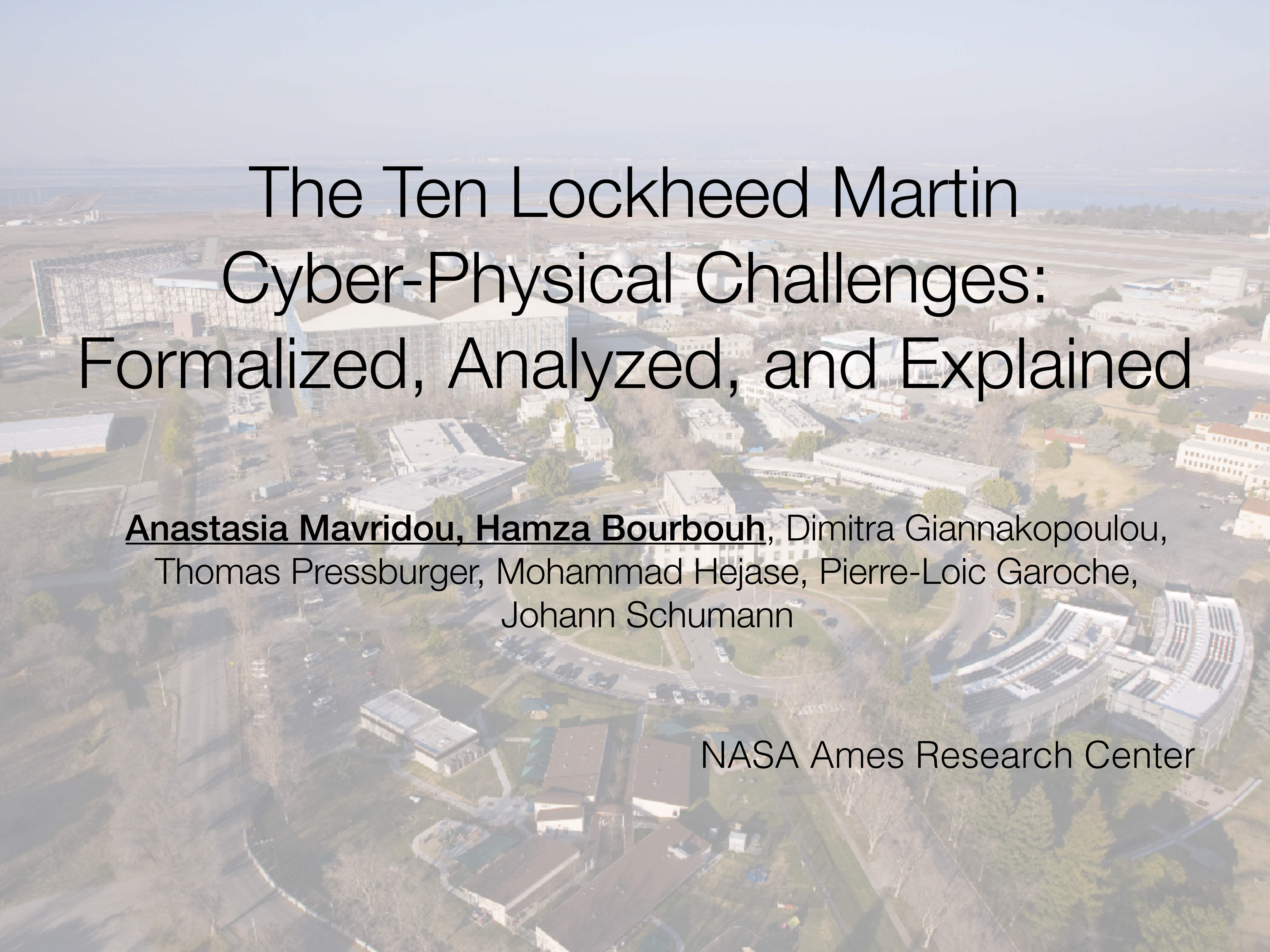


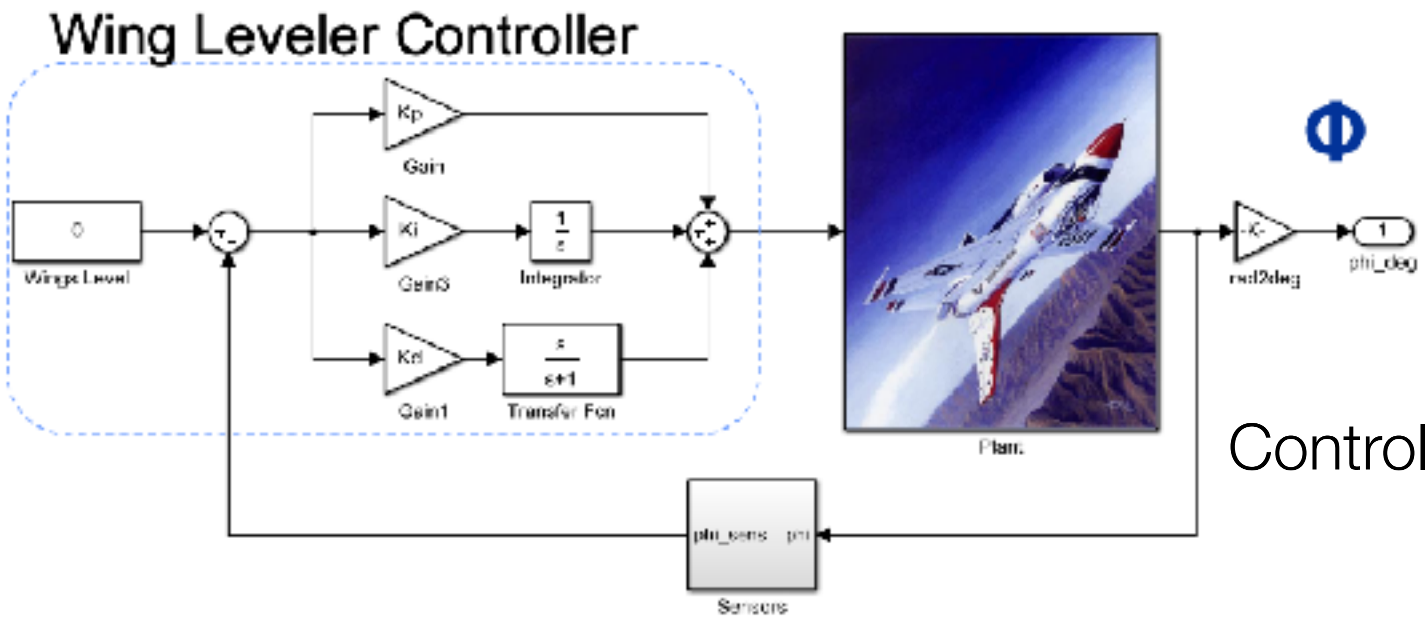
The Ten Lockheed Martin Cyber-Physical Challenges: Formalized, Analyzed, and Explained

Anastasia Mavridou, Hamza Bourbouh, Dimitra Giannakopoulou, Thomas Pressburger, Mohammad Hejase, Pierre-Loic Garoche, Johann Schumann

NASA Ames Research Center

Cyber-Physical Systems (CPS)

- Integrate computation with physical processes
- More than 60% of engineers use Simulink for the development and simulation of CPS [1]



Controller picture from [2]

- Ensure that bugs are identified as early as possible
    - It is paramount to check requirements against models

[1] Nejati, Shiva, Khouloud Gaaloul, Claudio Menghi, Lionel C. Briand, Stephen Foster, and David Wolfe. "Evaluating model testing and model checking for finding requirements violations in Simulink models." In *Proceedings of the 2019 27th ESEC/FSE,* 2019.
[2] Chris Elliot, "On Examples Models and Challenges Ahead for the Evaluation of Complex Cyber-Physical with State of the Art Formal Methods V&V", S5 conference, 2015.

CPS Requirements

- CPS requirements are usually expressed in natural language
    - Riddled with ambiguities

Lockheed Martin Cyber-Physical System Challenges

Autopilot component:

“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.”

CPS Requirements

- CPS requirements are usually expressed in natural language
    - Riddled with ambiguities

Lockheed Martin Cyber-Physical System Challenges

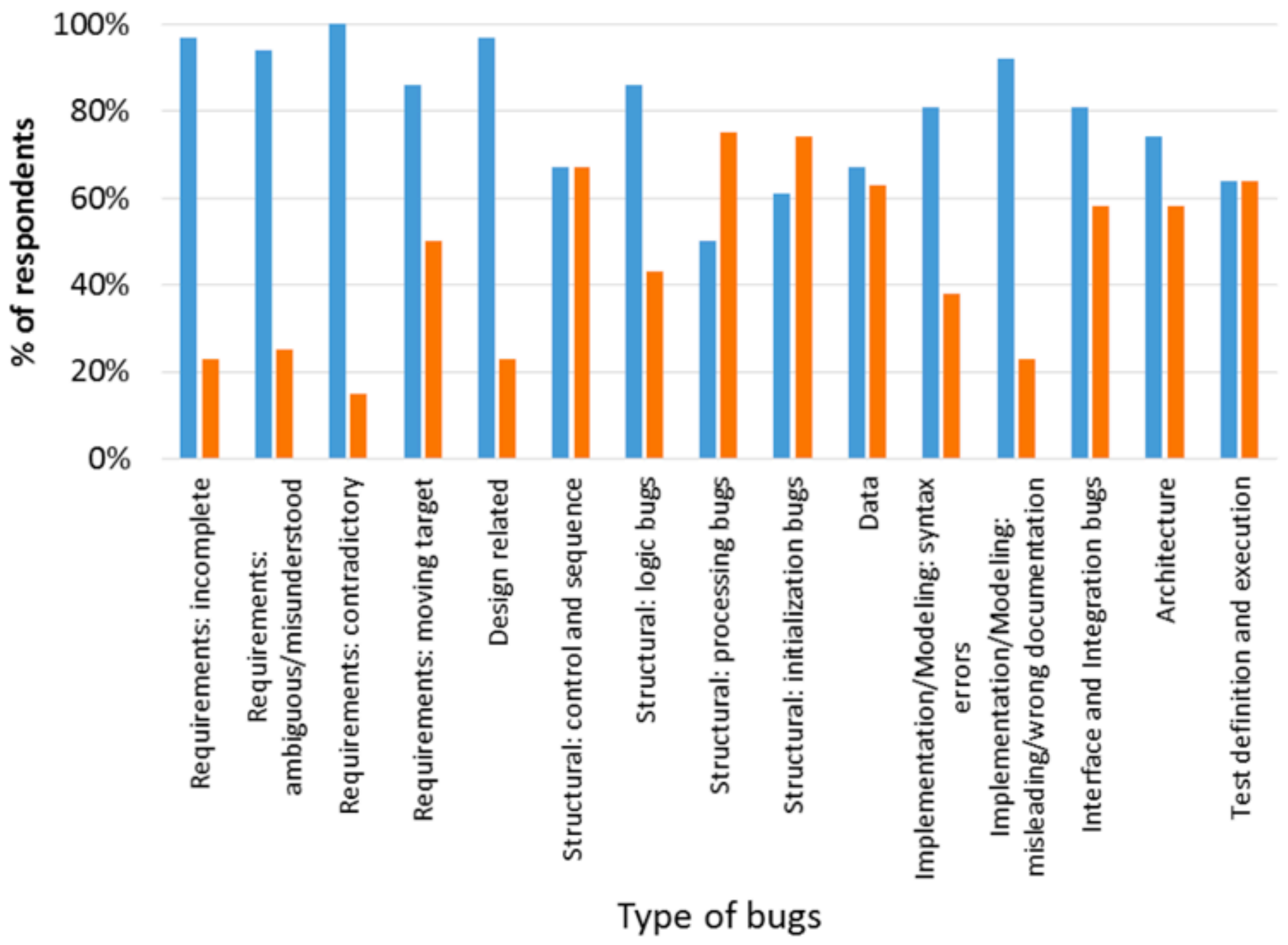
Autopilot component:

> *"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."*

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

Does my model satisfy this requirement?

