Formal Methods for V&V and Certification

Dr. Laura Humphrey

Team Lead, V&V of Complex & Autonomous Systems

Aerospace Systems Directorate, Autonomous Controls Branch (AFRL/RQQA)

October 2019

Outline

- Motivation
- Overview of formal methods
- Formal methods for autonomous systems
- Supplementary & alternative approaches
- Where our group is going
- Successes of formal methods
- Resources

Outline

- Motivation
- Overview of formal methods
- Formal methods for autonomous systems
- Supplementary & alternative approaches
- Where our group is going
- Successes of formal methods
- Resources
- Intersperse answers throughout

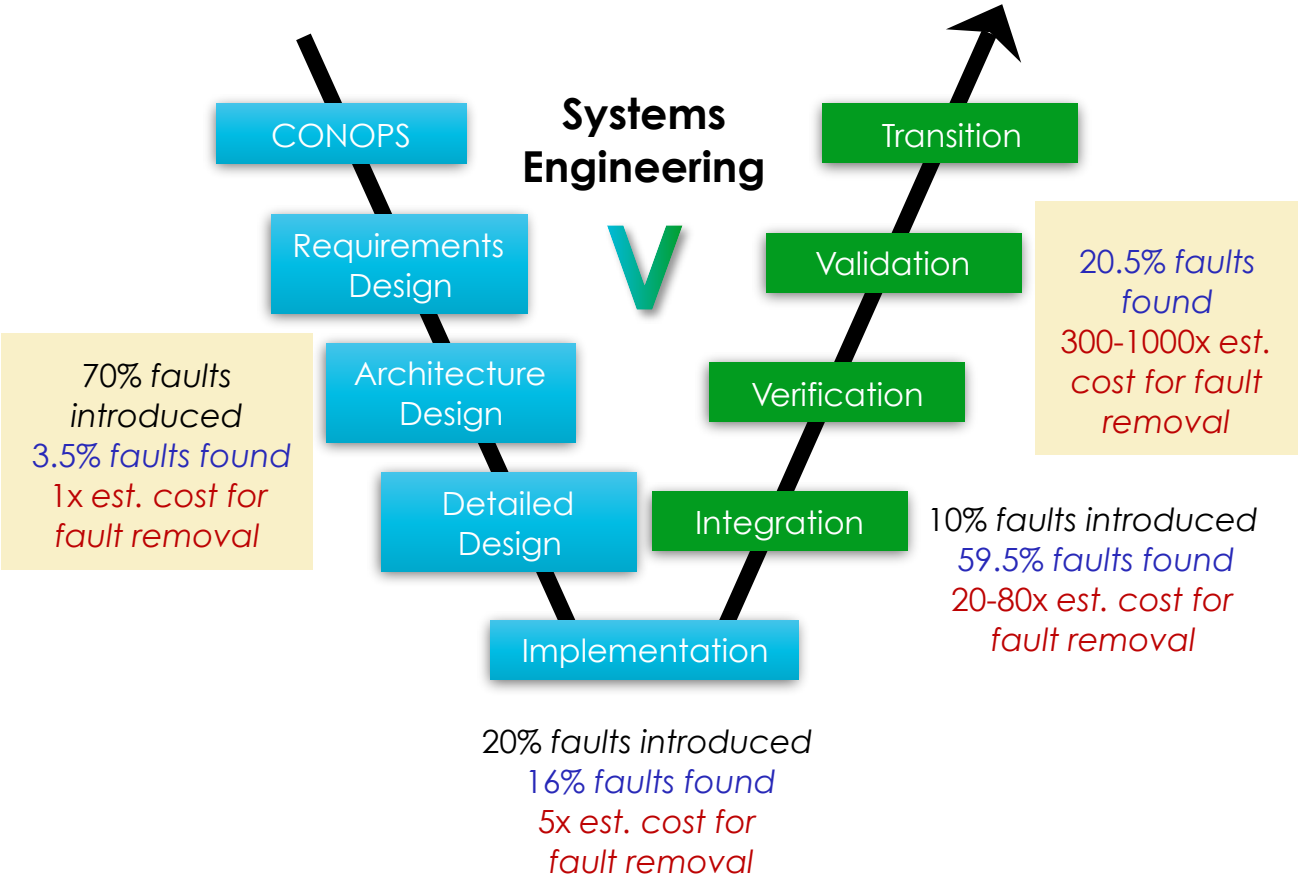
How is Assured Autonomy Interpreted in My Area?

