Runtime Verification with R2U2

14^{th} Summer School on Formal Techniques

Kristin Yvonne Rozier
Iowa State University of Science and Technology
May 27&29, 2025

A Recent Motivation. . .
Crash of Lion Air's Flight 610 Boeing 737-8 MAX

- October 29, 2018

A Recent Motivation. . .
Crash of Lion Air's Flight 610 Boeing 737-8 MAX

- October 29, 2018
- nose-up tendency
- Maneuvering Characteristics Augmentation System (MCAS) programmed to trim
- Left Primary Flight Display (PFD) fixed October 28

A Recent Motivation. . .
Crash of Lion Air's Flight 610 Boeing 737-8 MAX

- October 29, 2018
- nose-up tendency
- Maneuvering Characteristics Augmentation System (MCAS) programmed to trim
- Left Primary Flight Display (PFD) fixed October 28
- AoA sensor passed pre-flight check

A Recent Motivation. . .

Crash of Lion Air's Flight 610 Boeing 737-8 MAX

- October 29, 2018
- nose-up tendency
- Maneuvering Characteristics Augmentation System (MCAS) programmed to trim
- Left Primary Flight Display (PFD) fixed October 28
- AoA sensor passed pre-flight check
- AoA sensor detected approaching stall
- Left control column stick shaker alert

A Recent Motivation. . .
Crash of Lion Air's Flight 610 Boeing 737-8 MAX

- October 29, 2018
- nose-up tendency
- Maneuvering Characteristics Augmentation System (MCAS) programmed to trim
- Left Primary Flight Display (PFD) fixed October 28
- AoA sensor passed pre-flight check
- AoA sensor detected approaching stall
- Left control column stick shaker alert
- PIC diagnosed *left PFD fault*
  - continuous 20° displacement of left AoA
  - left and right instruments indicating different altitudes
  - faulty AoA, altitude ⇒ stall range
- automatic AND trim (*Aircraft Nose Down*)

A Recent Motivation. . .
Crash of Lion Air's Flight 610 Boeing 737-8 MAX

- October 29, 2018
- nose-up tendency
- Maneuvering Characteristics Augmentation System (MCAS) programmed to trim
- Left Primary Flight Display (PFD) fixed October 28
- AoA sensor passed pre-flight check
- AoA sensor detected approaching stall
- Left control column stick shaker alert
- PIC diagnosed *left PFD fault*
  - continuous 20° displacement of left AoA
  - left and right instruments indicating different altitudes
  - faulty AoA, altitude $\Rightarrow$ stall range
- automatic AND trim (*Aircraft Nose Down*)
- Crash into Java Sea 11 min after takeoff, killing all 189 on-board

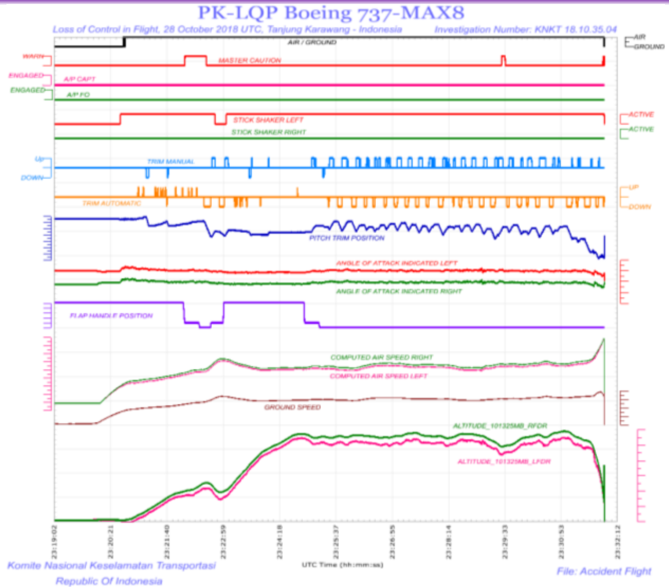


Figure 5: The significant parameters from the accident flight

14

A Recent Motivation. . .
Crash of Lion Air's Flight 610 Boeing 737-8 MAX

This is Unsafe Autonomy!

- The 737-8 MAX tends to pitch up
- High AoA + low airspeed + low altitude = possible stall, crash
- Left and right sensors tend to agree; only need to check one
- Fixed sensors for left PFD passed pre-flight check; tend to function correctly

A Recent Motivation. . .
Crash of Lion Air's Flight 610 Boeing 737-8 MAX

This is Unsafe Autonomy!

- The 737-8 MAX tends to pitch up
- High AoA + low airspeed + low altitude = possible stall, crash
- Left and right sensors tend to agree; only need to check one
- Fixed sensors for left PFD passed pre-flight check; tend to function correctly

Design was done by humans!
Is there hope for AI/learned autonomous systems?

A Recent Motivation. . .
Crash of Lion Air's Flight 610 Boeing 737-8 MAX

How can pilots compensate for a malfunction if they are not aware of how it works?[1]

What we want:

"I'm doing this because the **left AoA sensor** indicates AoA is **above the threshold** of X; at **low altitude threshold** Y from **left altitude sensor** this indicates a **likely stall**."

[1] Les Abend. "Lion Air crash: Is it safe to get on a Boeing 737 MAX plane?" CNN Opinion, Updated 11:17 PM ET, November 28, 2018.

A Recent Motivation...
Crash of Lion Air's Flight 610 Boeing 737-8 MAX

"Sanity Check" Specifications Relevant to this Mission:

- The AoA cannot be 20° different between two sides of the aircraft
- The altitude cannot be multiple values simultaneously.
- Altitude, airspeed verified by ATC to pilots, not autonomous system
- The pilot should not be fighting the stick in manual flight mode.
- The left and right PFD should agree; PIC and SIC should not have different control column modes like stick shake

A Recent Motivation. . .
Crash of Lion Air's Flight 610 Boeing 737-8 MAX

"Sanity Check" Specifications Relevant to this Mission:

- The AoA cannot be 20° different between two sides of the aircraft
- The altitude cannot be multiple values simultaneously.
- Altitude, airspeed verified by ATC to pilots, not autonomous system
- The pilot should not be fighting the stick in manual flight mode.
- The left and right PFD should agree; PIC and SIC should not have different control column modes like stick shake

These *sanity checks* might have prevented the crash

A Recent Motivation. . .
Crash of Lion Air's Flight 610 Boeing 737-8 MAX

"Sanity Check" Specifications Relevant to this Mission:

- The AoA cannot be 20° different between two sides of the aircraft
- The altitude cannot be multiple values simultaneously.
- Altitude, airspeed verified by ATC to pilots, not autonomous system
- The pilot should not be fighting the stick in manual flight mode.
- The left and right PFD should agree; PIC and SIC should not have different control column modes like stick shake

These *sanity checks* might have prevented the crash

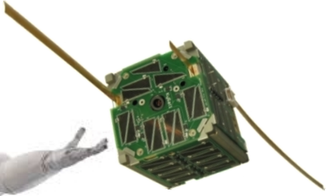
Need to continuously check safety specifications during operation

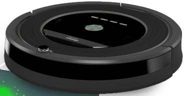
How is **Flight Software**

How is **Flight Software** Different from **Software**?

How is **Flight Software** Different from **Software**?

- **Has to** work





How is **Flight Software** Different from **Software**?

- **Has to** work
- Need capabilities for **independent checks**

How is **Flight Software** Different from **Software**?

- **Has to** work
- Need capabilities for **independent checks**
- Need **transparent** ties to **verification** tasks

Runtime Verification: Required for Autonomy & Future CPS

How do we fit RV into resources on-board already-flying CPS?

Satisfying Requirements

Satisfying Requirements

RESPONSIVE
REALIZABLE
UNOBTRUSIVE
Unit

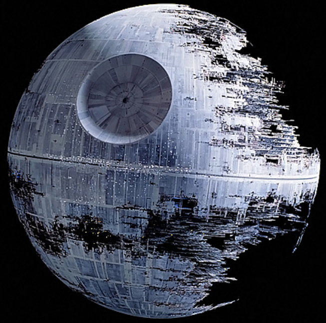
R2U2

Runtime Monitoring On-Board

Adding currently available runtime monitoring capabilities to the UAS would change its flight certification.

"Losing flight certification is like moving over to the dark side: once you go there you can never come back."

— Doug McKinnon,
NASA Ames' UAS Crew Chief



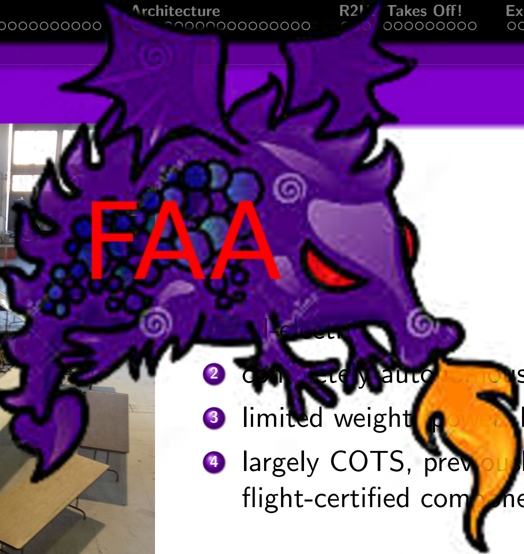
Swift UAS

Swift in Hanger Annex N211-A

1. all-electric
2. completely autonomous
3. limited weight, power, hardware
4. largely COTS, previously flight-certified components (cheap)

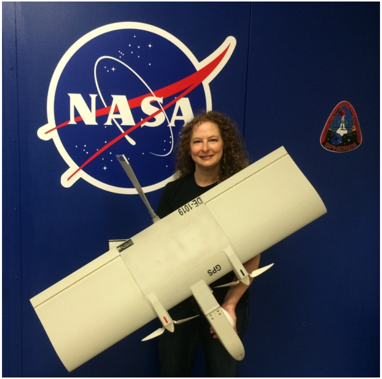
Swift UAS

Swift in Hanger Annex N211-A



1. l-electri
2. con etely auto us
3. limited weight hardware
4. largely COTS, prev ly flight-certified com nents (cheap)

How It All Began



NASA Ames Director's Colloquium: "No More Helicopter Parenting: Intelligent Autonomous UAS."
NASA Ames Research Center Director's Colloquium, June 10, 2014

