Rockets, Route-Analyzers, Rotorcraft, and Robonaut2: Intelligent, On-board Runtime Reasoning

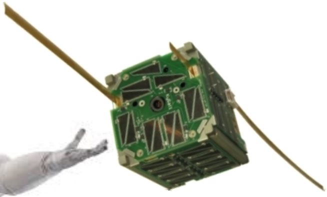
Kristin Y. Rozier
Iowa State University

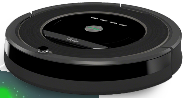
University of Bristol

June 7, 2022

How is **Flight Software**

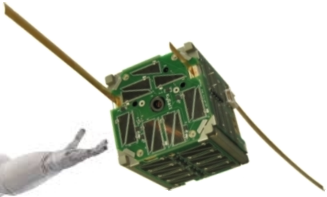
How is **Flight Software** Different from **Software**?

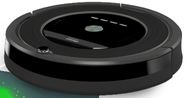





How is **Flight Software** Different from **Software**?

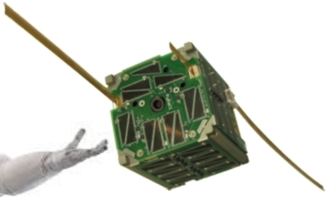
- **Has to** work





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

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*.

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

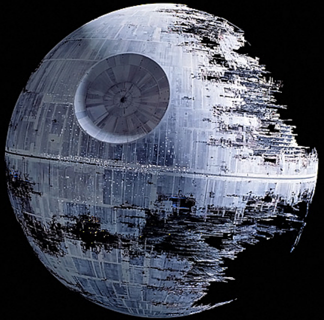
R2U2

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



Requirements

REALIZABILITY:

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

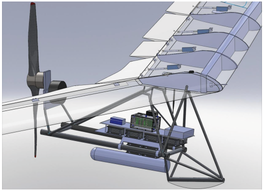
RESPONSIVENESS:

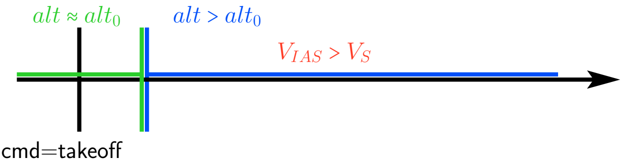
- *continuously monitor* the system
- *detect deviations* in *real time*
- *enable mitigation* or rescue measures

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

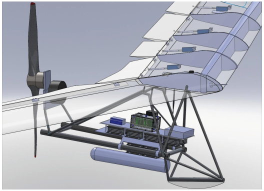
Runtime Observers for the Swift UAS

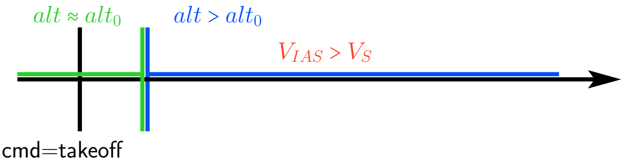




Whenever the Swift UAS is in the air, its indicated airspeed (V_{IAS}) must be greater than its stall speed V_S. The UAS is considered to be air-bound when its altitude *alt* is larger than that of the runway alt_0.

Runtime Observers for the Swift UAS

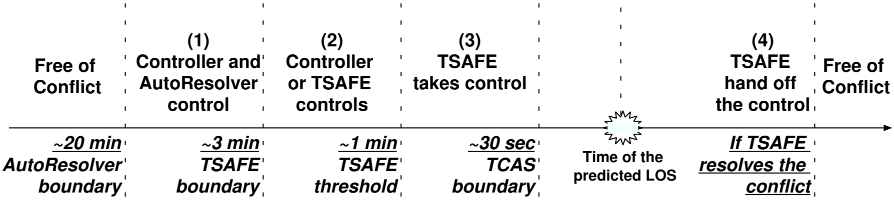




Whenever the Swift UAS is in the air, its indicated airspeed (V_{IAS}) must be greater than its stall speed V_S. The UAS is considered to be air-bound when its altitude alt is larger than that of the runway alt_0.

$$\text{ALWAYS}((alt > alt_0) \rightarrow (V_{IAS} > V_S))$$

Automated Airspace Concept High-Level Architecture[1234]



1 H. Erzberger, K. Heere, "Algorithm and operational concept for resolving short-range conflicts," Proc. IMechE G J. Aerosp. Eng. 224 (2) (2010) 225243

2 Y. Zhao, K.Y.Rozier, "Formal Specification and Verification of a Coordination Protocol for an Automated Air Traffic Control System." *Science of Computer Programming Journal*, vol 96 (3), 2014.

