Highlights of Model Checking and Runtime Verification of Aerospace Systems

Verifiability Seminar

UKRI
Verifiability
Node

Kristin Yvonne Rozier
Iowa State University of Science and Technology

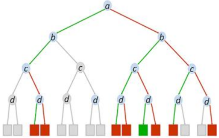
July 6, 2023

Research Interests

Automated Reasoning

- Avionics/Flight Software
- Satisfiability (SAT)/SMT
- AI/Algorithms
- Explainability

Formal Specification

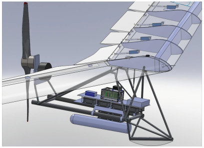


- Specification Patterns
- Specification Debugging
- Consistency/Temporal Satisfiability Checking

Design-time Safety Analysis

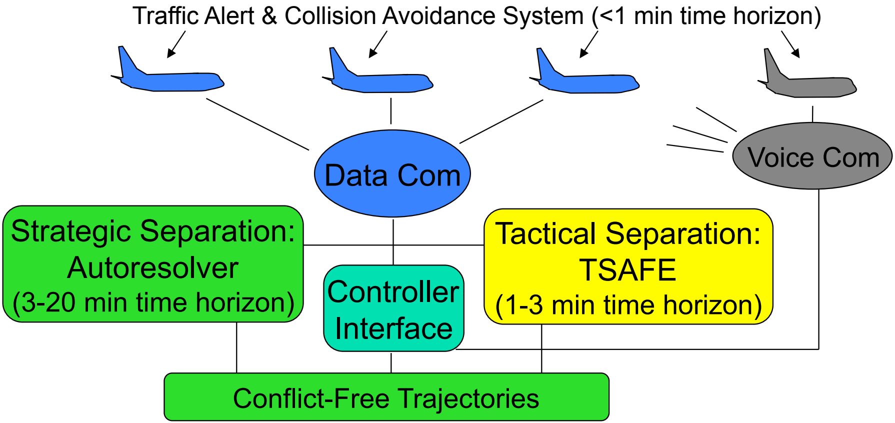
- Model Checking (Explicit and Symbolic)
- Model Based Design
- Requirements Elicitation
- Temporal Logic Encoding

Runtime Verification



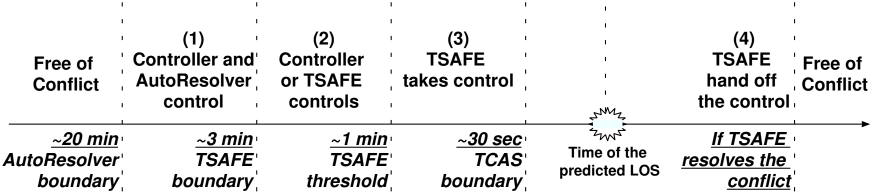
- R2U2 Engine
- System Health Management
- Resource-limited Sanity Checking
- Automated Diagnostics/Prognostics
- Real-time Intelligent Sensor Fusion

Automated Airspace Concept High-Level Architecture[1]


Traffic Alert & Collision Avoidance System (<1 min time horizon)
Data Com
Voice Com
Strategic Separation: Autoresolver (3-20 min time horizon)
Controller Interface
Tactical Separation: TSAFE (1-3 min time horizon)
Conflict-Free Trajectories

[1] 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[2]



Free of Conflict

(1) Controller and AutoResolver control

(2) Controller or TSAFE controls

(3) TSAFE takes control

(4) TSAFE hand off the control

Free of Conflict

~20 min AutoResolver boundary

~3 min TSAFE boundary

~1 min TSAFE threshold

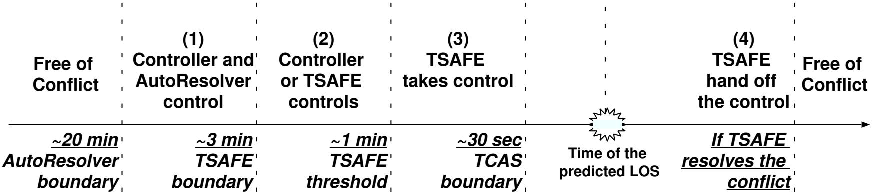
~30 sec TCAS boundary

Time of the predicted LOS

If TSAFE resolves the conflict

[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]