- Autonomy
  - General word cloud: non-deterministic, reactive, adaptive
  - Unique part of my definition: too complex to V&V through current approaches

- Assured (formal methods)
  - Having mathematically rigorous evidence of safety and correctness for critical functionality
  - Potentially less rigorous or thorough evidence for less critical functionality (i.e., commensurate with risk)

Motivation

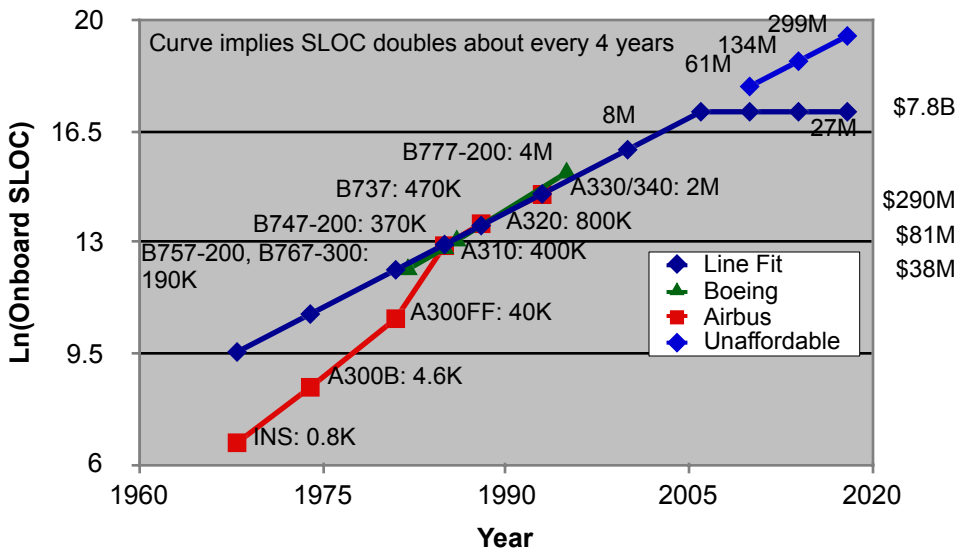
Rigorous



Rigorous design to decrease errors & cost[1,2,3]

Rapid

- Exponential software growth is making V&V through testing unaffordable



- Boeing & Airbus[4]: 27M+ SLOC → $10B+
- Driverless car testing[5]: 11B miles, $6B, 500y

Automated analysis to speed up V&V

What are Formal Methods?[6,7]

- Mathematically-based techniques for specification, design, and verification of software & hardware
  - Formal logic
  - Discrete mathematics
  - Computer-readable languages
- Two major activities
  - Modeling – high- & low-level requirements, architecture, source code, object code
  - Analysis – consistency, traceability, compliance, robustness, completeness
- <u>Formal</u> analysis must be <u>sound</u> (never say a property is true if it is not)
- Three categories
  - Theorem proving – use axioms & inference rules to prove properties of a model
  - Model checking – exhaustively search a model for a counterexample to a property
  - Abstract interpretation – construct & analyze conservative representations of software

Benefits of Formal Methods[6,7]

- Remove ambiguity (rigorous) → enable semi- or automated analysis (rapid)

- Analysis of general properties such as
  - Compliance: low-level model complies with higher-level model/architecture/ requirement
  - Traceability: all low-level model aspects trace to a higher-level model/architecture/ requirement & vice versa
  - Consistency: absence of conflicts
  - Completeness: output specified for all input conditions & vice versa
  - Robustness: nominal & off-nominal conditions

- Analysis of software properties such as
  - Freedom from runtime exceptions
  - Freedom from deadlock
  - Non-interference between different levels of criticality
  - Worst case execution time (WCET)
  - Bounds on stack size during execution
  - Freedom from unintended function
  - Correct synchronous or asynchronous behavior

A Simple Example

- Consider a function that returns (a + b)/c bounded by a threshold
- What could go wrong with this simple function?

```
function foo(a, b, c, threshold : Float) return Float is
      x : Float;
begin
   x := (a + b) / c;
   if x <= threshold and x >= -threshold then
      return x;
   elsif x > threshold then
      return threshold;
   else
      return -threshold;
   end if;
end;
```

A Simple Example

- Consider a function that returns (a + b)/c bounded by a threshold
- What could go wrong with this simple function?

What if a + b overflows?

What if c = 0?

What if (a + b)/c overflows?

```
function foo(a, b, c, threshold : Float) return Float is
      x : Float;
begin
   x := (a + b) / c;
   if x <= threshold and x >= -threshold then
      return x;
   elsif x > threshold then
      return threshold;
   else
      return -threshold;
   end if;
end;
```

- Maybe verifying this function is not so simple after all!

