



Alwyn Goodloe
NASA Langley Research Center



Overview

- Background
- Avionics and certification
- Runtime verification
- Aircraft self-separation
- Formal-Methods for numerical software
- Edge aircraft
- Software change management



NASA R&D in Formal Methods

- NASA Langley Research Center (LARC) - Safety Critical Avionics Systems Branch
- NASA Ames Research Center (ARC)– Robust Software Engineering Group
- Jet Propulsion Laboratory (JPL/FFRDC) – Laboratory for Reliable Software
- NASA Dryden is beginning to explore formal approaches
- NASA Marshall Spaceflight Center, NASA Kennedy Spaceflight Center, and NASA Johnson Spaceflight Center have efforts applying model checking to small projects



NASA Langley

- LaRC created in 1917 as the first National Advisory Committee for Aeronautics (NACA) research facility
 - Located in Hampton, Virginia
 - Between Norfolk and Colonial Williamsburg
- LaRC became a NASA lab in 1958
 - The Mercury program began at LaRC
- Research areas of focus: Aeronautics, Atmospheric Sciences, and Exploration
- I belong to Safety-Critical Avionics Systems Branch
 - Research focus is targeted to safety
 - We do much of our research in-house
 - With support from contractors and Universities



Organization/Mission Perspective



Research Directorate
(Jill Marlowe)

Safety Critical Avionics
Systems Branch
(Ray Calloway)

New methodologies and tools for designing, verifying, validating, and assuring high confidence software-intensive systems to improve safety, reliability, and capacity in mission- or life-critical aerospace systems

NASA Missions



- Aviation Safety Program
- Airspace Systems Program
- Integrated Systems Research Program



- Exploration Systems Development

Reimbursable

- Aircraft Certification Service





Branch Projects

- **Vehicle Systems Safety Technology**
 - Improved Crew Decision-Making and Response in Complex Situations
 - Assure Effectiveness through Comprehensive Test and Evaluation
- **NextGen Airspace - Concepts and Technology Development Project**
 - Criteria-based Policy for Conflict Detection and Resolution
- **UAS Integration in the NAS**
 - Avionics certification
- **System-Wide Safety Assurance Technologies**
 - Assurance of Flight Critical Systems
 - Discovery of Precursors to Safety Incidents
 - Prognostic Algorithm Design for Safety Assurance
 - Assuring Safe Human – Systems Interaction
- **Atmospheric Environment Safety Technologies**
 - EM Computations
- **FAA Interagency Agreement on V&V for Complex Systems**
 - Regulatory Considerations for Adaptive Systems
 - Assurance Case Applicability to Digital Systems
- **FAA Interagency Agreement on Closely Spaced Parallel Operations (CSPO)**
 - Simplified Aircraft-based Paired Approach (SAPA) - Validate and produce a preliminary safety analysis for an alerting algorithm and a concept of operations that enables coupled approaches in low visibility conditions
- **Center Innovation Fund**
 - Achieving Mass Self-Synchrony In GPS Denied Environment



Our In-House Work

- Developing and verifying algorithms, designs, and procedures
 - State-based conflict detection & resolution (CD&R)
 - Parallel Runway Alerting algorithm
 - Strategic CD&R (Stratway)
 - Distributed Systems (Merging/Spacing Terminal Area)
 - Priority Rules for Autonomous Flight Rules (AFR)
 - Formally verified implicit coordination (self-separation)
- Advanced tools for formal verification
 - Verification of floating point software (automated error analysis)
 - Mathematical libraries for PVS theorem prover (Hypatheon)
 - Auto Generation of Verified Code (Java, C)
 - Enhanced Unit Testing through symbolic execution
 - Runtime verification of Edge Aircraft Software
- Advanced methods in automated deduction
 - Fully automated global optimization
 - (provably correct, Bernstein polynomials)



Relation to the Year of Proof

- As you can see already most of what we do involves evidence for assurance
- The FAA set criteria for safety
 - Test, formal methods, ...
 - Our research aids this
- How do we construct a safety assurance argument
 - Some focus on mathematical proof
 - Some on experimental evidence
 - Some on the broader notion of safety and assurance
 - Civil aviation is very safe
 - Technical advances must make it more so to be accepted
 - How to prove



So You Want to Build an Airplane

- Form a startup and start hacking - just like silicon valley right?
 - Not so fast!
- Process starts off with a notification of intent to the FAA
 - A minuet begins between the company and the regulators
 - For a Part 25 aircraft they will tell you over 1500 safety criteria you must meet
 - Autos and medical devices are easy in comparison
 - DoD aircraft not subject to these regulations
- The FAA must certify the aircraft
 - Designated Engineering Representative (DER)
- The cyber-physical component is one of the largest risk factors
- You can choose to do things your own way and make an argument to the FAA that the aircraft is safe OR you can follow approved guidelines
 - Very process oriented



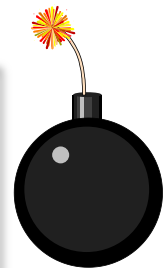
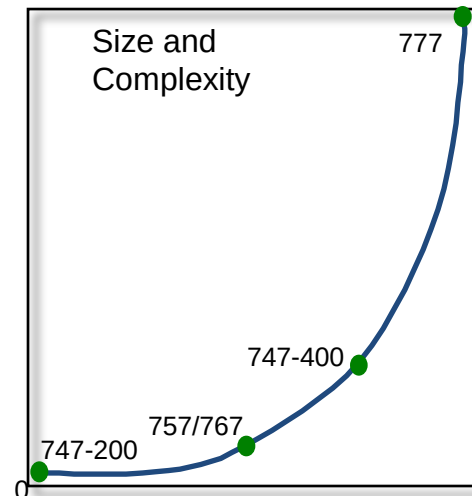
Ultra-Reliability is Hard

We are very good at building complex software systems that work **95% of the time**---but, we do not know how to build complex software systems that are ultra-reliably safe.

What, then has saved us in the past?

