

Capturing and Analyzing Requirements with FRET

Anastasia Mavridou

KBR Inc. / NASA Ames Research Center

How developers write requirements

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

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 **STANDBY** to **NOMINAL** when the pilot is in control (standby).
- The autopilot shall change states from **NOMINAL** to **MANEUVER** 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.
- ...

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

What analysis tools understand

```
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));
```

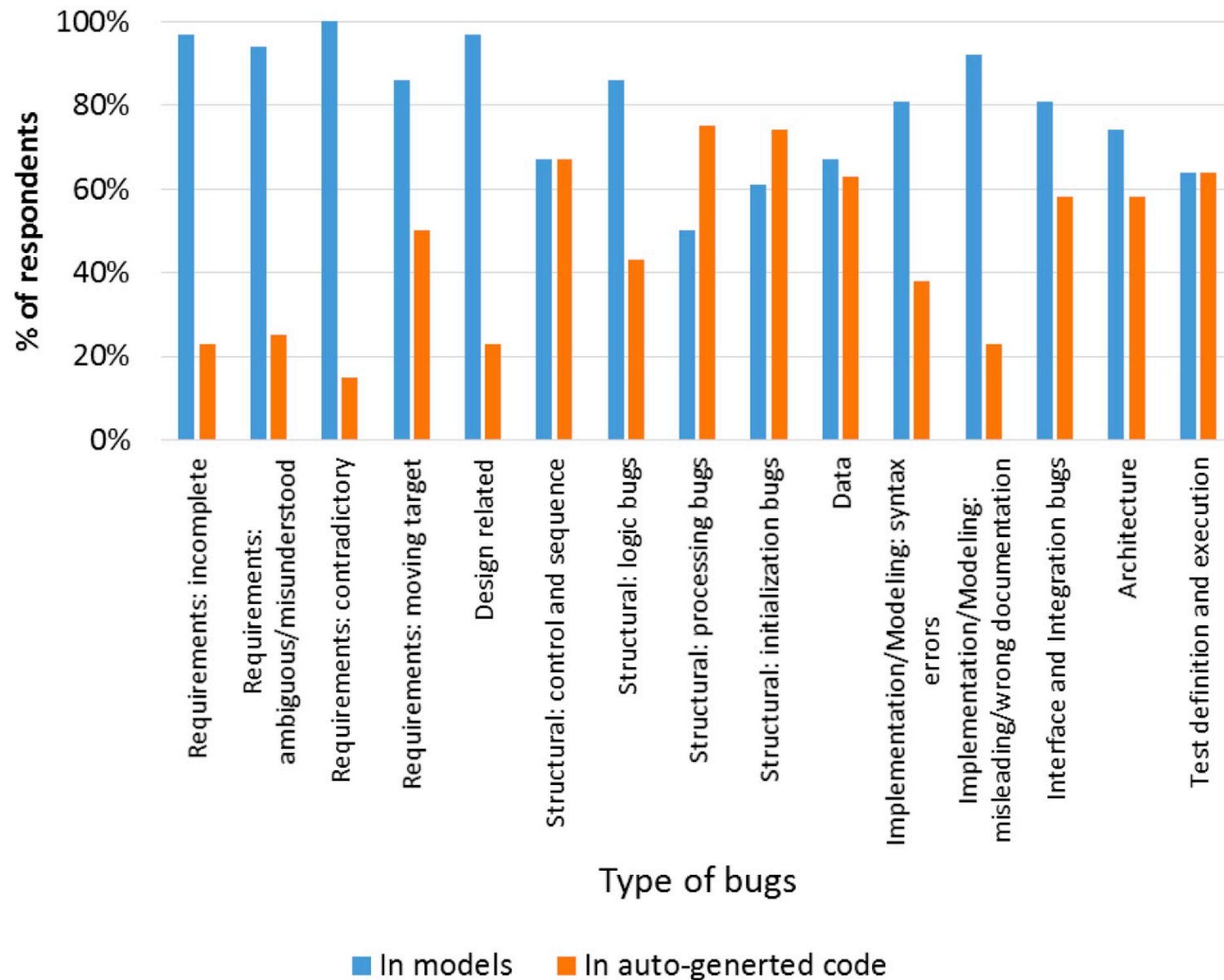
What analysis tools understand

Temporal logics

PAST TIME FORMULA

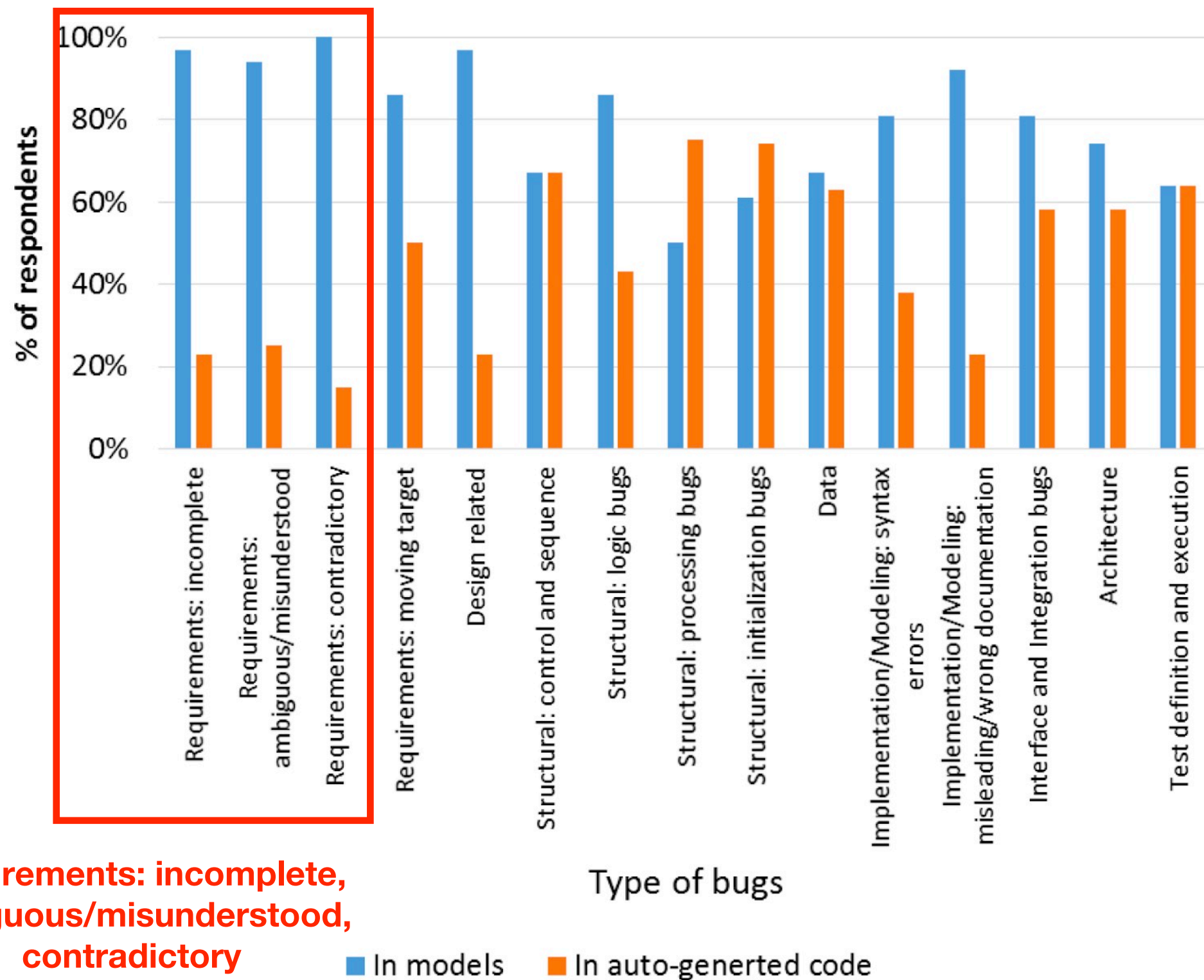
```
((H ((Lin_roll_hold & (! FTP)) -> (Y (((! (steady_state)) S ((! (steady_state)) & Fin_roll_hold)) | ((abs_roller_err <= 1.0) S ((abs_roller_err <= 1.0) & ((steady_state) & ((Y (! (steady_state))) | Fin_roll_hold)))) S (((! (steady_state)) S ((! (steady_state)) & Fin_roll_hold)) | ((abs_roller_err <= 1.0) S ((abs_roller_err <= 1.0) & ((steady_state) & ((Y (! (steady_state))) | Fin_roll_hold)))) & Fin_roll_hold)))) & (((! Lin_roll_hold) S ((! Lin_roll_hold) & Fin_roll_hold)) -> (((! (steady_state)) S ((! (steady_state)) & Fin_roll_hold)) | ((abs_roller_err <= 1.0) S ((abs_roller_err <= 1.0) & ((steady_state) & ((Y (! (steady_state))) | Fin_roll_hold)))) S (((! (steady_state)) S ((! (steady_state)) & Fin_roll_hold)) | ((abs_roller_err <= 1.0) S ((abs_roller_err <= 1.0) & ((steady_state) & ((Y (! (steady_state))) | Fin_roll_hold)))) & Fin_roll_hold))))
```

What types of bugs are found in models and code?