Free of Conflict

(1) Controller and AutoResolver control

(2) Controller or TSAFE controls

(3) TSAFE takes control

(4) TSAFE hand off the control

Free of Conflict

~20 min AutoResolver boundary

~3 min TSAFE boundary

~1 min TSAFE threshold

~30 sec TCAS boundary

Time of the predicted LOS

If TSAFE resolves the conflict

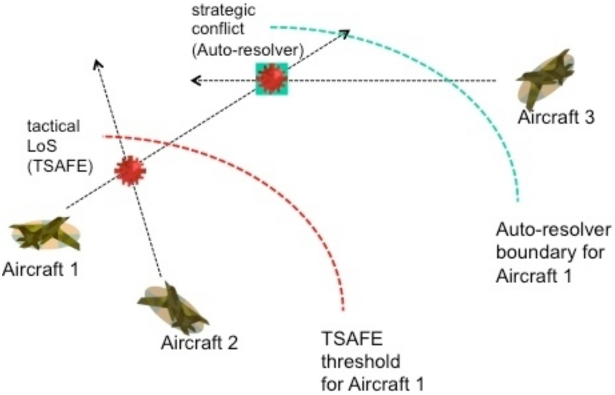
LTL Model Checking triggered system design changes[2]

[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.

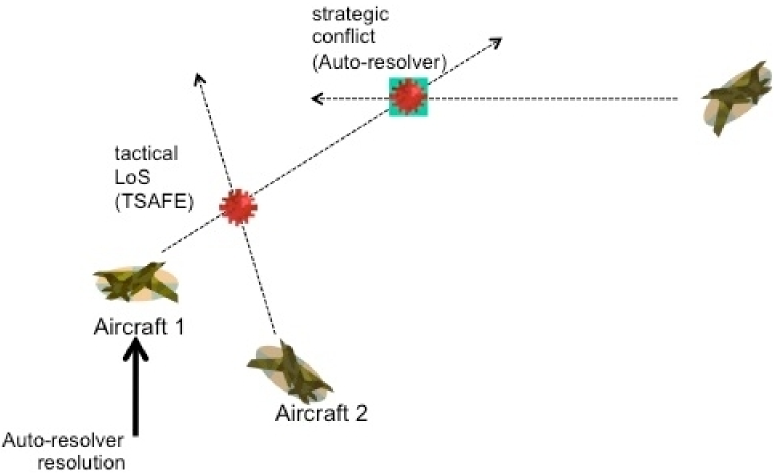
[3] H Erzberger, K Heere. "Algorithm and operational concept for resolving short-range conflicts." Proc. IMechE G J. Aerosp. Eng. 224 (2) (2010) 225–243.

Counterexample

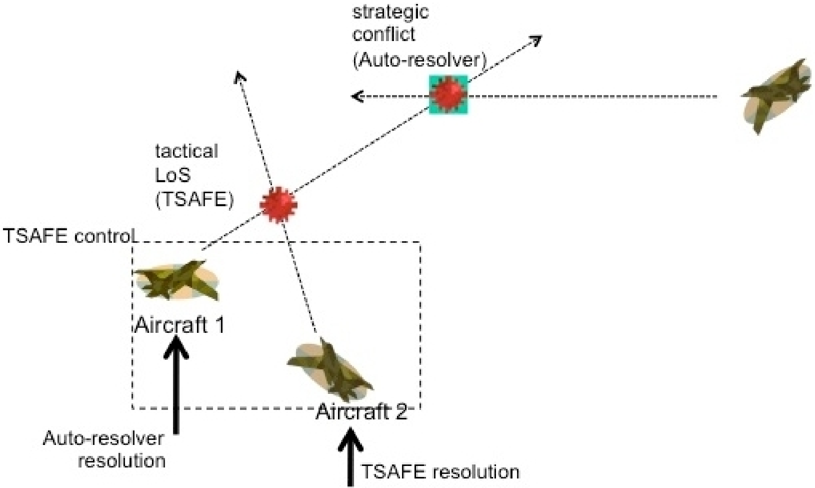
Specification: "If the controller hands off the control of an aircraft to TSAFE, this aircraft will not execute commands from the controller or Autoresolver."



