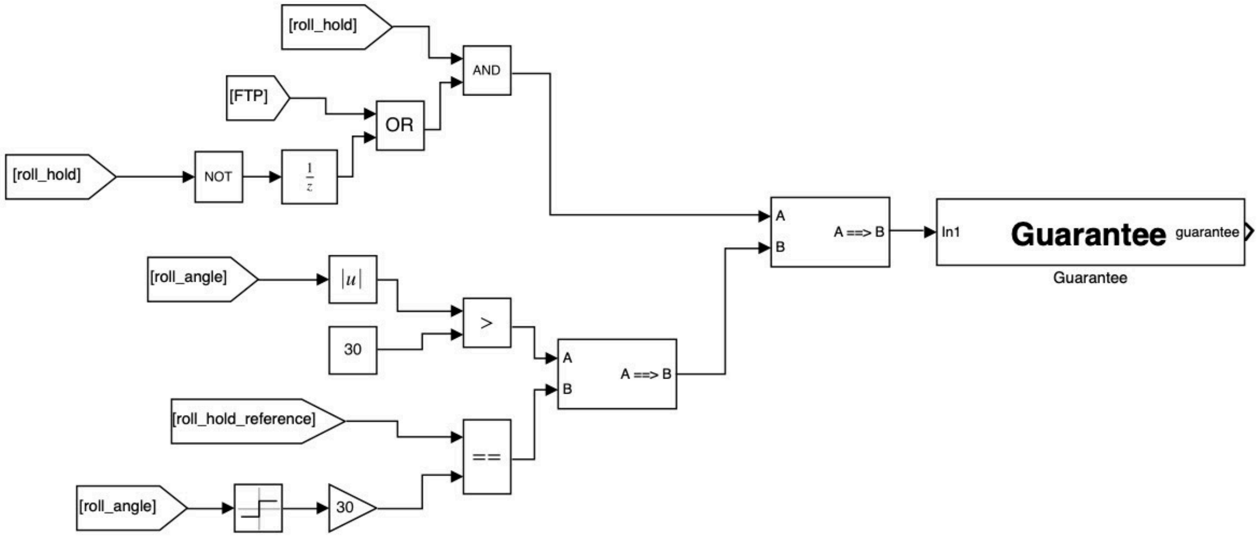
FRETish:

when in roll_hold_mode autopilot shall immediately satisfy if roll_angle > 3 then roll_hold_reference = 3