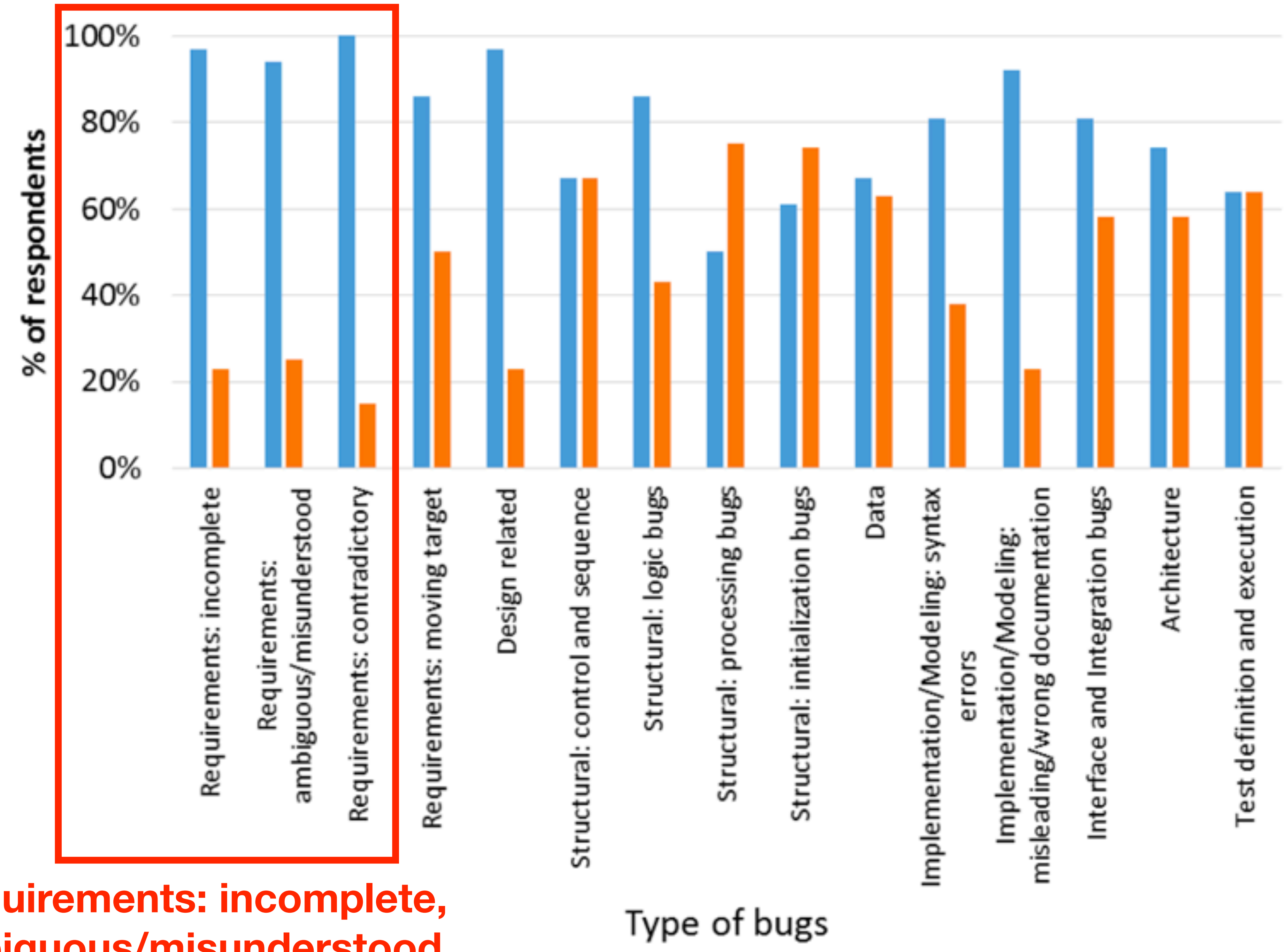
What types of bugs are found in models and code?



🟦 **in models** 🟧 **in auto-generated 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 Technical Memorandum, NASA/TM-2016-219443, 2016.

What types of bugs are found in models and code?



Requirements: incomplete, ambiguous/misunderstood, contradictory

🟦 **in models** 🟧 **in auto-generated code**

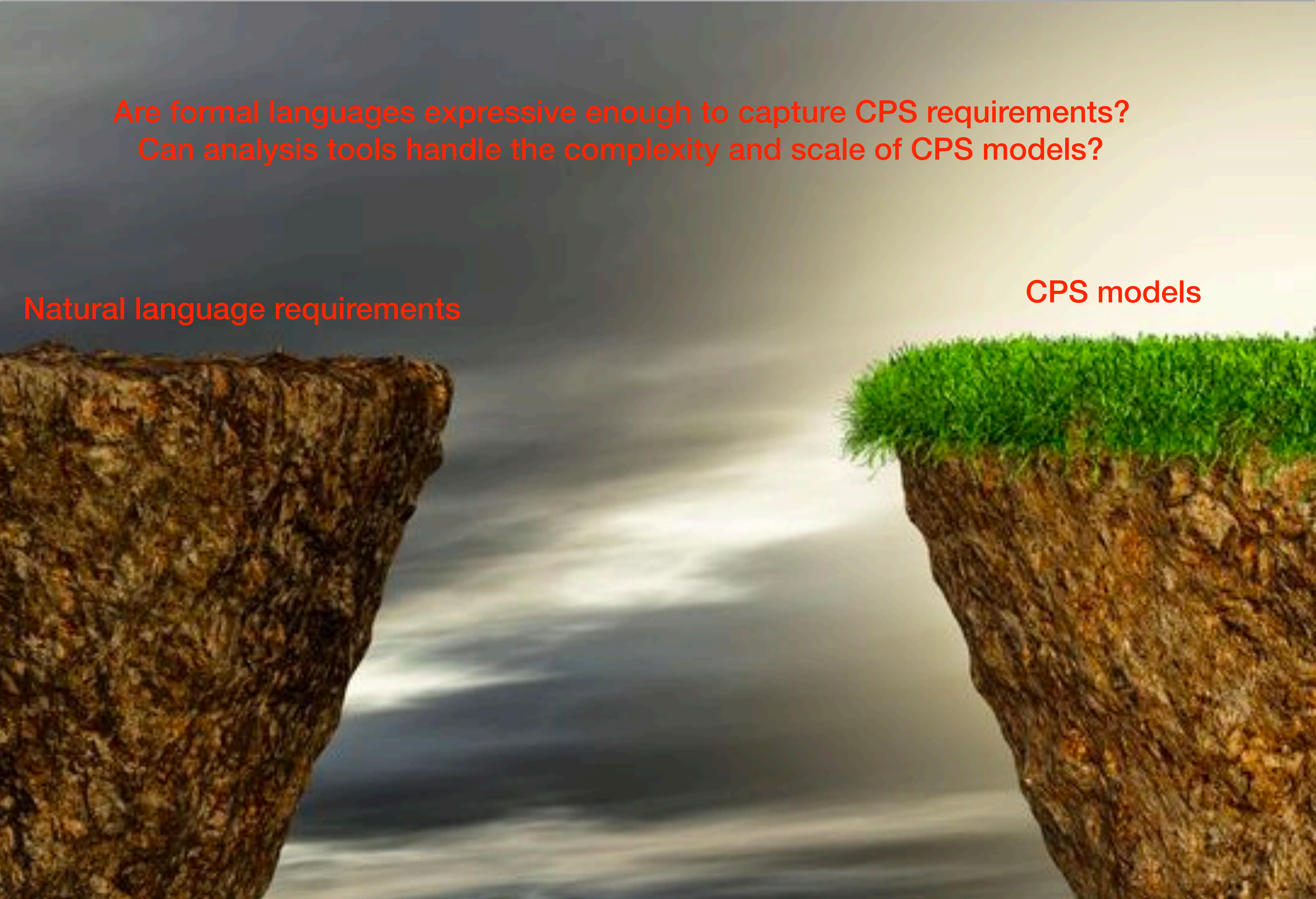
[3] Johann Schumann, Matt Knudsen, Teme Kahsai, Noble Nkwocha, Katerina Goseva-Popstojanova, Thomas Kyanko, "Report: Survey on Model-Based Software Engineering and Auto-Generated Code", NASA Technical Memorandum, NASA/TM-2016-219443, 2016.

CPS requirements and models are hard to analyze

CPS requirements and models are hard to analyze

Are formal languages expressive enough to capture CPS requirements?
Natural language requirements
CPS models

CPS requirements and models are hard to analyze

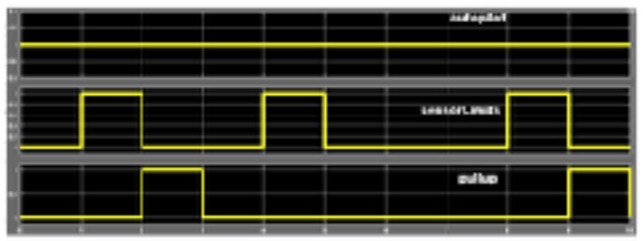


Are formal languages expressive enough to capture CPS requirements?
Can analysis tools handle the complexity and scale of CPS models?

Our Approach: End-to-end analysis of CPS

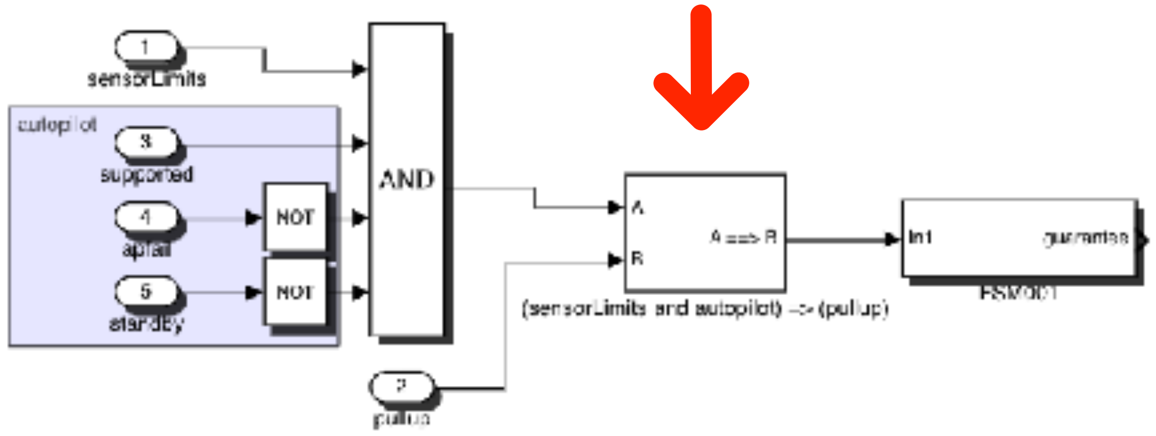
- Evaluate the feasibility and benefits
  - Application of **automated** tools to perform analysis
  - Our **end-to-end approach** involves:
    - Requirement elicitation, formalization, and analysis
    - Model analysis against formalized requirements

FSM shall always satisfy if sensorLimits & autopilot then pullup



```
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
```

Inputs	T=0	T=1	T=2	T=3
standby	F	F	F	F
apfail	F	F	F	F
supported	T	T	T	T
sensorLimits	T	F	T	F
Outputs				
pullup	F	T	F	F



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/hbourbouh/lm_challenges

The Ten Lockheed Martin CPS Challenge Problems (LMCPS)

- Requirements and models are typical of CPS systems
  - Inputs and outputs are modeled through signals
  - LMCPS models are highly numeric and often exhibit non-linear behavior

- Examples of LMCPS challenges:
  - Tustin Integrator (TUI)
  - Feedforward Cascade Connectivity Neural Network (NN)
  - 6DoF with DeHavilland Beaver Autopilot (AP)

The Ten Lockheed Martin CPS Challenge Problems (LMCPS)

- Requirements and models are typical of CPS systems
  - Inputs and outputs are modeled through signals
  - LMCPS models are highly numeric and often exhibit non-linear behavior