A Simple Example of Functional Verification in SPARK[8,9]

- SPARK is a programming language & associated set of verification tools
- Subset of Ada
- Performs several types of analyses, including functional verification
  - Absence of runtime errors (robustness)
  - Code satisfies user-specified low-level requirements written as contracts, e.g. pre- & post-conditions (compliance)
  - Low-level requirements are consistent
  - Certain types of low-level requirements are complete
- Functional verification achieves full coverage over all specified inputs

Specification:
Low-Level Requirements as Pre- & Post-Conditions

```
function foo(a, b, c, threshold : Float) return Float
with Pre => a > -Float'Last/2.0 and a < Float'Last/2.0 and
            b > -Float'Last/2.0 and b < Float'Last/2.0 and
            c >= 1.0 and threshold > 0.0,
    Post => foo'Result <= threshold and
            foo'Result >= -threshold;
```

Code Implementation

```
function foo(a, b, c, threshold : Float) return Float is
      x : Float;
begin
   x := (a + b) / c;
   if x <= threshold and x >= -threshold then
      return x;
   elsif x > threshold then
      return threshold;
   else
      return -threshold;
   end if;
end;
```

A Simple Example of Dependency Contracts in SPARK[8,9]

- SPARK can also perform analysis of dependency contracts
- Data dependency contracts
  - What global data is read and/or written
  - All read global data is initialized
  - All written global data is used
- Flow dependency contracts
  - Which output variables depend on which input variables
- Contracts help specify the architecture & low-level requirements
- Verifies all data is traceable to the architecture & low-level requirements

Specification for a Procedure with a Data & Flow Dependency Contract

```
procedure Append_To_Stream(Message : in String;
                           Status  : out Boolean)
with Global => (In_out => Message_Stream),
     Depends => (Message_Stream => (Message_Stream,
                                    Message)
                 Status =>         (Message_Stream,
                                    Message));
```

DO-333 Certification Case Studies[10]

- Formal methods applied to flight control system (FCS) with redundant flight guidance systems (FGSs)

•Models & longer report publicly available from NASA

Theorem Proving

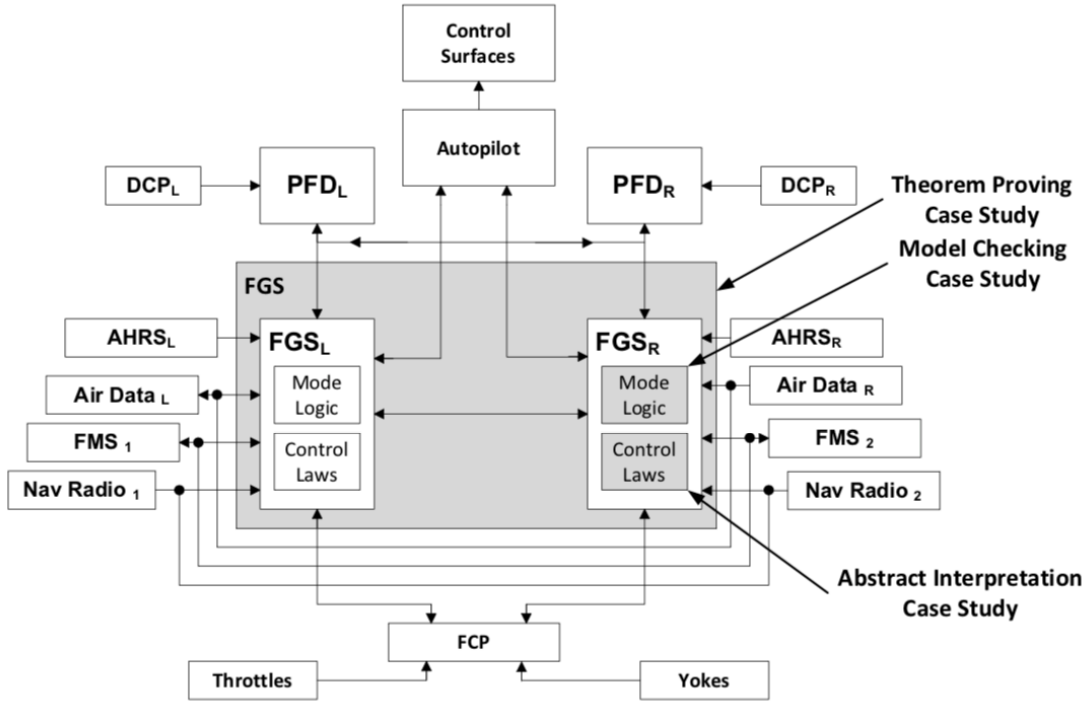
- Formalized system, high-level requirements (HLR) & architecture for FGS switching logic
  - System requirements: (1) Only one FGS active, (2) At least one FGS active, (3) Pressing transfer switch switches FGS ...
  - Architecture: Communications bus, clocks, switch, FGS
  - HLR: initial values & state update equations for all components
- Proved completeness & consistency of HLR & architecture, compliance with system requirements

Model Checking

- Formalized HLR and low-level requirements (LLR) for FGS logic for selecting lateral & vertical mode
  - HLR: (1) At least one vertical mode active, (2) At most one vertical mode active, (3) *Vertical Approach* cannot be active until *Lateral Approach* has become active ...
  - LLR: Simulink/Stateflow models of FGS mode logic
- Proved compliance of LLR with HLR