`https://www.youtube.com/watch?v=FTxT-fbCleA`

Requirements

Realizability:

- easy, *expressive* specification language
- *generic* interface to connect to a wide variety of systems
- *adaptable* to missions, mission stages, platforms

Responsiveness:

- *continuously monitor* the system
- *detect deviations* in *real time*
- *enable mitigation* or rescue measures

Unobtrusiveness:

- *functionality*: not change behavior
- *certifiability*: avoid re-certification of flight software/hardware
- *timing*: not interfere with timing guarantees
- *tolerances*: obey size, weight, power, telemetry bandwidth constraints
- *cost*: use commercial-off-the-shelf (COTS) components

REALIZABILITY:

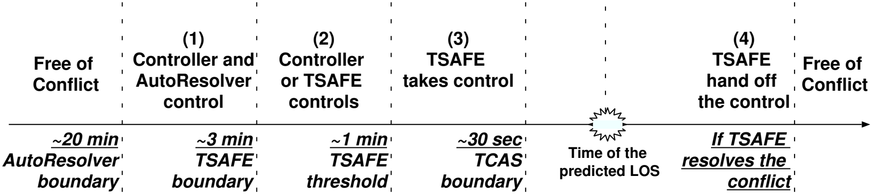
- easy, *expressive* specification language
- *generic* interface to connect to a wide variety of systems
- *adaptable* to missions, mission stages, platforms

REALIZABILITY:

- easy, *expressive* specification language
- *generic* interface to connect to a wide variety of systems
- *adaptable* to missions, mission stages, platforms

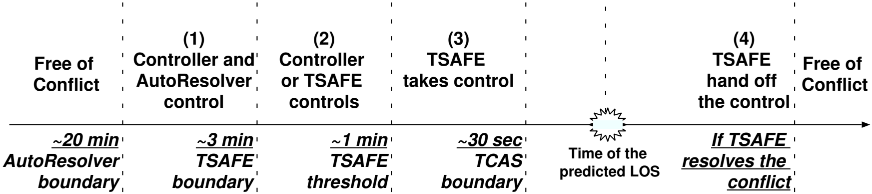
. . . So how do we do that?

AAC Operational Concept[2]



[2] H Erzberger, K Heere. "Algorithm and operational concept for resolving short-range conflicts." Proc. IMechE G J. Aerosp. Eng. 224 (2) (2010) 225–243.

AAC Operational Concept[3]



LTL Model Checking triggered system design changes[2]

[2] Y. Zhao and K.Y. Rozier. "Formal Specification and Verification of a Coordination Protocol for an Automated Air Traffic Control System." SCP Journal, vol-96, no-3, pg 337-353, 2014.

[3] H Erzberger, K Heere. "Algorithm and operational concept for resolving short-range conflicts." Proc. IMechE G J. Aerosp. Eng. 224 (2) (2010) 225–243.

Is LTL All We Need?

ATC never turns off . . .

Is LTL All We Need?

ATC never turns off . . .

LTL intuitively describes it

Is LTL All We Need?

ATC never turns off . . .

LTL intuitively describes it

Aircraft have finite missions; ATC has finite modes. . .

Is LTL All We Need?

ATC never turns off . . .

LTL intuitively describes it

Aircraft have finite missions; ATC has finite modes. . .

LTLf?

Is LTL All We Need?

ATC never turns off . . .

LTL intuitively describes it

Aircraft have finite missions; ATC has finite modes. . .

LTLf?

But there are numerical bounds on the timelines. . .

Is LTL All We Need?

ATC never turns off . . .

LTL intuitively describes it

Aircraft have finite missions; ATC has finite modes. . .

LTLf?

But there are numerical bounds on the timelines. . .

MTL? STL?

MTL: Many Variations[4]

- **Semantics:** continuous vs pointwise
- **Traces:** finite vs infinite
- **Intervals:**
  - infinite vs finite vs bounded (specific bounds)
  - open, closed, half-open
  - punctual (singleton allowed) or not
  - start with 0 or end with ∞: MTL_0; $MTL_{0,\infty}$
- **Interval types:** integer vs real numbers

[4] Ouaknine & Worrell. "Some Recent Results in Metric Temporal Logic." FORMATS 2008.

STL: Made for Describing CPS[6]

STL adds an **analog layer** to MTL,
reasons over **real-valued predicates** with **real-time intervals**[5]

STL Semantics: the satisfaction of an STL formula φ by a signal $\mathrm{x} = (x_1, \ldots, x_n)$ at time t is

$$(\mathrm{x}, t) \models \mu \Leftrightarrow f(x_1[t], \ldots, x_n[t]) > 0$$

$$(\mathrm{x}, t) \models \varphi \wedge \psi \Leftrightarrow (\mathrm{x}, t) \models \varphi \wedge (\mathrm{x}, t) \models \psi$$

$$(\mathrm{x}, t) \models \neg\varphi \Leftrightarrow \neg((x, t) \models \varphi)$$

$$(\mathrm{x}, t) \models \varphi \mathcal{U}_{[a,b]} \psi \Leftrightarrow \exists t' \in [t + a,\ t + b] \text{ such that } (\mathrm{x},\ t') \models \psi \wedge \forall t'' \in [t,\ t'], (\mathrm{x},\ t'') \models \varphi$$

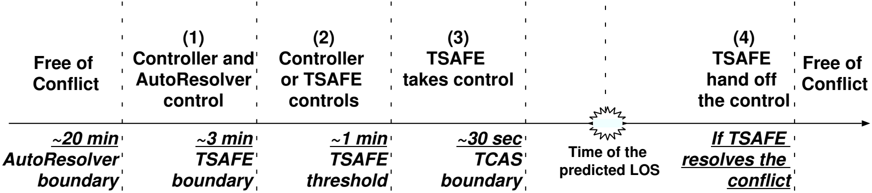
[5] Donzè. "On Signal Temporal Logic." RV, 2013.

[6] Maler & Nickovic. "Monitoring Temporal Properties of Continuous Signals." FORMATS 2004.

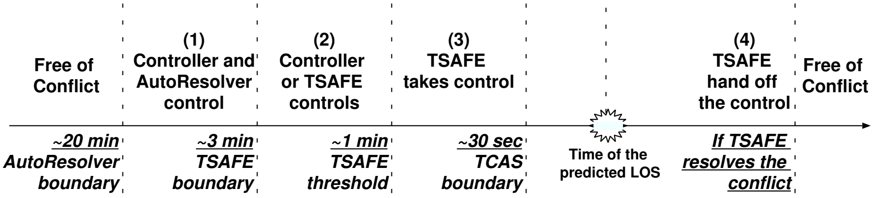
STL: With Great Power Comes. . .

- **Complexity**: MTL satisfiability and model checking are **undecidable**; STL?
- **Confusion**: LTL is hard to write correctly, validate; STL?
- **Precision**: STL needs details not present in the system

AAC Operational Concept



AAC Operational Concept



Times are discrete, rough estimates. . .

STL: Not a Good Fit

- Humans have to write the specifications; **writing formal properties is hard.**[7]
- Humans have to **validate** the specifications; need to check satisfiability efficiently
- Certification requires **explainability**
- Many domains (ISS) require **adaptability**

[7] K. Y. Rozier. "Specification: The Biggest Bottleneck in Formal Methods and Autonomy." VSTTE Keynote, 2016.

STL continued ... [8]

- Requires or assumes details not present in the system
- Continuous-time reasoning might not match up with discrete systems
- “Preciseness” $\rightarrow \uparrow$ complexity; won’t fit in tight spaces
- Formulas are too specific; not re-usable
- Hard to validate (hard for humans to understand)
- Specifications not robust to realistic system changes

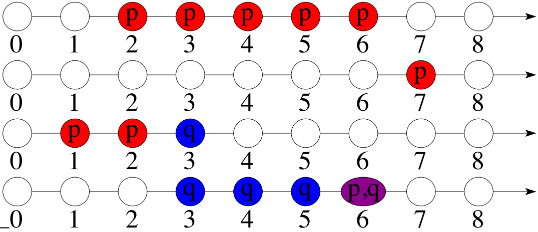
[8] K.Y.Rozier, “On the Effectiveness of Mission-time Linear Temporal Logic (MLTL) in AI Applications.” AAAI Symposium “On the Effectiveness of Temporal Logics on Finite Traces in AI,” San Francisco, California, USA, March 27–29, 2023. https://aaai.org/Symposia/Spring/sss23.php

MLTL: A Good Specification Language[9]

Mission-Time Temporal Logic (MLTL) reasons about *integer-bounded* timelines:

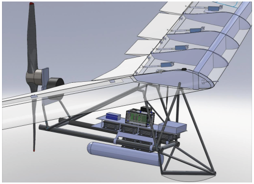
- finite set of atomic propositions {p q}
- Boolean connectives: $\neg$, $\wedge$, $\vee$, and $\rightarrow$
- temporal connectives *with time bounds*:

Symbol	Operator	Timeline
$\Box_{[2,6]} p$	ALWAYS$_{[2,6]}$	
$\Diamond_{[0,7]} p$	EVENTUALLY$_{[0,7]}$	
$p \mathcal{U}_{[1,5]} q$	UNTIL$_{[1,5]}$	
$p \mathcal{R}_{[3,8]} q$	RELEASE$_{[3,8]}$	



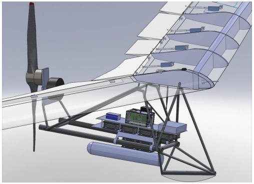
9 T. Reinbacher, K.Y. Rozier, J. Schumann. "Temporal-Logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems." TACAS 2014.

Runtime Monitoring for the Swift UAS



After receiving a command (cmd) for takeoff, the Swift UAS must reach an altitude of 600ft within 40 seconds.

Runtime Monitoring for the Swift UAS



After receiving a command (cmd) for takeoff, the Swift UAS must reach an altitude of 600ft within 40 seconds.

$$(\text{cmd} == \text{takeoff}) \rightarrow \Diamond_{[0,40]}(alt \geq 600 \text{ ft})$$

Runtime Monitoring for the Swift UAS

All messages sent from the guidance, navigation and control (GN&C) component to the Swift actuators must be logged into the on-board file system (FS). Logging has to occur before the message is removed from the queue. In contrast to the requirements stated above, this flight rule specifically concerns properties of the flight software.

$$\square((\text{addToQueue}_{GN\&C} \wedge \Diamond \text{removeFromQueue}_{Swift}) \rightarrow \neg\text{removeFromQueue}_{Swift} \; \mathcal{U} \; \text{writeToFS})$$

Hint: need domain constraints. . .