- Examples of LMCPS challenges:
  - Tustin Integrator (TUI) 🟦
  - Feedforward Cascade Connectivity Neural Network (NN) 🟦
  - 6DoF with DeHavilland Beaver Autopilot (AP) 🟦

🟦 **Vectors and Matrices**

The Ten Lockheed Martin CPS Challenge Problems (LMCPS)

- Requirements and models are typical of CPS systems
  - Inputs and outputs are modeled through signals
  - LMCPS models are highly numeric and often exhibit non-linear behavior

- Examples of LMCPS challenges:
  - Tustin Integrator (TUI) 🟦 🟪
  - Feedforward Cascade Connectivity Neural Network (NN) 🟦 🟪
  - 6DoF with DeHavilland Beaver Autopilot (AP) 🟦 🟪

🟦 **Vectors and Matrices**

🟪 **Non Linear and Non Algebraic Blocks**

The Ten Lockheed Martin CPS Challenge Problems (LMCPS)

- Requirements and models are typical of CPS systems
  - Inputs and outputs are modeled through signals
  - LMCPS models are highly numeric and often exhibit non-linear behavior

- Examples of LMCPS challenges:
  - Tustin Integrator (TUI) 🟦 🟪
  - Feedforward Cascade Connectivity Neural Network (NN) 🟦 🟪
  - 6DoF with DeHavilland Beaver Autopilot (AP) 🟦 🟪 🟩

🟦 **Vectors and Matrices**

🟩 **Continuous time blocks**

🟪 **Non Linear and Non Algebraic Blocks**

The Ten Lockheed Martin CPS Challenge Problems (LMCPS)

- Requirements and models are typical of CPS systems
  - Inputs and outputs are modeled through signals
  - LMCPS models are highly numeric and often exhibit non-linear behavior

- Examples of LMCPS challenges:
  - Tustin Integrator (TUI) 🟦 🟪
  - Feedforward Cascade Connectivity Neural Network (NN) 🟦 🟪
  - 6DoF with DeHavilland Beaver Autopilot (AP) 🟦 🟪 🟩 🟥

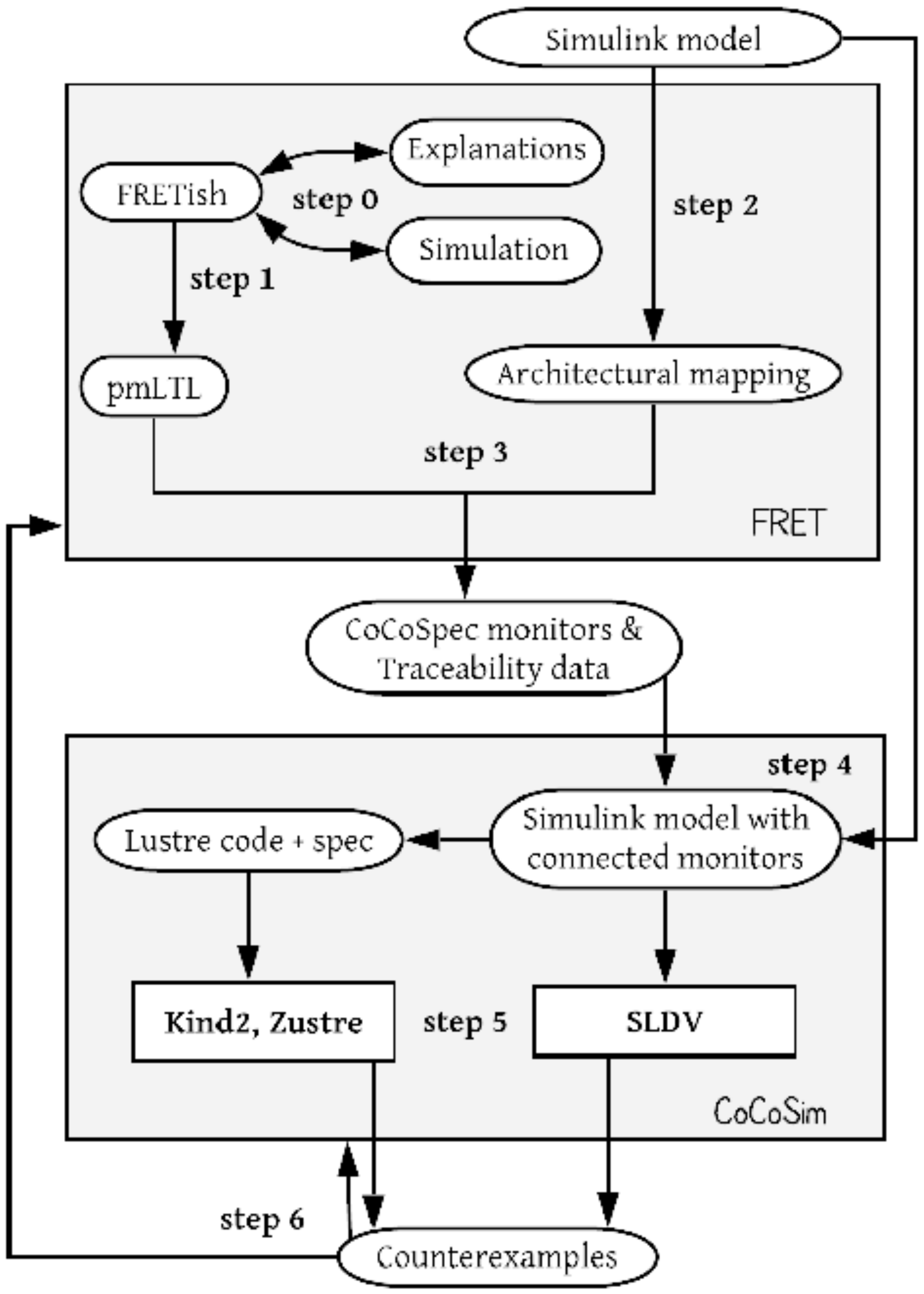
🟦	**Vectors and Matrices**	🟩	**Continuous time blocks**
🟪	**Non Linear and Non Algebraic Blocks**	🟥	**Complex req. formalizations**

The FRET-CoCoSim Integrated Framework

- Elicit, explain, and formalize the semantics of the given natural language requirements
  (Steps: 0, 1)

- Generate verification code and monitors that can be automatically attached to the Simulink models
  (Steps: 2, 3, 4)

- Perform verification by using Lustre-based model checkers or SLDV
  (Steps: 5, 6)



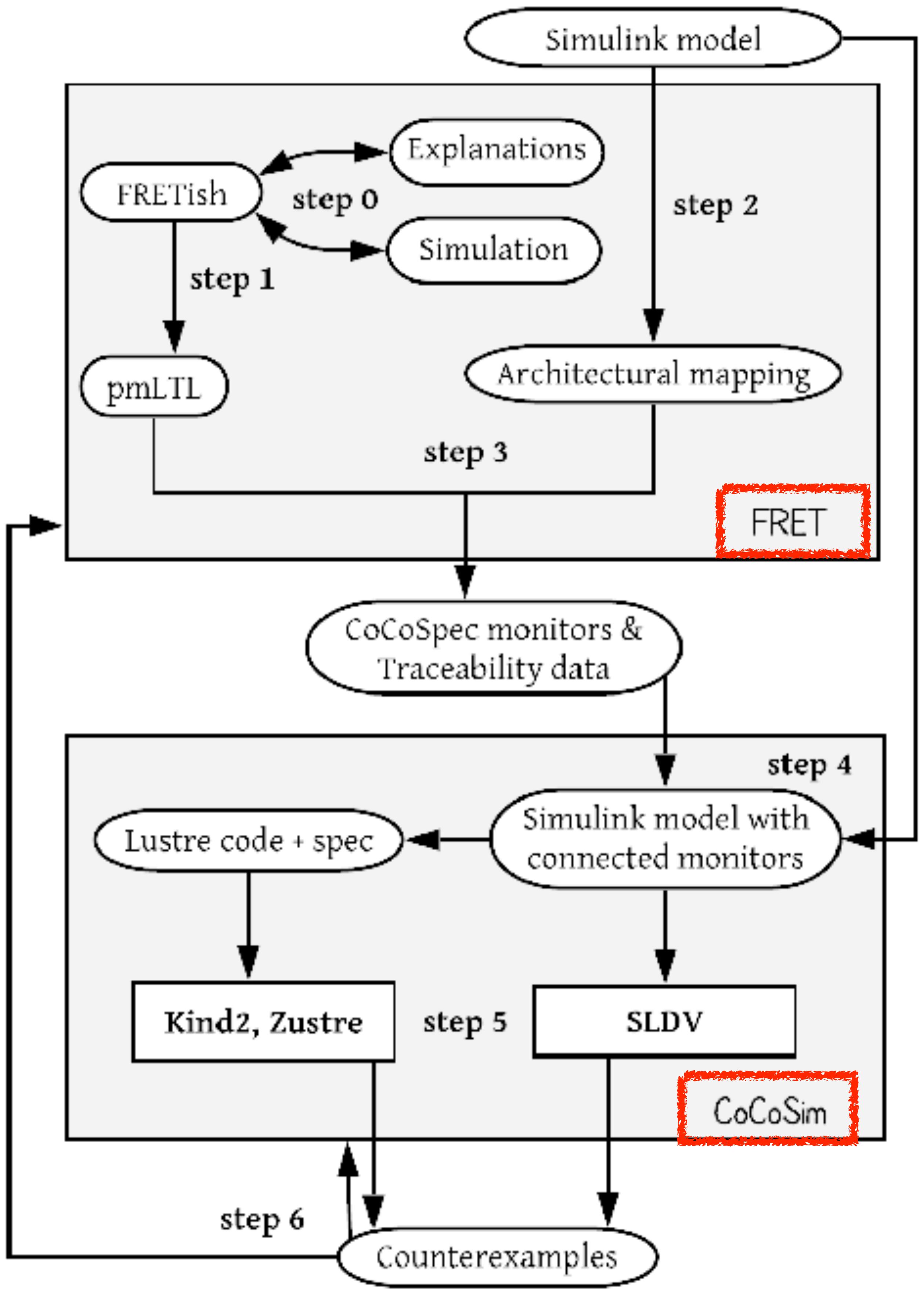
The FRET-CoCoSim Integrated Framework

- Elicit, explain, and formalize the semantics of the given natural language requirements
  (Steps: 0, 1)

- Generate verification code and monitors that can be automatically attached to the Simulink models
  (Steps: 2, 3, 4)

- Perform verification by using Lustre-based model checkers or SLDV
  (Steps: 5, 6)

All tools are open source and developed at NASA Ames



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 abs(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 abs(roll_angle) >30 & 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 abs(roll_angle) >30 & roll_hold_mode_engagement,
Autopilot shall immediately satisfy
roll_hold_reference = 30*sign(roll_angle)

FRETish fields:

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

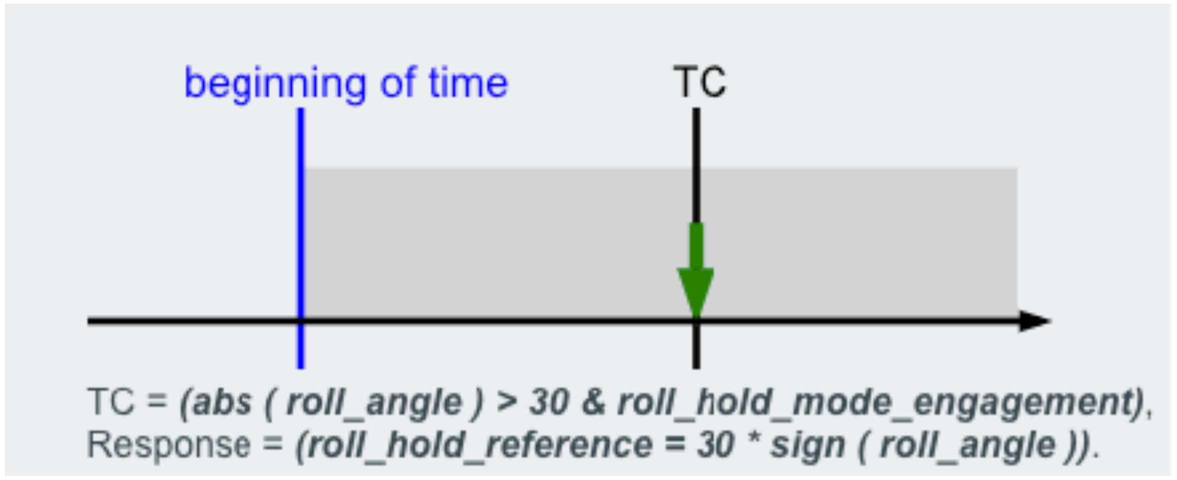
Step 0: Requirement Elicitation

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 abs(roll_angle) >30 & roll_hold_mode_engagement, Autopilot shall immediately satisfy roll_hold_reference = 30*sign(roll_angle)



TC = *(abs (roll_angle) > 30 & roll_hold_mode_engagement)*,
Response = *(roll_hold_reference = 30 * sign (roll_angle))*.