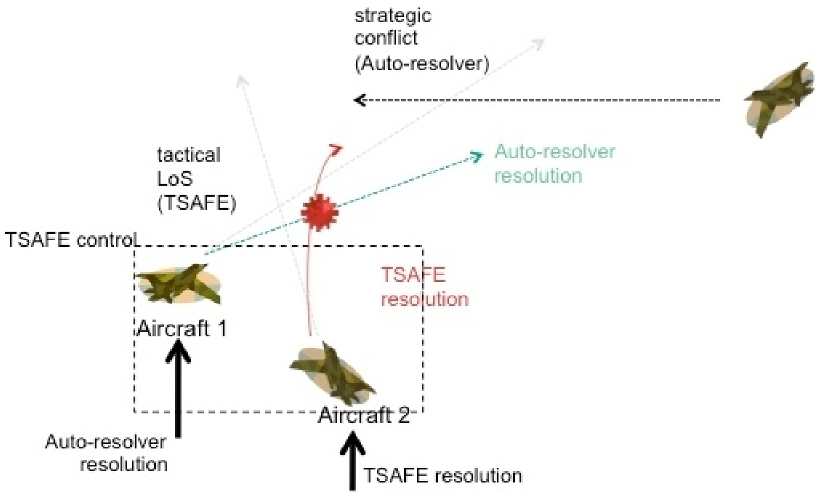
Counterexample



Counterexample

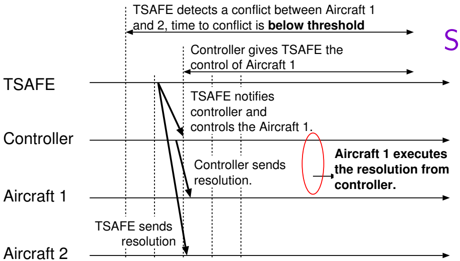


Counterexample



Counterexample: Fixed![4]

Specification: "If the controller hands off the control of an aircraft to TSAFE, this aircraft will not execute commands from the controller or Autoresolver."



TSAFE detects a conflict between Aircraft 1 and 2, time to conflict is **below threshold**

Controller gives TSAFE the control of Aircraft 1

TSAFE

TSAFE notifies controller and controls the Aircraft 1.

Controller

Controller sends resolution.

Aircraft 1 executes the resolution from controller.

Aircraft 1

TSAFE sends resolution

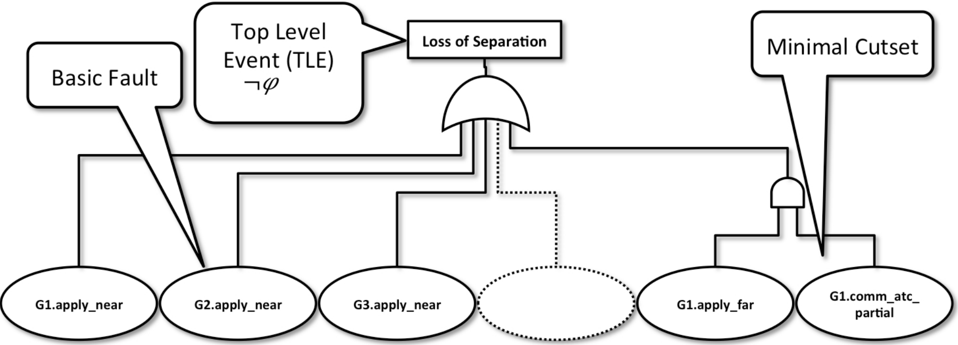
Aircraft 2

Solution:

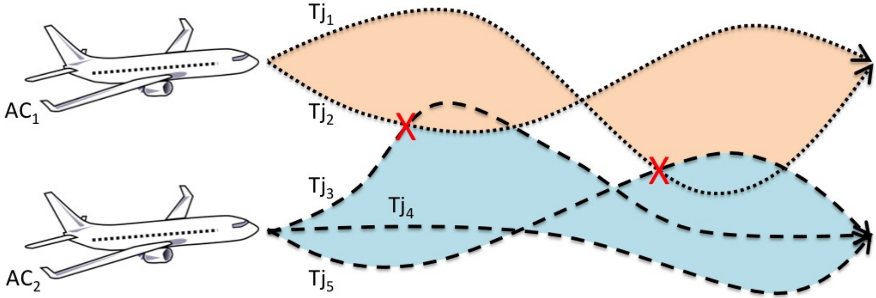
- A1 receives notice of control transfer and "hold current route" resolution from TSAFE
- AR/controller's command will be superseded and ignored

[4] Zhao, Yang, and Rozier, Kristin Yvonne. "Formal Specification and Verification of a Coordination Protocol for an Automated Air Traffic Control System." In AVoCS 2012.