- minimal amount of software that is safety-critical
- simple designs
- enormously expensive verification and certification processes
- backups that are not software, e.g.
 - hardware interlocks
 - human intervention

All sectors of aerospace are increasingly relying on software to perform safety-critical functions



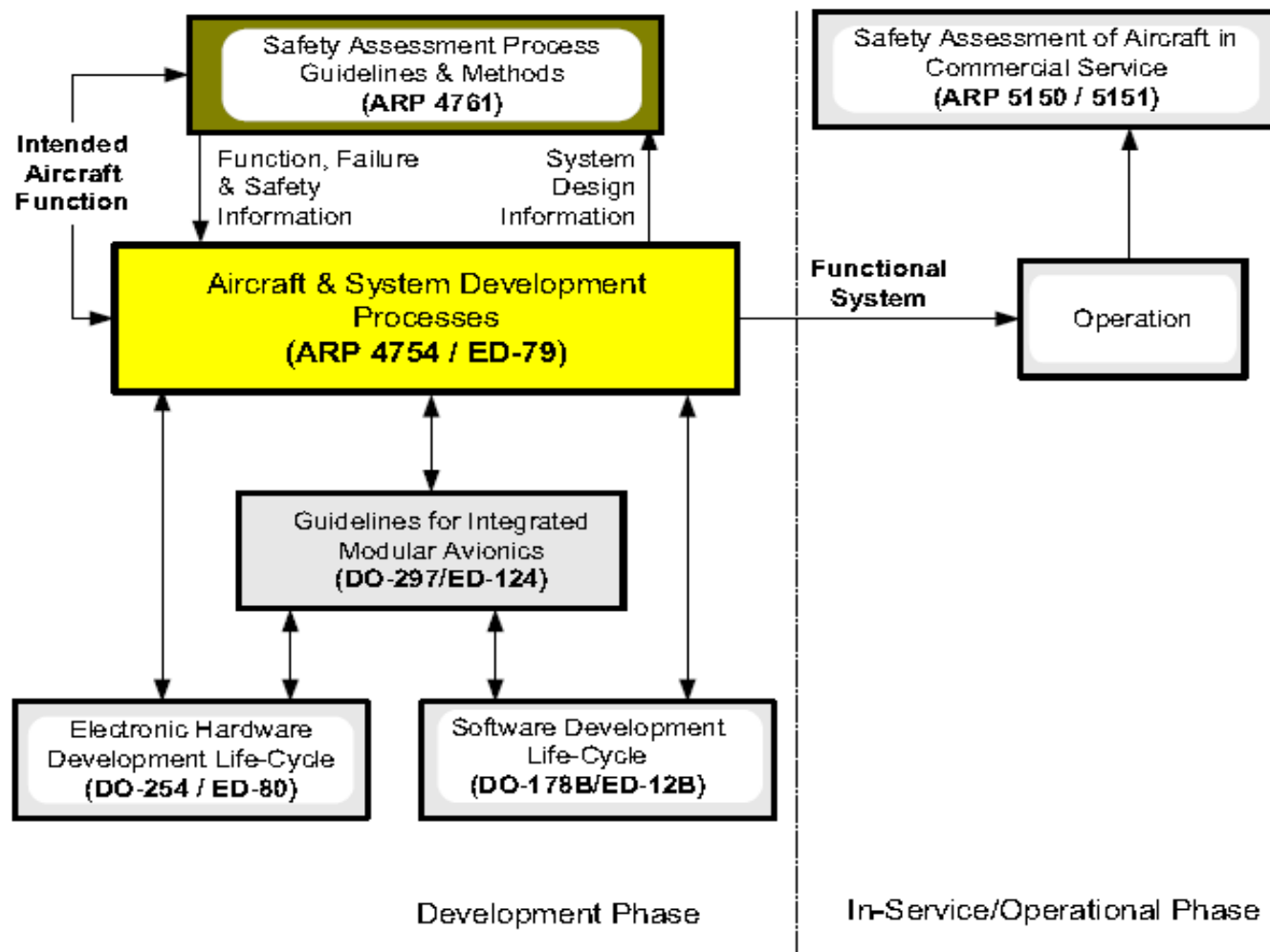


Eliminating Common Mode Errors

- Independence – A concept to minimize the likelihood of common mode and cascade errors
- Diversity
 - HW, SW,
- Redundancy
 - Triple redundancy
 - Com/Mon
- Can mix techniques
 - Dissimilar com/mon



Guideline Documents





ARP 4761

- Aerospace Recommended Practice for performing safety assessments on civil aircraft
- Guidelines and methods of performing the safety assessment
- Functional Hazard Analysis (FHA)
- Preliminary Safety Assessment (PSA)
- System Safety Assessment (SSA)



ARP 4761 Contd.

- Safety assessment process
- Safety assessment overview
- Detailed method guidelines
 - Functional Hazard Assessment (FHA)
 - Fault-Tree Analysis (FTA)
 - Failure Modes and Effects Analysis (FMEA)
 - Common Mode Analysis (CMA)
 - Zonal Safety Analysis (ZSA)



Functional Hazard Analysis

- Identifies and classifies the failure conditions associated with the aircraft functions and combinations of aircraft functions
 - Classification Levels: Minor (D), Major (C), Hazardous (B), Catastrophic (A)
 - Classifications establish safety objectives
 - Output starting point for generation and allocation of safety requirements



Preliminary System Safety Assessment

- Systematic examination of proposed system architecture
 - Used to complete the failure conditions list and the safety requirements
- Identify how failures lead to the hazards identified in FHA
 - Suggested analysis techniques such as FTA
- How FHA requirements can be met
 - Identify protective strategies
 - partitioning, dissimilarity, etc.



System Safety Assessment

- Systematic examination of system, architecture, and installation to show compliance with safety requirements
- A SSA done for each PSSA
- Verification that the design requirements established at system level
- Verification that safety requirements derived from requirements are met
- Verification design requirements in CCA met
- Linkage system level SSA to aircraft level FHA



4754A

- Input is the function, failure, and safety info from 4761
- Iterative process as design is refined and the analysis process prescribed by 4761 is repeated
- Functional Design Assurance Levels (FDAL) assigned
- FDAL assigned to systems from aircraft architecture based on Preliminary Aircraft Safety Assessment (PASA)
- Item Design Assurance Level (IDAL) done in refinement



4754A Contd.