Abstract Interpretation

- Analyzed for runtime errors (robustness) & unreachable code (traceability) in C code generated by Simulink model of Heading Control Law

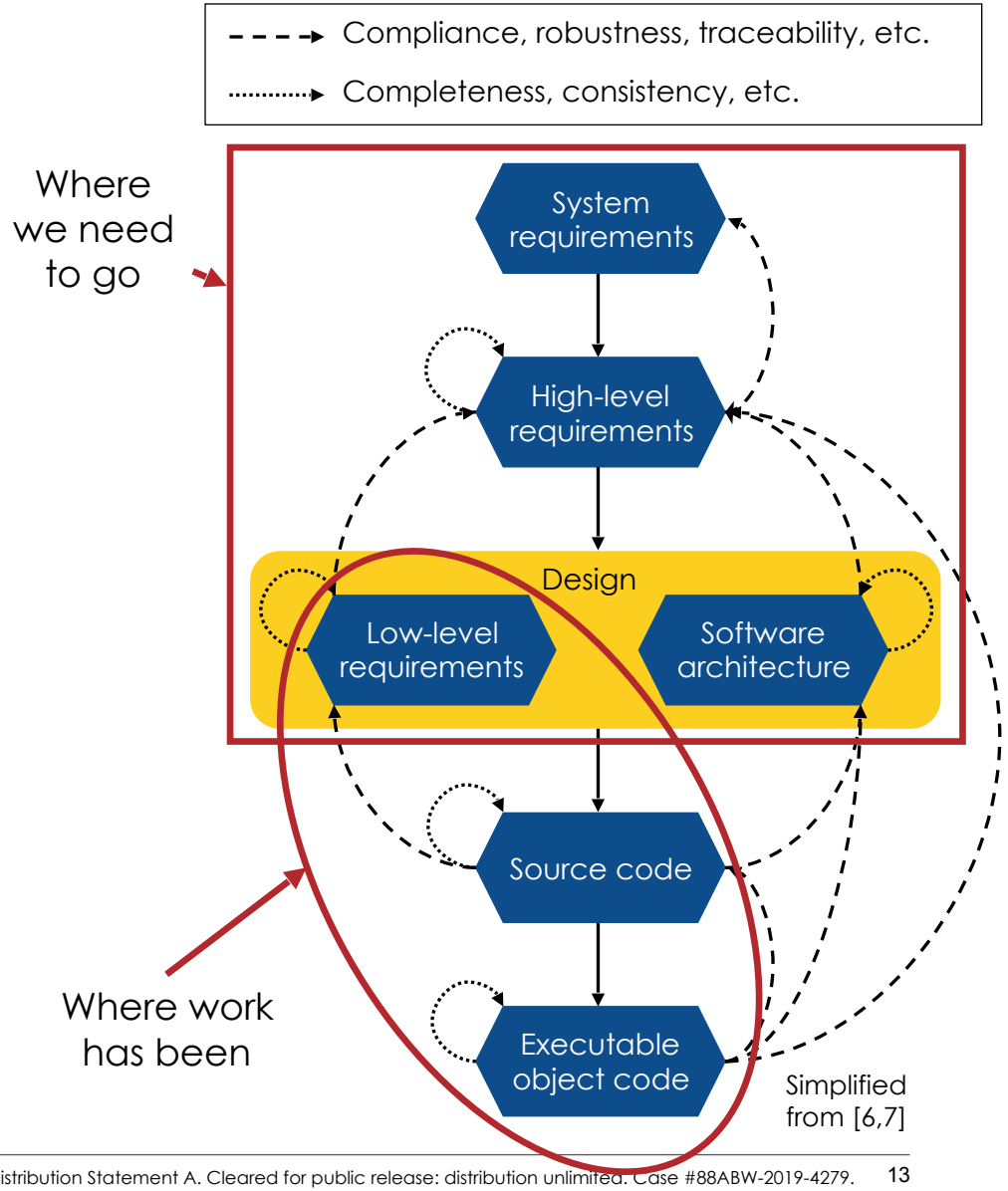


Formal Methods for Autonomous Systems

- Need automation to help build autonomy!

- Current work has focused on source code & low-level requirements, but still need to
  - Integrate these methods into existing workflows better
  - Increase education & provide good case studies

- Autonomy requirements & architecture will be more complex, so we & others have
  - Developed tools for architectures, LLR & HLR[11,12]
  - Used tools to find errors in an autonomous protocol for controlling teams of unmanned air vehicles[13]
  - Developed tools to automate design from LLR, HLR & architectures, e.g. synthesize code skeletons or source code[14, 15, 16]



What are Key Challenges your Field Thinks About?

- Formalizing to enable automated rigorous analysis
  - Richer requirements: more focused on safety & high-level goals, possibly re-evaluated by system online
  - Flexible architectures: aid decomposition, enable "plug & play" composability, yet still provide strong evidence
- Handling complexity
  - Horizontally (compositional verification)
  - Vertically (end-to-end verification)

What Advances are Needed?