[3] Johann Schumann, Matt Knudsen, Teme Kahsai, Nolie Inkwocha, Katerina Goseva-Popstojanova, Thomas Kyanko, "Report: Survey on Model-Based Software Engineering and Auto-Generated Code", NASA

What types of bugs are found in models and code?



[3] Johann Schumann, Matt Knudsen, Teme Kahsai, Noble Nkwocha, Katerina Goseva-Popstojanova, Thomas Kyanko, "Report: Survey on Model-Based Software Engineering and Auto-Generated Code", NASA

How do we bridge this gap?

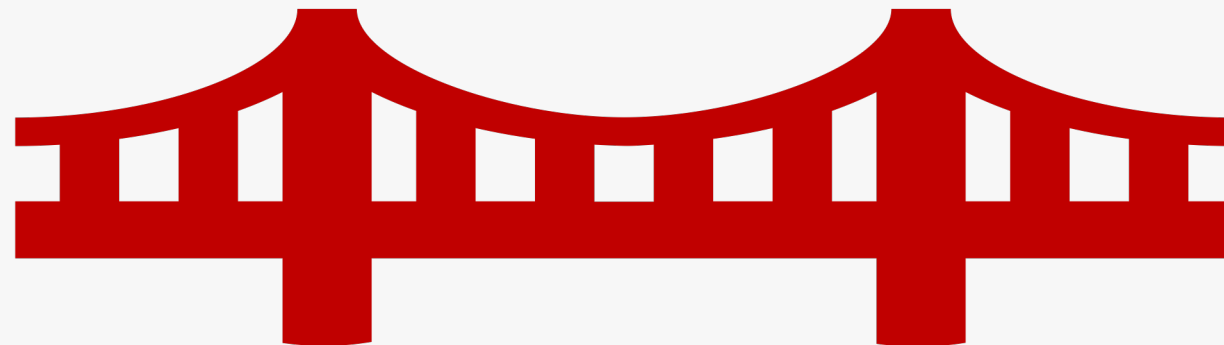
Natural language requirements

Rigorous formalizations



welcome to FRET!

<https://github.com/NASA-SW-VnV/fret>



<https://github.com/NASA-SW-VnV/fret>

The Formal Requirements Elicitation Tool (FRET) Team



Dimitra Giannakopoulou



Tom Pressburger

Anastasia Mavridou



Johann Schumann



Andreas Katis

Khank Trinh

<https://github.com/NASA-SW-VnV/fret>

The Formal Requirements Elicitation Tool (FRET)

Captures requirements

when in cruising mode, the altitude_hold_;

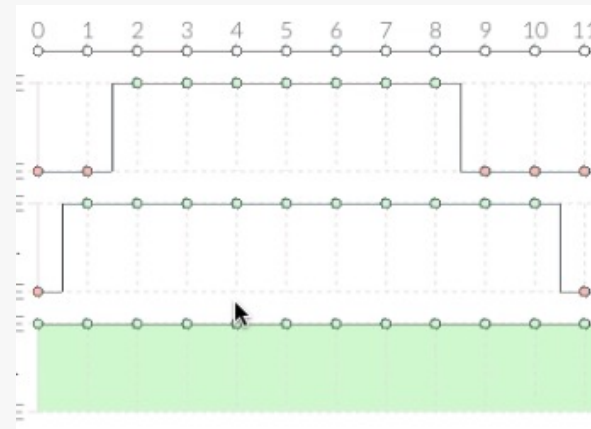
<https://github.com/NASA-SW-VnV/fret>

The Formal Requirements Elicitation Tool (FRET)

Captures requirements

Explains their meaning

when in cruising mode, the altitude_hold_;



ENFORCED: in every interval where *cruising* holds. TRIGGER: first point in the interval. REQUIRES: for every trigger, RES must hold at all time points between (and including) the trigger and the end of the interval.

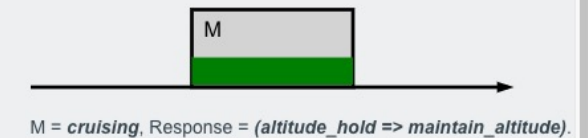


Diagram Semantics

M Mode of operation (mentioned in Scope)

Intervals

Scope interval

Response must hold at least somewhere in this interval

Negation of response must hold at least somewhere in this interval

Response must hold everywhere in this interval

<https://github.com/NASA-SW-VnV/fret>

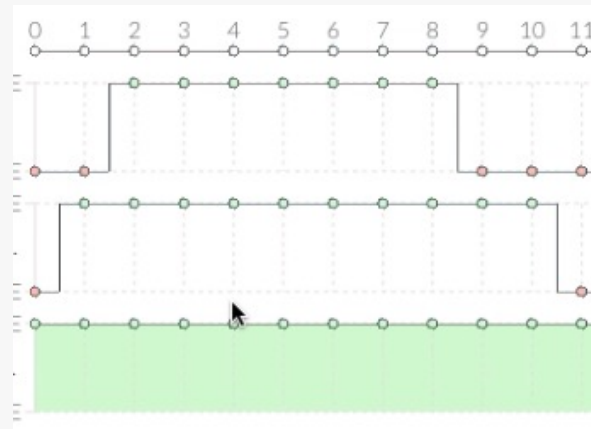
The Formal Requirements Elicitation Tool (FRET)

Captures requirements

Explains their meaning

Produces formalizations

when in cruising mode, the altitude_hold_;



ENFORCED: in every interval where *cruising* holds. TRIGGER: first point in the interval. REQUIRES: for every trigger, RES must hold at all time points between (and including) the trigger and the end of the interval.



Future Time LTL

```
(LAST V (cruising -> (altitude_hold -> maintain_altitude)))
```

Target: *altitude_hold_autopilot* component.

Past Time LTL

<https://github.com/NASA-SW-VnV/fret>

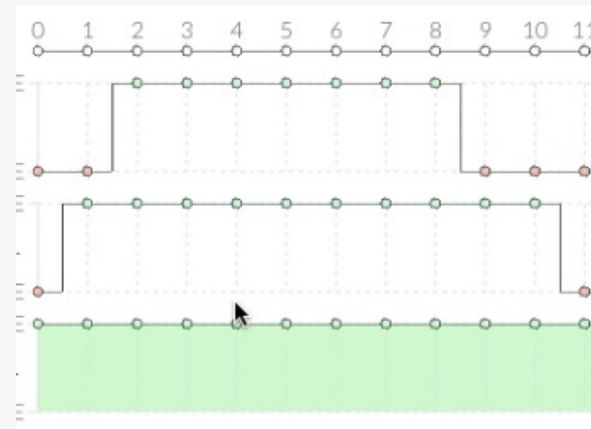
The Formal Requirements Elicitation Tool (FRET)

Captures requirements

Explains their meaning

Produces formalizations

when in cruising mode, the altitude_hold_



ENFORCED: in every interval where **cruising** holds. TRIGGER: first point in the interval. REQUIRES: for every trigger, RES must hold at all time points between (and including) the trigger and the end of the interval.

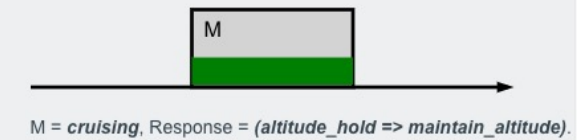
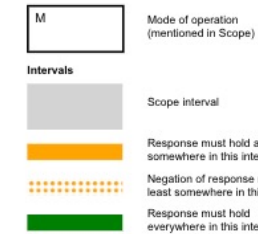
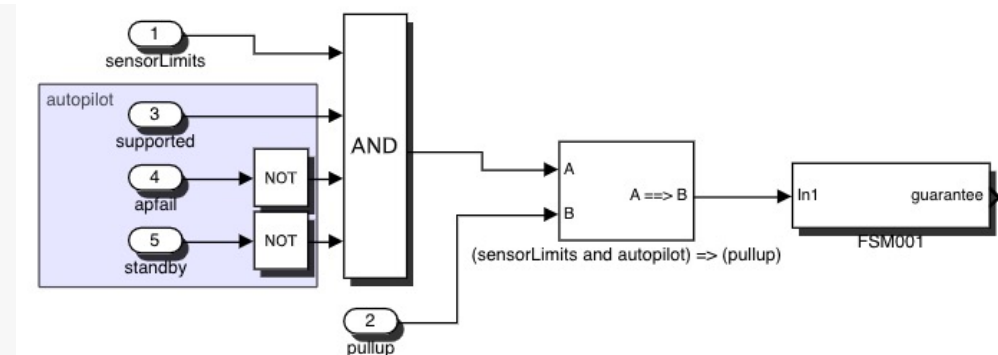


Diagram Semantics



Connects with analysis tools

```
contract FSMSpec(apfail:bool; sensorLimits:bool; standby:bool; supported:bool; ) returns (
  pullup: bool; );
let
var autopilot:bool=supported and not apfail and not standby;
guarantee "FSM001" H ((sensorLimits and autopilot) => (pullup));
tel
```



Future Time LTL

```
(LAST V (cruising -> (altitude_hold ->
  maintain_altitude)))
```

Target: *altitude_hold_autopilot* component.