Fault Tree Analysis

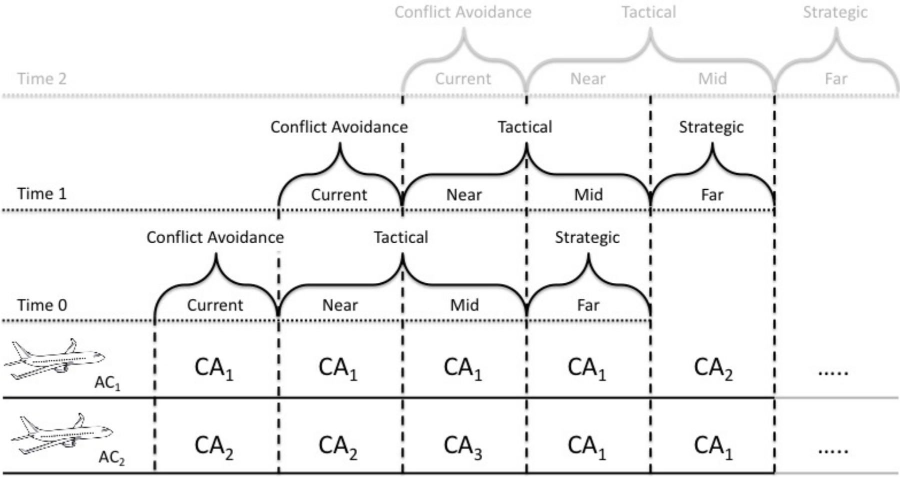


Formal Modeling: Conflict Areas[5]

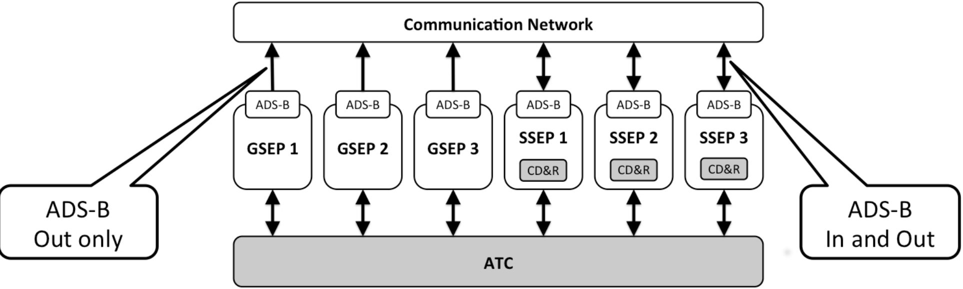


[5] Cristian Mattarei, Alessandro Cimatti, Marco Gario, Stefano Tonetta and Kristin Y. Rozier. "Comparing Different Functional Allocations in Automated Air Traffic Control Design." In Formal Methods in Computer-Aided Design (FMCAD), IEEE/ACM, 2015.

Formal Modeling: Time Windows

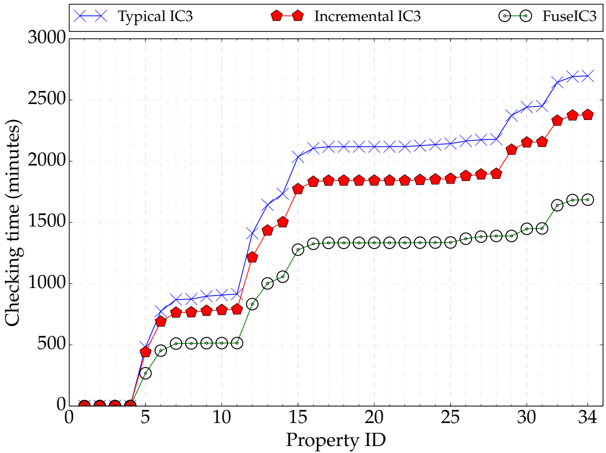


Formal Modeling: System Components[6]



[6] Marco Gario, Alessandro Cimatti, Cristian Mattarei, Stefano Tonetta and Kristin Y. Rozier. "Model Checking at Scale: Automated Air Traffic Control Design Space Exploration." In *Computer Aided Verification (CAV)*, 2016.