- Scalability
  - Through the right decompositions (art)
  - Practical ways to use human intuition to guide proof
  - Using machine learning to speed up analysis
- Usability
  - Incorporating formal methods into practical workflows
  - Enabling iterative processes
  - Increasing education

Supplementary & Alternative Approaches

Hybrid Systems

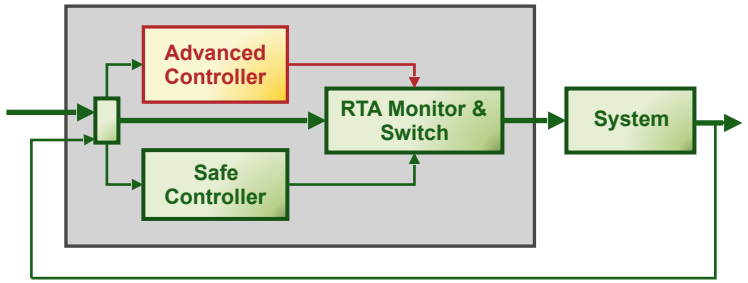
- Hybrid systems have both discrete logic & continuous dynamics
- "Autonomy in motion"
- Tools to check whether hybrid system can reach unsafe state from initial conditions[17]

Assurance Cases[18, 19]

- No "one size fits all" approach to TEV&V of autonomous systems
- Assurance cases: "Structured arguments supported by evidence that a system is safe in a given environment"
- Approach & tools to allow designers to build custom but rigorous TEV&V arguments

Runtime Assurance (RTA)[20]

- Some systems too complex to verify with sufficient confidence (formally or informally)
- Develop simpler but verifiable "safe controller" as fallback
- Requires "monitor" to know when to switch
- Caveats:
  - Interactions between RTA controllers can result in unexpected actions
  - Developing correct monitors can be challenging



Where our Group is Going

Unmanned Systems Autonomy Services (UxAS)[21]

- Service-oriented framework for mission-level autonomy for teams of unmanned air vehicles
- Use as case study for formal methods
- Re-coding & verifying services in SPARK (LLR)
- Want to formalize HLR & architecture, then develop workflow to automate parts of design
- Want to address cyber security concerns through crypto libraries, taint analysis with SPARK

Air & Space Collision Avoidance

- Formalize requirements for air & space collision avoidance
- Develop faster approaches for hybrid systems analysis
- Use a mix of formal methods & more traditional methods like modeling & simulation to design collision avoidance systems
- Consider cyber security concerns

Verified Algorithm Implementations

- Formalize requirements for core algorithms used in autonomy: branch & bound, search trees, etc.
- Develop verified algorithm implementations, similar to certified machine learning algorithm in [22]
- Automatically synthesize verified implementations from requirements as an extension to [15]

Assurance Cases

- Develop tools to graphically represent assurance cases & organize evidence electronically
- Apply to problems of interest

Successes of Formal Methods

Airbus A380 & A400M software modules[23]

- Determined WCET & max stack usage (required minimal expertise)
- Proved functional correctness (more expertise & manual effort)

Dassault-Aviation flight control software[7]

- Replaced software robustness testing with deductive verification & abstract interpretation
- Saved person-month of effort per software release

Microsoft[24]

- Developed Static Driver Verifier (SDV) tool to verify third-party drivers
- Verifying HTTPS ecosystem

Amazon Web Services[25,26,27]

- Developed cloud computing tools to
  - Verify user access control polices meet user security requirements
  - Determine if computers are reachable from outside world
- Automated continuous formal verification of s2n TLS library

DARPA High Assurance Cyber Military Systems (HACMS)[28]

- Proved cyber security guarantees with "safe" programming languages, formalized architecture & seL4
- Secured Boeing's Unmanned Little Bird (ULB) helicopter

seL4 microkernel[29]

- Proved security properties like integrity & confidentiality
- Verified functional correctness of kernel capabilities in source & object code

Collins Aerospace[30]

- Aircraft systems status display
  - Formalized requirements
  - Auto-generated source code from requirements
  - Verified compliance of source code with requirements
  - Saved 10-15 weeks of labor
- Crew alerting system
  - Auto-generated test cases for coverage from requirements