Runtime Monitoring for the Swift UAS

All messages sent from the guidance, navigation and control (GN&C) component to the Swift actuators must be logged into the on-board file system (FS). Logging has to occur before the message is removed from the queue. In contrast to the requirements stated above, this flight rule specifically concerns properties of the flight software.

$$\Box((\text{addToQueue}_{\text{GN\&C}} \wedge \Diamond\text{removeFromQueue}_{\text{Swift}}) \rightarrow \neg\text{removeFromQueue}_{\text{Swift}}\ \mathcal{U}\ \text{writeToFS})$$

Hint: need domain constraints...

MLTL: Not MTL-over-naturals[10]

Some important differences:

- Finite traces
- Finite intervals
- **U-semantics**: $\pi \models \varphi\,\mathcal{U}_{[a,b]}\psi$ iff $|\pi| > a$ and, $\exists i \in [a,\ b], i < |\pi|$ such that $\pi, i \models \psi$ and $\forall j \in [a,\ b], j < i$ it holds that $\pi, j \models \varphi$
- Intervals are closed, unit-less (generic)
- Signal processing compartmentalized

[10] Li, Vardi, Rozier. "Satisfiability checking for mission-time LTL." CAV, 2019.

MLTL is Unusually Effective![11]

- Easier to **accurately represent timing constraints** of real systems (e.g., ATC bounds)
- Easier to **validate** (matches real system better)
- Low **complexity/memory** to check
- **Generic, reusable** specifications are robust to hardware substitutions/clock changes
  - Can **tune timescales** for resource trade offs on embedded systems
- **Separation of concerns** (Boolean testers, temporal logic, intervals)
  - Allows **re-use of processed signals** outside of logic
  - Retain **validation/complexity/size benefits** while separating out extensions
- **Fits into business processes for real CPS development**

[11] K.Y.Rozier, "On the Effectiveness of Mission-time Linear Temporal Logic (MLTL) in AI Applications." AAAI Symposium "On the Effectiveness of Temporal Logics on Finite Traces in AI," San Francisco, California, USA, March 27–29, 2023. https://aaai.org/Symposia/Spring/sss23.php

A Recent Motivation. . .
Crash of ESA's ExoMars Schiaparelli Lander

- October 19, 2016

A Recent Motivation. . .
Crash of ESA's ExoMars Schiaparelli Lander

- October 19, 2016
- parachute deployed at:
  - altitude of 7.5 miles (12 km)
  - speed of 1,1075 mph (1,730 km/h)

A Recent Motivation...
Crash of ESA's ExoMars Schiaparelli Lander

- October 19, 2016
- parachute deployed at:
  - altitude of 7.5 miles (12 km)
  - speed of 1,1075 mph (1,730 km/h)
- heat shield ejected at altitude of 4.85 miles (7.8 km)

A Recent Motivation...
Crash of ESA's ExoMars Schiaparelli Lander

- October 19, 2016
- parachute deployed at:
  - altitude of 7.5 miles (12 km)
  - speed of 1,1075 mph (1,730 km/h)
- heat shield ejected at altitude of 4.85 miles (7.8 km)
- IMU miscalculated saturation-maximum period (by 1 sec)

A Recent Motivation...
Crash of ESA's ExoMars Schiaparelli Lander

- October 19, 2016
- parachute deployed at:
  - altitude of 7.5 miles (12 km)
  - speed of 1,1075 mph (1,730 km/h)
- heat shield ejected at altitude of 4.85 miles (7.8 km)
- IMU miscalculated saturation-maximum period (by 1 sec)
- Navigation system calculated a *negative altitude*
  - premature release of parachute & backshell
  - firing of braking thrusters
  - activation of on-ground systems at 2 miles (3.7 km) altitude

A Recent Motivation...
Crash of ESA's ExoMars Schiaparelli Lander

- October 19, 2016
- parachute deployed at:
  - altitude of 7.5 miles (12 km)
  - speed of 1,1075 mph (1,730 km/h)
- heat shield ejected at altitude of 4.85 miles (7.8 km)
- IMU miscalculated saturation-maximum period (by 1 sec)
- Navigation system calculated a *negative altitude*
  - premature release of parachute & backshell
  - firing of braking thrusters
  - activation of on-ground systems at 2 miles (3.7 km) altitude
- Crash at 185 mph (300 km/h)

A Recent Motivation. . .
Crash of ESA's ExoMars Schiaparelli Lander

Sanity Checks
Relevant to this Mission:

- The altitude cannot be negative.
- The rate of change of descent can't be faster than gravity.
- The δ altitude must be within nominal parameters; it cannot change from 2 miles to a negative value in one time step.
- The saturation-maximum has an a priori known temporal bound.

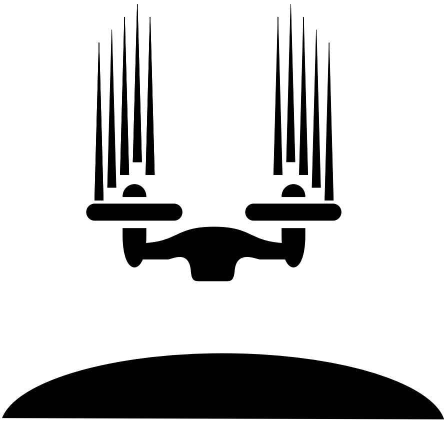
These *sanity checks* could have prevented the crash.

Capability of such observations is *required for autonomy*.

There's a pattern here!

Runtime Functional Specification Patterns[12]

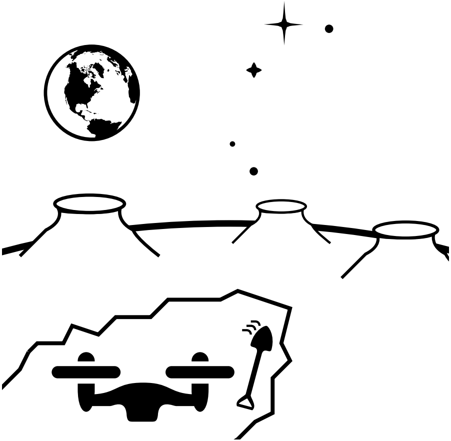
- **Rates**
- **Ranges**
- **Relationships**
- **Control Sequences**
- **Consistency Checks**



[12] K.Y.Rozier. "Specification: The Biggest Bottleneck in Formal Methods and Autonomy." VSTTE, 2016.

Runtime Functional Specification Patterns[12]

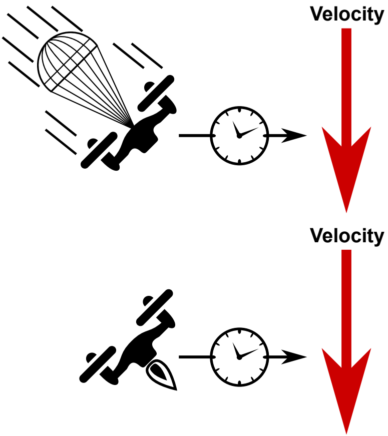
- Rates
- Ranges
- Relationships
- Control Sequences
- Consistency Checks



12 K.Y.Rozier. "Specification: The Biggest Bottleneck in Formal Methods and Autonomy." VSTTE, 2016.

Runtime Functional Specification Patterns[12]

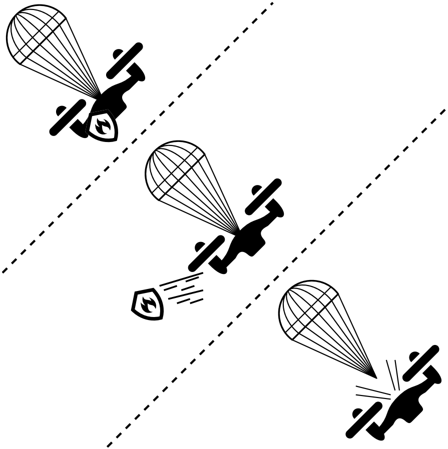
- Rates
- Ranges
- Relationships
- Control Sequences
- Consistency Checks



[12] K.Y.Rozier. “Specification: The Biggest Bottleneck in Formal Methods and Autonomy.” VSTTE, 2016.

Runtime Functional Specification Patterns[12]

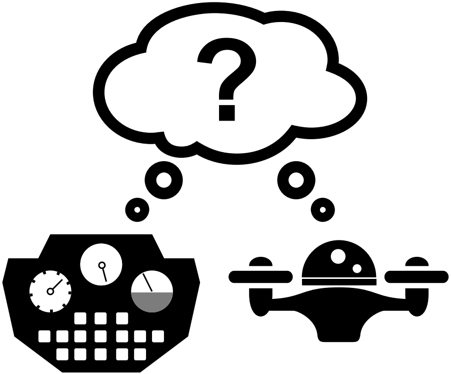
- Rates
- Ranges
- Relationships
- Control Sequences
- Consistency Checks



[12] K.Y.Rozier. "Specification: The Biggest Bottleneck in Formal Methods and Autonomy." VSTTE, 2016.

Runtime Functional Specification Patterns[12]

- Rates
- Ranges
- Relationships
- Control Sequences
- Consistency Checks



[12] K.Y.Rozier. "Specification: The Biggest Bottleneck in Formal Methods and Autonomy." VSTTE, 2016.

RESPONSIVENESS:

- *continuously monitor* the system
- *detect deviations* in *real time*
- *enable mitigation* or rescue measures

Responsiveness:

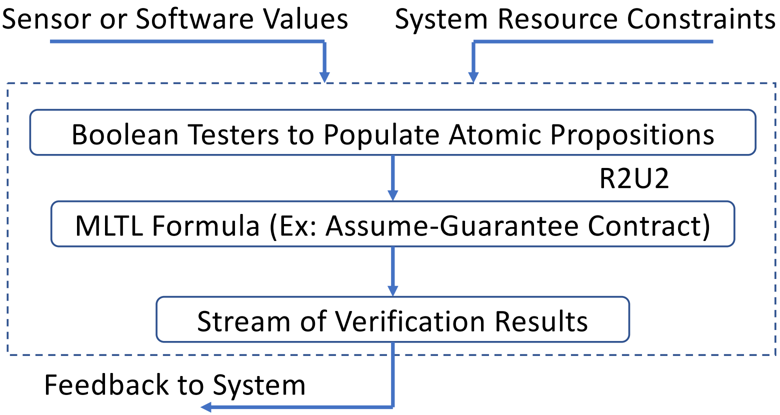
- *continuously monitor* the system
- *detect deviations* in *real time*
- *enable mitigation* or rescue measures

. . . So how do we do that?