3 C. Mattarei, A. Cimatti, M. Gario, S. Tonetta, K.Y. Rozier, "Comparing Different Functional Allocations in Automated Air Traffic Control Design," *Formal Methods in Computer-Aided Design (FMCAD)*, 2015.

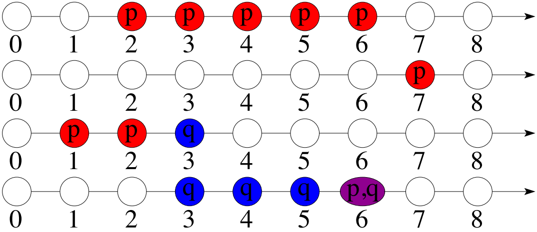
4 M. Gario, A. Cimatti, C. Mattarei, S. Tonetta, K.Y. Rozier, "Model Checking at Scale: Automated Air Traffic Control Design Space Exploration," *Computer Aided Verification (CAV)*, 2016.

Encoding Timelines: Linear Temporal Logic

Mission-time LTL (MLTL) reasons about *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$_{[3,8]}$	

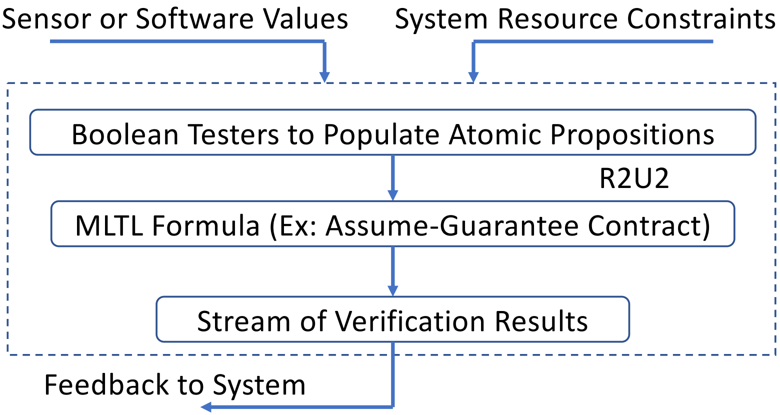


Mission-bounded LTL is an over-approximation for mission time τ

Asynchronous Observers (aka event-triggered)

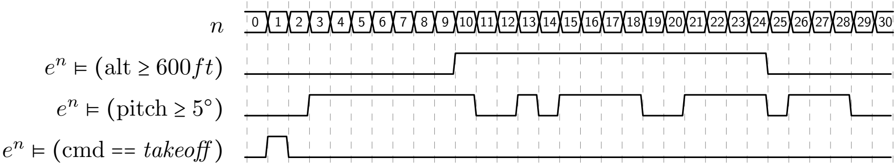
- *evaluate with every new input*
- 2-valued output: {true; false}
- resolve φ *as early as possible* (a priori known time)
- for each clock tick, may resolve φ for clock ticks prior to the current time n if the information required for this resolution was not available until n

R2U2 High-Level Architecture[5]



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

Asynchronous Observers Example

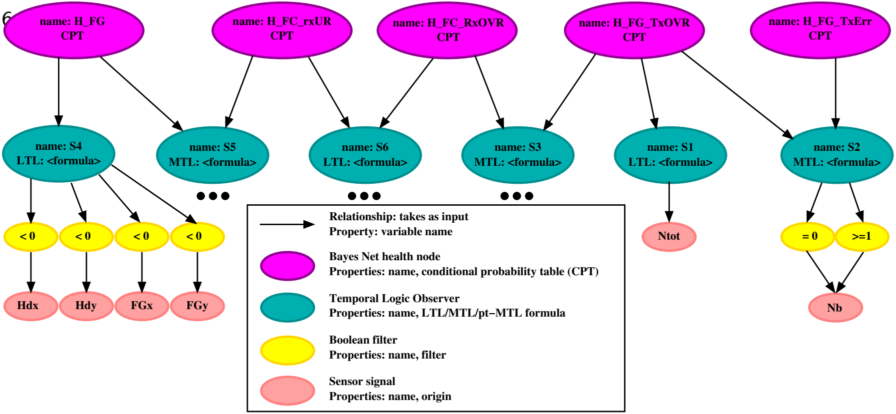


$\text{ALWAYS}_{[5]}(\text{pitch} \geq 5°)$

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	(␣, ␣)