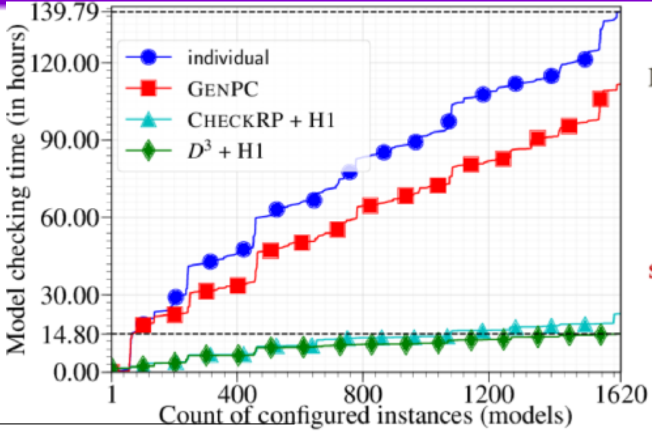
FuseIC3: An Algorithm for Checking Large Design Spaces[7]



Model checking **34 formulas** over **1,620 models** is **5.48x faster**

[7] Rohit Dureja and Kristin Yvonne Rozier. "FuseIC3: An Algorithm for Checking Large Design Spaces." In Formal Methods in Computer-Aided Design (FMCAD), IEEE/ACM, Vienna, Austria, October 2-6, 2017.

More Scalable LTL Model Checking via Discovering Design-Space Dependencies (D^3)[8]

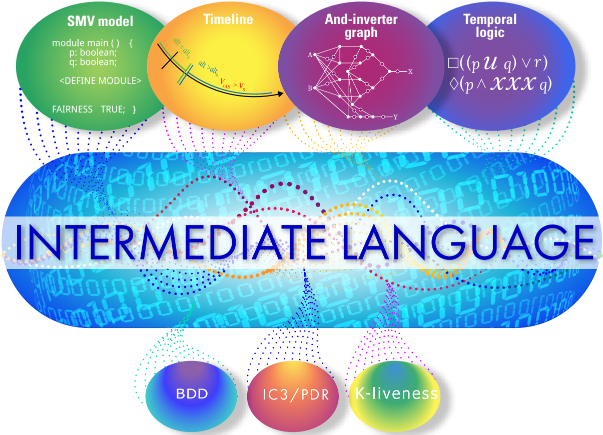


UNSAT properties

9.0x speedup

[8] Rohit Dureja and Kristin Yvonne Rozier. "More Scalable LTL Model Checking via Discovering Design-Space Dependencies (D^3)." In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, part I, volume 10805 of Springer LNCS, pages 309-327, Springer-Verlag, Thessaloniki, Greece, 14-21 April 2018.

Developing an Open-Source, State-of-the-Art Symbolic Model-Checking Framework for the Research Community[9]



```
module main ( )  {
    p: boolean;
    q: boolean;

  <DEFINE MODULE>

FAIRNESS   TRUE;  }
```

[9] NSF CCRI (modelchecker.temporallogic.org), **PI**

Necessity of Specification Debugging

Verification (e.g., model checking) finds disagreements between the system model and the formal specification.

If there is disagreement, which one has the error?

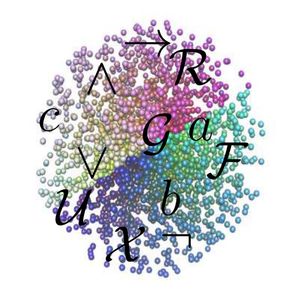
If there is agreement, it does not mean there is no error.

ALWAYS $(A \rightarrow$ EVENTUALLY $B)$

- A *valid* specification is true in *all* models.
  - Ex: A and B are logically equivalent.
- An *unsatisfiable* specification is *never* true.
  - Ex: A and EVENTUALLY B are contradictory.

We Need to Establish Rigorous Benchmarks [10]

Random Formulas: 60,000



Counter Formulas: ~60 (4 types)

00 01 10 11 . . .

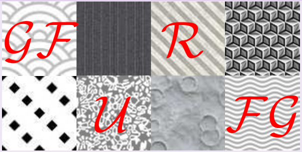
000 001 010 011 100 . . .