- FDAL considers functional independence of aircraft/system functions
- IDAL considers design independence of items
- Assertion of independence must be substantiated
 - Verify no common mode introduced
- IDALs are assigned to items then fed back to analysis
- During allocation of top-level function into two or more independent sub-functions
 - One sub-function cannot itself cause top-level hazard



Independence Can Be Your Friend

- Architectural strategies incorporating independence, redundancy, and dissimilarity can be a powerful means of reducing the potential for errors in requirements or in design implementation
- The people writing the standards have built these architectures
 - They do it for Boeing, Airbus, etc.
- The justifications and arguments for safety are found in certification documents
- Engineers and certification bodies lack guidelines and examples



Yet

- The effectiveness of particular architectural strategies, introduced to allow the allocation of lower item risk level, generally cannot be quantified
 - John Downer's work redundancy in engineering
- As a consequence, the justification to support such allocation necessarily involves some degree of engineering judgment by the applicant and the certification authorities
- Do existing architectural patterns and arguments work on newer more complex systems

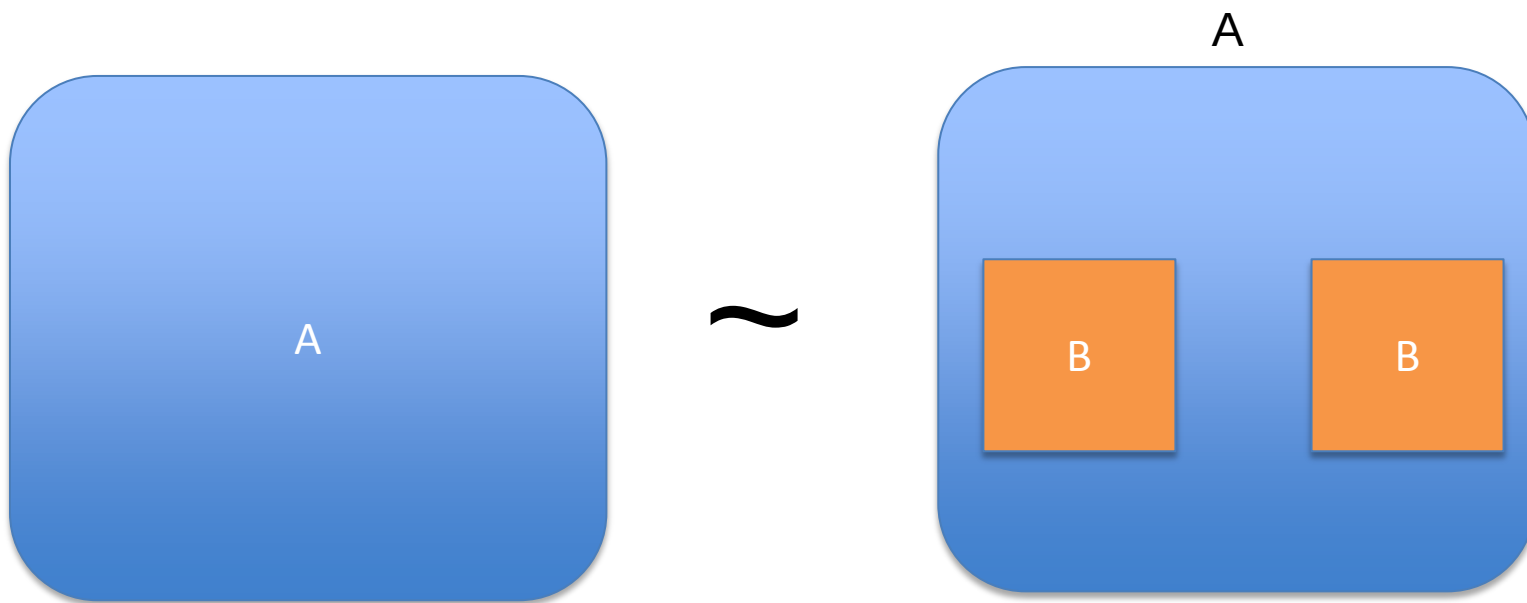


Challenges

- Over many decades Langley built up the NACA catalogue of airfoils
 - If you build a wing, go to the catalogue and find out it's properties
- Is there an analogue to the airfoil catalog?
- Can we help designers of avionics and just as importantly certification agents
 - We want to take the first steps in this area
 - Won't be as complete
- Short term we hope to do one or two case studies



How to Show





Much Work TBD

- Safety Cases
- Formal Methods
 - Fault Models
 - Formalisms
 - Proofs
- Experiments on an avionics testbed
 - Last summer recent Penn graduate Kevin Conley led the effort to stand the testbed
 - There is a lot of work for students here