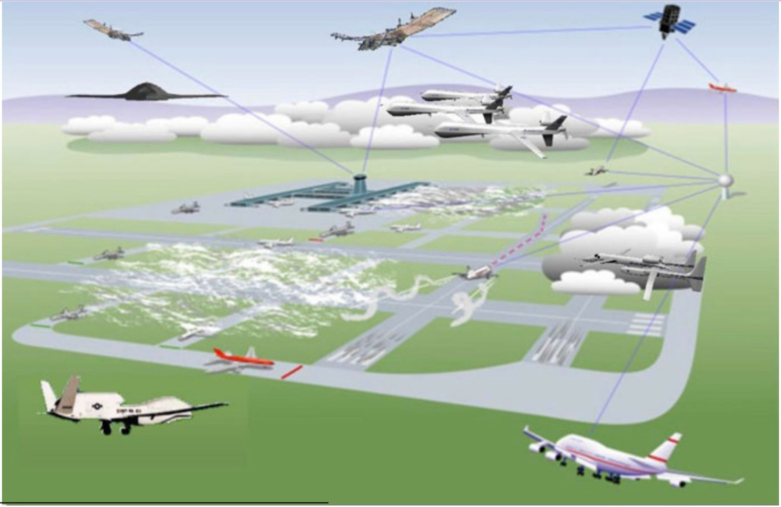
R2U2 Observation Tree (Specification)

[6]



[6] 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 2017), Springer-Verlag, Seattle, Washington, USA, September 13–16, 2017.

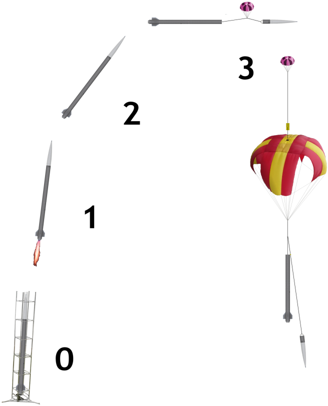
Adding UAS into the NAS?[7]

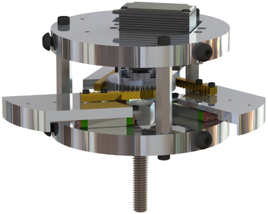


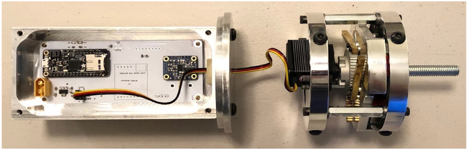
[7] Matthew Cauwels, Abigail Hammer, Benjamin Hertz, Phillip Jones, and Kristin Yvonne Rozier. "Integrating Runtime Verification into an Automated UAS Traffic Management System." *DETECT* 2020

https://www.youtube.com/watch?v=p6dwT0sTdH0

Cyclone Rocketry's Sounding Rocket[8]



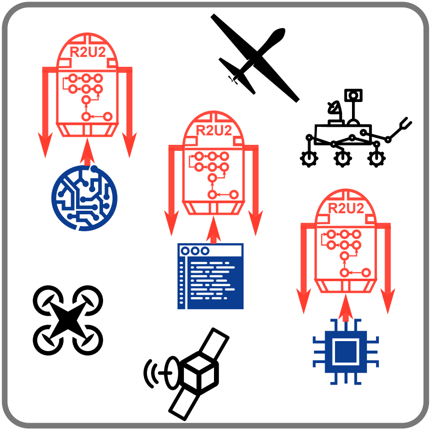




Left: Rocket mission states: *Launch Pad* (0), *Boost* (1), *Coast* (2), *Descent* (3). Right Top: Model of *Nova Somnium*'s ACS, Right Bottom: the physical ACS.

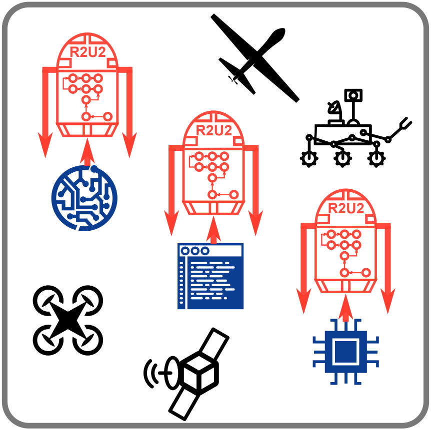
[8] B. Hertz, Z. Luppen, K.Y. Rozier. "Integrating Runtime Verification into a Sounding Rocket Control System." *NASA Formal Methods Symposium (NFM)*, 2021.