Asynchronous Observers (aka event-triggered)

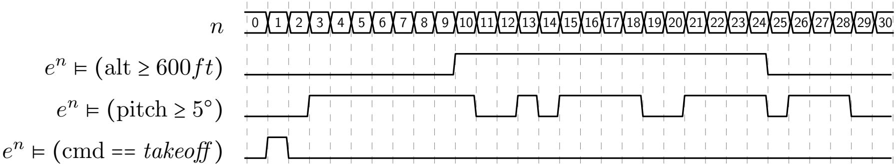
- *evaluate with every new input*
- 2-valued output: {true; false}
- resolve φ *as early as possible* (a priori known time)
- for each clock tick, may resolve φ for clock ticks prior to the current time n if the information required for this resolution was not available until n

R2U2 High-Level Architecture[13]



13 Rozier, Kristin Y., and Johann Schumann. "R2U2: tool overview." (2017)

Asynchronous Observers Example



$\text{ALWAYS}_{[5]}(\text{pitch} \geq 5°)$

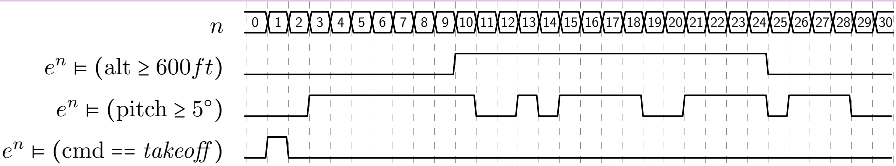
0	(false,0)	8	(true,3)	
1	(false,1)	9	(true,4)	
2	(false,2)	10	(true,5)	
3	(␣, ␣)	11	(false,11)	Resynchronized!
4	(␣, ␣)	12	(false,12)	
5	(␣, ␣)	13	(␣, ␣)	
6	(␣, ␣)	14	(false,14)	Resynchronized!
7	(␣, ␣)	15	(␣, ␣)	

Synchronous Observers (aka time-triggered)

- update continuously
- 3-valued output: {true; false; maybe}
- small hardware footprints
  ($\leq$ 11 two-input gates/operator)

Synchronous observers update at every tick of the system clock
...enabling probabilistic system diagnosis!

Synchronous Observers Example

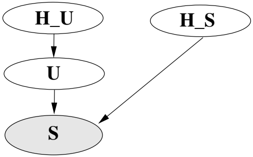


$$\text{ALWAYS}_{[5]}((\text{alt} \geq 600ft) \wedge (\text{pitch} \geq 5°))$$

0	(false,0)	8	(false,8)
1	(false,1)	9	(false,9)
2	(false,2)	10	(maybe,10)
3	(false,3)	11	(false,11)
4	(false,4)	12	(false,12)
5	(false,5)	13	(maybe,13)
6	(false,6)	14	(false,14)
7	(false,7)	15	(maybe,15)

Bayesian Reasoning

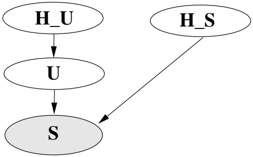
- Our Bayesian Networks (BNs) contain
  - (observable) sensor nodes S
  - (unobservable) status nodes U
  - health nodes H_S, H_U



- Discrete sensor values & outputs of LTL/MTL formulas $\rightarrow$ S nodes
- *Posteriors* of the health nodes H_U, H_S reflect the *most likely* health status of the component

Bayesian Reasoning

- Our Bayesian Networks (BNs) contain
  - (observable) sensor nodes S
  - (unobservable) status nodes U
  - health nodes H_S, H_U



- Discrete sensor values & outputs of LTL/MTL formulas $\rightarrow$ S nodes

- *Posteriors* of the health nodes H_U, H_S reflect the *most likely* health status of the component

In our framework we do not use Dynamic BNs as temporal aspects are handled by the temporal observers.

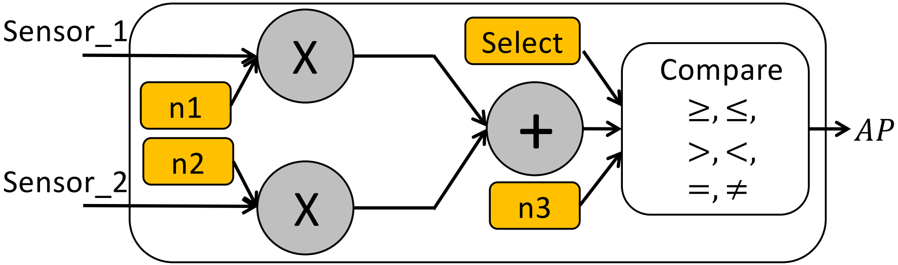
R2U2: Realizable, Responsive, Unobtrusive[14]

R2U2 specification format:

1. **Signal Processing**: Preparation of sensor readings
   - **Filtering**: processing of incoming data
   - **Discretization**: generation of Boolean outputs
2. **Temporal Logic (TL) Observers**: Efficient temporal reasoning
   1. **Asynchronous**: output $\langle t, \{0,1\}\rangle$
   2. **Synchronous**: output $\langle t, \{0,1,?\}\rangle$
   - **Logics**: MTL, pt-MTL, Mission-time LTL
   - **Variables**: Booleans (from system bus), sensor filter outputs
3. **Bayes Nets**: Efficient decision making
   - **Variables**: outputs of TL observers, sensor filters, Booleans
   - **Output**: most-likely status + probability

[14] Kristin Yvonne Rozier, and Johann Schumann. "R2U2: Tool Overview." In International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools (RV-CUBES), held in conjunction with the 17th International Conference on Runtime Verification (RV), Kalpa Publications, Seattle, Washington, USA, September 13-16, 2017.

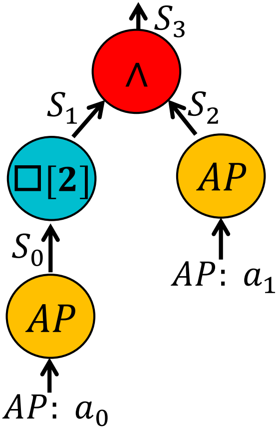
Signal Processing[15]



Orange blocks are configurable online. Example: $n1$ is the scale factor, and $n2$ is the configurable offset. The final AP output is the Boolean result of comparison with the $n3$ reference value.

15 B.Kempa, P.Zhang, P.H.Jones, J.Zambreno, K.Y.Rozier. "Embedding Online Runtime Verification for Fault Disambiguation on Robonaut2." FORMATS, LNCS vol 12288, 2020.

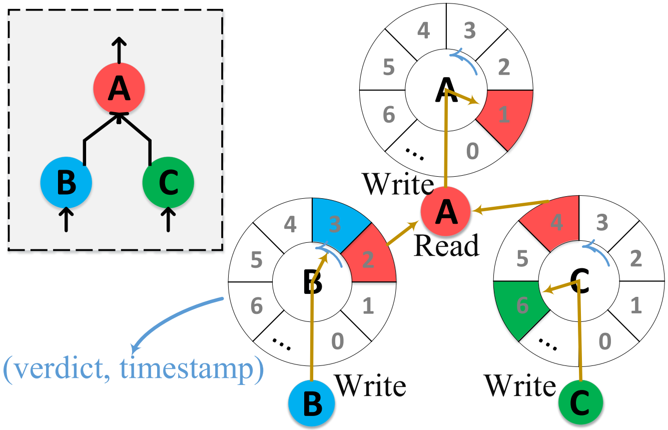
Temporal Logic (TL) Observers: Not Automata; ASTs[16]



$$(\Box_{[0,2]} a_0) \wedge (a_1)$$

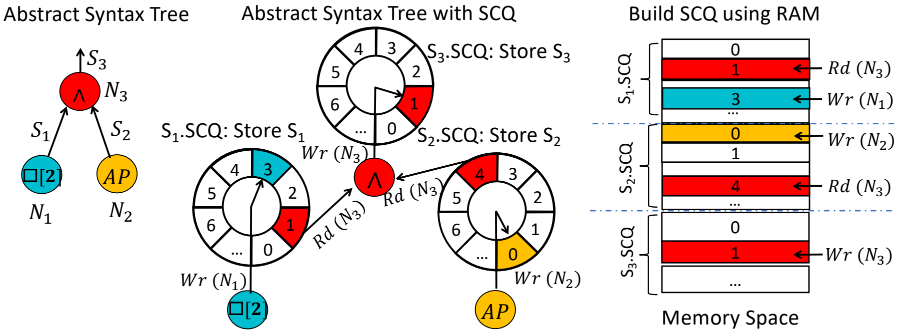
16 B.Kempa, P.Zhang, P.H.Jones, J.Zambreno, K.Y.Rozier. "Embedding Online Runtime Verification for Fault Disambiguation on Robonaut2." FORMATS, LNCS vol 12288, 2020.

Temporal Logic (TL) Observers: Not Automata; ASTs with SCQs[17]



17 B.Kempa, P.Zhang, P.H.Jones, J.Zambreno, K.Y.Rozier. "Embedding Online Runtime Verification for Fault Disambiguation on Robonaut2." FORMATS, LNCS vol 12288, 2020.

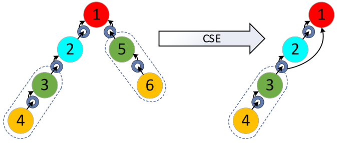
Abstract Syntax Trees with Shared Connection Queues[18]



18 B.Kempa, P.Zhang, P.H.Jones, J.Zambreno, K.Y.Rozier. "Embedding Online Runtime Verification for Fault Disambiguation on Robonaut2." FORMATS, LNCS vol 12288, 2020.