FRETish fields:

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

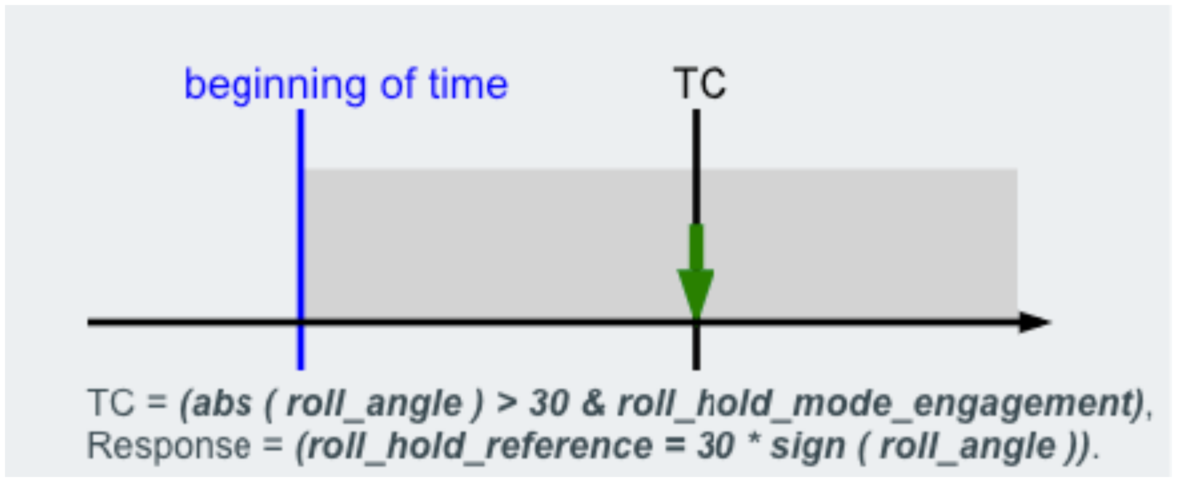
Step 0: Requirement Elicitation

Natural language requirement:

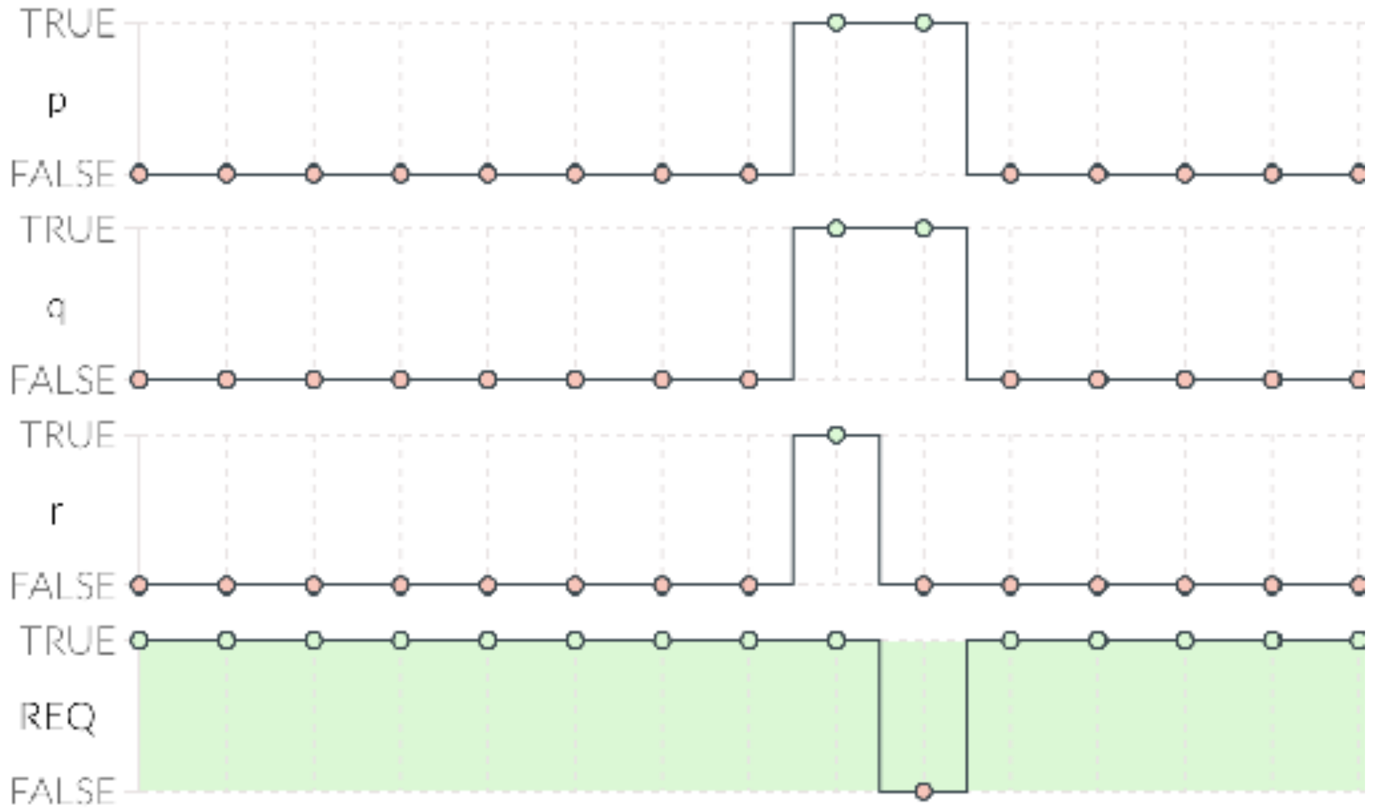
[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

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



TC = *(abs (roll_angle) > 30 & roll_hold_mode_engagement)*,
Response = *(roll_hold_reference = 30 * sign (roll_angle))*.



FRETish fields:

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

p: abs(roll_angle)
q: roll_hold_mode_engagement
r: response

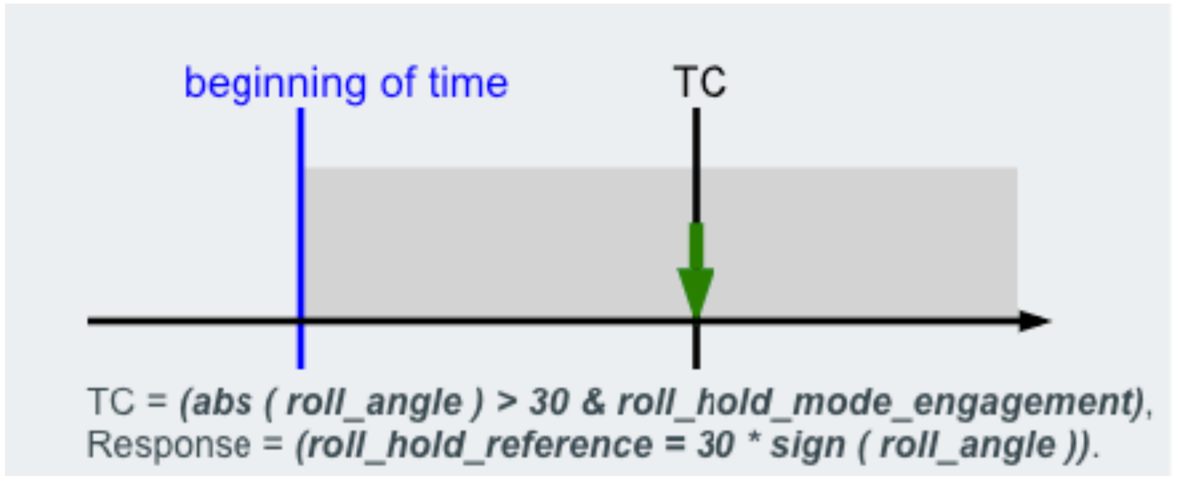
Step 0: Requirement Elicitation

Natural language requirement:

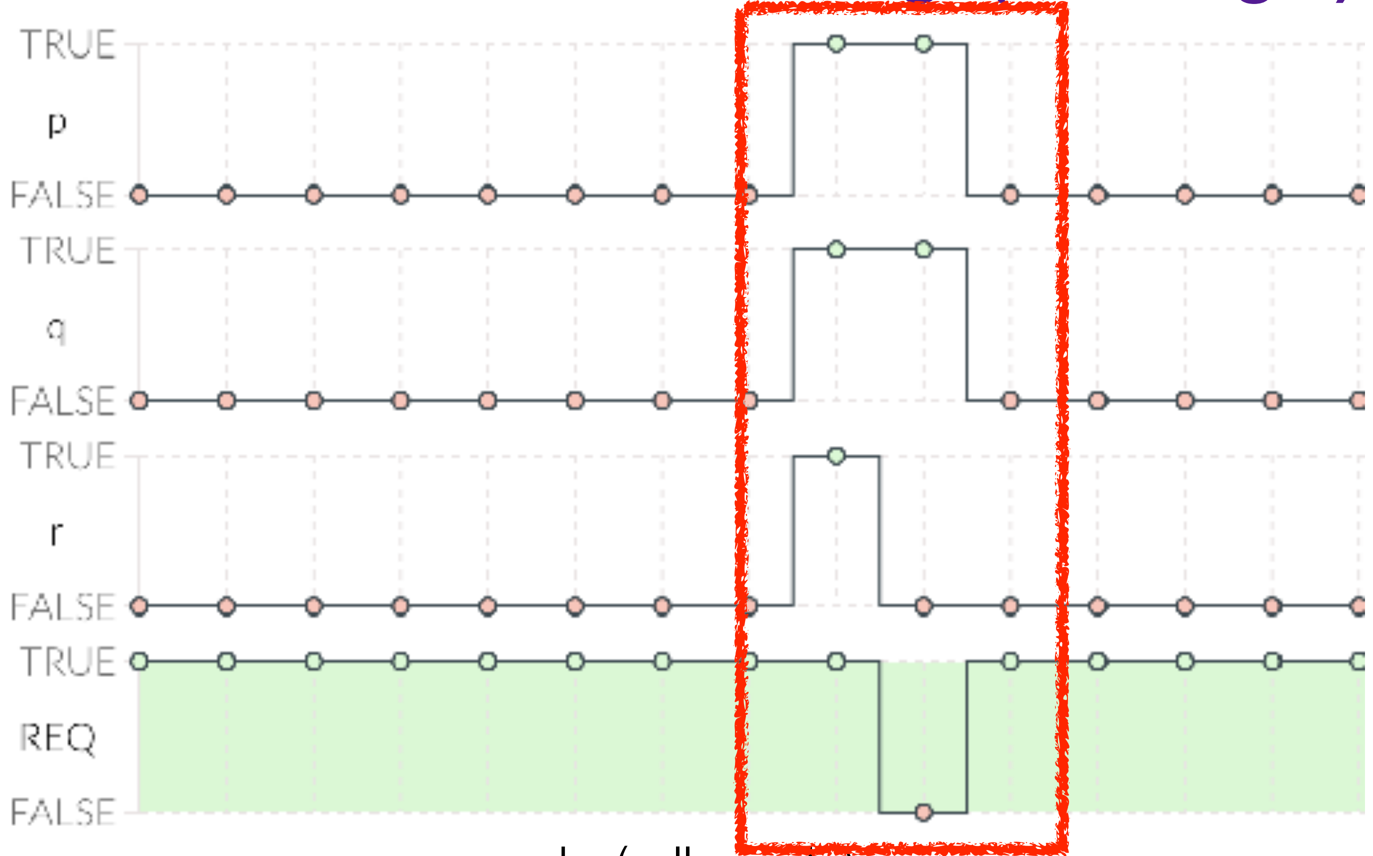
[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

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



TC = *(abs (roll_angle) > 30 & roll_hold_mode_engagement)*,
Response = *(roll_hold_reference = 30 * sign (roll_angle))*.