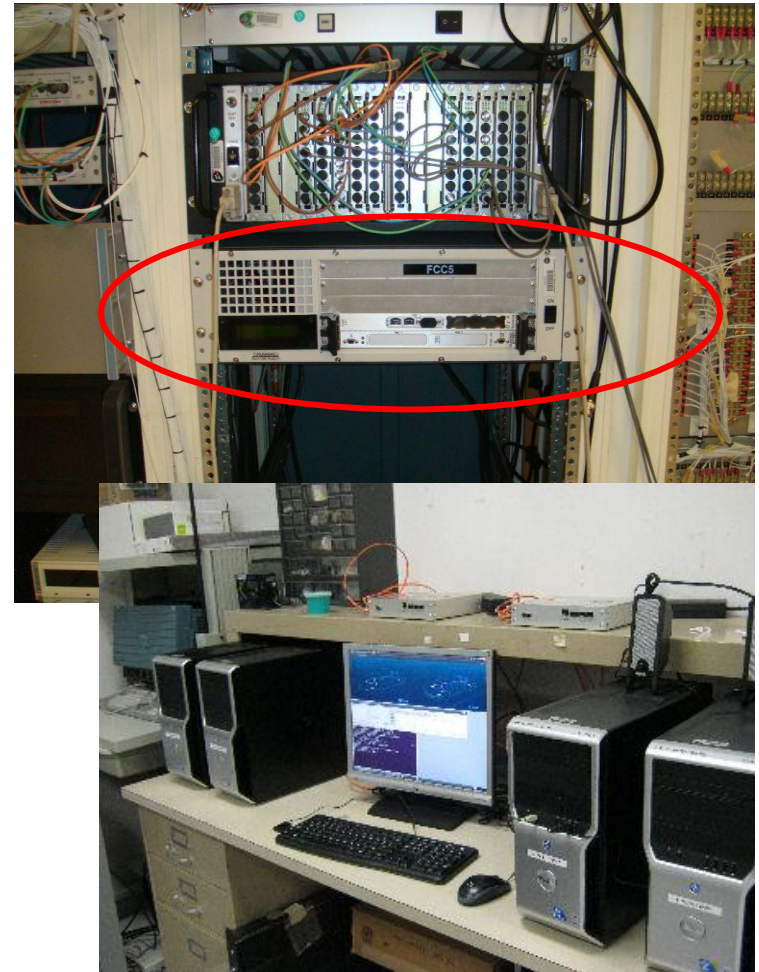


Digital Avionics Platform

Problem: Lack of a state-of-the-art avionics testbed providing researchers the means to evaluate concepts and algorithms designed to prevent aircraft LOC and improve the performance and reliability of flight-critical systems

Objective: Develop a representative avionics platform supporting V&V of supervisory and health monitoring technologies for complex civil aviation systems.

This platform, called the Assessment Environment (AE), consists of embedded computers running an ARINC 653 compliant, multi-tasking, real-time operating system, communicating over a modern avionics bus

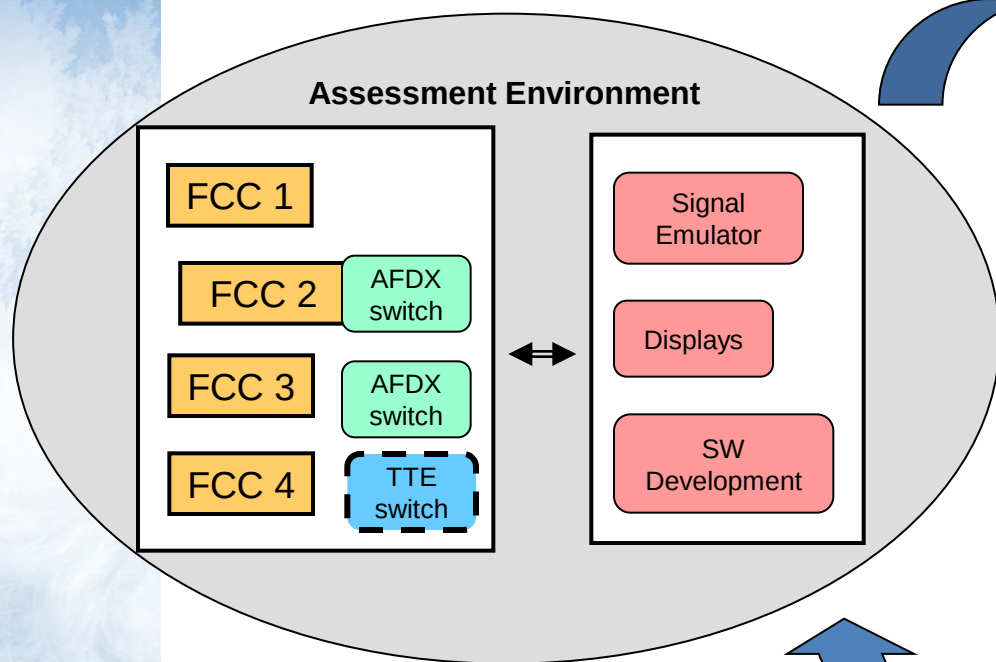




ARINC-653 Assessment Environment

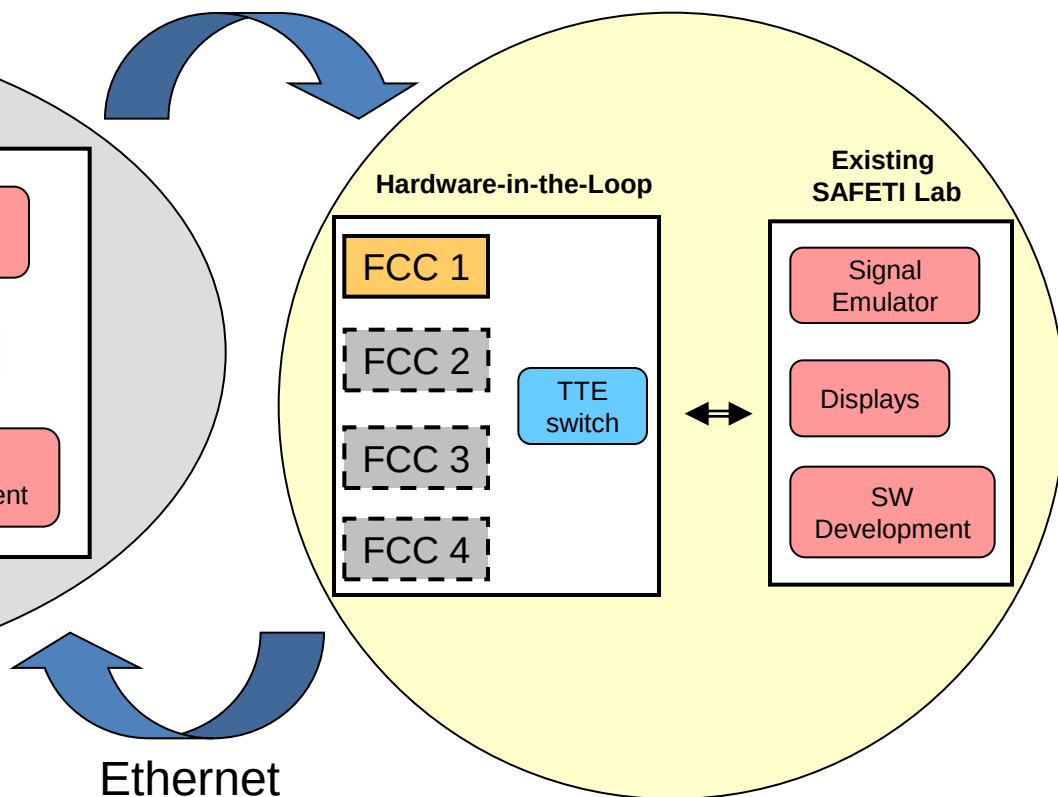
Dryden

Quadruplex FCC avionics target
for V&V methods evaluation



LaRC

Integrate w/ existing Avionics
Test & Eval Platform





Safety Cases

- SCASB has an ongoing research effort researching safety cases
 - Collaborators in the UK, Germany, ...
- Requires a broad a approach to problem solving
 - Safety is almost never just a technical problem
- Engineering students who focus on this during their internship are forced to reason about systems in a way that they usually do not have to in coursework