Common Subexpression Elimination (CSE)[19]

$load\ \mathcal{AP}(a0)$
$f_{\square[0,3]}(S0)$
$f_{\neg}(S1)$
$load\ \mathcal{AP}(a0)$
$f_{\square[0,3]}(S3)$
$f_{\mathcal{U}[0,5]}(S2, S4)$



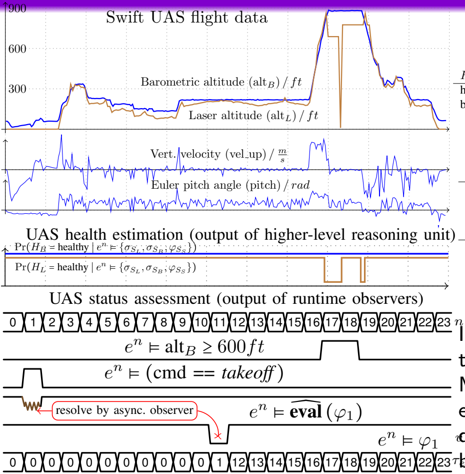
$$\Big(\overbrace{\neg}^{2}\ \overbrace{\square_{[0,3]}}^{3}\ \overbrace{a_0}^{4}\Big)\ \overbrace{\mathcal{U}_{[0,5]}}^{1}\ \Big(\overbrace{\square_{[0,3]}}^{5}\ \overbrace{a_0}^{6}\Big)$$

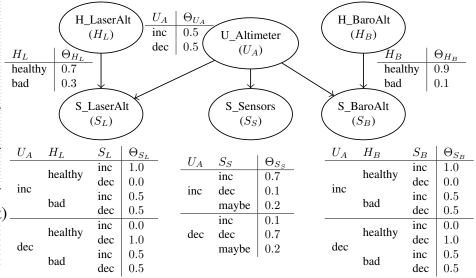
$load\ \mathcal{AP}(a0)$
$f_{\square[0,3]}(S0)$
$f_{\neg}(S1)$
$f_{\mathcal{U}[0,5]}(S2, S1)$

CSE on an MLTL formula where nodes 3 and 5 have identical children.
LHS: AST and resulting R2U2 instruction representing the formula.
RHS: AST and instructions produced by applying CSE.
Sharing the output of node 3 removes one repetition of this sequence, saving two queues and two instructions.

19 B.Kempa, P.Zhang, P.H.Jones, J.Zambreno, K.Y.Rozier. "Embedding Online Runtime Verification for Fault Disambiguation on Robonaut2." FORMATS, LNCS vol 12288, 2020.

R2U2 Finds Laser Altimeter Failure





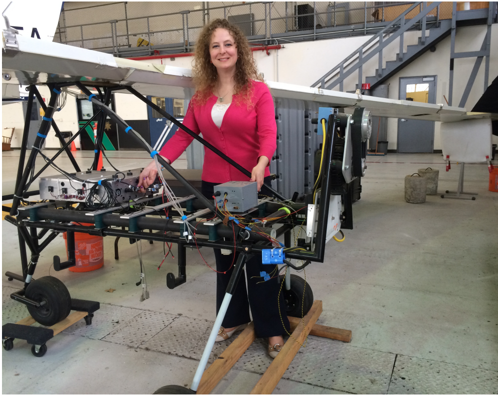
Inputs to R2U2 are flight data, sampled in real time; a health model as BN, right; and an MLTL specification φ. Outputs: health estimation (posterior marginals of H_L and H_B, quantifying the health of the laser and barometric altimeter) and the UAS status.

After receiving a command (cmd) for takeoff, the Swift UAS must reach an altitude of 600ft within 10 seconds

Satisfying Requirements

Unobtrusiveness:

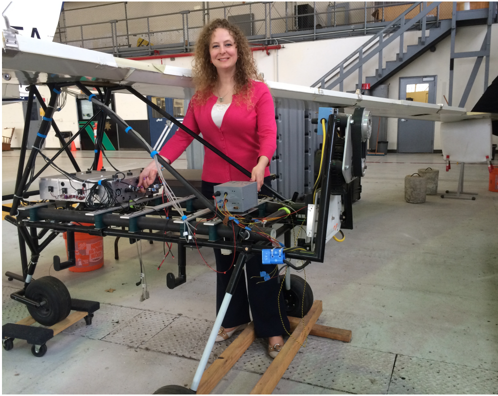
- *functionality*: not change behavior
- *certifiability*: avoid re-certification of flight software/hardware
- *timing*: not interfere with timing guarantees
- *tolerances*: obey size, weight, power, telemetry bandwidth constraints
- *cost*: use commercial-off-the-shelf (COTS) components



Satisfying Requirements

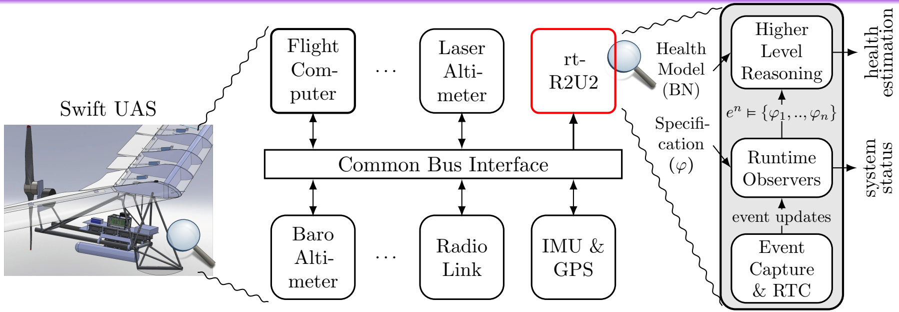
UNOBTRUSIVENESS:

- *functionality*: not change behavior
- *certifiability*: avoid re-certification of flight software/hardware
- *timing*: not interfere with timing guarantees
- *tolerances*: obey size, weight, power, telemetry bandwidth constraints
- *cost*: use commercial-off-the-shelf (COTS) components



. . . So how do we do that?

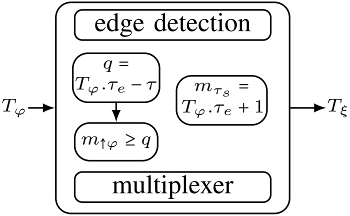
R2U2: Realizable, Responsive, Unobtrusive[20]

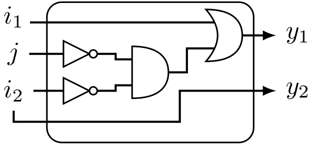


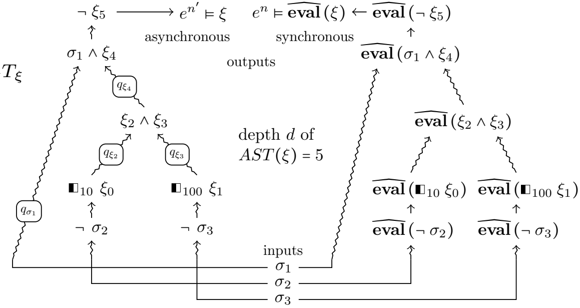
- synthesis and integration of new paired synchronous/asynchronous observer framework
- enables discrete Bayesian Network-based reasoning
- modular hardware implementation: FPGA

[20] Johann Schumann, Kristin Y. Rozier, Thomas Reinbacher, Ole J. Mengshoel, Timmy Mbaya, and Corey Ippolito. "Towards Real-time, On-board, Hardware-supported Sensor and Software Health Management for Unmanned Aerial Systems." In *International Journal of Prognostics and Health Management (IJPHM)*, volume 6, number 1, pages 1–27, PHM Society, June, 2015.

FPGA Implementation of Temporal Observers[21]





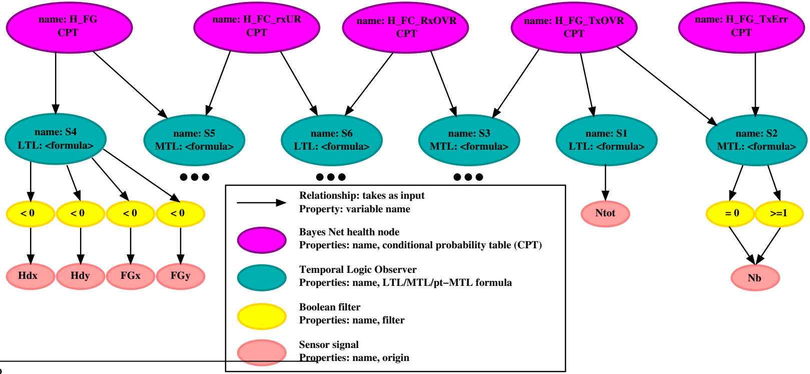


- asynchronous observers: substantial hardware complexity
- synchronous observers: small HW footprint

[21] Thomas Reinbacher, Kristin Y. Rozier, and Johann Schumann. "Temporal-Logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems." In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 8413 of Lecture Notes in Computer Science (LNCS), pages 357–372, Springer-Verlag, April 2014.

R2U2 Observation Tree (Specification)[22]

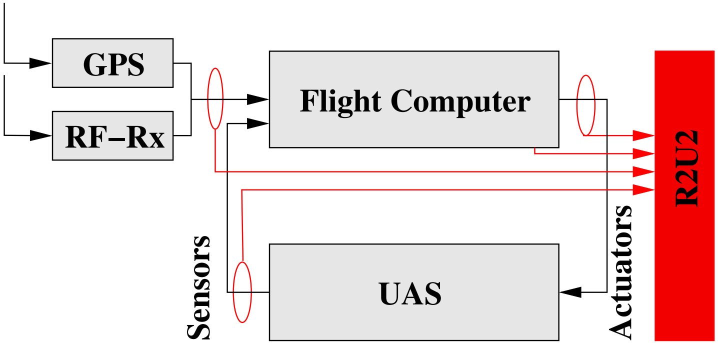
Health Nodes / Failure Modes	
H_FG	**Magnetometer sensor**
H_FC_RxUR	Receiver underrun
H_FC_RxOVR	Receiver overrun
H_FG_TxOVR	**Transmitter overrun in sensor**
H_FG_TxErr	Transmitter error in in sensor



[22] Rozier & Schumann. "R2U2: Tool Overview." In *International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools (RV-CUBES)*, 2017.