References

1. NIST, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, Planning Report 02-3, May 2002.
2. B.W. Boehm, *Software Engineering Economics*, Prentice Hall (1981).
3. J. Delange and P. H. Feiler, "Supporting the ARP4761 Safety Assessment Process with AADL," Software Engineering Institute, CMU.
4. D. Redman, "The System Architecture Virtual Integration (SAVI) Project," In *Safe & Secure Systems & Software Symposium*, 2010.
5. N. Kalra & S. M. Paddock, "Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?," RAND Technical Report RR-1478-RC, 2016.
6. RTCA Special Committee 205, *Formal Methods Supplement to DO-178C and DO-278A*, RTCA DO-333, 13 December 2011.
7. Y. Moy et al, "Testing or Formal Verification: DO-178C Alternatives and Industrial Experience," *IEEE Software* 30(3): 50-57, 2013.
8. J. W. McCormick & P. C. Chapin, *Building High Integrity Applications with SPARK*, Cambridge University Press, 2015.
9. AdaCore, "Introduction to SPARK," learn.adacore.com, 2018.
10. D. Cofer & S. Miller, "DO-333 Certification Case Studies," In *NASA Formal Methods Symposium*, 2014.
11. A. Fifarek, L. Wagner, E. Hoffman, B. Rodes, M. A. Aiello, J. Davis, "SpeAR v2.0: Formalized Past LTL Specification and Analysis of Requirements," in *NASA Formal Methods Symposium*, 2017.
12. Collins Aerospace, "Assume Guarantee Reasoning Environment," online, 2019.
13. J. Davis, L. Humphrey, D. Kingston, "When Human Intuition Fails: Using Formal Methods to Find an Error in the 'Proof' of a Multi-Agent Protocol," in *Computer Aided Verification*, 2019.
14. AdaCore, "QGen," online, 2019.
15. T. Elliot, M. Alshiekh, L. Humphrey, L. Pike, U. Topcu, "Salty–A Domain Specific Language for GR(1) Specifications and Designs," in IEEE Int. Conf. Robotics and Automation, 2019.
16. A. Murugesan et al, "From Requirements to Code: Model Based Development of a Medical Cyber Physical System," In *Int. Workshop Software Engineering in Health Care*, 2014.
17. S. Bak, P. Duggirala, "HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems" in *Int. Conf. Hybrid Systems: Computation and Control*, 2017.
18. R. Hawkins et al, "Assurance Cases and Prescriptive Software Safety Certification: A Comparative Study," *Safety Science* 59:55-71, 2013.
19. Collins Aerospace, "Resolute: An Assurance Case Language for Architecture Models," online, 2019.
20. J. D. Schierman et al, "Runtime Assurance Framework Development for Highly Adaptive Flight Control Systems," USAF Technical Report AFRL-RQ-WP-TR-2016-0001, 2016.
21. S. Rasmussen, D. Kingston, L. Humphrey, "A Brief Introduction to Unmanned Systems Autonomy Services (UxAS)," in *Int. Conf. Unmanned Aircraft Systems*, 2018.
22. A. Bagnall et al, "Brief Announcement: Certified Multiplicative Weights Update: Verified Learning without Regret," in *ACM Symposium on Principles of Distributed Computing*, 2017.
23. J. Souyris et al, "Formal Verification of Avionics Software Products," In *International Symposium on Formal Methods*, 2009.
24. T. Ball et al, "SLAM and Static Driver Verifier: Technology Transfer of Formal Methods inside Microsoft," Microsoft Technical Report MSR-TR-2004-08, 2004.
25. J. Backes et al, "Semantic-based Automated Reasoning for AWS Access Policies using SMT," In *Formal Methods in Computer Aided Design*, 2018.
26. C. Dodge and S. Quigg, "A simpler way to assess the network exposure of EC2 instances," AWS Security Blog, 2018.
27. A. Chudnov et al, "Continuous Formal Verification of Amazon s2n," In *International Conference on Computer Aided Verification*, 2018.
28. D. Cofer et al, "Security Mathematically-Assured Composition of Control Models," DARPA USAF Technical Report AFRL-RI-RS-TR-2017-176, 2017.
29. NICTA & General Dynamics, "Mathematically Verified Software Kernels: Raising the Bar for High Assurance Implementations," online, 2014.
30. L. Wagner, "*Automating Avionics Certification Activities using Formal Methods*," In *High Confidence Systems and Software*, 2019.