Runtime Verification in Copilot

- Copilot is a runtime verification framework aimed at hard real-time systems
 - Co-developed by Lee Pike at Galois and myself
- Written as a Haskell EDSL that compiles to C99
- Although we have completed many exciting and fun experiments more work remains
- We now have a real avionics testbed
- Copilot has been ported and some very simple experiments done



Sample Copilot Specification

Haskell

```
fib :: [Word32]
```

```
fib = [0,1] ++ zipWith (+) (drop 1 fib)
```

Copilot

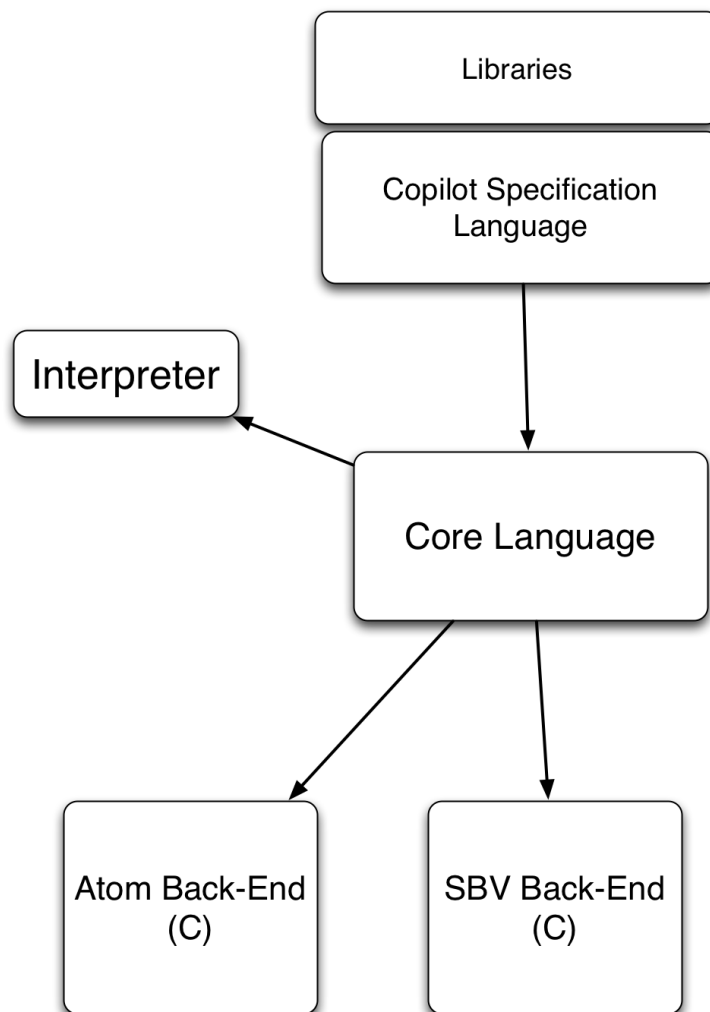
```
fib :: Stream Word32
```

```
fib = [0,1] ++ (fib + drop 1 fib)
```

Special constructs for input (sampling) and output (triggers)



Copilot Architecture





Copilot Experiments I

- Do you like functional programming
- Do you know something about RTOS scheduling
 - Rate-monotonic scheduling, etc.
- Would you want to get hands-on experience with the sort of architectures used on real aircraft
- Well I have a job for you 😊



Copilot Experiments II

Tasks

- We are continuing our investigations using the Edge aircraft
- An ever growing list of properties to monitor