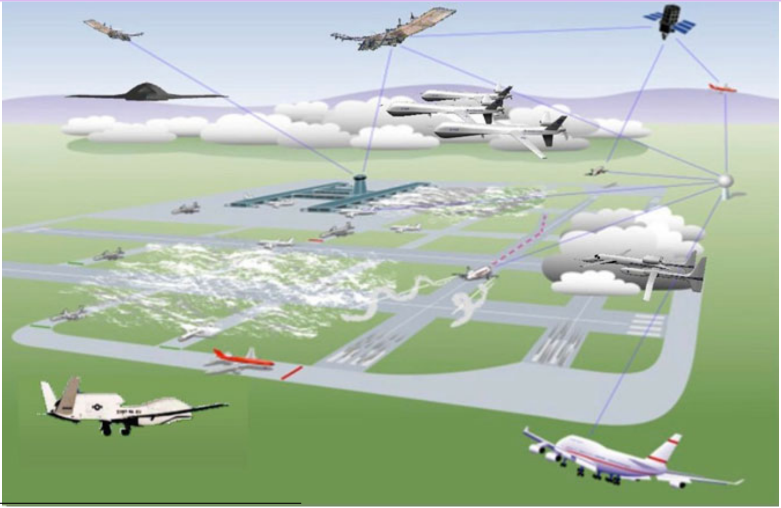
Monitoring and Diagnosis of Security Threats[23]

Threat detection: *attack monitoring*, *post-attack system behavior monitoring*, and *diagnosis*.



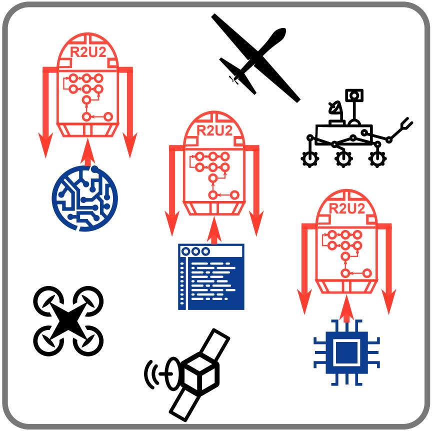
[23] Johann Schumann, Patrick Moosbrugger, Kristin Y. Rozier. "R2U2: Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems." In *Runtime Verification (RV15)*, Springer-Verlag, September, 2015.

Adding UAS into the NAS: A UTM First Step[24]

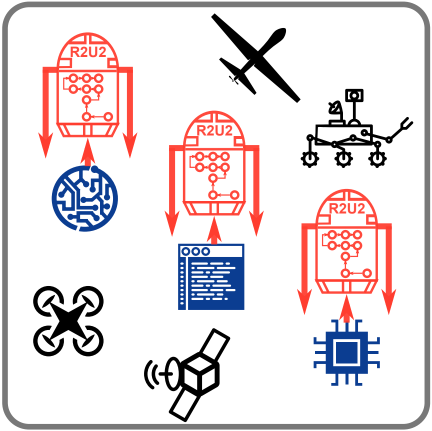


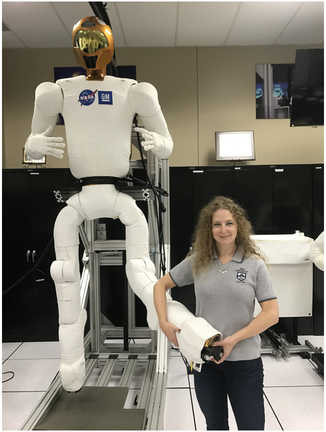
[24] Hammer, Cauwels, Hertz, Jones, Rozier. "Integrating runtime verification into an automated UAS traffic management system." *Innovations in Systems and Software Engineering*, 2021

Multi-Platform, Multi-Architecture Runtime Verification of Autonomous Space Systems



Multi-Platform, Multi-Architecture Runtime Verification of Autonomous Space Systems

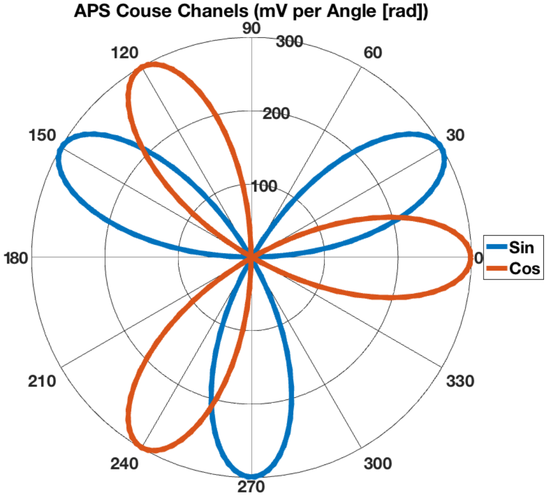




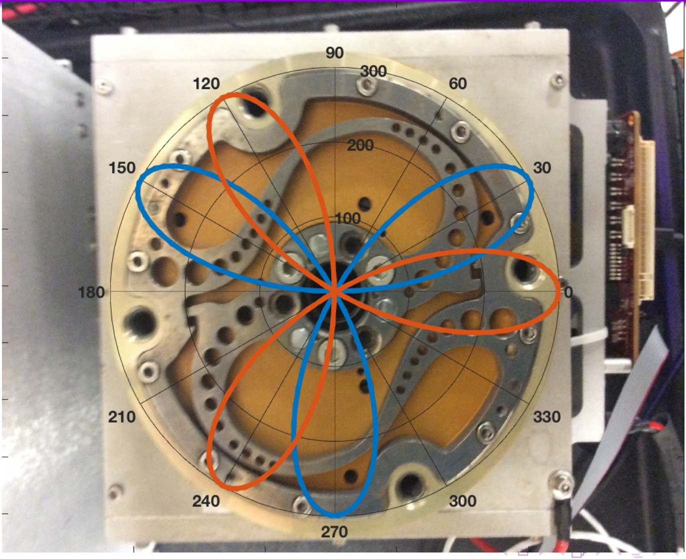
https://temporallogic.org/research/R2U2/FORMATS_18_teaser_BrianKempa.mp4

Robonaut2

Robonaut2's Knee



Robonaut2's Knee



http://temporallogic.org/research/R2U2/R2U2-on-R2_demo.mp4

Lifting Runtime Monitoring

Runtime Monitoring

[25] Falcone, Yliès, Srdan Krstić, Giles Reger, and Dmitriy Traytel. "A taxonomy for classifying runtime verification tools." *Runtime Verification*, 2018.

Lifting Runtime Monitoring

Temporal Fault Disambiguation

↑

Runtime Monitoring

[25] Falcone, Yliès, Srdan Krstić, Giles Reger, and Dmitriy Traytel. "A taxonomy for classifying runtime verification tools." *Runtime Verification*, 2018.

Lifting Runtime Monitoring

Temporal Fault Disambiguation

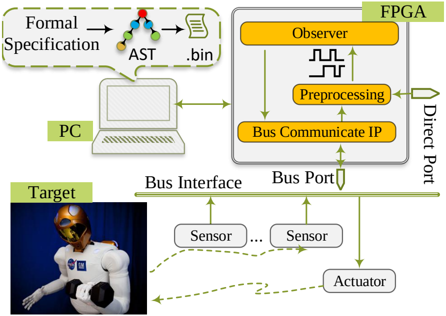
↑

Runtime Monitoring

"R2U2 breaks our taxonomy; it is entirely application driven."
— Giles Reger, 11/13/2018[25]

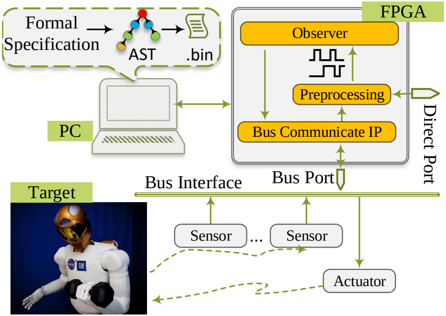
[25] Falcone, Yliès, Srdan Krstić, Giles Reger, and Dmitriy Traytel. "A taxonomy for classifying runtime verification tools." *Runtime Verification*, 2018.

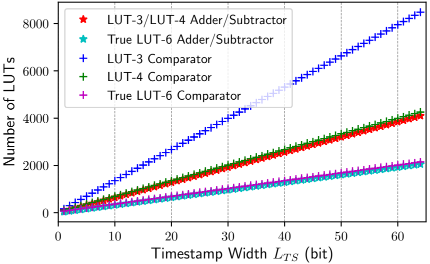
Resource Estimation and Improved Encoding Algorithms [26]



26 B.Kempa, P.Zhang, P.H.Jones, J.Zambreno, K.Y.Rozier. "Embedding Online Runtime Verification for Fault Disambiguation on Robonaut2." FORMATS, LNCS vol 12288, 2020.

Resource Estimation and Improved Encoding Algorithms [26]



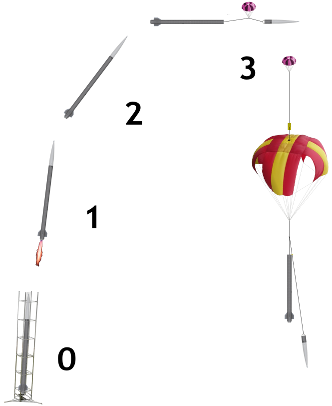


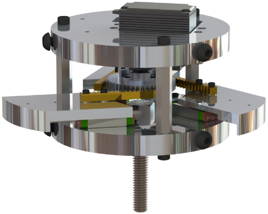
[26] B.Kempa, P.Zhang, P.H.Jones, J.Zambreno, K.Y.Rozier. "Embedding Online Runtime Verification for Fault Disambiguation on Robonaut2." FORMATS, LNCS vol 12288, 2020.