Past Time LTL

<https://github.com/NASA-SW-VnV/fret>

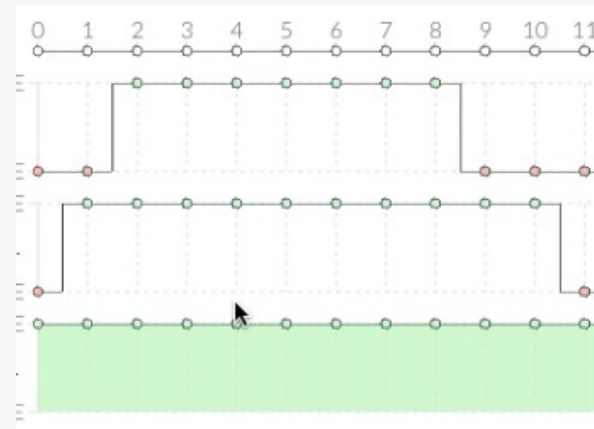
The Formal Requirements Elicitation Tool (FRET)

Captures requirements

Explains their meaning

Produces formalizations

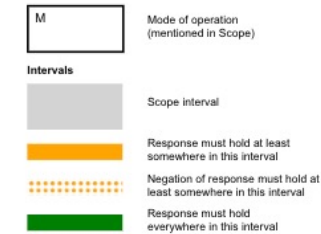
when in cruising mode, the altitude_hold_;



ENFORCED: in every interval where **cruising** holds. TRIGGER: first point in the interval. REQUIRES: for every trigger, RES must hold at all time points between (and including) the trigger and the end of the interval.



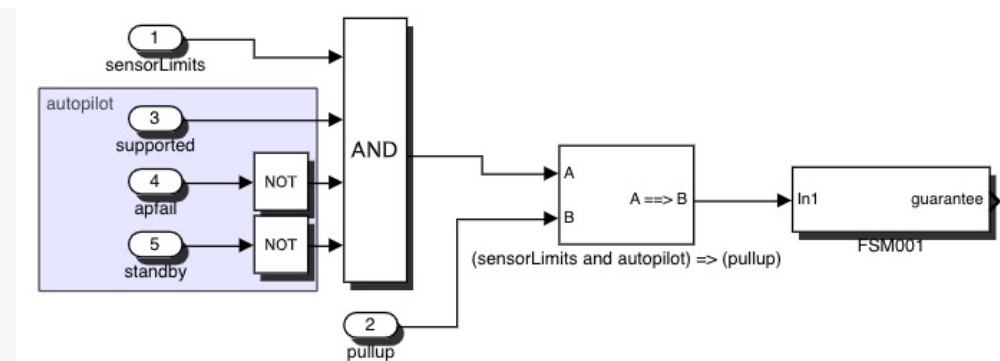
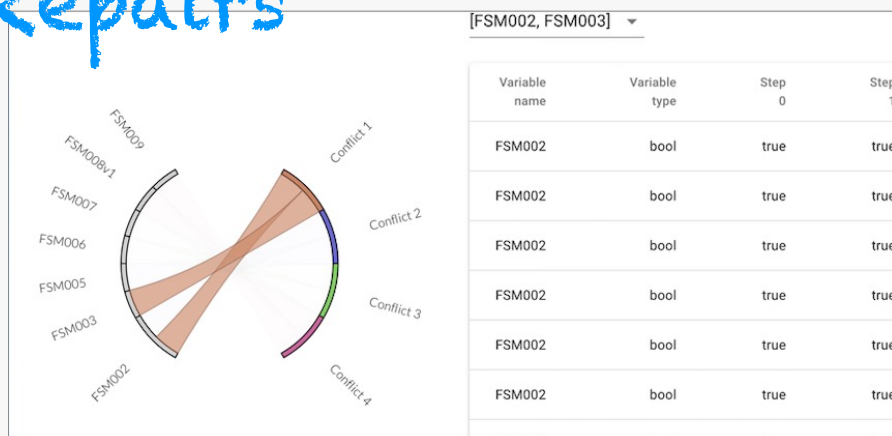
Diagram Semantics



Connects with analysis tools

```
contract FSMSpec(apfail:bool; sensorLimits:bool; standby:bool; supported:bool; ) returns (
  pullup: bool; );
let
var autopilot:bool=supported and not apfail and not standby;
guarantee "FSM001" H ((sensorLimits and autopilot) => (pullup));
tel
```

Repairs



The Ten Lockheed Martin CPS Challenge Problems (LMCPS)

- A set of industrial benchmarks. Each challenge includes:
 - Natural language requirements
 - A Simulink model
 - A set of parameters (in .mat format) for simulating the model

The Ten Lockheed Martin CPS Challenge Problems (LMCPS)

- A set of industrial benchmarks. Each challenge includes:
 - Natural language requirements
 - A Simulink model
 - A set of parameters (in .mat format) for simulating the model
- The challenges are:
 - Representative of flight-critical systems
 - Publicly available

https://github.com/hbourbough/lm_challenges

Step 0: Requirement Elicitation

Requirement from the 6 DoF DeHavilland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

Step 0: Requirement Elicitation

Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

Step 0 involves understanding the requirement and making it precise.

We start by identifying the **variables** involved.

Step 0: Requirement Elicitation

Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPs challenge

Natural language requirement:

[AP-003c]: *The **roll hold reference** shall be set to 30 degrees in the same direction as the actual **roll angle** if the actual roll angle is greater than 30 degrees at the time of **roll hold mode engagement**.*

Step 0 involves understanding the requirement and making it precise.

We start by identifying the **variables** involved.

Now we are ready to write the requirement in the **FRETish** language!

Step 0: Requirement Elicitation

Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish fields:

Scope, condition, component*, shall*, timing, response*

Step 0: Requirement Elicitation

Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

If $\text{abs}(\text{roll_angle}) > 30$ & roll_hold_mode_engagement,

FRETish fields:

Scope, condition, component*, shall*, timing, response*

Step 0: Requirement Elicitation

Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPs challenge

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

If $\text{abs}(\text{roll_angle}) > 30$ & $\text{roll_hold_mode_engagement}$,
Autopilot shall

FRETish fields:

Scope, condition, component*, shall*, timing, response*

Step 0: Requirement Elicitation

Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees **at the time of** roll hold mode engagement.*

FRETish version:

If **abs(roll_angle) >30 & roll_hold_mode_engagement**,
Autopilot shall **immediately**

FRETish fields:

Scope, **condition**, **component***, shall*, **timing**, **response***

Step 0: Requirement Elicitation

Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPs challenge

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

If $\text{abs}(\text{roll_angle}) > 30$ & $\text{roll_hold_mode_engagement}$,
Autopilot shall immediately satisfy
 $\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle})$

FRETish fields:

Scope, condition, component*, shall*, timing, response*

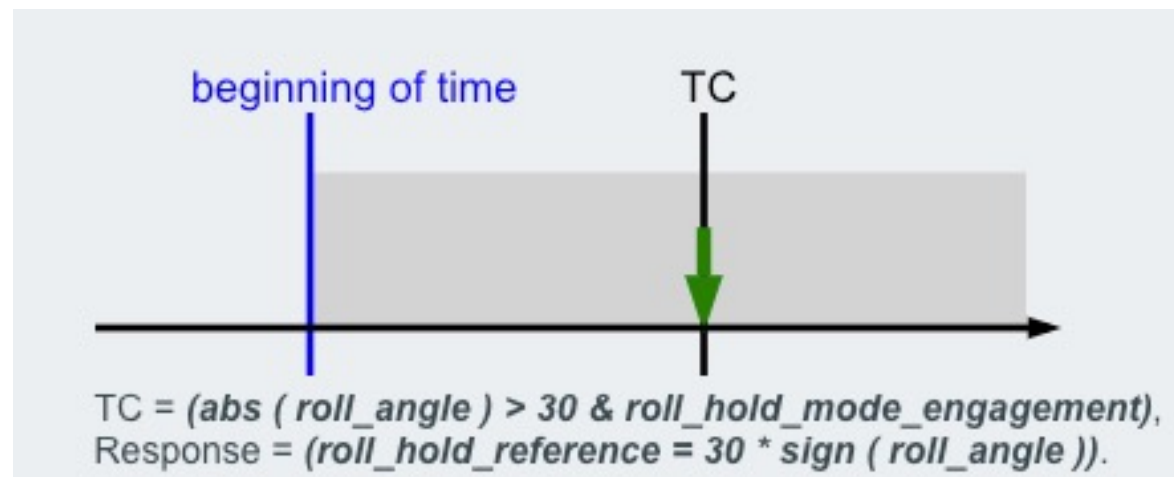
Step 0: Requirement Elicitation

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v1]: If $\text{abs}(\text{roll_angle}) > 30$ & $\text{roll_hold_mode_engagement}$, Autopilot shall immediately satisfy $\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle})$



FRETish fields:

Scope, condition, component*, shall*, timing, response*

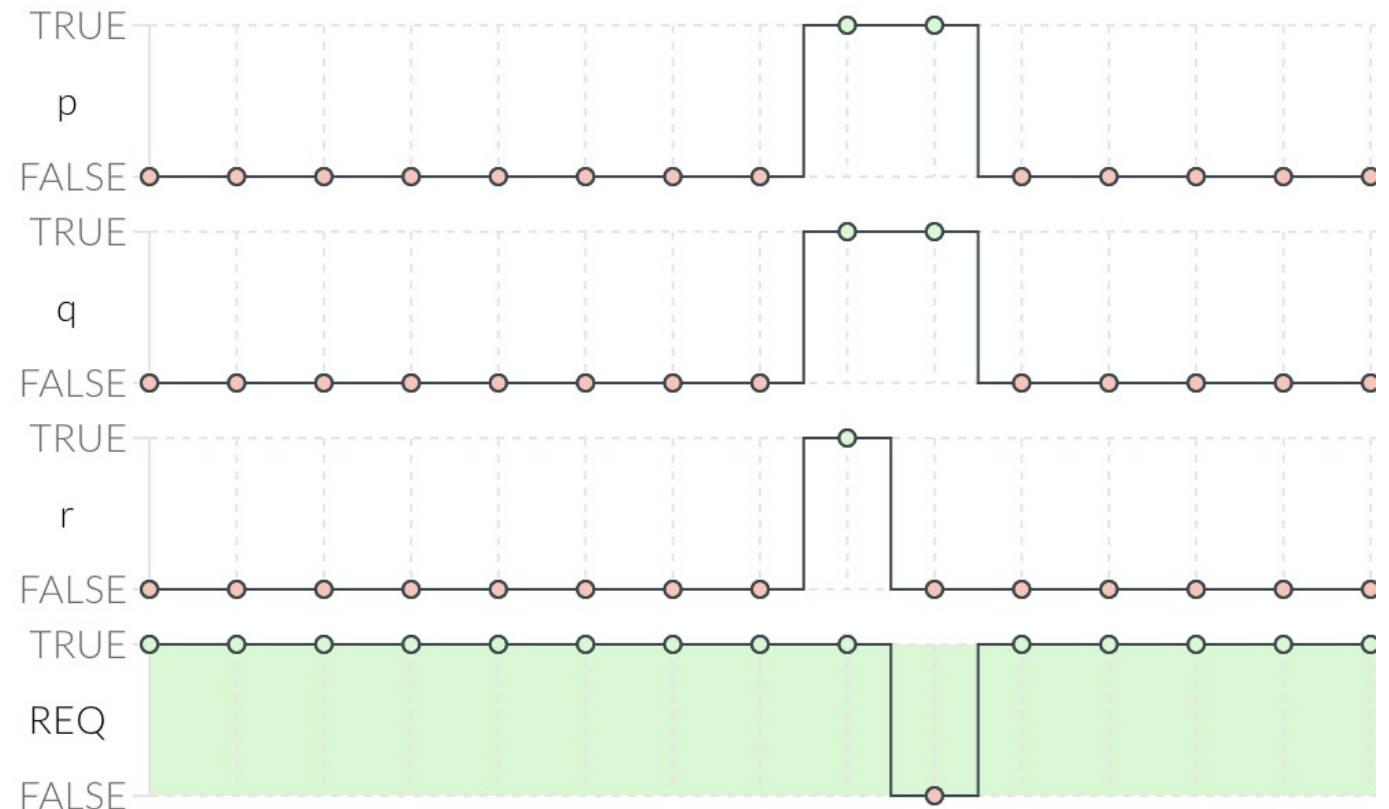
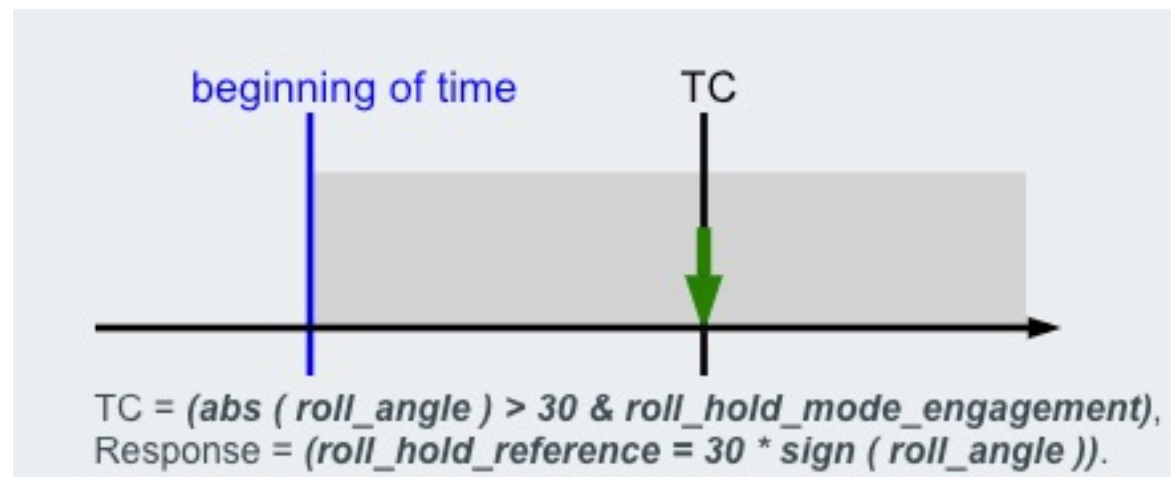
Step 0: Requirement Elicitation

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v1]: If $\text{abs}(\text{roll_angle}) > 30$ & $\text{roll_hold_mode_engagement}$, Autopilot shall immediately satisfy $\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle})$



FRETish fields:

Scope, condition, component*, shall*, timing, response*

p: $\text{abs}(\text{roll_angle})$

q: $\text{roll_hold_mode_engagement}$

r: response

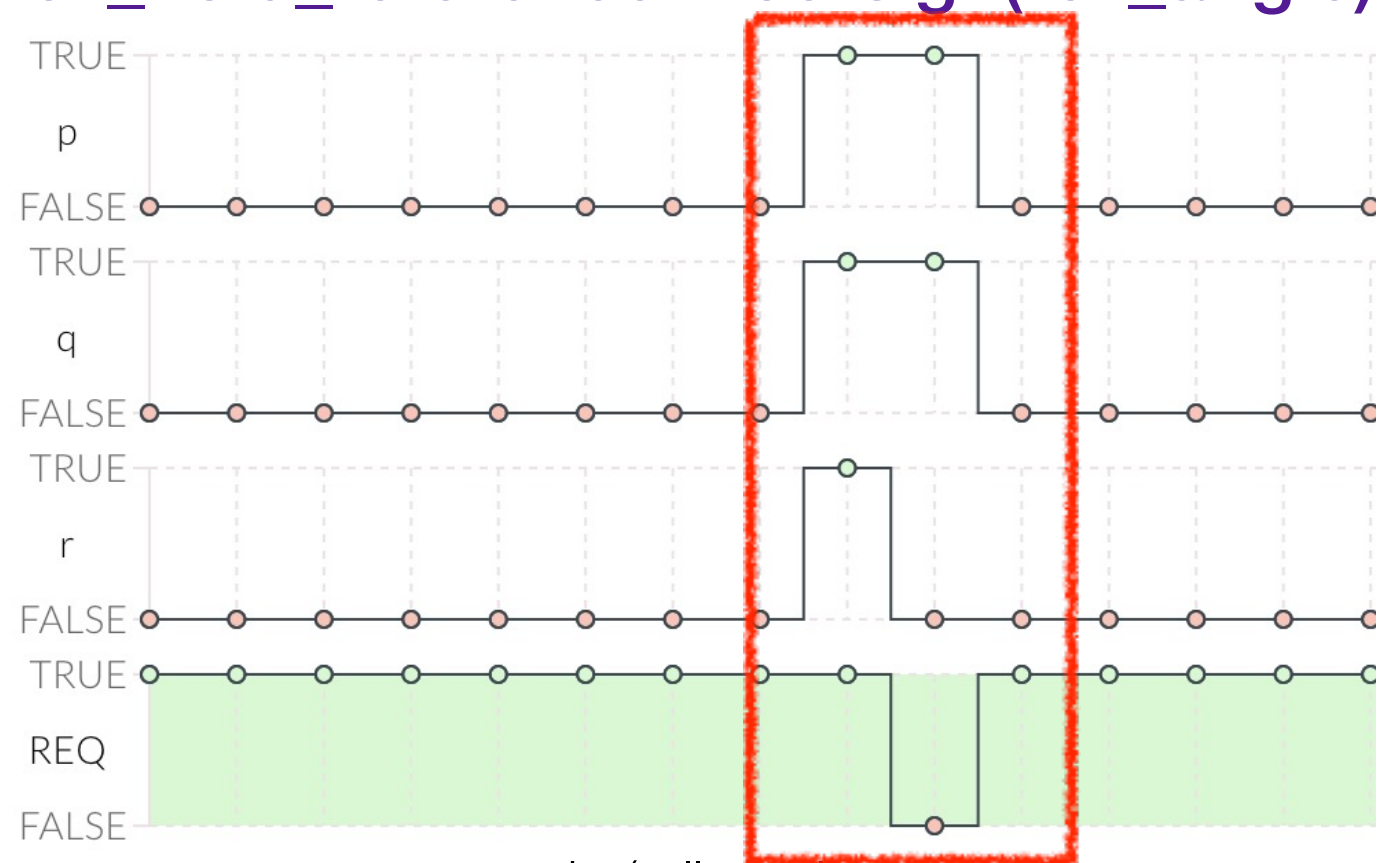
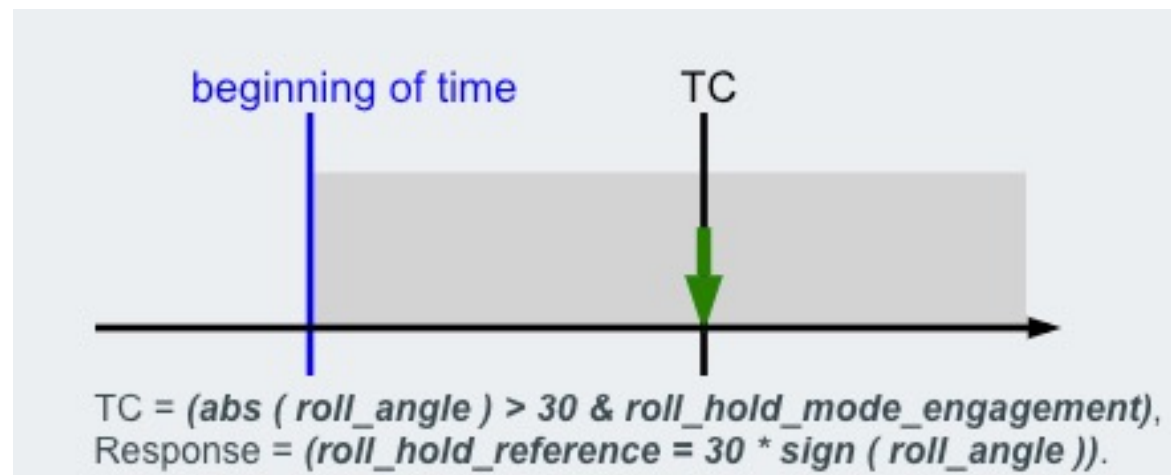
Step 0: Requirement Elicitation