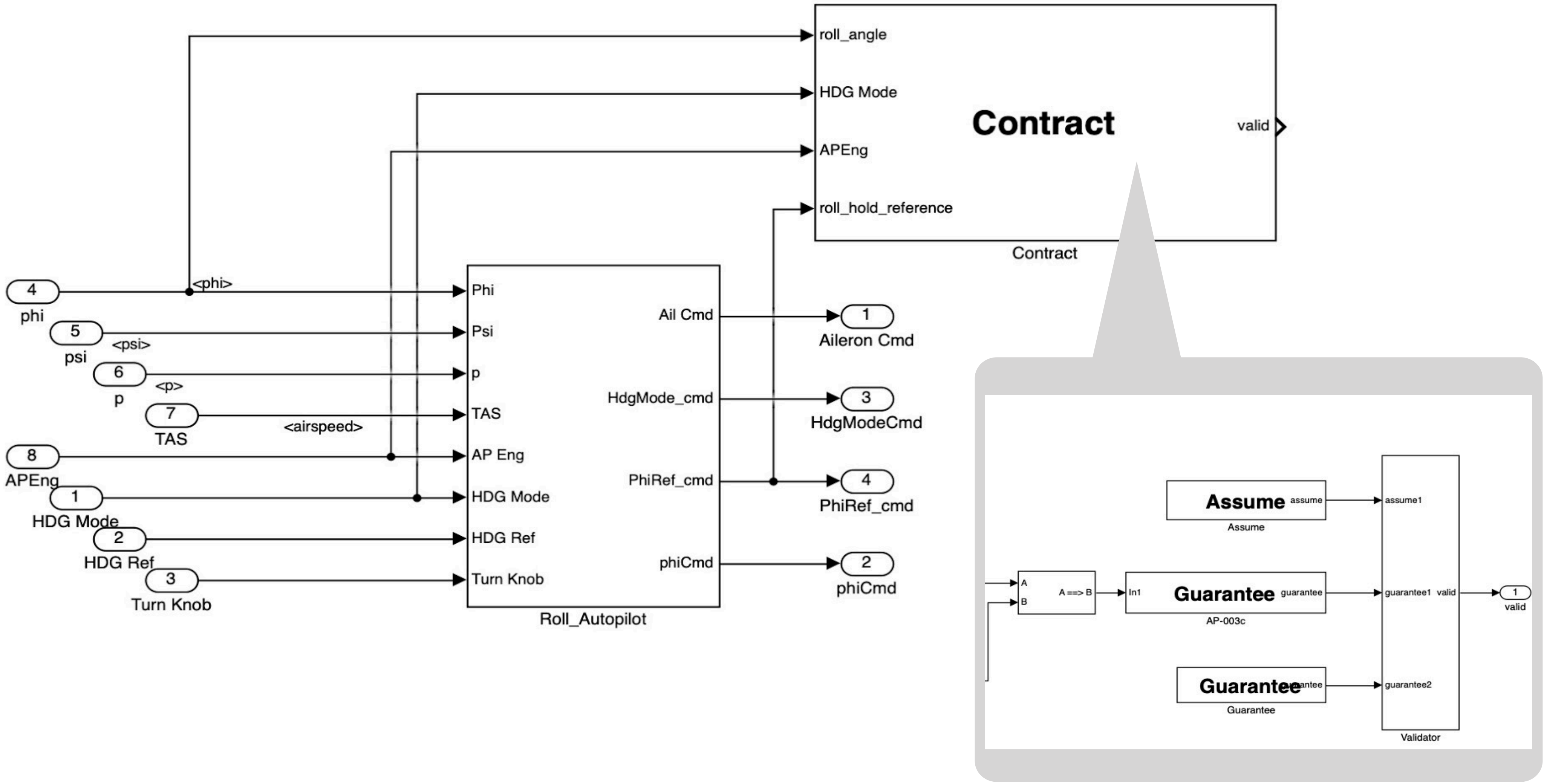
Lustre specification:

```
-- AP-003c-v3 requirement in CoCoSpec
guarantee H((roll_hold and (FTP or (pre (not roll_hold))))
            => abs(roll_angle) > 30 =>
            roll_hold_reference = 30 * sign(roll_angle))
```