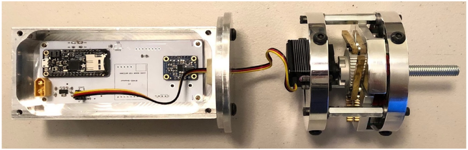
Cyclone Sounding Rocket!

https://www.youtube.com/watch?v=p6dwT0sTdH0&t=158s

Cyclone Rocketry's Sounding Rocket[27]



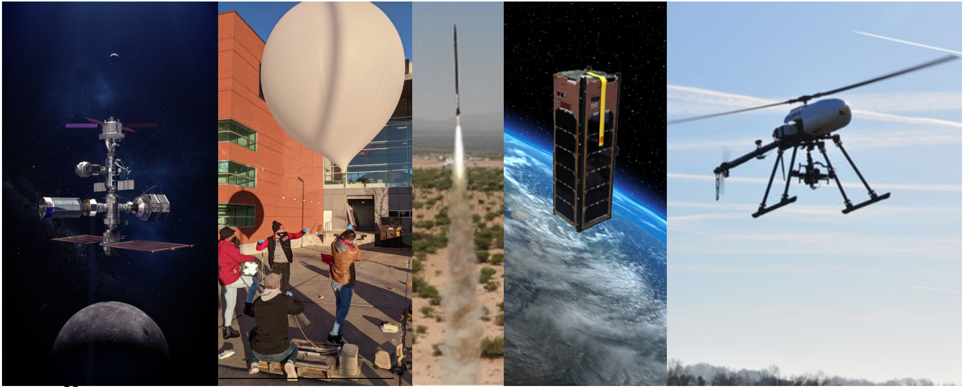




Left: Rocket mission states: *Launch Pad* (0), *Boost* (1), *Coast* (2), *Descent* (3). Right Top: Model of *Nova Somnium*'s ACS, Right Bottom: the physical ACS.

27 B. Hertz, Z. Luppen, K.Y. Rozier. "Integrating Runtime Verification into a Sounding Rocket Control System." *NASA Formal Methods Symposium (NFM)*, 2021.

Flight-Certification == Proofs that Fly! [28] [29] [30] [31]



[28] Hariharan, Kempa, Wongpiromsarn, Jones, Rozier. “MLTL Multi-type (MLTLM): A Logic for Reasoning about Signals of Different Types.” NSV 2022.

[29] Luppen, Jacks, Baughman, Hertz, Cutler, Lee, Rozier. “Elucidation and Analysis of Specification Patterns in Aerospace System Telemetry.” NFM 2022.

[30] Hertz, Luppen, Rozier. “Integrating Runtime Verification into a Sounding Rocket Control System.” NFM 2021.

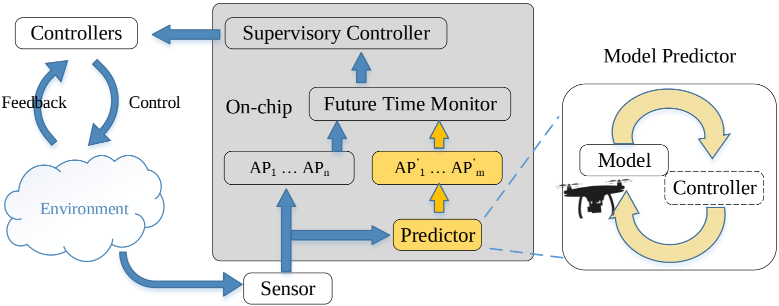
[31] Hammer, Cauwels, Hertz, Jones, Rozier. “Integrating Runtime Verification into an Automated UAS Traffic Management System.” *Innovations in Systems and Software Engineering: A NASA Journal* 2021.

Model Predictive Runtime Verification[32]

How do we make decisions with incomplete information?

[32] Pei Zhang, Alexis Aurandt, Rohit Dureja, Phillip Jones, and Kristin Y. Rozier. "Model Predictive Runtime Verification for Cyber-Physical Systems with Real-Time Deadlines." FORMATS, 2023.

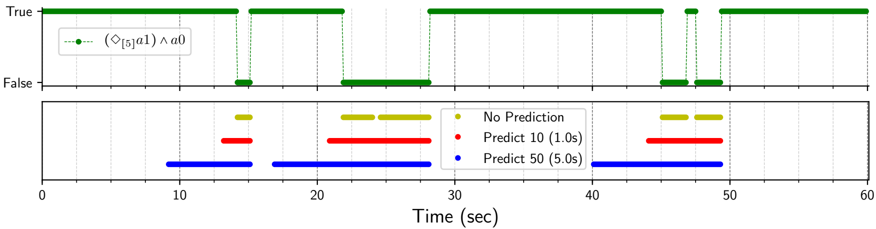
Model Predictive Runtime Verification[33]

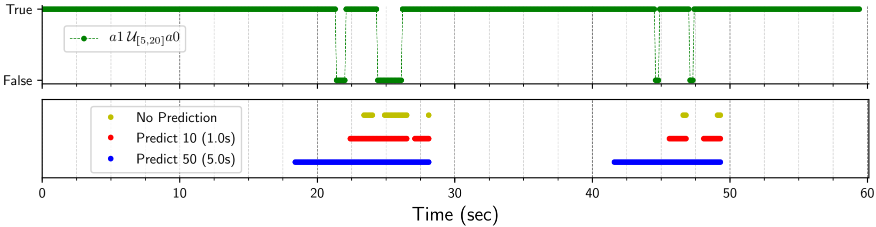


MPRV processing flow

[33] Pei Zhang, Alexis Aurandt, Rohit Dureja, Phillip Jones, and Kristin Y. Rozier. "Model Predictive Runtime Verification for Cyber-Physical Systems with Real-Time Deadlines." FORMATS, 2023.

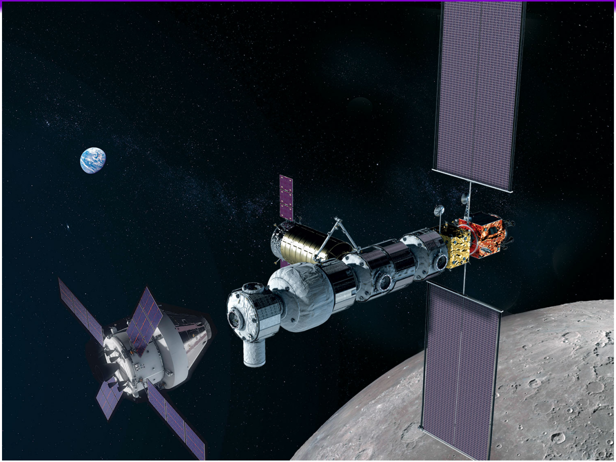
Model Predictive Runtime Verification





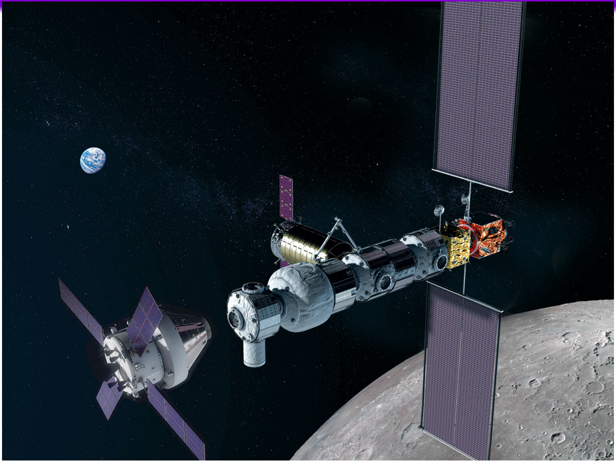
MPRV responsiveness for different prediction horizons: 0, 10 steps (1s), 50 steps (5s).

NASA Lunar Gateway: Assume-Guarantee Contracts[34]



[34] Dabney, James B., Julia M. Badger, and Pavan Rajagopal. "Adding a Verification View for an Autonomous Real-Time System Architecture." In AIAA Scitech 2021 Forum, p. 0566. 2021.

NASA Lunar Gateway: Assume-Guarantee Contracts[34]



$$(CMD == START) \rightarrow (\Diamond_{[0,5]}(ActionHappens\& \Diamond_{[0,2]}(CMD = END)))$$

[34] Dabney, James B., Julia M. Badger, and Pavan Rajagopal. "Adding a Verification View for an Autonomous Real-Time System Architecture." In AIAA Scitech 2021 Forum, p. 0566. 2021.

NASA Lunar Gateway V&V Using MLTL

Adding a Verification View for an Autonomous Real-Time System Architecture
James B. Dabney, Julia M. Badger, Pavan Rajagopal
AIAA 2021-0566; SE-05:Systems Engineering V, 12 January 2021
https://doi.org/10.2514/6.2021-0566
Video: https://doi.org/10.2514/6.2021-0566.vid

FSW 2021: Using Assume-Guarantee Contracts In Autonomous Spacecraft - James Bruster Dabney
https://www.youtube.com/watch?v=zrtyiyNf674

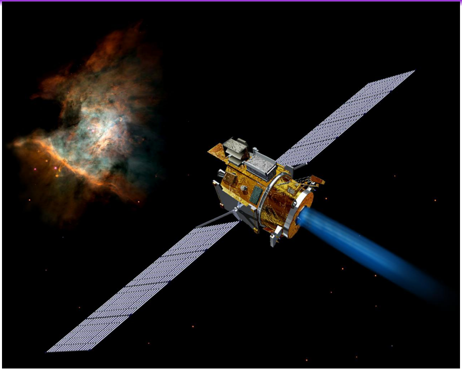
FSW 2022: Using Assume-Guarantee Contracts for Developmental Verification of Autonomous Spacecraft
https://www.youtube.com/watch?v=HFnn6TzblPg

Generating Contracts: From Textual Requirements to MLTL[35]

1. When Executive receives a task command, Executive shall respond with accept/reject within 5 seconds
2. Executive shall provide task updates at 0.1 Hz
3. After accepting command, Executive shall respond with completion message within 10 seconds

[35] FSW 2021: Using Assume-Guarantee Contracts In Autonomous Spacecraft - James Bruster Dabney

MLTL Multi-type (MLTLM): A Logic for Reasoning About Signals of Different Types[37]



The spacecraft maintenance cycle runs at least once a month over the five-year mission.

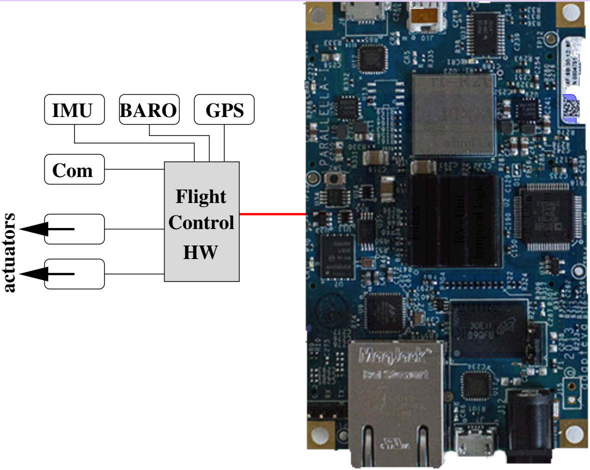
Monthly course corrections never involve burning the thrusters more than 3 seconds at a time.

$$\Box_{[0,5,year]}[(\Diamond_{[0,30,day]} maintenance) \wedge (\neg\Box_{[0,3,sec]} thrusters)]^{36}$$

[36] Formula simplified for illustration.