Publications

- J. Davis, L. Humphrey, D. Kingston, "When Human Intuition Fails: Using Formal Methods to Find an Error in the 'Proof' of a Multi-Agent Protocol," in *Computer Aided Verification*, 2019.
- T. Elliot, M. Alshiekh, L. Humphrey, L. Pike, U. Topcu, "Salty–A Domain Specific Language for GR(1) Specifications and Designs," in *IEEE Int. Conf. Robotics and Automation*, 2019.
- K. Hobbs, I. Perez, A. Fifarek, E.M. Feron, "Formal Specification and Verification of System States for Spacecraft Automatic Maneuvering," in *AIAA Information Systems-AIAA Infotech@Aerospace*, 2019.
- H. Tran, W. Xiang, S. Bak, T. Johnson, "Reachability Analysis for One Dimensional Linear Parabolic Equations," in *IFAC Conf. Analysis and Design of Hybrid Systems* 2018.
- S. Rasmussen, D. Kingston, L. Humphrey, "A Brief Introduction to Unmanned Systems Autonomy Services (UxAS)." in *Int. Conf. Unmanned Aircraft Systems*, 2018.
- P. Heidlauf, A. Collins, M. Bolender, S. Bak, "Verification Challenges in F-16 Ground Collision Avoidance and Other Automated Maneuvers," in *Int. Workshop Applied Verification for Continuous and Hybrid Systems*, 2018.
- K. Hobbs, P. Heidlauf, A. Collins, S. Bak, "Space Debris Collision Detection using Reachability," in *5th Int. Workshop Applied Verification for Continuous and Hybrid Systems*, 2018.
- L. V. Nguyen, K. A. Hoque, S. Bak, S. Drager, T. Johnson, "Cyber-physical Specification Mismatches," *ACM Trans. on Cyber-Physical Systems* 2(4):23, 2018.
- Y. Wang, L. R. Humphrey, Z. Liao, H. Zheng, "Trust-Based Multi-Robot Symbolic Motion Planning with a Human-in-the-Loop," *ACM Transactions on Interactive Intelligent Systems* 8(4):31, 2018.
- B. Könighofer, M. Alshiekh, R. Bloem, L. Humphrey, R. Könighofer, U. Topcu, C. Wang, "Shield Synthesis," *Formal Methods in System Design*, 51:450–462, 2017.
- S. Bak, O. A. Beg, S. Bogomolov, T. Johnson, L. V. Nguyen, C. Schilling, "Hybrid Automata: From Verification to Implementation," *Int. J. Software Tools for Technology Transfer*, 2017.
- K. Gross, M. Clark, J. Hoffman, E. Swenson, A. Fifarek, "Run-Time Assurance and Formal Methods Analysis Nonlinear System Applied to Nonlinear System Control," *AIAA J. Aerospace Information Systems* 14(4), 2017.
- A. Fifarek, L. Wagner, E. Hoffman, B. Rodes, M. A. Aiello, J. Davis, "SpeAR v2.0: Formalized Past LTL Specification and Analysis of Requirements," in *NASA Formal Methods Symp.*, 2017.
- S. Bak, S. Bogomolov, M. Althoff, "Time-Triggered Conversion of Guards for Reachability Analysis of Hybrid Automata," in *Int. Conf. Formal Modelling and Analysis of Timed Systems*, 2017.
- S. Bak, P. Duggirala, "Rigorous Simulation-Based Analysis of Linear Hybrid Systems," in *Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems*, 2017.
- S. Bak, P. Duggirala, "Simulation-Equivalent Reachability of Large Linear Systems with Inputs," in *Int. Conf. Computer-Aided Verification*, 2017.
- S. Bak, P. Duggirala, "HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems" in *Int. Conf. Hybrid Systems: Computation and Control*, 2017.
- C. Rothwell, M. Patzek, L. Humphrey, "Verifiable Task Assignment and Scheduling Controller," 711HPW/AFRL/USAF, Tech. Rep. AFRL-RH-WP-TR-2017-0045, 2017.
- S. Bak, S. Bogomolov, T. Henzinger, A. Kumar "Challenges and Tool Implementation of Hybrid Rapidly-Exploring Random Trees," in *Int. Workshop Numerical Software Verification*, 2017.
- S. Bak, P. Duggirala, "Direct Verification of Linear Systems with over 10000 Dimensions," in *Int. Workshop Applied Verification for Continuous and Hybrid Systems*, 2017.
- K. H. Gross, "Formal Specification and Analysis Approaches for Spacecraft Attitude Control Requirements," in *IEEE Aerospace Conf.*, 2017.
- L. Feng, C. Wiltsche, L. Humphrey, U. Topcu, "Synthesis of Human-in-the-Loop Control Protocols for Autonomous Systems," *IEEE Trans. Automation Science and Engineering* 13(2): 450–462, 2016.
- L. Humphrey, B. Könighofer, U. Topcu, "Synthesis of Admissible Shields," in *Haifa Verification Conf.*, 2016.
- D. Kingston, S. Rasmussen, L. Humphrey, "Automated UAV Tasks for Search and Surveillance," *in IEEE Multi-Conf. Systems and Controls*, 2016.
- T. Johnson, S. Bak, M. Caccamo, L. Sha, "Real-Time Reachability for Verified Simplex Design," *ACM Trans. Embedded Computing Systems* 15(2), 2016.
- V. Horan, S. Adachi, S.Bak, "A Comparison of Approaches for Finding Minimum Identifying Codes on Graphs", *Quantum Information Processing*, 2016.
- D. A. Spencer, Y. Wang & L. Humphrey, "Trust-based human-robot interaction for multi-robot symbolic motion planning," in *IEEE/RSJ Inter. Conf. Intelligent Robots and Systems*, 2016.
- Gross, K. H., "Evaluation of Verification Approaches Applied to Nonlinear System Control," Master's Thesis, Department of Aeronautics and Astronautics, Air Force Institute of Technology, March 2016.
- L. Feng, L. Humphrey, I. Lee, U. Topcu, "Human-Interpretable Diagnostic Information for Robotic Planning Systems," in *IEEE/RSJ Inter. Conf. Intelligent Robots and Systems*, 2016.
- T. Apker, B. Johnson, L. Humphrey, "LTL Templates for Play-Calling Supervisory Control," in *AIAA SciTech*, 2016.