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v1]: If $\text{abs}(\text{roll_angle}) > 30$ & $\text{roll_hold_mode_engagement}$, Autopilot shall immediately satisfy $\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle})$



FRETish fields:

Scope, condition, component*, shall*, timing, response*

p: $\text{abs}(\text{roll_angle})$
q: $\text{roll_hold_mode_engagement}$
r: response

Step 0: Requirement Elicitation

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v2]: Autopilot shall immediately satisfy if $(\text{abs}(\text{roll_angle}) > 30 \ \& \ \text{roll_hold_mode_engagement})$ then $(\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle}))$

FRETish fields:

Scope, condition, component*, shall*, timing, response*

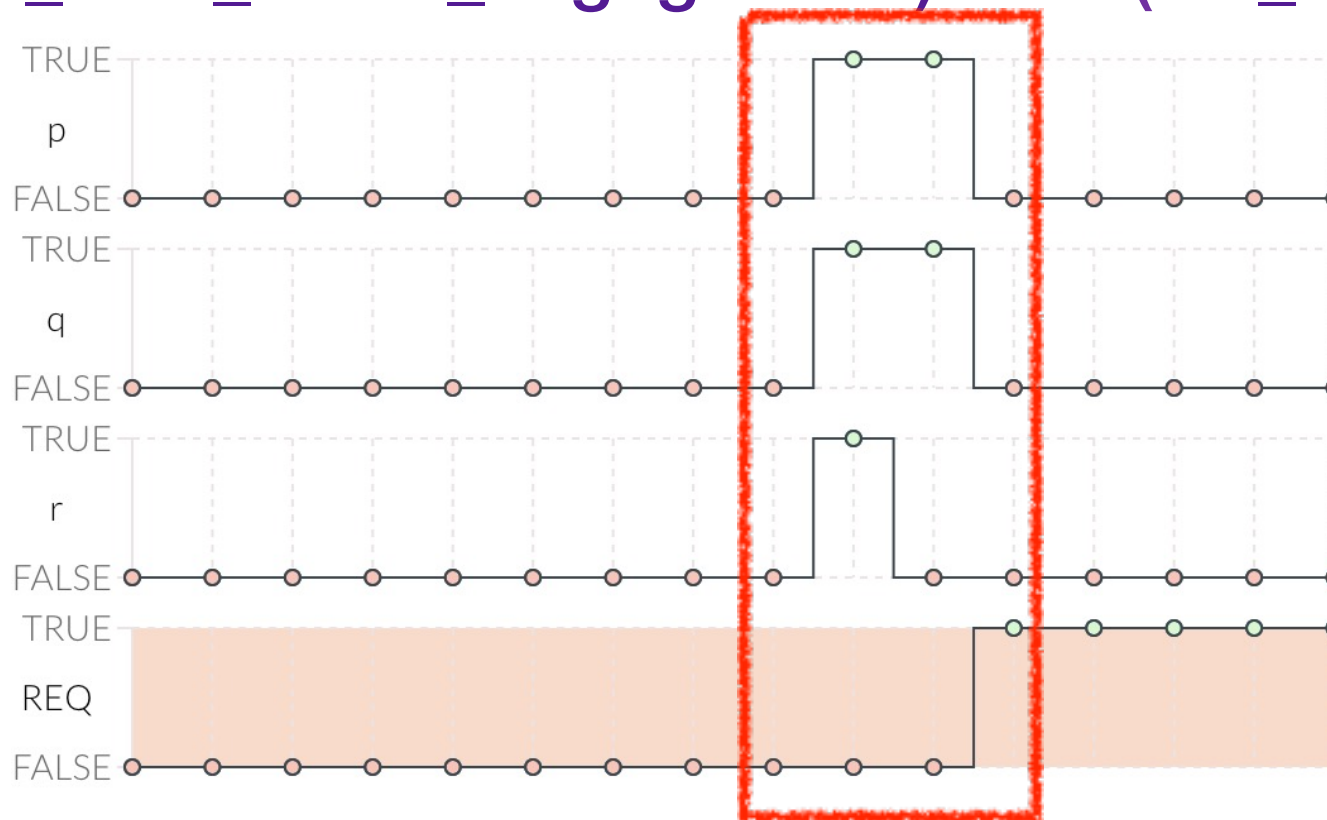
Step 0: Requirement Elicitation

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v2]: Autopilot shall immediately satisfy if $(\text{abs}(\text{roll_angle}) > 30 \ \& \ \text{roll_hold_mode_engagement})$ then $(\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle}))$



p: $\text{abs}(\text{roll_angle})$
q: $\text{roll_hold_mode_engagement}$
r: $\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle})$

FRETish fields:

Scope, condition, component*, shall*, timing, response*

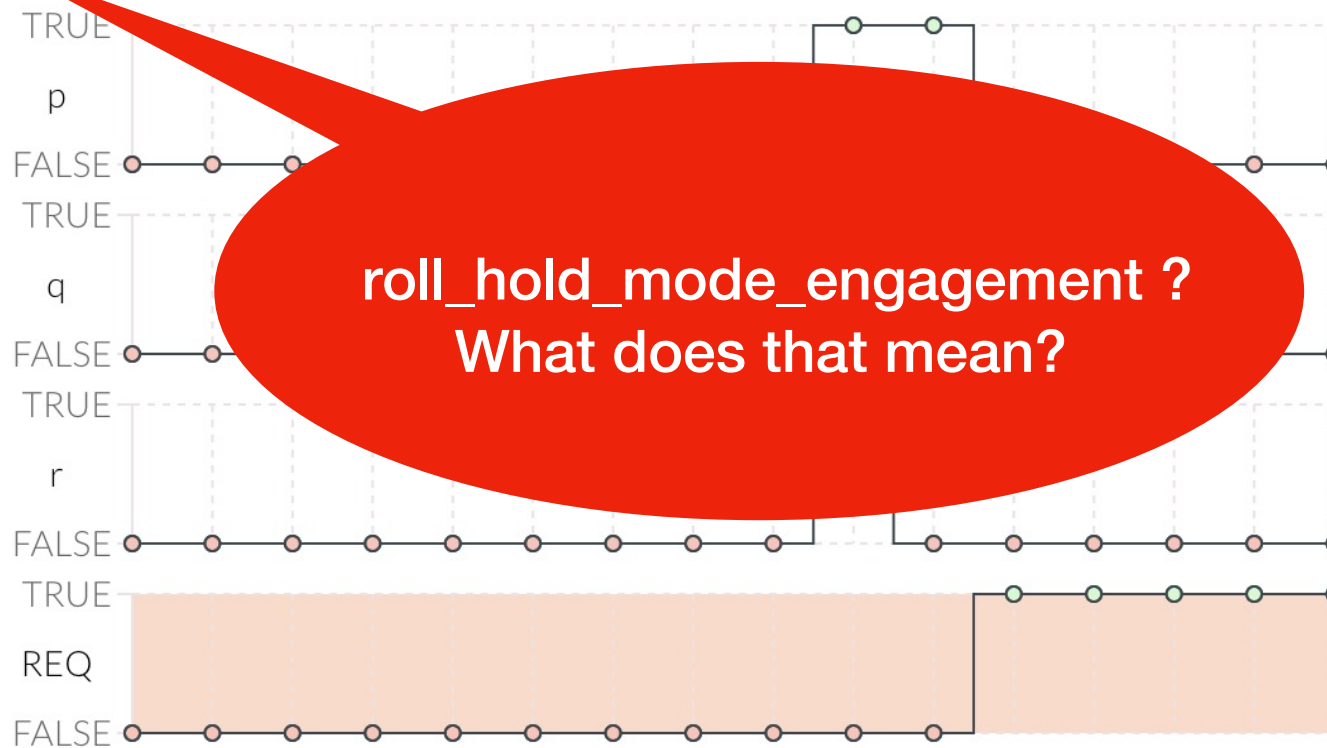
Step 0: Requirement Elicitation

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v2]: Autopilot shall immediately satisfy if $(\text{abs}(\text{roll_angle}) > 30 \ \& \ \text{roll_hold_mode_engagement})$ then $(\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle}))$