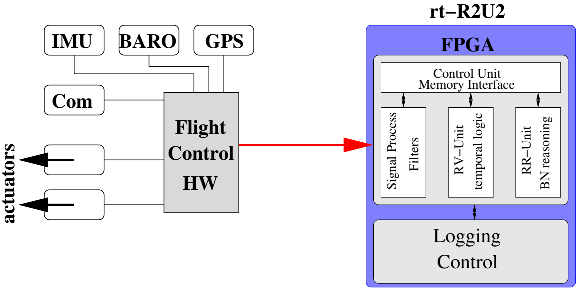
[37] Hariharan, Kempa, Wongpiromsarn, Jones, Rozier, NSV 2022

Hard- and Software Architecture: Resource Estimation



- How do we fit in the resources left over?
- Choose between 3 R2U2 implementations:
  - Hardware: FPGA
  - Software: C emulation of FPGA
  - Software: Object-oriented C++

Hard- and Software Architecture: Resource Estimation



- How do we fit in the resources left over?
- Choose between 3 R2U2 implementations:
  - Hardware: FPGA
  - Software: C emulation of FPGA
  - Software: Object-oriented C++

Need a Configuration Compiler for Property Organization (C2PO)

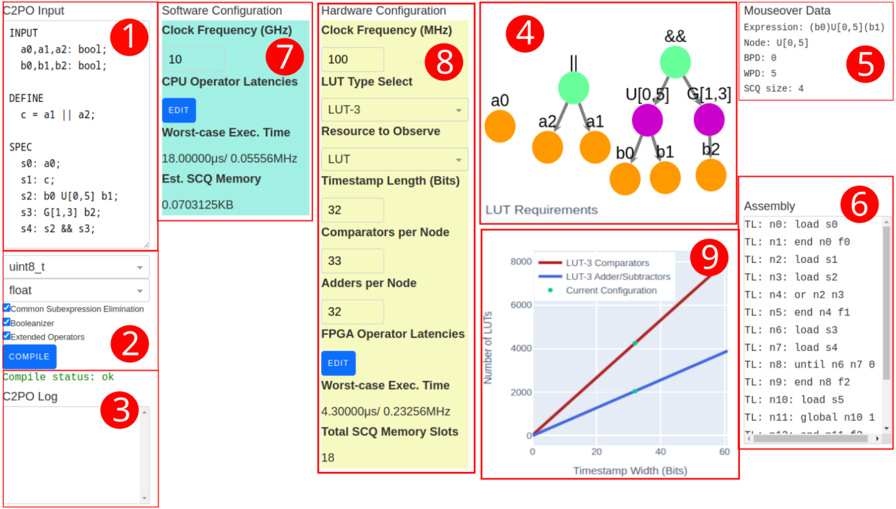
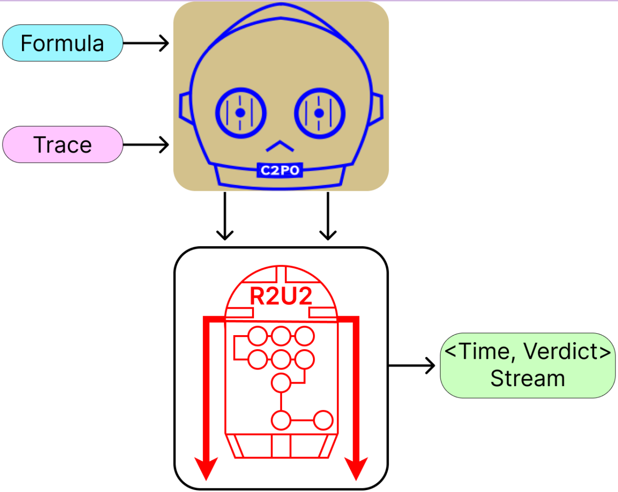


Figure: R2U2 Configuration Explorer web application: 1) C2PO specification input; 2) C2PO options; 3) C2PO output; 4) AST visualization; 5) AST node data; 6) R2U2 instruction; 7) C engine speed and memory calculator; 8) FPGA speed and size calculator; 9) FPGA design size vs maximum timestamp value.

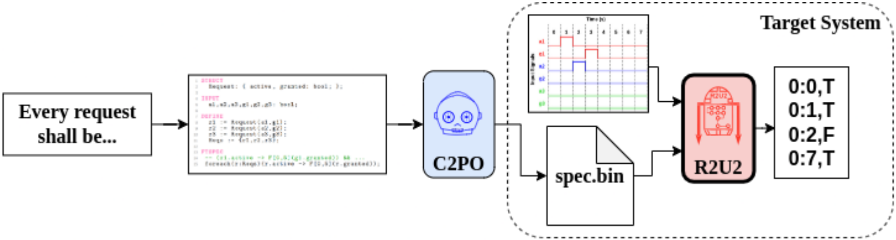


39 40

[39] C. Johannsen, P. H.Jones, B. Kempa, K. Y. Rozier, P. Zhang. "R2U2 Version 3.0: Re-imagining a Toolchain for Specification, Resource Estimation, and Optimized Observer Generation for Runtime Verification in Hardware and Software." CAV, 2023.

[40] C. Johannsen, B. Kempa, P. H. Jones, K. Y. Rozier, T. Wongpiromsarn. "Impossible Made Possible: Encoding Intractable Specifications via Implied Domain Constraints." FMICS, 2023.

R2U2 "Offline" Verification[41]



[41] C. Johannsen, P.H.Jones, B.Kempa, K.Y.Rozier, P.Zhang. "R2U2 Version 3.0: Re-imagining a Toolchain for Specification, Resource Estimation, and Optimized Observer Generation for Runtime Verification in Hardware and Software." CAV 2023.

R2U2 Example: Arbiter

English Requirement

Every request shall be granted within 5 seconds; only 3 requests can be queued at once.

STRUCT

```
Request: { active, granted: bool; };
```

INPUT

```
a1,a2,a3,g1,g2,g3: bool;
```

DEFINE

```
r1  := Request(a1,g1);
r2  := Request(a2,g2);
r3  := Request(a3,g3);
Reqs := {r1,r2,r3};
```

FTSPEC

```
-- (r1.active -> F[0,5](g1.granted)) && ...
foreach(r:Reqs)(r.active -> F[0,5](r.granted));
```

Domain-Driven Adaptations

“Every file that gets opened eventually gets closed.”

[42] C. Johannsen, B. Kempa, P. H. Jones, K. Y. Rozier, T. Wongpiromsarn. “Impossible Made Possible: Encoding Intractable Specifications via Implied Domain Constraints.” FMICS, 2023.

Domain-Driven Adaptations

"Every file that gets opened eventually gets closed."

"At most X of these hold at the same time."

[42] C. Johannsen, B. Kempa, P. H. Jones, K. Y. Rozier, T. Wongpiromsarn. "Impossible Made Possible: Encoding Intractable Specifications via Implied Domain Constraints." FMICS, 2023.

Domain-Driven Adaptations

"Every file that gets opened eventually gets closed."

"At most X of these hold at the same time."

Need a Configuration Compiler for Property Organization (C2PO) [42]

[42] C. Johannsen, B. Kempa, P. H. Jones, K. Y. Rozier, T. Wongpiromsarn. "Impossible Made Possible: Encoding Intractable Specifications via Implied Domain Constraints." FMICS, 2023.

Impossible Temporal Logic Specifications [43]

Lesson: Use Implied Domain Constraints

- unboundedness
- self-reference
- explicit counting

[43] C.Johannsen, B.Kempa, P.H.Jones, K.Y.Rozier, T.Wongpiromsarn. "Impossible Made Possible: Encoding Intractable Specifications via Implied Domain Constraints." FMICS 2023.

Impossible Temporal Logic Specifications [43]

Lesson: Use Implied Domain Constraints

- unboundedness
- self-reference
- explicit counting

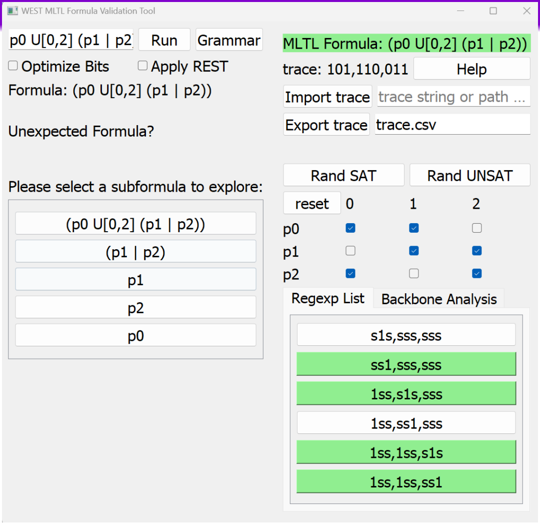
Implied domain constraints are often essential . . .

[43] C.Johannsen, B.Kempa, P.H.Jones, K.Y.Rozier, T.Wongpiromsarn. "Impossible Made Possible: Encoding Intractable Specifications via Implied Domain Constraints." FMICS 2023.

What Is Wrong With This Picture?

Aircraft visualization generated exclusively from explicit constraints

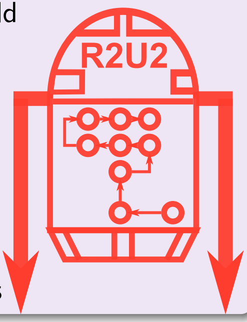
WEST [44]



[44] J. Elwing, L. Gamboa-Guzman, J. Sorkin, C. Travesset, Z. Wang, K.Y.Rozier. "Mission-Time LTL (MLTL) Formula Validation via Regular Expressions." iFM, Leiden, The Netherlands, November 13-15, 2023.

R2U2: Realizable Responsive Unobtrusive Unit

- Data Integrity: data is consistent, coherent, within expectations
- Sanity Checking: common-sense assumptions hold
- Fault Mitigation: real-time monitoring for fault signatures
- Security Monitoring: complex temporal patterns indicative of breaches
- Mission Integration: automatically catch mis-configured, or otherwise tenuous/faulty connections that elude system integration checks



http://r2u2.temporallogic.org/