0000 0001 0010 0011 0100 0101 . . .

00000 00001 00010 00011 00100 00101 00110 . . .

⋮

Pattern Formulas: ~8,000 (9 patterns)



[10] K.Y. Rozier and M.Y. Vardi. “LTL Satisfiability Checking.” SPIN’07.

We Must Check Specifications for Satisfiability![11]

1. Find real specification errors
2. Build into model checking

Helios flying wing over Hawaii, just before it crashed.

Cited in over 250 publications:

- Our benchmarks are now a de facto standard.
- Evaluation of LTL encoding algorithms changed.
- Benchmarks & code integrated into industrial tool SPOT.

[11] K.Y. Rozier and M.Y. Vardi. "LTL Satisfiability Checking." STTT, 2010.

LTL Satisfiability Checking With Fairness

LTL formula f

Fairness constraint c

ALWAYS EVENTUALLY $c \rightarrow f$

An overstrict c can effectively cause f to be valid!

Example:

Specification: "All TSAFE alerts will be eventually resolved."

Fairness Constraint: Progress between TSAFE alerts

Wrong: `FAIRNESS (TSAFE_Alert = Non);`

Right: `FAIRNESS (TSAFE_Alert != AT);`

PANDA: A Multi-Encoding Approach to LTL Satisfiability Checking [12]

PANDA (Portfolio Approach to Navigate the Design of Automata)

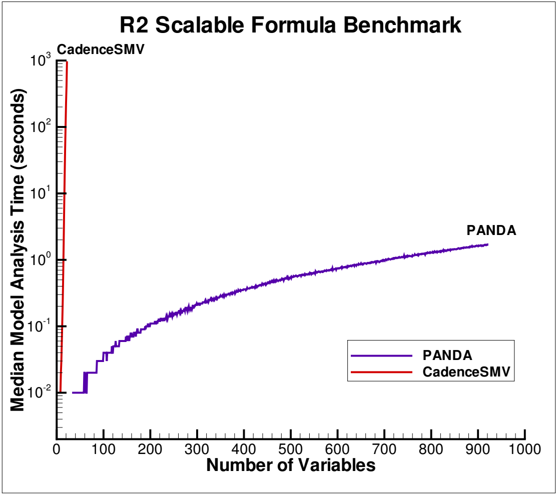
- 30 parallel LTL encodings
- Up to exponentially faster than the best tool (SMV) alone

New uses today. . .

LTL satisfiability checking pinpointed overconstrained specifications

[12] Rozier, Kristin Y., and Vardi, Moshe Y. "A Multi-Encoding Approach for LTL Symbolic Satisfiability Checking." In 17th International Symposium on Formal Methods (FM), LNCS, Springer, 2011.

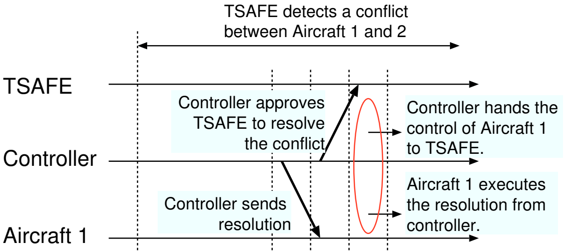
PANDA Can Be Exponentially Faster



$$R_2(n) = (..(p_1 \ \mathcal{R} \ p_2) \ \mathcal{R} \ \dots) \ \mathcal{R} \ p_n.$$

Specification Debugging Changes Requirements

Example: If the controller hands off the control of an aircraft to TSAFE, the aircraft will not execute commands from the AR/controller.


TSAFE detects a conflict between Aircraft 1 and 2
TSAFE
Controller
Aircraft 1
Controller approves TSAFE to resolve the conflict
Controller sends resolution
Controller hands the control of Aircraft 1 to TSAFE.
Aircraft 1 executes the resolution from controller.