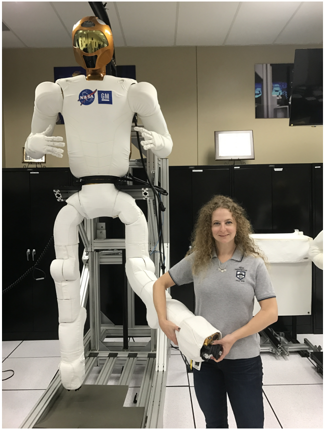
Multi-Platform, Multi-Architecture Runtime Verification of Autonomous Space Systems[9]



[9] NASA ECF Award

Multi-Platform, Multi-Architecture Runtime Verification of Autonomous Space Systems[9]

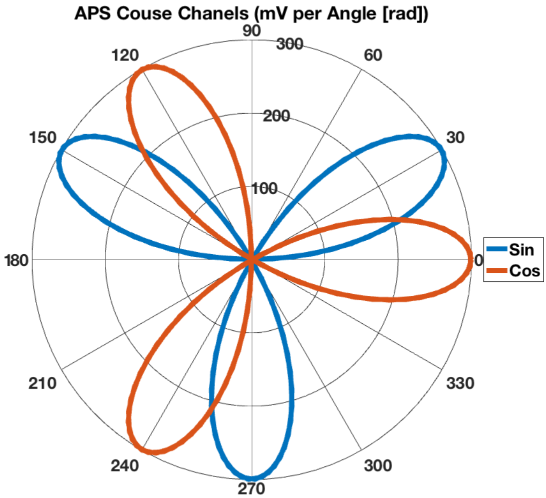




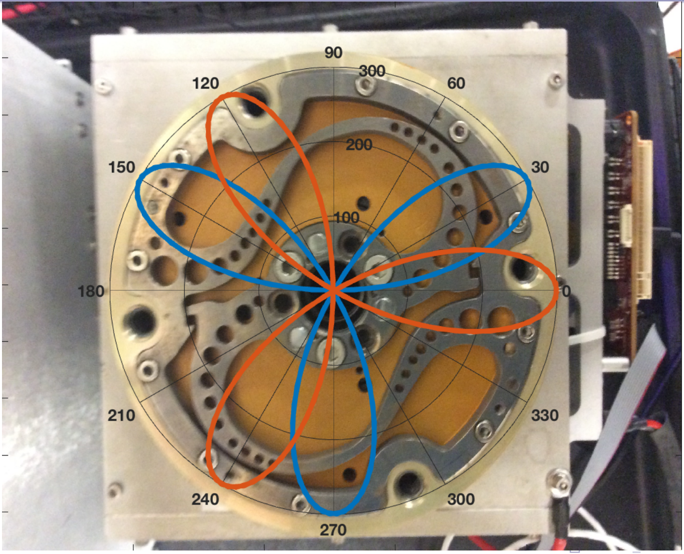
[9] NASA ECF Award

Robonaut2

Robonaut2's Knee



Robonaut2's Knee



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

Lifting Runtime Monitoring

Runtime Monitoring

[10] Falcone, Ylis, Sran Krsti, Giles Reger, and Dmitriy Traytel. "A taxonomy for classifying runtime verification tools." In International Conference on Runtime Verification, pp. 241-262. Springer, Cham, 2018.

Lifting Runtime Monitoring

Temporal Fault Disambiguation

↑

Runtime Monitoring

[10] Falcone, Ylis, Sran Krsti, Giles Reger, and Dmitriy Traytel. "A taxonomy for classifying runtime verification tools." In International Conference on Runtime Verification, pp. 241-262. Springer, Cham, 2018.

Lifting Runtime Monitoring

Temporal Fault Disambiguation

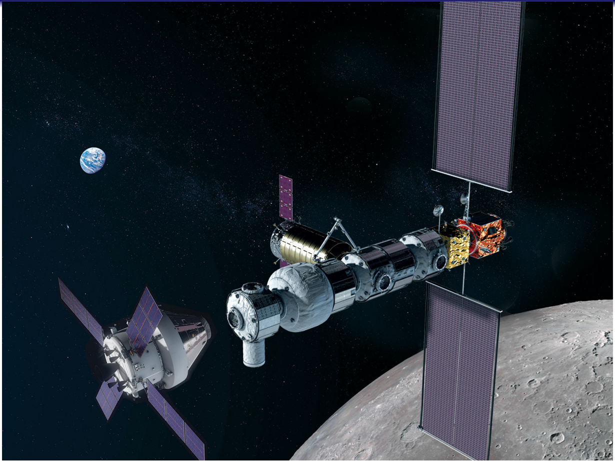
↑

Runtime Monitoring

"R2U2 breaks our taxonomy; it is entirely application driven."
— Giles Reger, 11/13/2018[10]

[10] Falcone, Ylis, Sran Krsti, Giles Reger, and Dmitriy Traytel. "A taxonomy for classifying runtime verification tools." In International Conference on Runtime Verification, pp. 241-262. Springer, Cham, 2018.