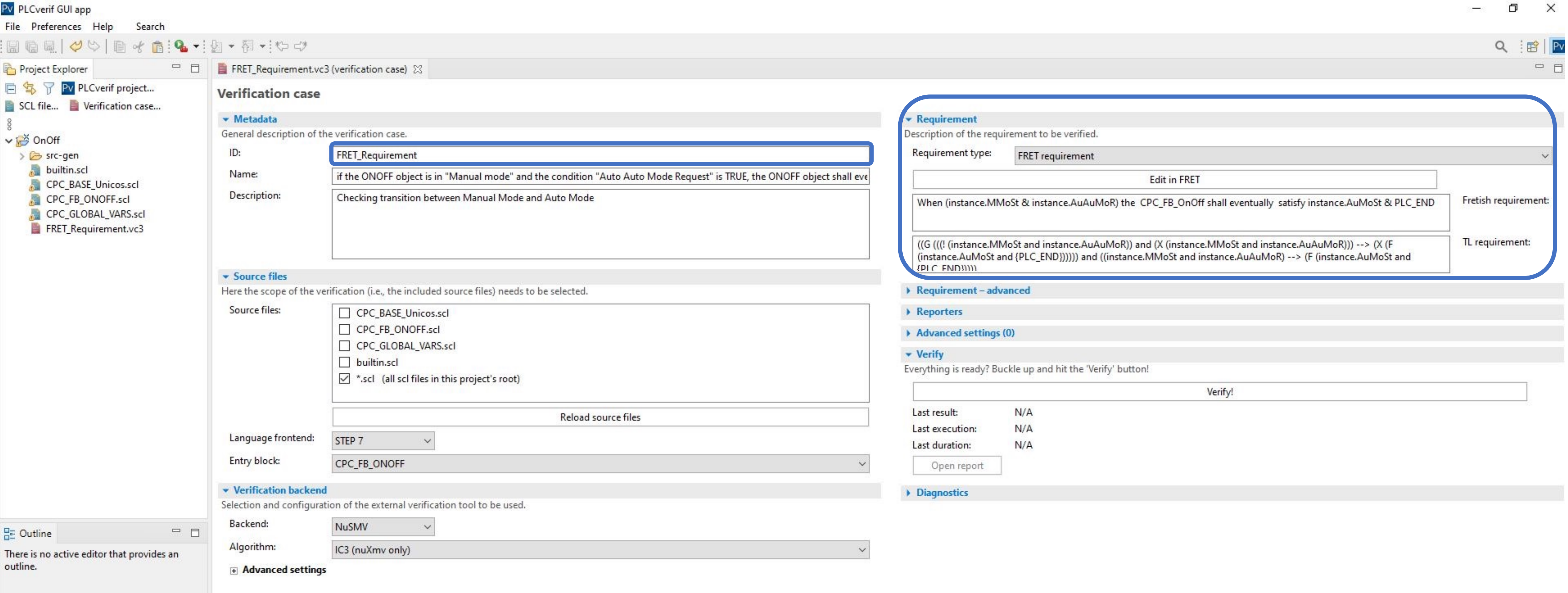
Simulink monitor



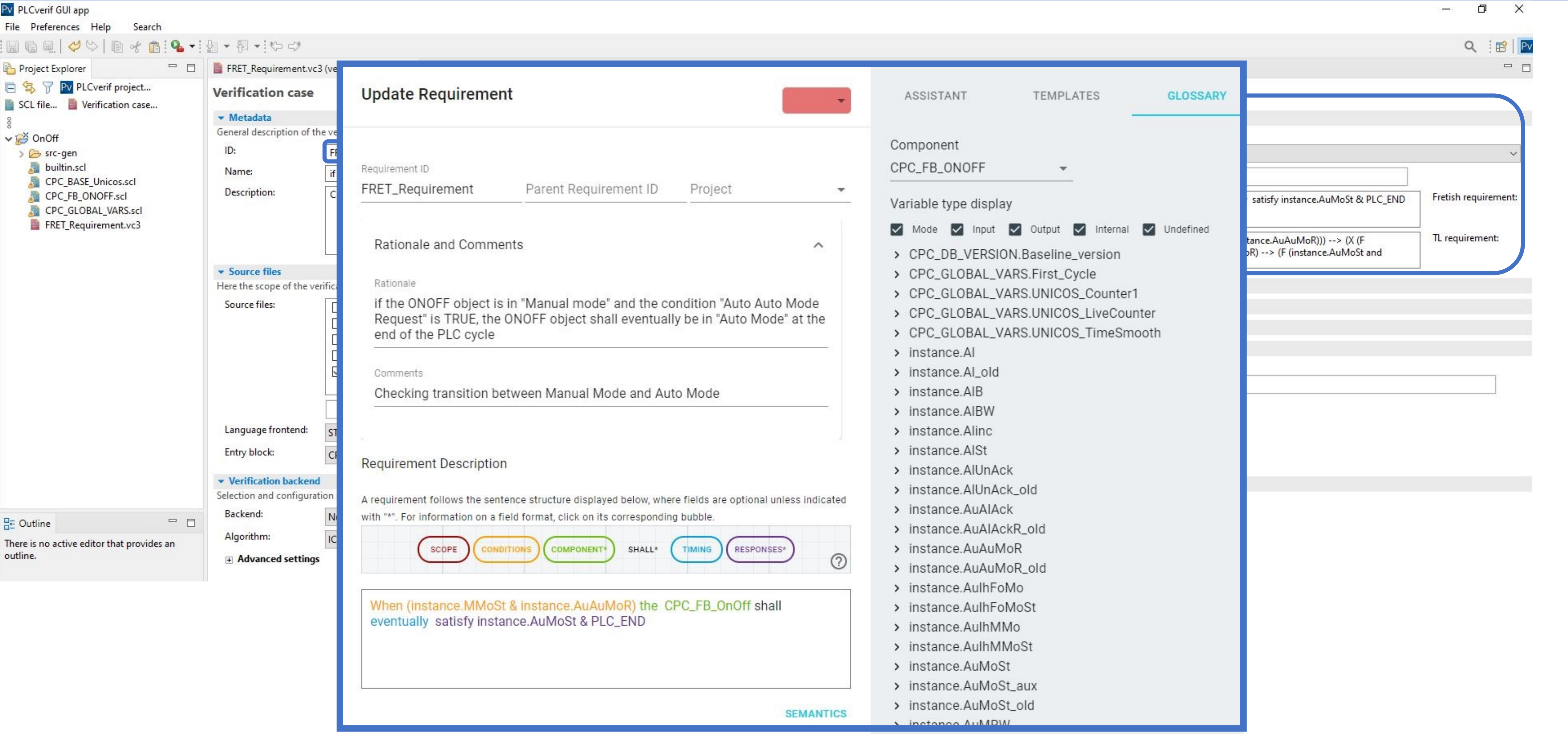
model checking Simulink models



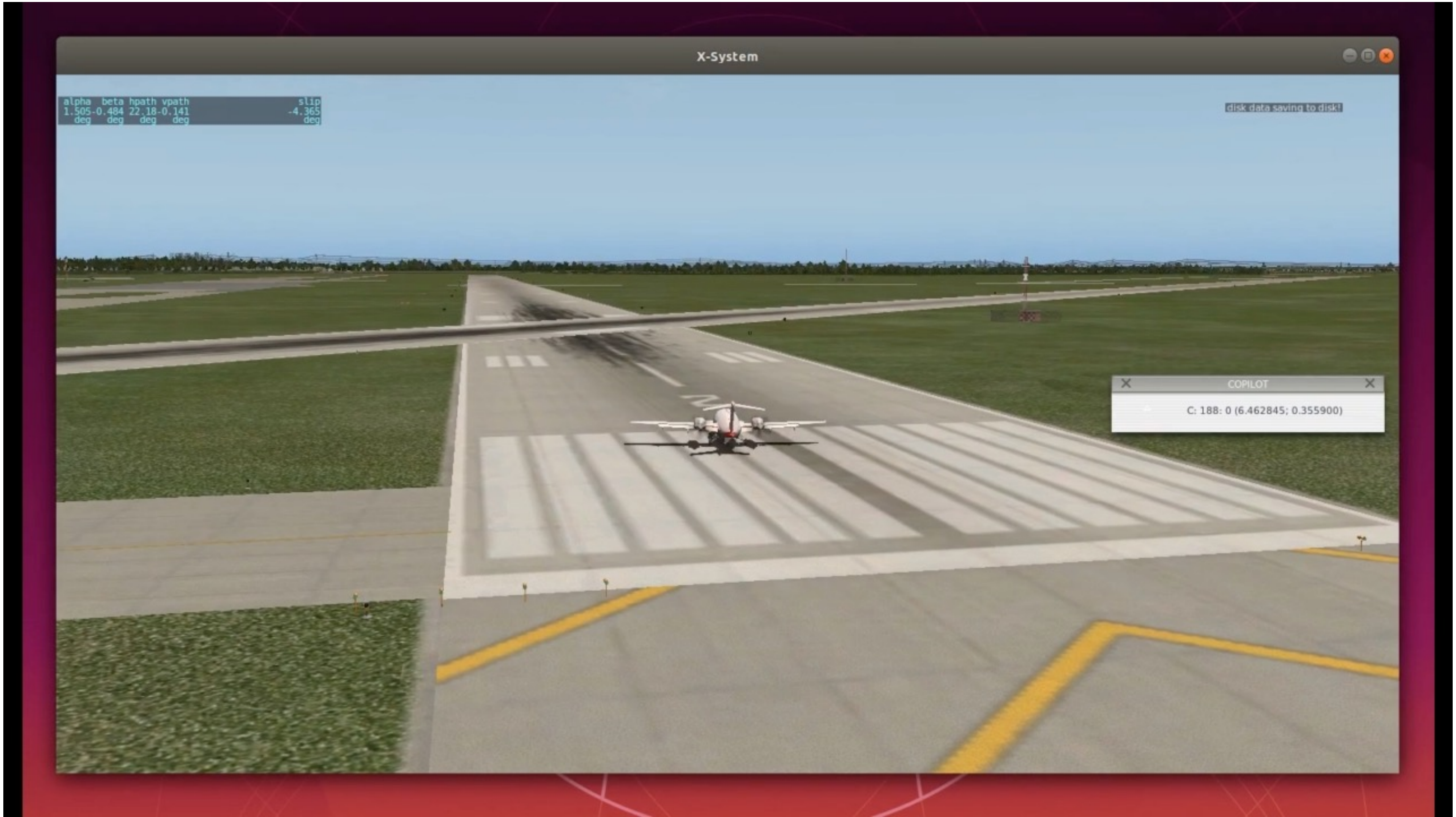
model checking PLC code



model checking PLC code



runtime monitoring



examples of case studies/projects that use FRET

|galois|

High Assurance Rigorous Digital Engineering for Nuclear Safety (HARDENS)