FRETish fields:

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

p: abs(roll_angle)
q: 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 (abs(roll_angle) >30 & roll_hold_mode_engagement) => roll_hold_reference = 30*sign(roll_angle)

FRETish fields:

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

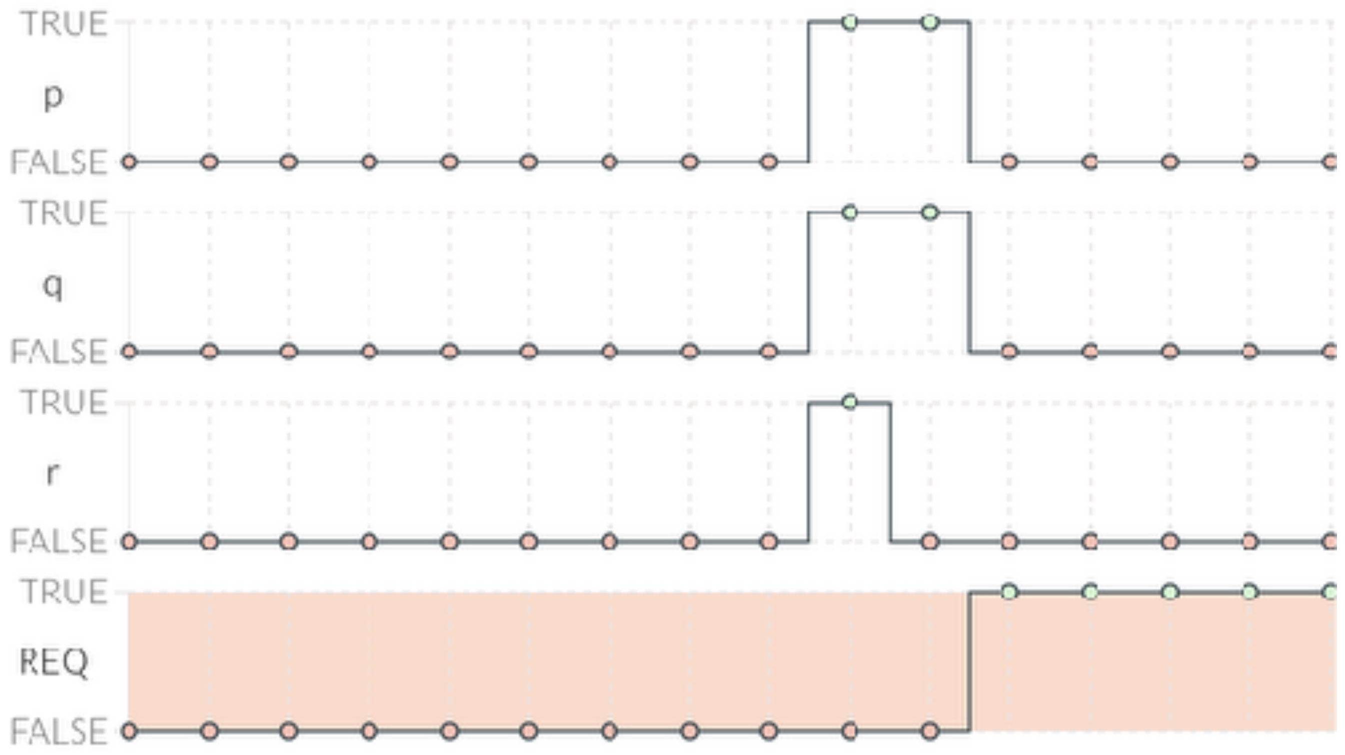
Step 0: Requirement Elicitation

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 (abs(roll_angle) >30 & roll_hold_mode_engagement) => roll_hold_reference = 30*sign(roll_angle)



p: abs(roll_angle)
q: roll_hold_mode_engagement
r: roll_hold_reference = 30 * sign(roll_angle)

FRETish fields:

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

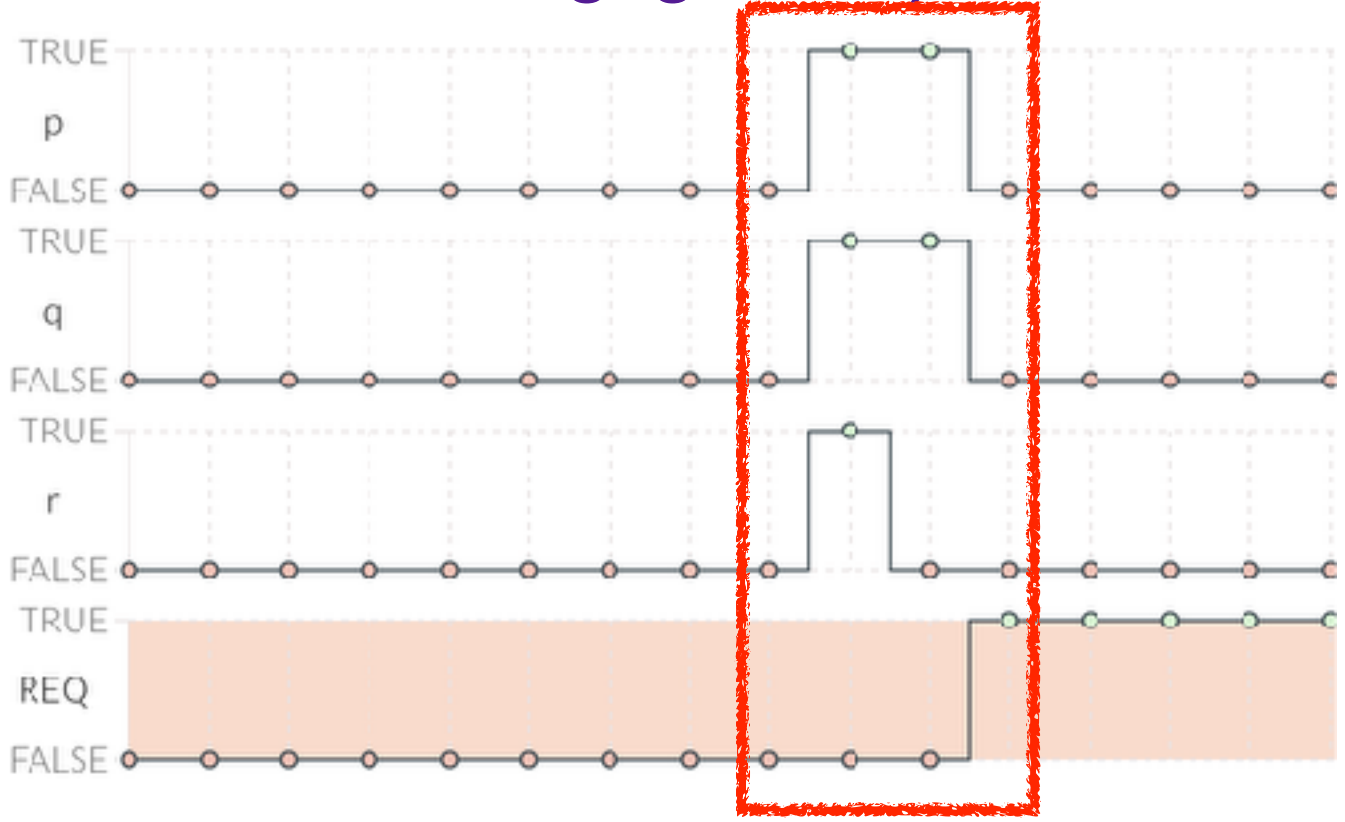
Step 0: Requirement Elicitation

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 (abs(roll_angle) >30 & roll_hold_mode_engagement) => roll_hold_reference = 30*sign(roll_angle)



p: abs(roll_angle)
q: roll_hold_mode_engagement
r: roll_hold_reference = 30 * sign(roll_angle)

FRETish fields:

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

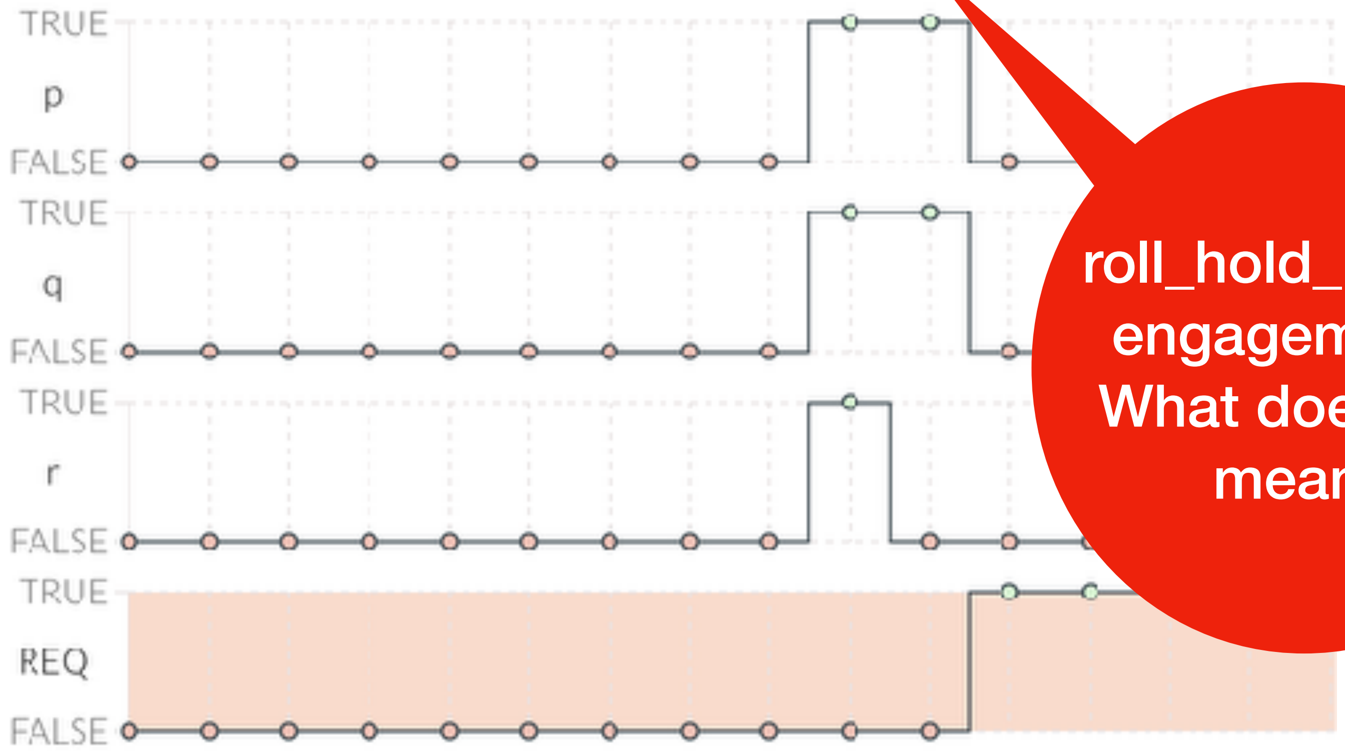
Step 0: Requirement Elicitation

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 (abs(roll_angle) >30 & roll_hold_mode_engagement) => roll_hold_reference = 30*sign(roll_angle)



roll_hold_mode_ engagement ? What does that mean?

bs(roll_angle)
oll_hold_mode_engagement
oll_hold_reference = 30 * sign(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 (abs(roll_angle) >30) => roll_hold_reference = 30*sign(roll_angle)

FRETish fields:

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

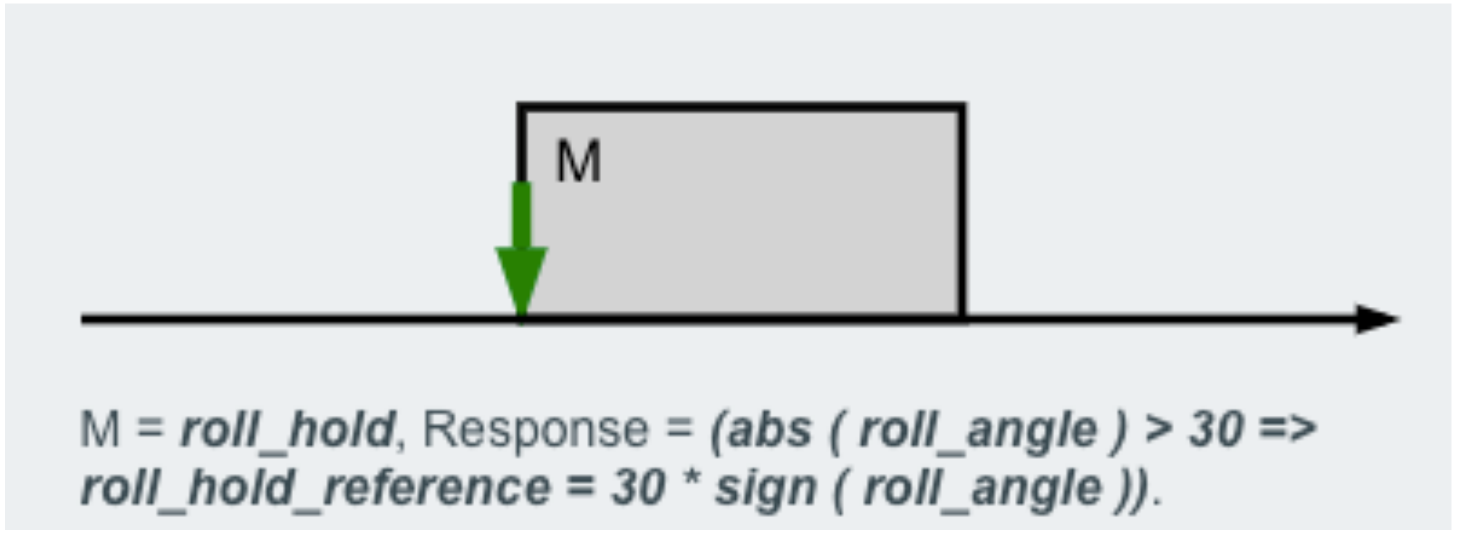
Step 0: Requirement Elicitation

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 (abs(roll_angle) >30) => roll_hold_reference = 30*sign(roll_angle)



M = ***roll_hold***, Response = ***(abs (roll_angle) > 30 => roll_hold_reference = 30 * sign (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 (abs(roll_angle) >30) => roll_hold_reference = 30*sign(roll_angle)

Past Time Linear Temporal Logic formula:

```
H(roll_hold & (! FTP | (Y (! roll_hold))) => abs(roll_angle >
30 => roll_hold_reference = 30 * sign(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(roll_hold & (! FTP | (Y (! roll_hold)) => abs(roll_angle >
30 => roll_hold_reference = 30 * sign(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))
```

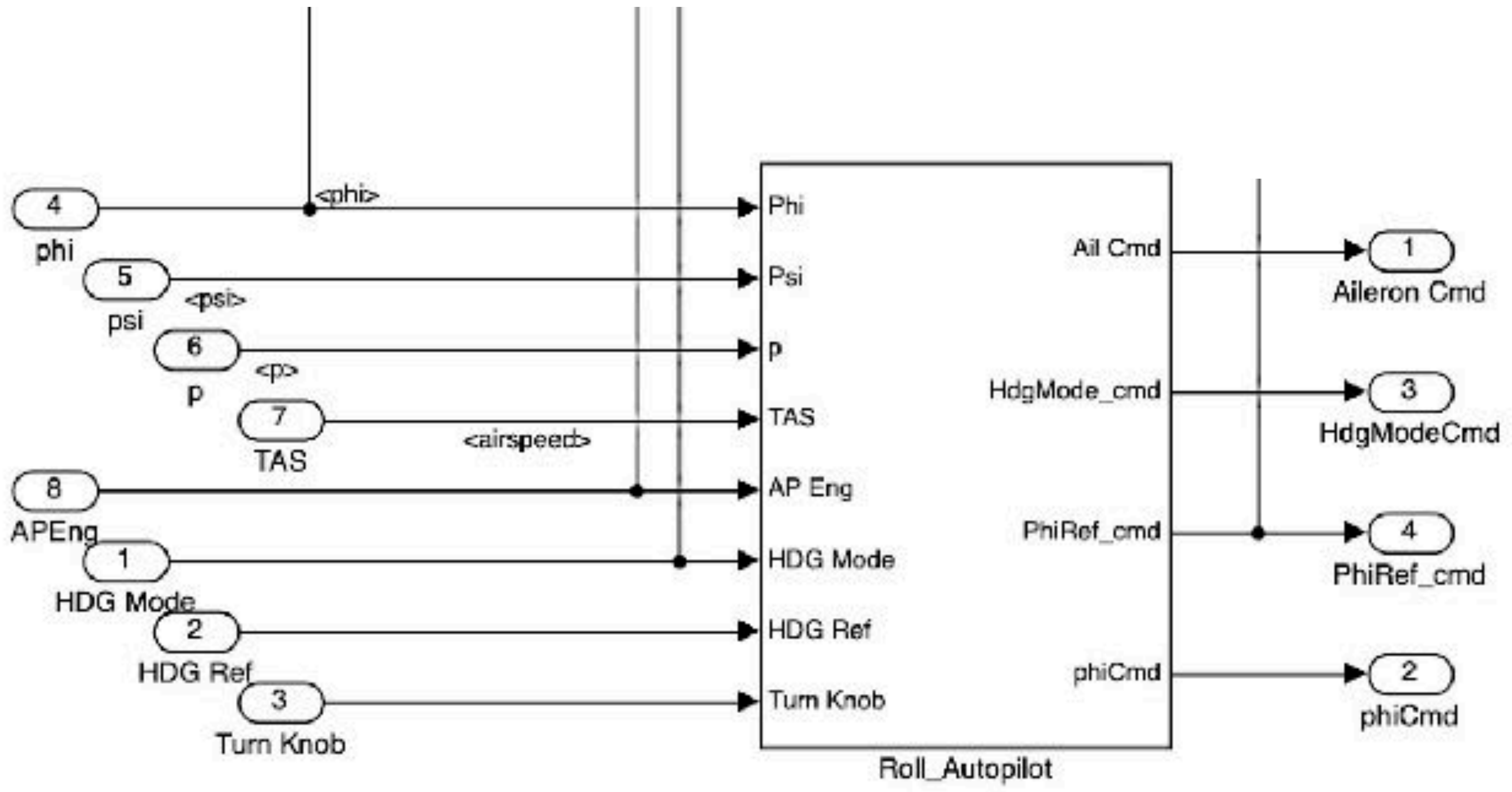
Step 3: Automated Architectural Mapping

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 (abs(roll_angle) >30) => roll_hold_reference = 30*sign(roll_angle)



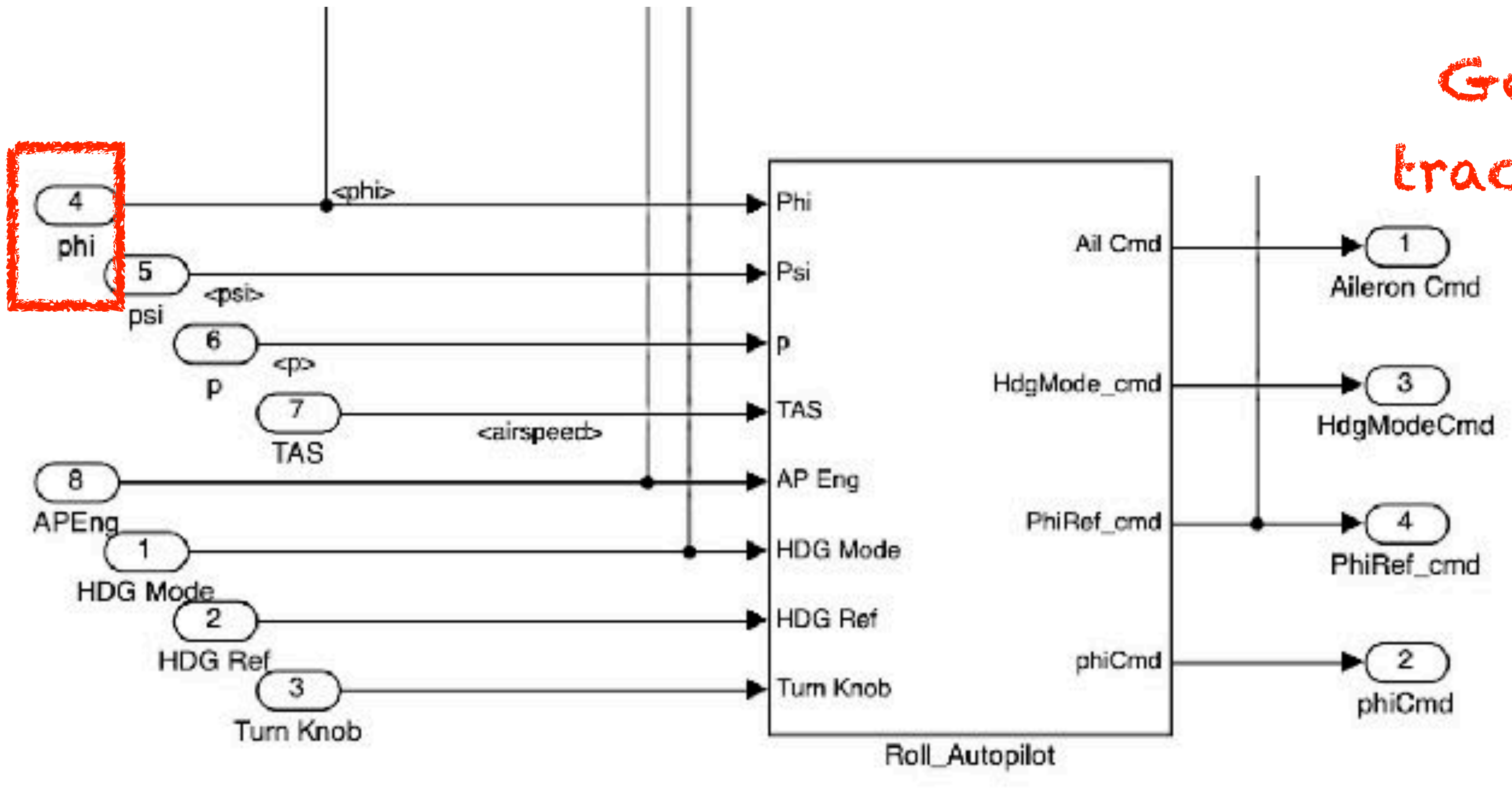
Step 4: Automated Architectural Mapping

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 (abs(roll_angle) >30) => roll_hold_reference = 30*sign(roll_angle)



Generation of traceability data!