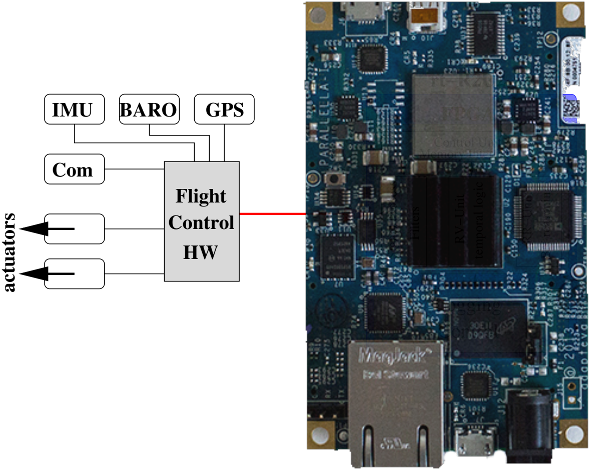
NASA Lunar Gateway: Assume-Guarantee Contracts[11]



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

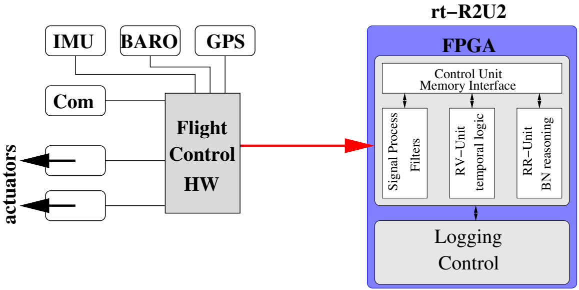
[11] Dabney, James B., Julia M. Badger, and Pavan Rajagopal. "Adding a Verification View for an Autonomous Real-Time System Architecture." In AIAA Scitech 2021 Forum, p. 0566. 2021.

Hard- and Software Architecture: Resource Estimation



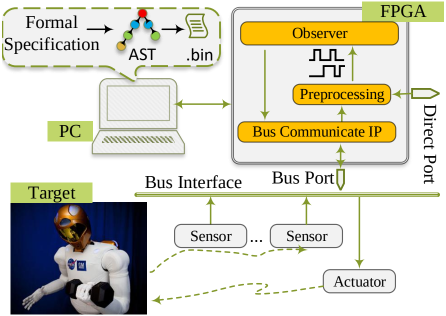
- How do we fit in the resources left over?
- Choose between 3 R2U2 implementations:
  - Hardware: FPGA
  - Software: C emulation of FPGA
  - Software: Object-oriented C++

Hard- and Software Architecture: Resource Estimation

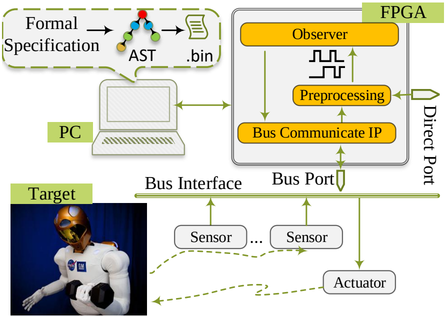


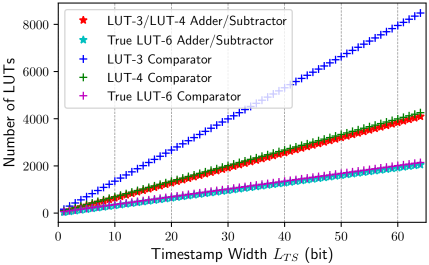
- How do we fit in the resources left over?
- Choose between 3 R2U2 implementations:
  - Hardware: FPGA
  - Software: C emulation of FPGA
  - Software: Object-oriented C++