Wrong: `!(!CTR_control & aircraft.CTR_cmd_done)`

Flight-Certifiable Runtime Verification[13]

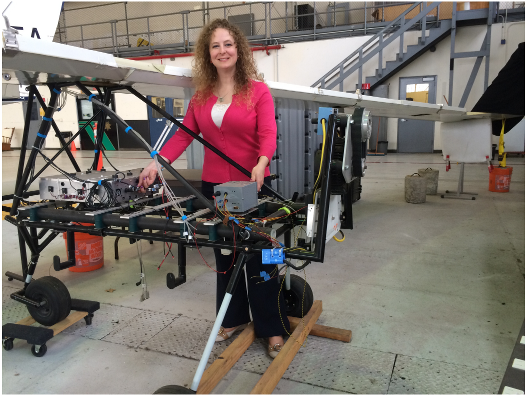
RESPONSIVE

REALIZABLE

UNOBTRUSIVE

Unit

R2U2



https://r2u2.temporallogic.org/

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

R2U2: Realizable, Responsive, Unobtrusive[14]

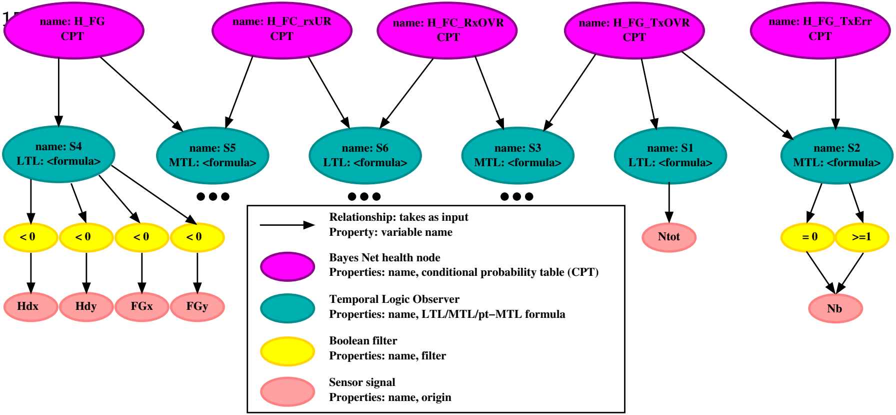
1. **Signal Processing**: Preparation of sensor readings

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**: Mission-time LTL (MLTL) (plus pt-MLTL, set-wise reasoning)

3. **Bayes Nets**: Efficient decision making
   - **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.

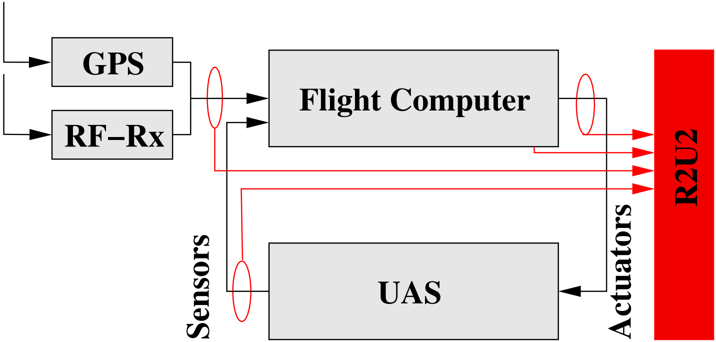
R2U2 Observation Tree (Specification)



[15] 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.

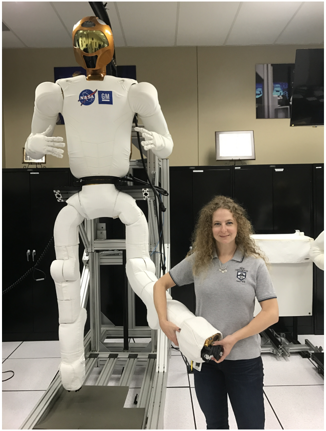
Monitoring and Diagnosis of Security Threats[16]

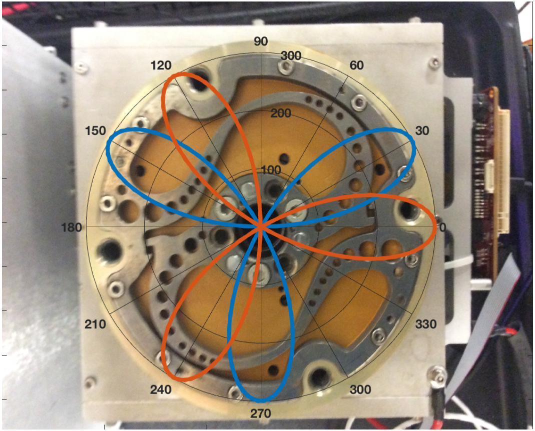
Threat detection: *attack monitoring*, *post-attack system behavior monitoring*, and *diagnosis*.