Step 4: Generation of Simulink Monitors

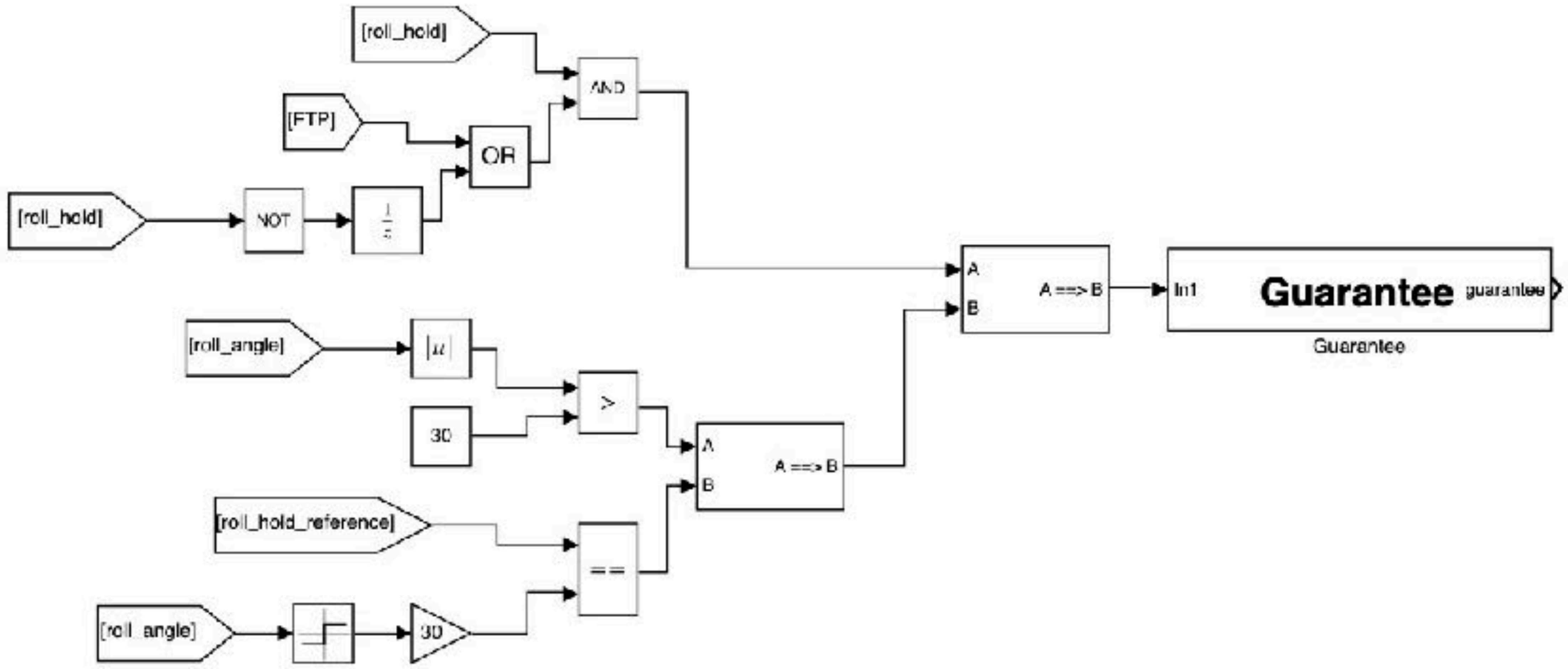
Natural language requirement:

[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:

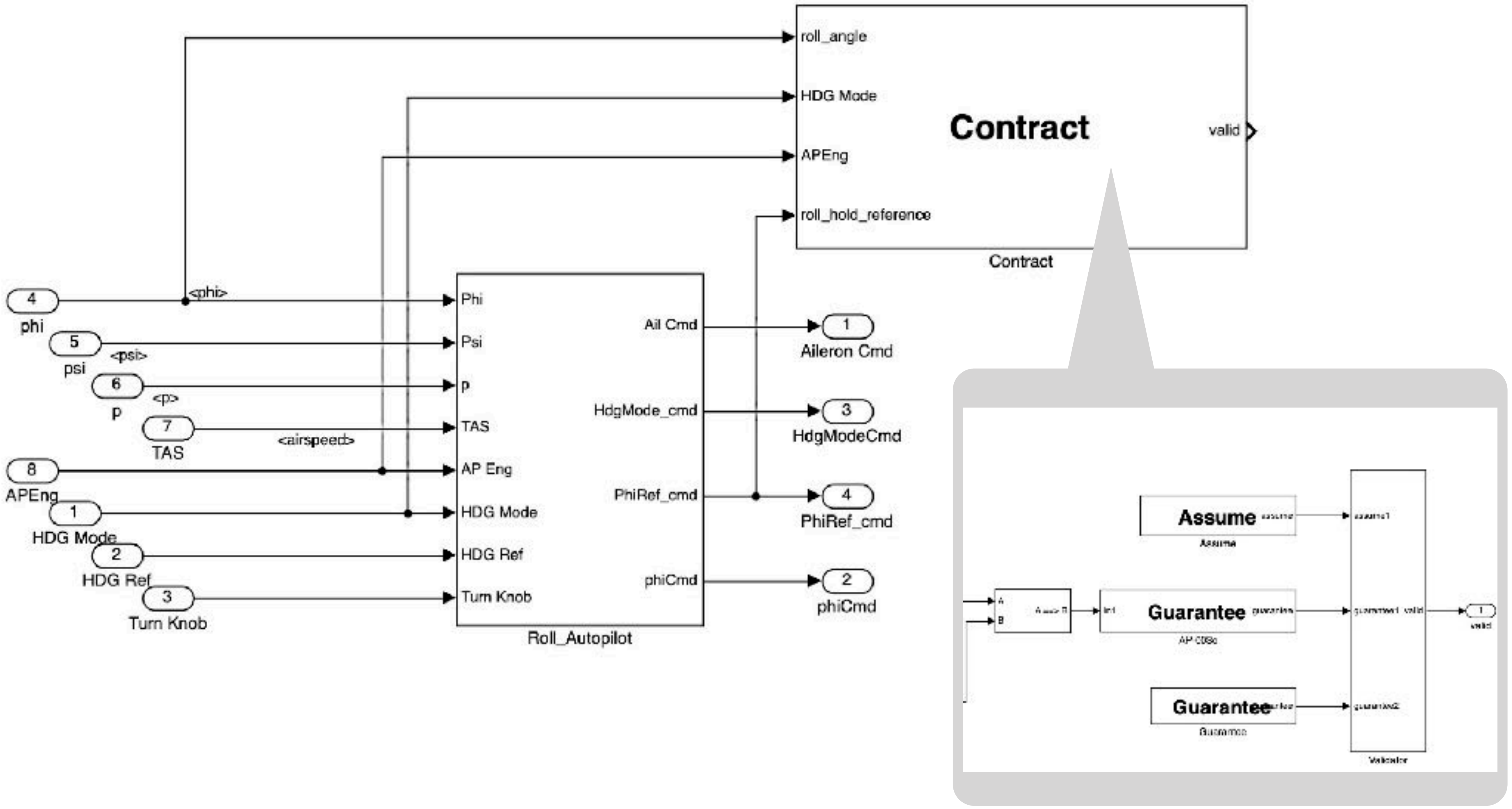
```
-- AP-003c-v3 requirement in CoCoSpec
guarantee H((roll_hold and (FTP or (pre (not roll_hold))))
            => abs(roll_angle) > 30 =>
            roll_hold_reference = 30 * sign(roll_angle))
```

Simulink monitor:



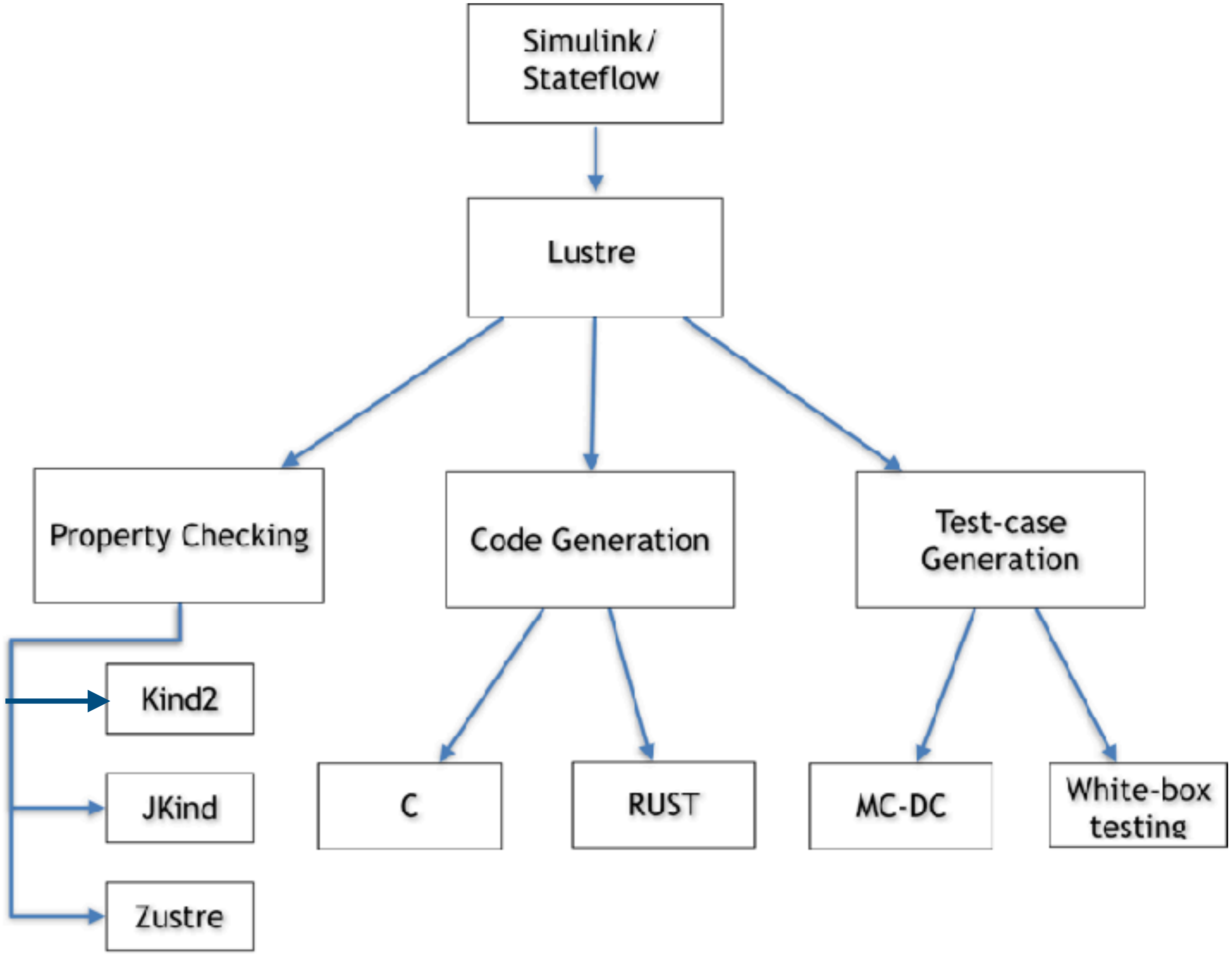
Step 4: Generation of Simulink Monitors

Simulink monitor automatically attached on the model:

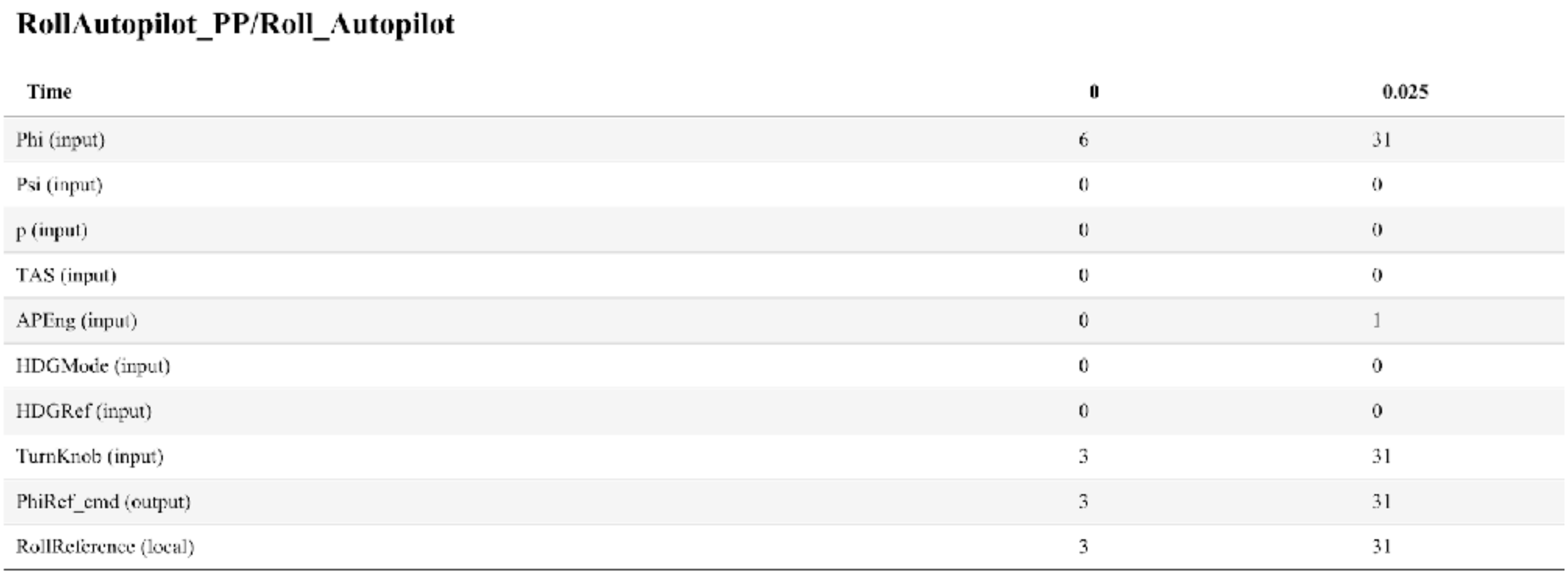


The CoCoSim Tool

CoCoSim (Contract based Compositional verification of Simulink models) is used to verify user-supplied requirements.