Publications

- S.Bak, S. Chaki, "Verifying Cyber-Physical Systems by Combining Software Model Checking with Hybrid Systems Reachability", In *ACM SIGBED Int. Conf. Embedded Software*, 2016.
- F. Abdi, R. Mancuso, S. Bak, O. Dantsker, M. Caccamo, "Reset-Based Recovery for Real-Time Cyber-Physical Systems with Temporal Safety Constraints", in *IEEE Int. Conf. Emerging Technology & Factory Automation*, 2016
- S. Bak, S. Bogomolov, C. Schiling, "High-level Hybrid Systems Analysis with Hypy", in *Applied Verification for Continuous and Hybrid Systems*, 2016
- S. Bak, S. Bogomolov, T. Henzinger, T. Johnson, P. Prakash, "Scalable Static Hybridization Methods for Analysis of Nonlinear Systems", in Int. Conf. Hybrid Systems: Computation and Control, 2016.
- K. H. Gross, M. Clark, J. Hoffman, A. Fifarek, K. Rattan, E. Swenson, M. Whalen, L. Wagner, "Formally Verified Run Time Assurance Architecture of a 6U CubeSat Attitude Control Subsystem," in *AIAA SciTech*, 2016.
- K. H. Gross, R. Patrick, E. Swenson, J. Agte, "Optimal Attitude Control of a 27U CubeSat with a Four-Wheel Pyramid Reaction Wheel Array and Magnetic Torque Coils," in *AIAA SciTech*, 2016.
- K. H. Gross, J. Hoffman, A. Fifarek, "Incremental Formal Methods Based Design Approach Demonstrated on a Coupled Tanks Control System," in *IEEE High Assuance Systems Engineering*, 2016.
- S. Regisford, B. Hulbert, A. Fifarek, "Compositional Architecture Design for Fuel Tank Thermal Systems" in *IEEE High Assurance Systems Engineering*, 2016.
- A. Burkhard, M. Clark, "Applicability of MIL-HDBK-516B to Certifying Autonomous Decision Making Air Vehicle Systems," J. Systems Safety, 2015.
- K. Gross, J. Hoffman, M. Clark, E. Swenson, R. Cobb, M. Whalen, L. Wagner, "Application and Evaluation of Formal Methods Applied to a 6U CubeSat Attitude Control System," AIAA Space, 2015.
- L. Feng, C. Wiltsche, L. Humphrey, U. Topcu, "Controller Synthesis for Autonomous Systems Interacting with Human Operators," in ACM/IEEE Int. Conf. Cyber-Physical Systems, 2015.
- K. Rattan, M. Clark, J. Hoffman, "Design and Analysis of a Multistage Fuzzy PID Controller," in American Control Conference, 2015.
- X. Zhang, M. Clark, K. Rattan, J. Muse, "Controller Verification in Adaptive Learning Systems Towards Trusted Autonomy," in ACM/IEEE Int. Conf. Cyber-Physical Systems, 2015.
- M. Clark, K. Rattan, "Hybrid Representation of Rule-Based Systems," in Int. Conf. Hybrid Systems: Computation and Control, 2015.
- A. Fifarek, L. Wagner, "Formal Requirements of a Simple Turbofan Using the SpeAR Framework," in Int. Symposium on Air Breathing Engines, 2015.
- Assistant Secretary of Defense / Research and Engineering (ASD/R&E), Autonomy Community of Interest (COI), "DoD Autonomy Test and Evaluation, Verification and Validation (TEVV) Investment Strategy", Alexandria, VA, Jun. 2015.
- L. Humphrey, E. Wolff, U. Topcu, "Formal Specification and Synthesis of Mission Plans for Unmanned Aerial Vehicles," in *AAAI Spring Symposium*, 2014.
- B. U. Kim, L. Humphrey, "Satisfiability Checking of LTL Specifications for Verifiable UAV Mission Planning," in *AIAA Aerospace Sciences Meeting*, 2014.
- L. Humphrey, "Model Checking for Verification in UAV Cooperative Control Applications," in Recent Advances in Research on Unmanned Aerial Vehicles, ser. LNCS, F. Fahroo, L. Wang, and G. Yin, Eds. Springer-Verlag, 2013, vol. 444, pp. 69–117.
- L. Humphrey, "Verification of a Leader Election Protocol for Autonomous Intruder Detection," in *AIAA Infotech@Aerospace*, 2012.