[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's Knee[17]



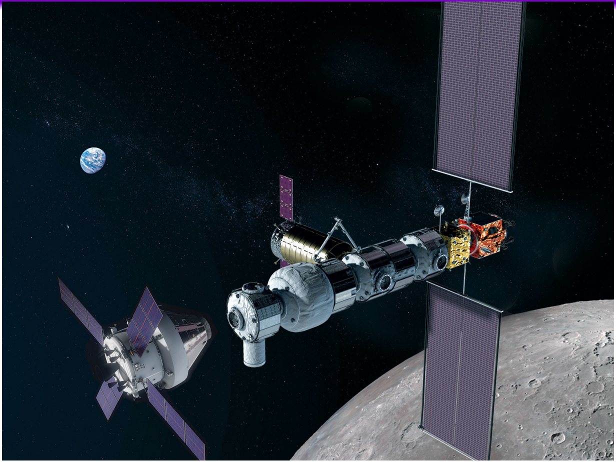


[17] Kempa, Zhang, Jones, Zambreno, Rozier. "Embedding Online Runtime Verification for Fault Disambiguation on Robonaut2." FORMATS, 2020.

Robonaut2's Knee

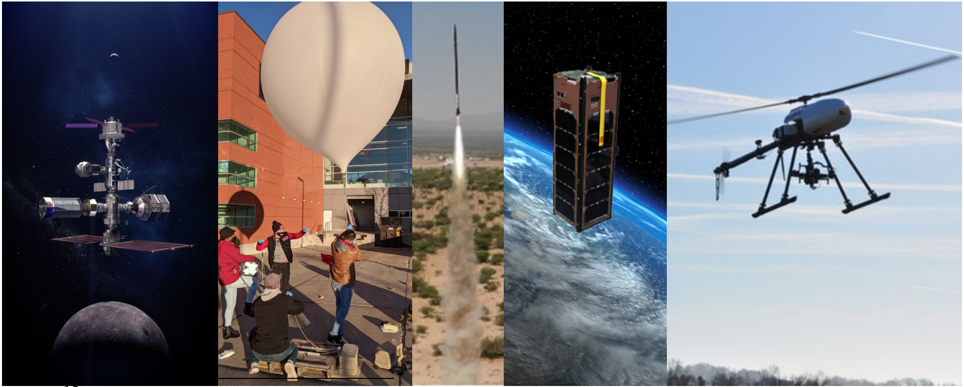
http://temporallogic.org/research/R2U2/R2U2-on-R2_demo.mp4

NASA Lunar Gateway: Assume-Guarantee Contracts→R2U2



[18] 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.

Flight-Certifible Runtime Verification [19] [20] [21] [22]



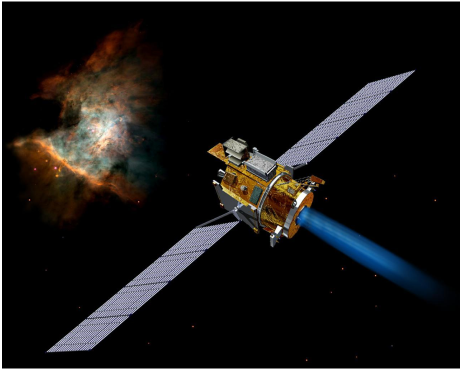
[19] Hariharan, Kempa, Wongpiromsarn, Jones, Rozier. "MLTL Multi-type (MLTLM): A Logic for Reasoning about Signals of Different Types." NSV 2022.

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

[21] Hertz, Luppen, Rozier. "Integrating Runtime Verification into a Sounding Rocket Control System." NFM 2021.

[22] 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.

MLTL Multi-type (MLTLM): A Logic for Reasoning About Signals of Different Types[23]



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.

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

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

Major Contributions

- Specification debugging via satisfiability checking: LTL LTLf, MTL
  - changed the requirements for the Automated Airspace Concept
- Benchmarks of temporal logic specifications, model-checking models
- Model Checking: algorithmic improvements, design-space analysis, international standards
- Runtime Verification: real-time responsiveness, flight-certifiable algorithms

`laboratory.temporallogic.org`