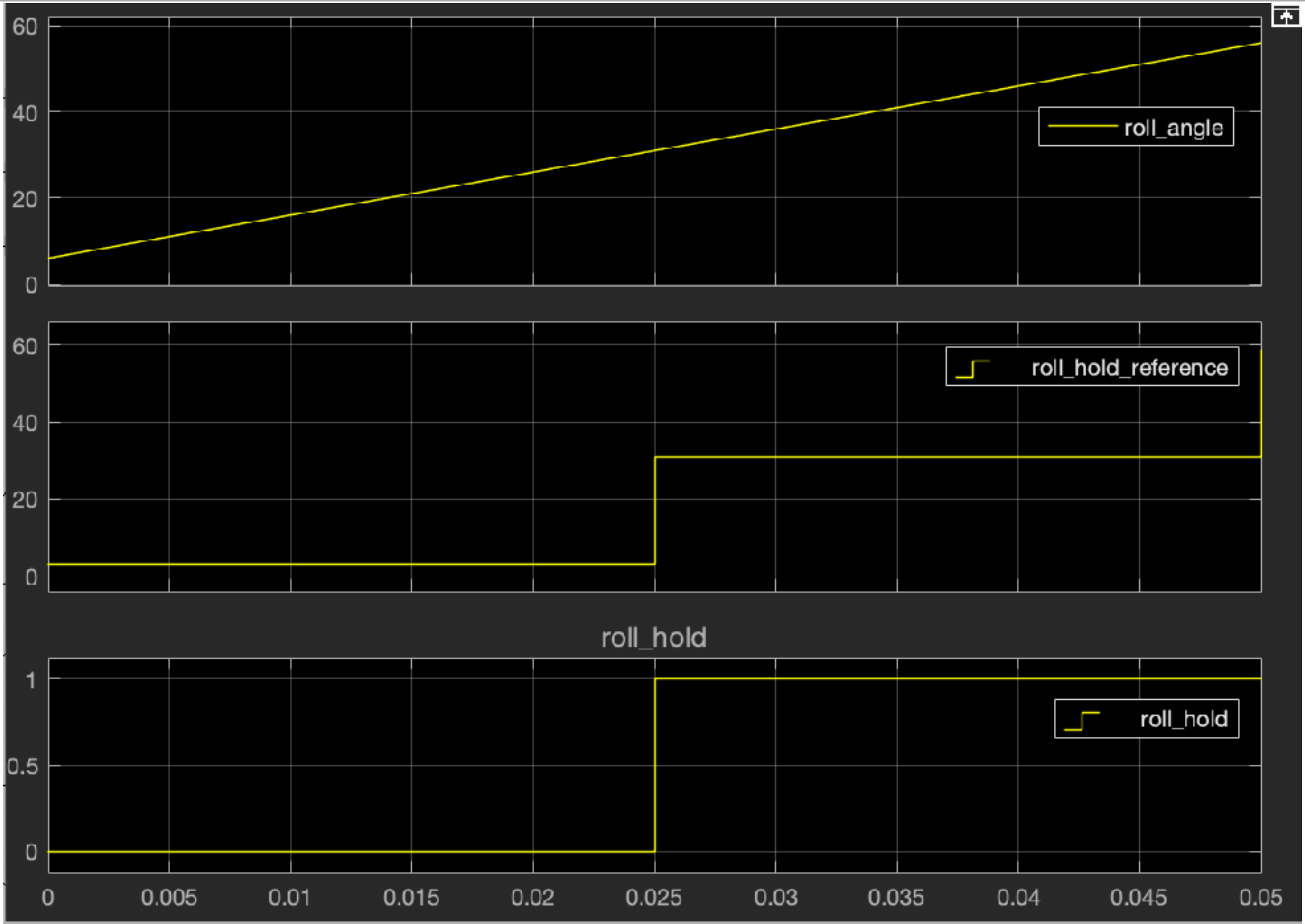
Step 5: Analysis and Counterexample Generation



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



Verification results

TABLE VII: LMCPS verification results. N_R: #requirements, N_F: #formalized requirements, N_A: #requirements analyzed by Kind2. Analysis results categorized by **V**alid/**IN**valid/**UN**decided. Timeout (TO) was set to 2 hours

Name	N_R	N_F	N_A	**Kind2 V/IN/UN**	**Kind2 t(s)**
TSM	6	6	6	5/1/0	37.7
FSM	13	13	13	7/6/0	141.1
TUI	4	3	3	2/1/0	19.2
REG	10	10	10	1/5/4	TO
NLG	7	7	7	0/0/7	TO
NN	4	4	4	0/0/4	TO
EB	5	3	3	0/0/3	TO
AP	14	13	8	5/3/0	40.6
SWIM	3	3	3	2/1/0	25
EUL	8	7	7	1/6/0	43
Total	74	69	64	23/23/18	

Lessons Learned

Can LMCPS requirements be captured in FRET?

- We captured 69/74 LMCPS requirements in FRET
  - We were not able to formalize requirement [TUI-004] that contains a temporal condition
  - The rest of the requirements that were not expressed were either out of scope of the Simulink model or unclear

Lessons Learned

Can LMCPS requirements be captured in FRET?

- We captured 69/74 LMCPS requirements in FRET
  - We were not able to formalize requirement [TUI-004] that contains a temporal condition
  - The rest of the requirements that were not expressed were either out of scope of the Simulink model or unclear

Is FRETish intuitive?

- Several requirements fall into recurring patterns.
- We are currently extending FRET with the capability of defining typical requirement patterns

Lessons Learned

Can LMCPS requirements be captured in FRET?

- We captured 69/74 LMCPS requirements in FRET
    - We were not able to formalize requirement [TUI-004] that contains a temporal condition
    - The rest of the requirements that were not expressed were either out of scope of the Simulink model or unclear

Is FRETish intuitive?

- Several requirements fall into recurring patterns.
- We are currently extending FRET with the capability of defining typical requirement patterns

Are FRET explanations useful?

- We extensively relied on the semantic descriptions and diagrams
- They helped us identify several semantic nuances

Lessons Learned

How effective is the FRET-CoCoSim integration?

- We were able to generate specifications and traceability data for all LMCPS challenges
- Simulink monitors were automatically generated and attached to the models

Lessons Learned

How effective is the FRET-CoCoSim integration?

- We were able to generate specifications and traceability data for all LMCPS challenges
- Simulink monitors were automatically generated and attached on the models

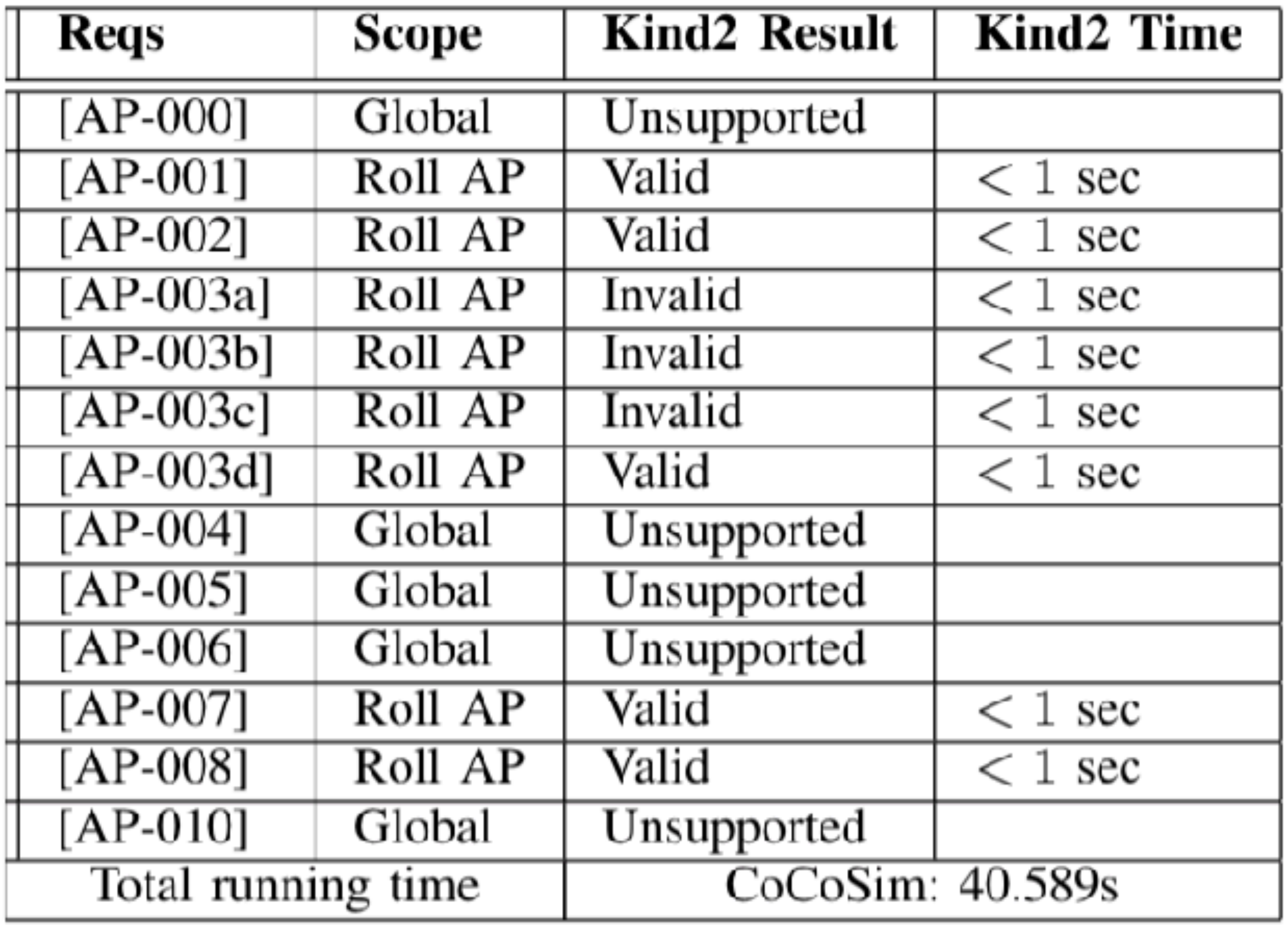
How did we deal with model and specification complexity?

- We performed modular analysis
- Simulink monitors were automatically deployed at different system levels

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	

Lessons Learned

Which types of property reasoning/checking did we find useful?

- Vacuity checking & simulation
- Check a weaker property
- Check feasibility with bounded model checking

What else was useful?

- Abstractions for non-linear functions

To summarize

- LMCPS provides a valuable case study to evaluate:
  - Requirements elicitation
  - Analysis tools
- 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
- CPS 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!