Resource Estimation and Improved Encoding Algorithms

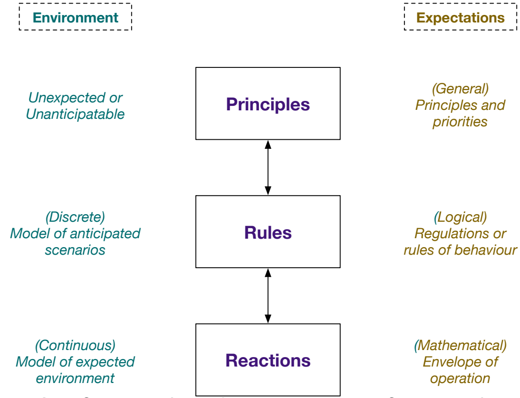


Resource Estimation and Improved Encoding Algorithms





Towards a Framework for Certification of Reliable Autonomous Systems[12]

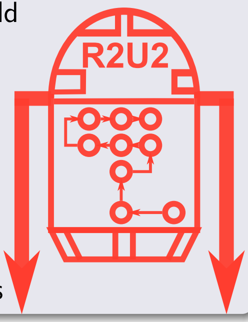


A reference three-layer autonomy framework

[12] M. Fisher, V. Mascardi, K.Y.Rozier, H. Schlingloff, M. Winikoff, N. Yorke-Smith, "Towards a Framework for Certification of Reliable Autonomous Systems," *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, vol 35 (8), 2021.

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

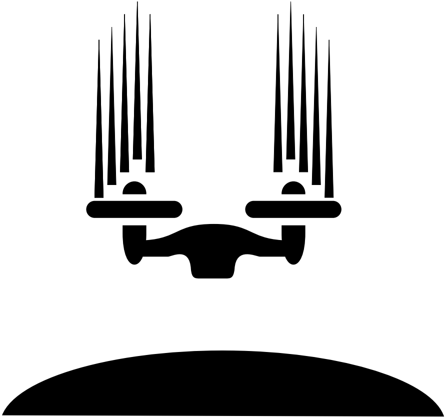


http://r2u2.temporallogic.org/

BACKUP SLIDES

Runtime Functional Specification Patterns[13]

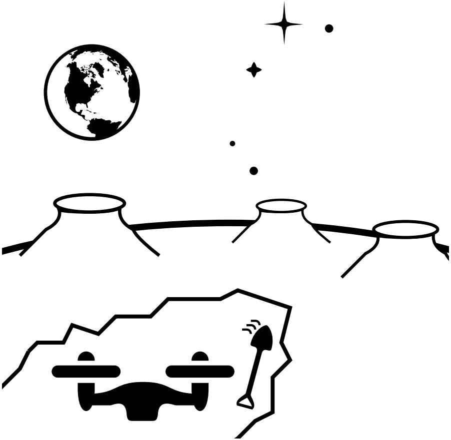
- Rates
- Ranges
- Relationships
- Control Sequences
- Consistency Checks



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

Runtime Functional Specification Patterns[13]

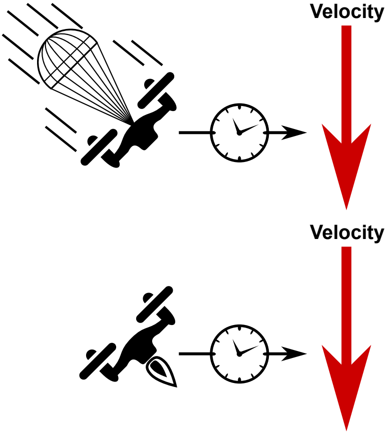
- Rates
- Ranges
- Relationships
- Control Sequences
- Consistency Checks



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

Runtime Functional Specification Patterns[13]

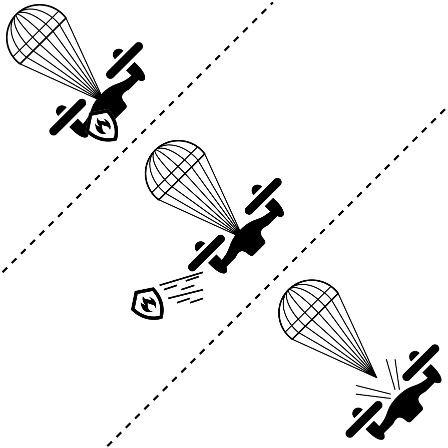
- **Rates**
- **Ranges**
- **Relationships**
- **Control Sequences**
- **Consistency Checks**



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

Runtime Functional Specification Patterns[13]