Edge

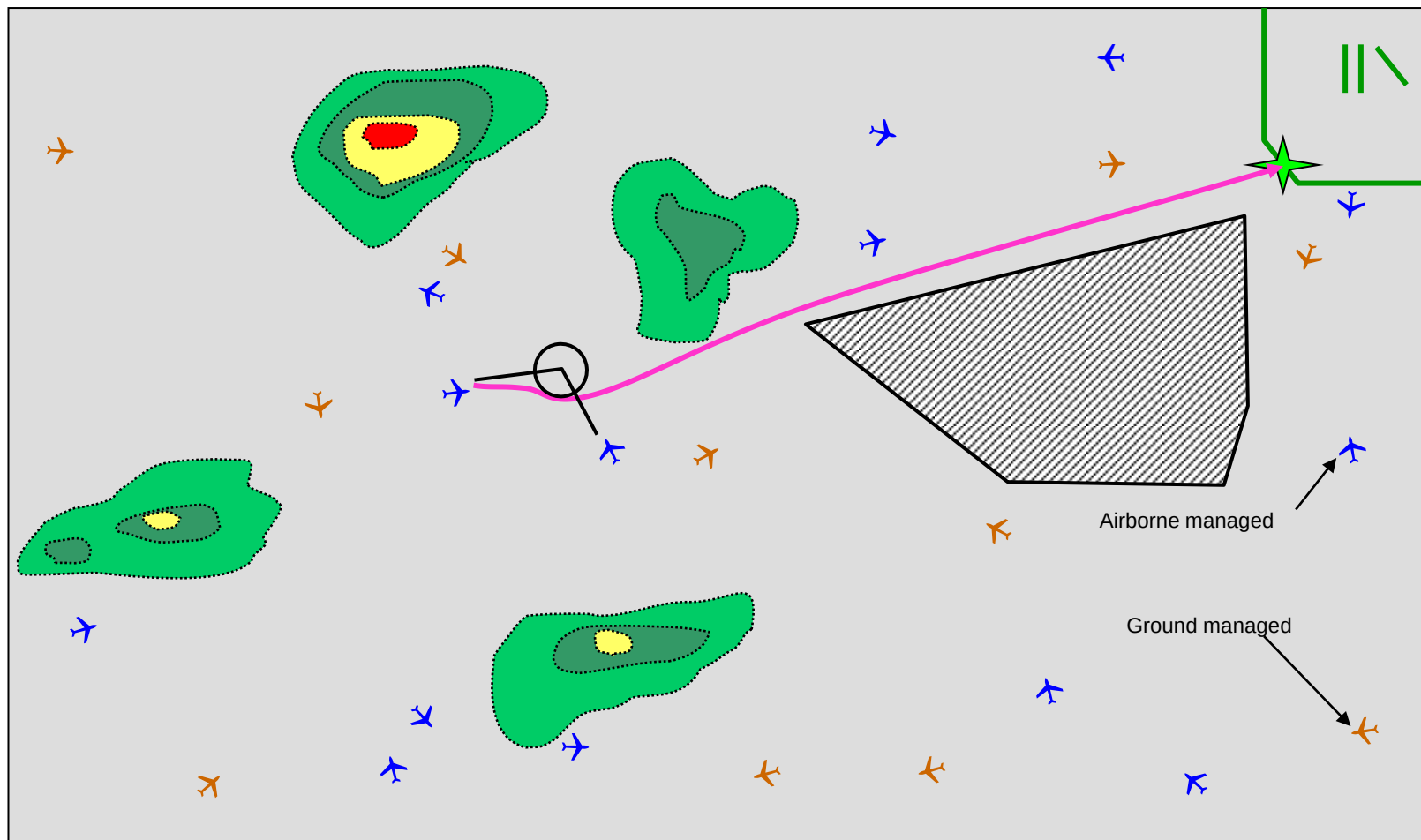




Aircraft Separation

- As part of congressional mandate, as a Joint Planning Development Office (JPDO) member, NASA is responsible for looking at futuristic ATM concepts
 - Including the lead on UAS in the NAS
- NASA is investigating a variety of air traffic management concepts to look at increasing capacity, efficiency, flexibility, etc.
- More controllers will not be able to achieve big gains in these parameters
 - Everything that NASA is looking at has a significant role for automation
- More automation doesn't remove safety issues, but simply shifts the risk from people to automation
- NASA is interested in new ways to analyze the safety of air traffic automation
- Enabled by Automatic Dependent Surveillance Broadcast (ADS-B)
- Warning: Civil aviation and military aviation do not treat these issues the same way

Self Separation Concept



Separation and Automation

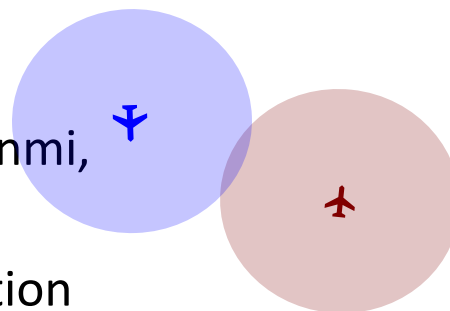
- Collision

- Scrape paint
- Avoid through pilot, controller, and TCAS



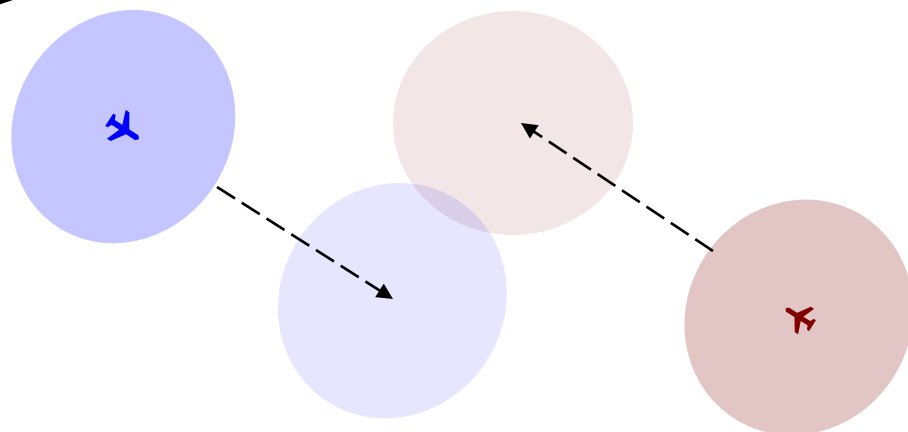
- Loss of Separation

- Separation standards are violated (5nmi, 1000ft)
- Avoid through human and/or automation decisions