p: $\text{abs}(\text{roll_angle})$
q: roll_hold_mode_engagement
r: $\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle})$

FRETish fields:

Scope, condition, component*, shall*, timing, response*

Step 0: Requirement Elicitation

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v3]: when in roll_hold mode Autopilot shall immediately satisfy if $(\text{abs}(\text{roll_angle}) > 30)$ then $(\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle}))$

FRETish fields:

Scope, condition, component*, shall*, timing, response*

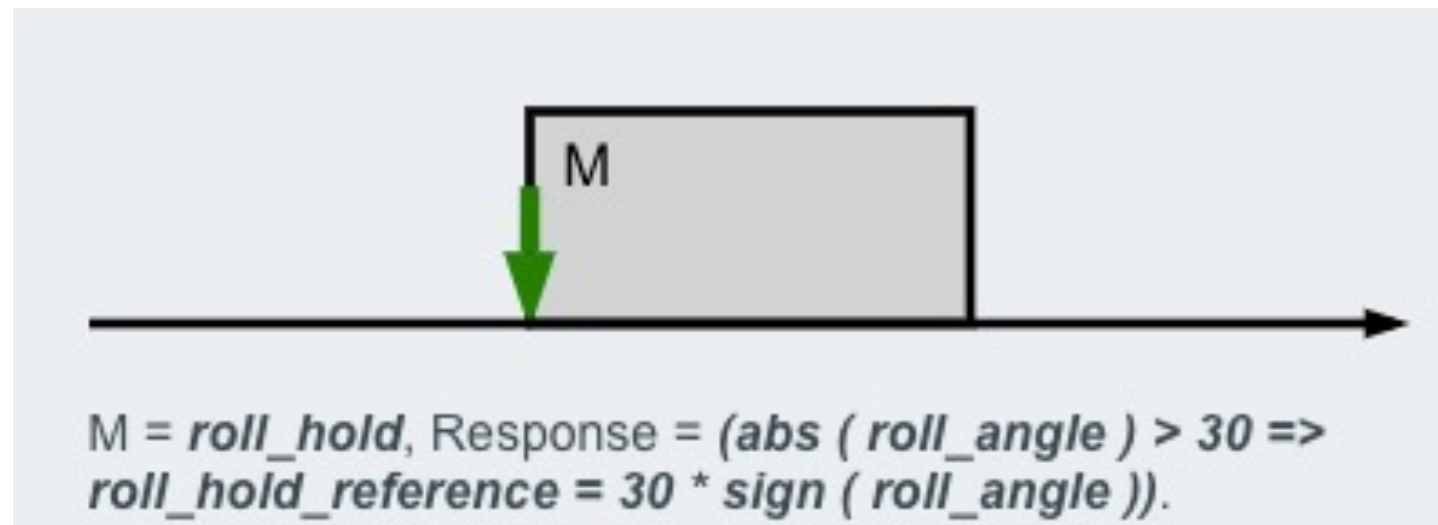
Step 0: Requirement Elicitation

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v3]: when in roll_hold mode Autopilot shall immediately satisfy if $(\text{abs}(\text{roll_angle}) > 30)$ then $(\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle}))$



FRETish fields:

Scope, condition, component*, shall*, timing, response*

Step 1: Requirement Formalization

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v3]: when in roll_hold mode Autopilot shall immediately satisfy $(\text{abs}(\text{roll_angle}) > 30) \Rightarrow (\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle}))$

Past Time Linear Temporal Logic formula:

$$H(\text{roll_hold} \ \& \ (! \text{FTP} \mid (Y \ (! \ \text{roll_hold}))) \Rightarrow \text{abs}(\text{roll_angle}) > 30 \Rightarrow \text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle}))$$

where FTP is a predicate that holds at the First Time Point of an execution

Step 2: Generation of Analysis Code

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

Past Time Linear Temporal Logic formula:

$$H(\text{roll_hold} \ \& \ (! \ \text{FTP} \mid (Y \ (! \ \text{roll_hold}))) \Rightarrow \text{abs}(\text{roll_angle} > 30 \Rightarrow \text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle})))$$

```
--Once
node O(X:bool) returns (Y:bool);
let
  Y = X or (false -> pre Y);
tel
--Y since X
node S(X,Y: bool) returns (Z:bool);
let
  Z = X or (Y and (false -> pre Z));
tel

--Historically
node H(X:bool) returns (Y:bool);
let
  Y = X -> (X and (pre Y));
tel
--Y since inclusive X
node SI(X,Y: bool) returns (Z:bool);
let
  Z = Y and (X or (false -> pre Z));
tel
```

CoCoSpec analysis code:

```
-- 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))
```

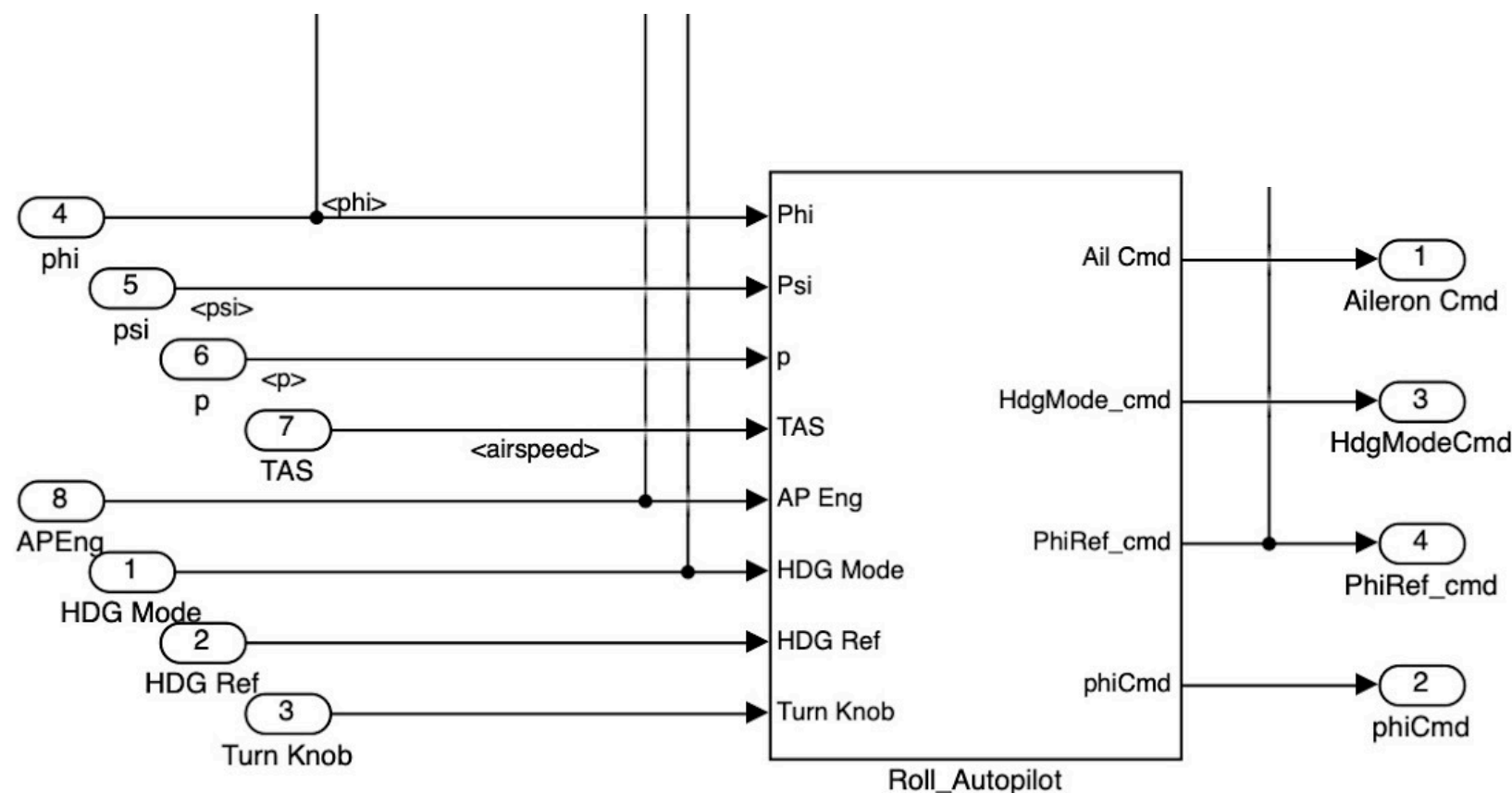
Step 3: Automated Architectural Mapping

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v3]: when in roll_hold mode Autopilot shall immediately satisfy $(\text{abs}(\text{roll_angle}) > 30) \Rightarrow (\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle}))$