Joe Kiniry, Galois (kiniry@galois.com)

May 2022

Theme: Driving FM to Practice

Keywords: digital engineering, model-based engineering, software engineering, hardware engineering, safety engineering, requirements engineering, formal verification, rigorous runtime verification, Cryptol, SAW, ACSL, SysML, FRET, RISC-V

This work is supported by the U.S. Nuclear Regulatory Commission (NRC), Office of Nuclear Regulatory Research, under contract/order number 31310021C0014.

related papers

Zsófia Ádám, Ignacio D. Lopez-Miguel, Anastasia Mavridou, Thomas Pressburger, Marcin Bęś, Enrique Blanco Viñuela, Andreas Katis, Jean-Charles Tournier, Khanh V. Trinh, Borja Fernandez Adiego. From Natural Language Requirements to the Verification of Programmable Logic Controllers: Integrating FRET into PLCverif, NFM 2023.

Joseph Kiniry, Alexanders Bakst, Simon Hansen, Michal Podhradsky, and Andrew Bivin. The HARDENS Final Report, Galois Inc Technical Report.

Thomas Pressburger, Andreas Katis, Aaron Dutle, Anastasia Mavridou. Authoring, Analyzing, and Monitoring Requirements for a Lift-Plus-Cruise Aircraft, REFSQ 2023.

Ivan Perez, Anastasia Mavridou, Tom Pressburger, Alwyn Goodloe, Dimitra Giannakopoulou. Automated Translation of Natural Language Requirements to Runtime Monitors, TACAS 2022.

Hamza Bourbouh, Marie Farrell, Anastasia Mavridou, Irfan Sljivo, Guillaume Brat, Louise A. Dennis, Michael Fisher. Integrating Formal Verification and Assurance: An Inspection Rover Case Study, NFM 2021.

Anastasia Mavridou, Hamza Bourbouh, Dimitra Giannakopoulou, Tom Pressburger, Pierre-Loic Garoche, Johann Schumann. The Ten Lockheed Martin Cyber-Physical Challenges: Formalized, Analyzed, and Explained, RE 2020, Industry track.

Anastasia Mavridou, Hamza Bourbouh, Pierre Loic Garoche, Dimitra Giannakopoulou, Thomas Pressburger, Johann Schumann. Bridging the Gap Between Requirements and Simulink Model Analysis, REFSQ 2020, Poster Paper.

Full list: https://github.com/NASA-SW-VnV/fret/blob/master/PUBLICATIONS.md

source code

FRET: https://github.com/NASA-SW-VnV/fret

CoCoSim: https://github.com/NASA-SW-VnV/CoCoSim

Ogma: https://github.com/nasa/ogma

PLCverif: https://gitlab.com/plcverif-oss

acknowledgements

Zsófia Ádám, Alexanders Bakst, Swee Balachandran, Milan Bhandari, Marcin Bęś, Enrique Blanco Viñuela, Geoffrey Biggs, David Bushnell, Maxime Artaud, Hamza Bourbouh, Guillaume Brat, Esther Conrad, Louise A. Dennis, Tanja DeJong, Michael Dille, Aaron Dutle, Marie Farrell, Borja Fernandez Adiego, Michael Fisher, Pierre-Loic Garoche, Dimitra Giannakopoulou, Alwyn Goodloe, Simon Hansen, Kelly Ho, Michael Jeronimo, George Karamanolis, Andreas Katis, Joseph Kiniry, David Kooi, Ignacio D. Lopez-Miguel, Carlos Mao de Ferro, Patrick J. Martin, Francisco Martins, Amalaye Oyake, Ivan Perez, Jessica Phelan, Tom Pressburger, Julian Rhein, Daniel Riley, Johann Schumann, Nija Shi, Irfan Sljivo, Laura Titolo, Jean-Charles Tournier, Khanh V. Trinh, Gricel Vazquez, Tim Wang, Michael W. Whalen, Alexander Will.

Thank you!