- Conflict

- Predicted loss of separation

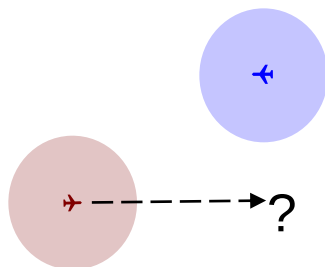




Separation Algorithms

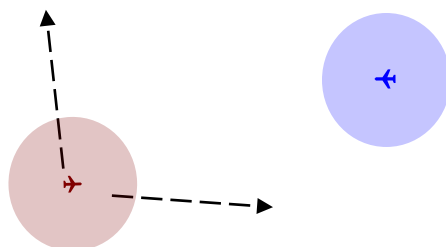
Conflict Detection

- Detect future loss of separation



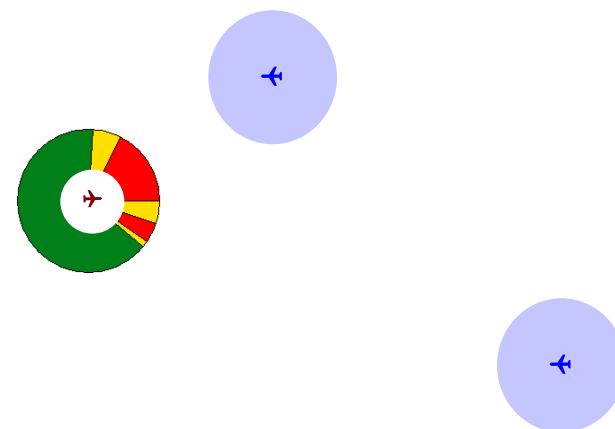
Conflict Resolution

- Suggest maneuvers to avoid a conflict



Conflict Prevention

- Provide conflict-free maneuvers

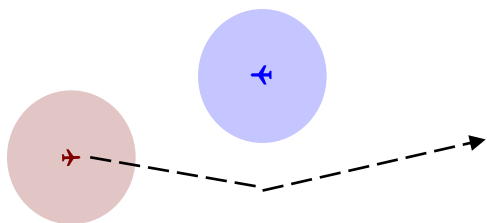




Recovery Algorithms

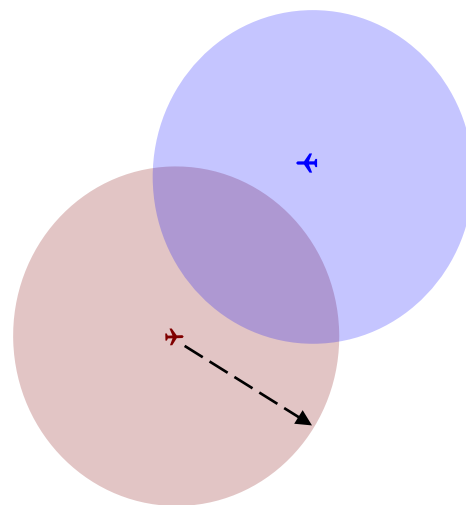
Conflict Recovery

- Suggest maneuvers to regain desired path



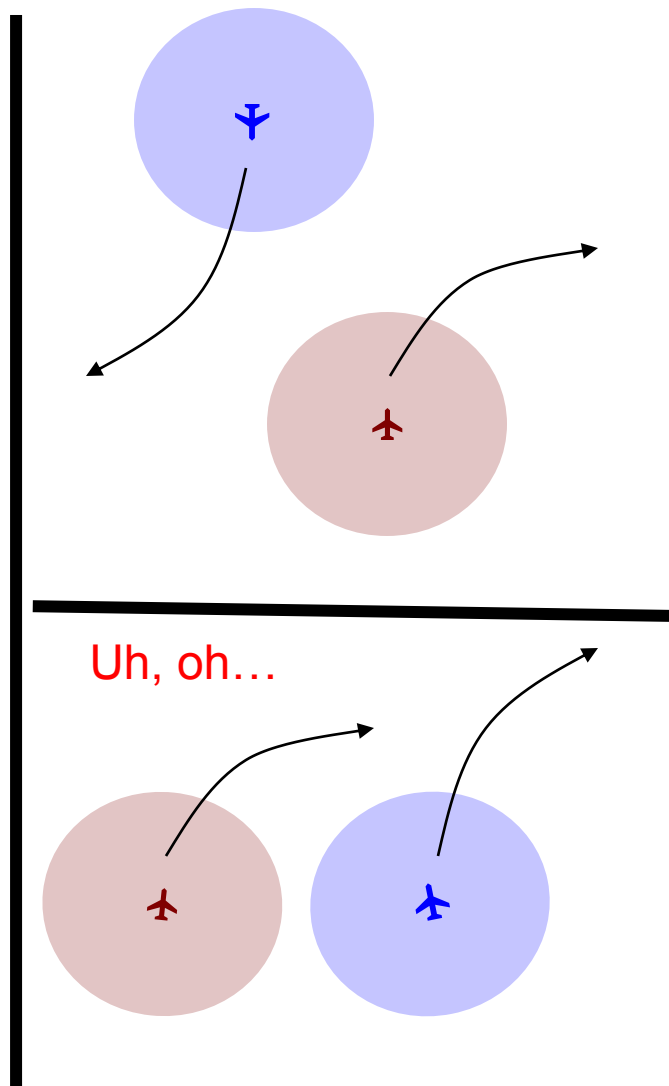
Loss of Separation Recovery

- For a variety of reasons separation may be lost
- Suggest a maneuver to regain separation



Conflict Resolution

- Each aircraft determines its own set of six maneuvers
 - Go right/left, Speed up/slow down, Go up/down
- Properties
 - Independence: free of conflicts if one aircraft maneuvers
 - Coordination: free of conflicts if both aircraft maneuver
- Requirements
 - No specific comm between aircraft
 - No unfair rules: lower aircraft ID goes first, etc.





Formal Statement of Properties

independent: THEOREM

precondition_ind?(s(a), s(b), v(a), v(b)) AND
(nva = cr3d_vertical_speed(a,b) OR
nva = cr3d_ground_speed(a,b) OR
nva = cr3d_heading(a,b)) AND

IMPLIES

NOT conflict?(s(a), s(b), nva-v(b))

coordinated: THEOREM

precondition_coord?(s(a), s(b), v(a), v(b)) AND
(nva = cr3d_vertical_speed(a,b) OR
nva = cr3d_ground_speed(a,b) OR
nva = cr3d_heading(a,b)) AND
(nvb = cr3d_vertical_speed(b,a) OR
nvb = cr3d_ground_speed(b,a) OR
nvb = cr3d_heading(b,a))