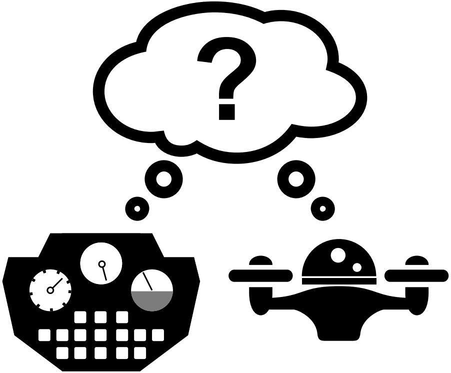
- **Rates**
- **Ranges**
- **Relationships**
- **Control Sequences**
- **Consistency Checks**



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

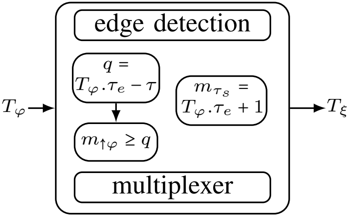
Runtime Functional Specification Patterns[13]

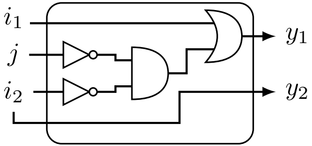
- Rates
- Ranges
- Relationships
- Control Sequences
- Consistency Checks

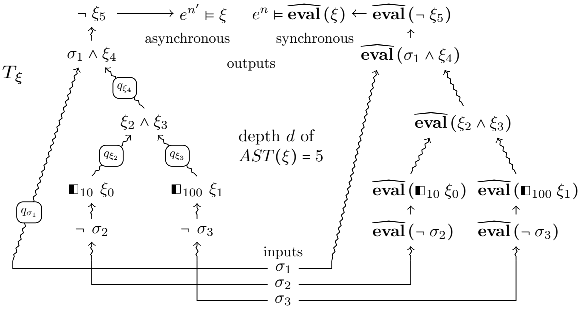


[13] K.Y.Rozier. "Specification: The Biggest Bottleneck in Formal Methods and Autonomy." VSTTE, 2016.

FPGA Implementation of Temporal Observers[14]







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

[14] 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.

Goals to Work Toward

Fault description: (1) identify when a "switch" happens from 1 of 3 positions (as it is at a discrete point during operation), and (2) to identify on the joint level which APS is at fault.
(1) is indicated by φ_1: do APS1 and APS2 disagree
(2) is indicated by the other two MLTL specs: φ_2, φ_3
If φ_1 is triggered but not φ_2 or φ_3 then we have a different fault; trigger standard error handling

Goal 1: detect this fault 100% of the time with no false positives

Goal 2: disambiguate between 3 actions:

1. Reinitialize assuming APS1 is bad
2. Reinitialize assuming APS2 is bad
3. No action: either there is no fault or a different fault has occurred

Goal 3: there is a precursor to this error whose cause is not known?

MLTL Specifications

Do APS1 and APS2 disagree by a large margin (2 radian threshold): indicates that there is a fault

$$THRESHOLD = (2.094 \pm 0.03 rad)$$

2.094 is the 120 separation; 0.03 is the range of the fine position sensing in APS

$$V_{threshold} = |r2.left\_leg.joint0.APS1 - r2.left\_leg.joint0.APS2| > (2.064)$$

$$\varphi_1 = G_{[0,3]}(V_{threshold})$$

Assumption: all faults occur in known transition modes so we can test the monitor with generated error traces for those scenarios

MLTL Specifications

Encoder drift fault occurs and encoder position agrees with APS2 (indicates fault occurred and APS1 is wrong)

$$AGREE_{Enc-APS2} = |r2.left\_leg.joint0.APS2 - r2.left\_leg.joint0.EncPos| < 0.01 rad$$

Assumption: this can be refined to represent encoder drift over time but this should be a good indication of agreement in general

$$\varphi_2 = [r2.left\_leg.joint0.FaultEncPos \wedge G_{[0,3]}(AGREE_{Enc-APS2})] \rightarrow APS1_{WRONG}$$

If there is disagreement but *not* encoder drift fault then assume APS2 is wrong:

$$\varphi_3 = G_{[0,3]}(V_{threshold}) \wedge !r2.left\_leg.joint0.FaultEncPos \rightarrow APS2_{WRONG}$$

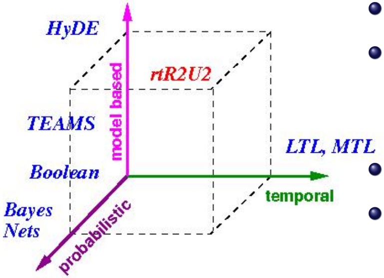
Assumption: the two agreeing sensors are correct {EncPos, APS1, APS2}
Assumption: all encoder faults are detected in r2.left_leg.joint0.FaultEncPos