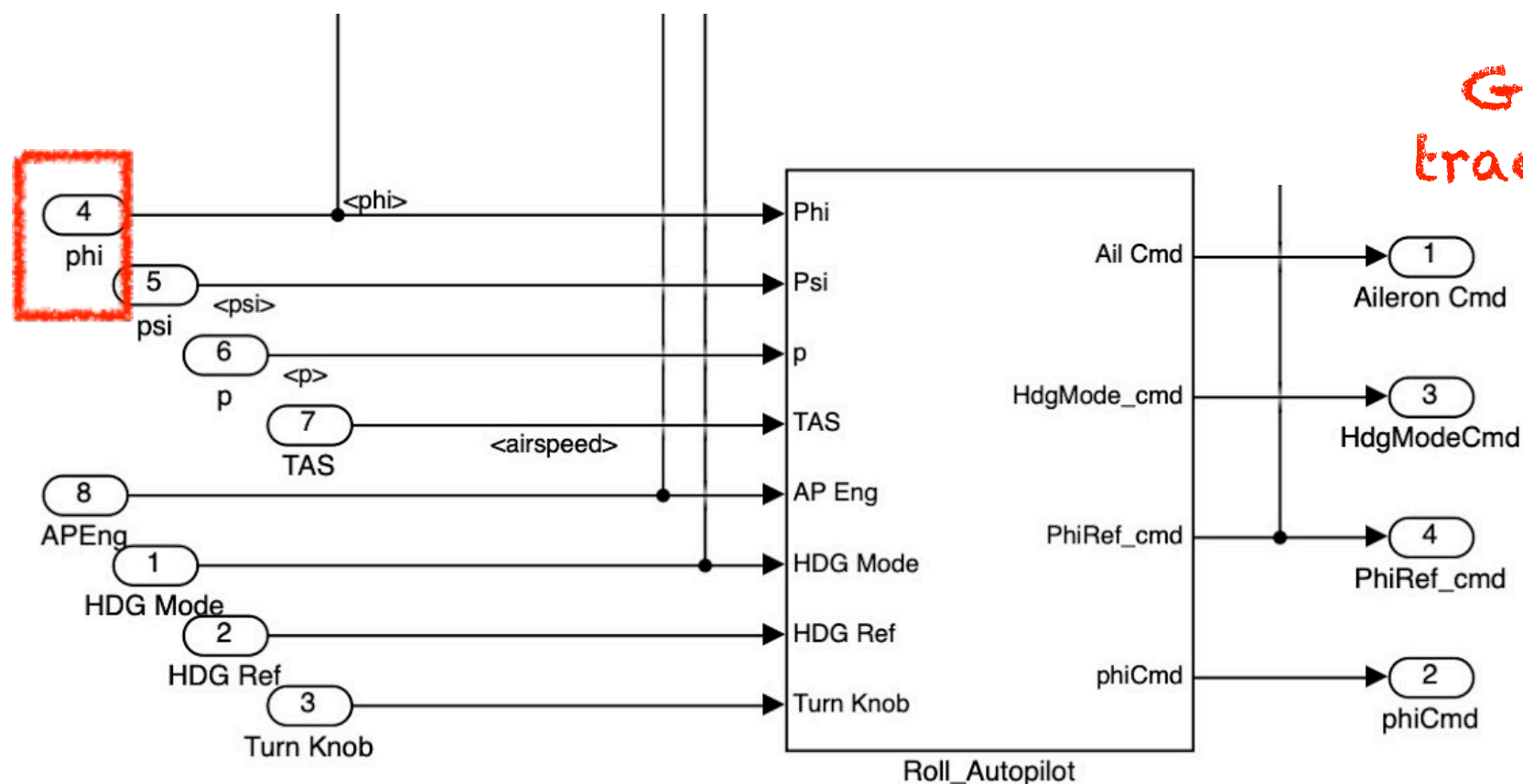
Step 3: Automated Architectural Mapping

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v3]: when in roll_hold mode Autopilot shall immediately satisfy $(\text{abs}(\text{roll_angle}) > 30) \Rightarrow (\text{roll_hold_reference} = 30 * \text{sign}(\text{roll_angle}))$



Step 4: Generation of Simulink Monitors

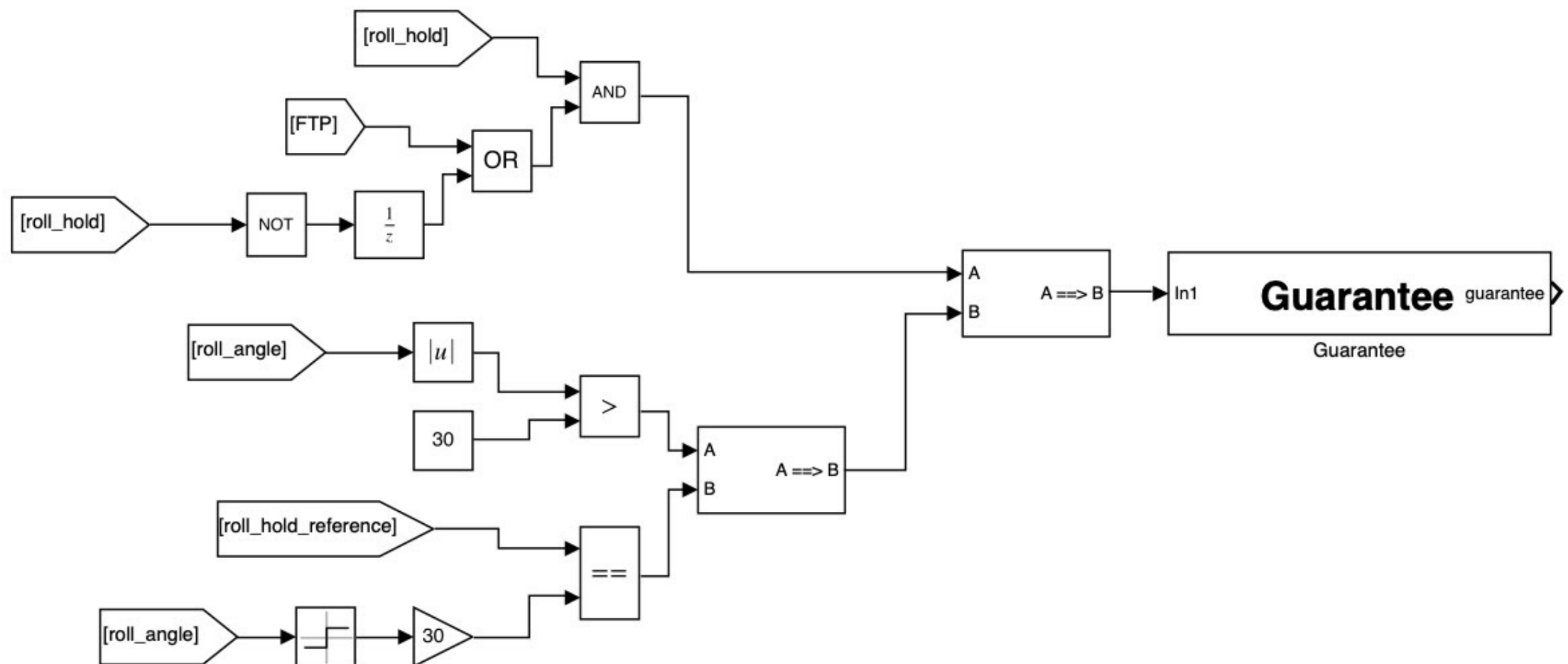
Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

CoCoSpec analysis code:

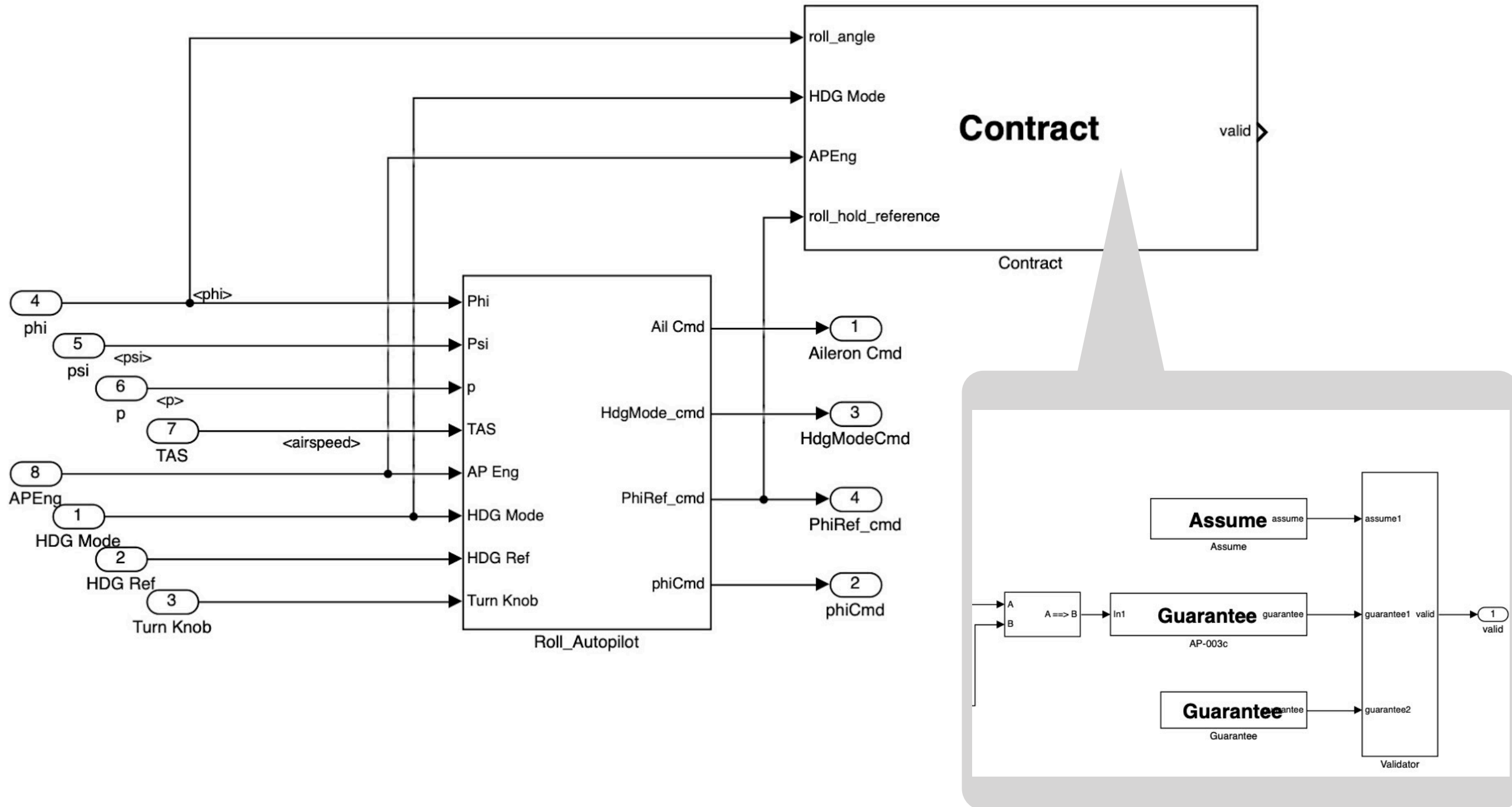
```
-- AP-003c-v3 requirement in CoCoSpec
guarantee H((roll_hold and (FTP or (pre (not roll_hold))))
=> abs(roll_angle) > 30 =>
roll_hold_reference = 30 * sign(roll_angle))
```

Simulink monitor:



Step 4: Generation of Simulink Monitors

Simulink monitors automatically attached on the model:

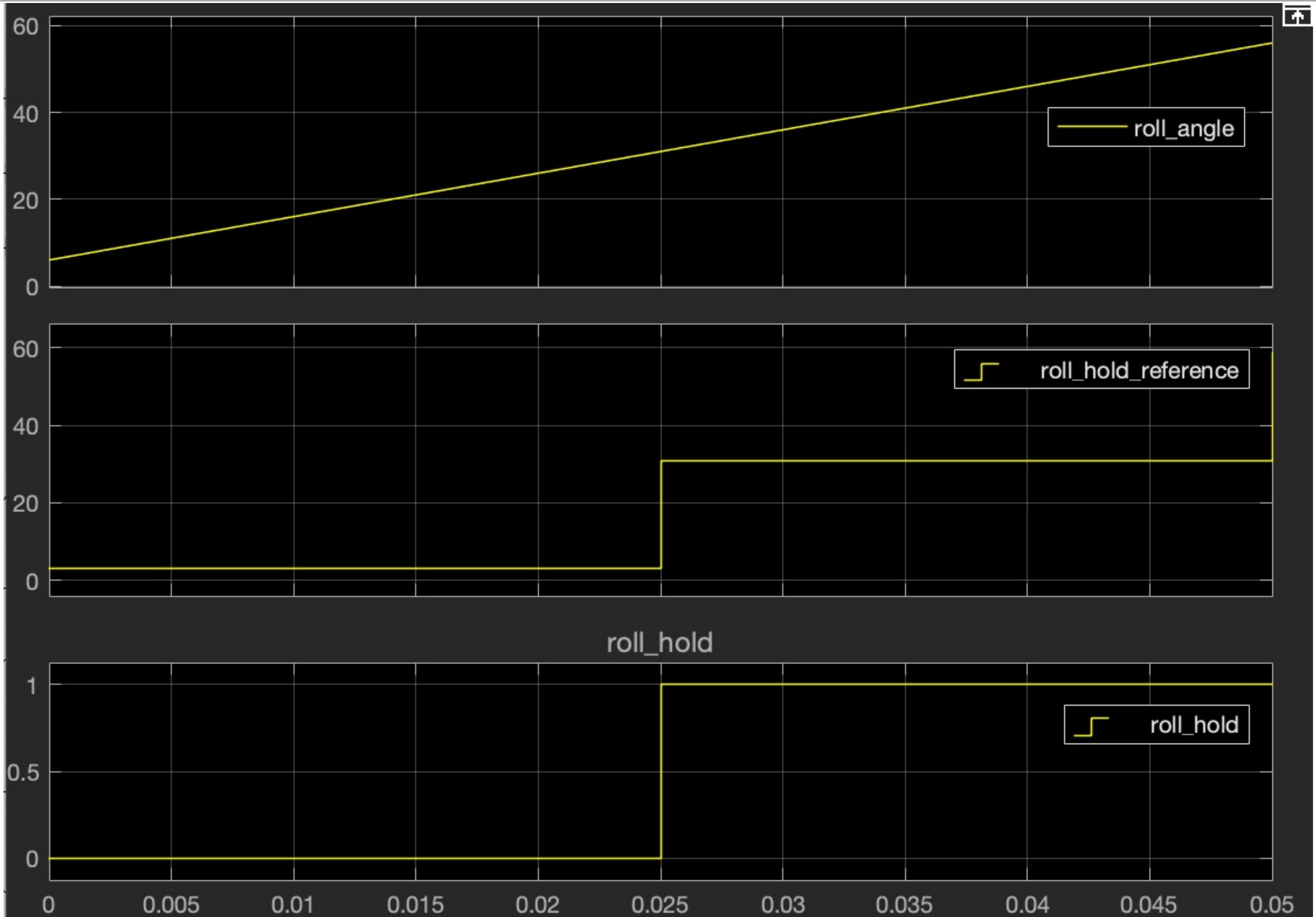


Step 5: Analysis and Counterexample Generation

RollAutopilot_PP/Roll_Autopilot

Time	0	0.025
Phi (input)	6	31
Psi (input)	0	0
p (input)	0	0
TAS (input)	0	0
APEng (input)	0	1
HDGMode (input)	0	0
HDGRef (input)	0	0
TurnKnob (input)	3	31
PhiRef_cmd (output)	3	31
RollReference (local)	3	31

Step 6: Simulation



Modular Analysis

- For the Autopilot challenge this proved particular useful:
 - We were able to analyze all properties that were specified at local level but none of the properties that were specified globally

TABLE VI: AP Analysis Results with Kind2

Reqs	Scope	Kind2 Result	Kind2 Time
[AP-000]	Global	Unsupported	
[AP-001]	Roll AP	Valid	< 1 sec
[AP-002]	Roll AP	Valid	< 1 sec
[AP-003a]	Roll AP	Invalid	< 1 sec
[AP-003b]	Roll AP	Invalid	< 1 sec
[AP-003c]	Roll AP	Invalid	< 1 sec
[AP-003d]	Roll AP	Valid	< 1 sec
[AP-004]	Global	Unsupported	
[AP-005]	Global	Unsupported	
[AP-006]	Global	Unsupported	
[AP-007]	Roll AP	Valid	< 1 sec
[AP-008]	Roll AP	Valid	< 1 sec
[AP-010]	Global	Unsupported	
Total running time		CoCoSim: 40.589s	

To summarize

- Using an end-to-end framework significantly simplifies requirements elicitation and model analysis
- Eliciting requirements with unambiguous and as-intended semantics is not an easy task
 - Explanations and interactive exploration of requirements helps
- Industrial requirements are complex to analyze
 - It is important to provide modular analysis

Our open source tools can be accessed on Github:

<https://github.com/NASA-SW-VnV/fret>

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

Thank you!