R2 Case Study Next Steps

- Are all assumptions correct?
- Where are we stuck?
- Can we get more traces to see if we're detecting all faults?
  - generate organically by triggering faulty behavior
  - manufacture (e.g., manually)
  - get from NASA?
- How do we optimize the encoding?
  - efficiently encoding subtraction and *abs*() Boolean testers
  - improve interface for Boolean tester specification
  - resource estimation of $\varphi_1, \varphi_2, \varphi_3$
- What else can we monitor for?

Fault Detection and Monitoring

Any diagnosis system works with an *abstracted* model of the actual system

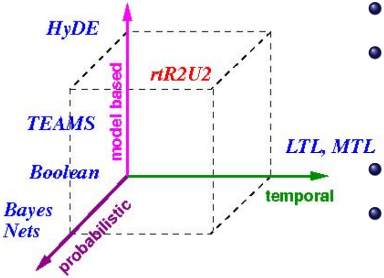


Typical Abstraction Dimensions

- Boolean conditions: "if-then-else" rules
- model-based: use hierarchical, multi-signal reachability (e.g., TEAMS) or simplified dynamic models (HyDE)
- temporal: use temporal logic
- probabilistic: use BN, or Fuzzy, or Neural Networks

Fault Detection and Monitoring

Any diagnosis system works with an *abstracted* model of the actual system

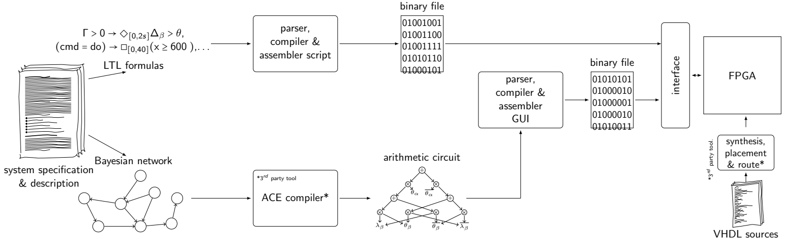


Typical Abstraction Dimensions

- Boolean conditions: "if-then-else" rules
- model-based: use hierarchical, multi-signal reachability (e.g., TEAMS) or simplified dynamic models (HyDE)
- temporal: use temporal logic
- probabilistic: use BN, or Fuzzy, or Neural Networks
- R2U2 combines model-based, temporal, and probabilistic paradigms for convenient modeling and high expressiveness

Tool Chain and FPGA Implementation of Bayes Nets[15]

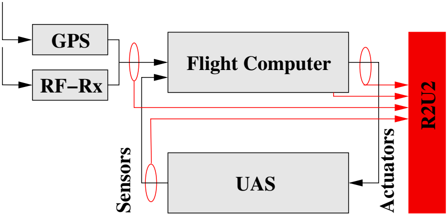
- Tool chain to translate SHM models into efficient FPGA-designs
- Bayesian Network models are compiled into arithmetic circuits that are evaluated by highly parallel special purpose execution units.



15 Johannes Geist, Kristin Yvonne Rozier, and Johann Schumann. "Runtime Observer Pairs and Bayesian Network Reasoners On-board FPGAs: Flight-Certifiable System Health Management for Embedded Systems." In *Runtime Verification (RV14)*, Springer-Verlag, September 22-25, 2014.

Monitoring and Diagnosis of Security Threats[16]

For threat detection we use R2U2 to perform *attack monitoring*, *post-attack system behavior monitoring*, and *diagnosis*.



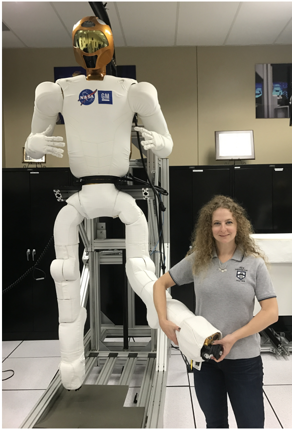
R2U2 monitors...

- Flight Software (FSW) inputs
  - GPS, GCS commands
  - sensor values
- actuator outputs
- important FSW variables

Monitoring of system inputs and analyzing post-attack behavior is not independent. We therefore model their interaction in R2U2.

[16] 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.

Robonaut2



Robonaut2