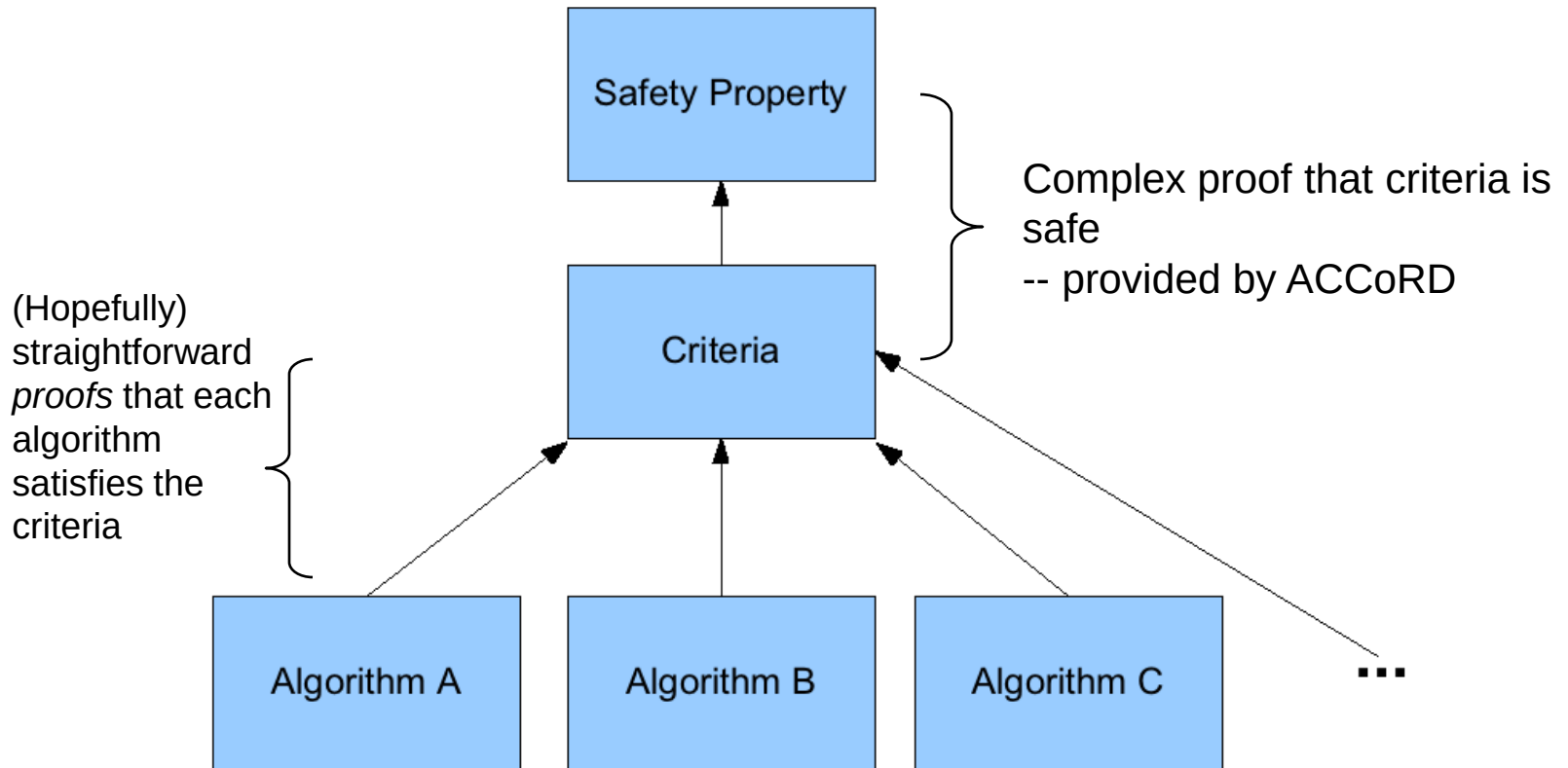
IMPLIES

NOT conflict?(s(a), s(b), nva-nvb)



ACCoRD Framework

Solution: ACCoRD – a verification framework for *classes* of separation algorithms





Criteria is Very General

- The criteria was developed to aid the verification process
- Criteria allows combinations of ground speed and vertical speed
- But even more, if different algorithms satisfy the criteria, then they will be coordinated with each other
- Self-separation does not rely on everyone running the same algorithm!



From Models to Programs

- Our models and proofs on aircraft separation are carried out on real numbers in PVS
- Computers running on aircraft do not execute exact real arithmetic!
- Recent effort to focus on these issues
 - A different perspective than our scientific computing and numerical analysis experts
 - Collaborators at CEA and INRIA



Frama-C Specification + Code

```
/*@ logic real tauR(real ux, real uy, real vx, real vy, real b, real t) =  
  @  dmin(dmax(b*sqvR(vx,vy), - dotR(ux,uy, vx,vy)), t*sqvR(vx, vy)) ; @*/  
/*@ requires -185200. <= s_x <= 185200. && -185200. <= s_y <= 185200. &&  
  @      -308. <= v_x <= 308. && -308. <= v_y <= 308. ;  
  @ ensures \abs(\result - tauR(s_x, s_y, v_x, v_y, Br, Tr)) <= 0x1.p-8; @*/
```

```
double tau_vv(double s_x, double s_y, double v_x, double v_y)  
{ return min(max((Br-epb)*sqv(v_x,v_y),  
  - dot(s_x,s_y, v_x, v_y)), (Tr+ept)*sqv(v_x, v_y));}
```



Non-Linear Arithmetic

- Heart Dipole Problem:
- $P(x_1, \dots, x_8) = -x_1x_6^3 + 3x_1x_6x_7^2 - x_3x_7^3 + 3x_3x_7x_6^2 - x_2x_5^3 + 3x_2x_5x_8^2 - x_4x_8^3 + 3x_4x_8x_5^2 - 0.9563453$
- $x_1 \in [-0.1, 0.4], x_2 \in [0.4, 1], x_3 \in [-0.7, -0.4], x_4 \in [-0.7, -0.4], x_5 \in [0.1, 0.2], x_6 \in [-0.1, 0.2], x_7 \in [-0.3, 1.1], x_8 \in [-1.1, -0.3]$
- Theorem: $\forall x: p(x_1, \dots, x_8) \geq -1.7435$
- Theorem: $\exists x: p(x_1, \dots, x_8) \leq -1.7434$



Bernstein

- A formal library in PVS for reasoning about (multivariate polynomials)
- Based on Bernstein polynomials
- Numeric constants are operated on using infinite-precision rational arithmetic
 - All results produced are free from numerical representation errors
 - Bernstein's results carry the weight of rigorously proved mathematical theorems.
- Proof strategies in PVS for automatically solving inequalities
- User friendly tools for formally solving global optimization



Kodiak

- A C++ library that implements Bernstein polynomial using an infinite precision arithmetic library (GiNaC/GMP)
- Intended for use in SMT solvers
- Last summer we had an intern from University of Nebraska that integrated this with SRI's YICES
- Opportunities exist in this area this summer



Mixed Quantification

Standard numerical methods can be used to prove

$$\forall x \in [0,1]: x(1-x) \leq 1/4$$

We can even do this in the PVS prover

However few tools can prove similar problems with multiple quantifiers

$$\forall x \in [0,1]: \exists y \in [0,1]: -x^{10} + x^3 \neq -y^{10} + y^3$$

Numerical and interval arithmetic can be extended to solve these problems, but difficult to see how to do so in a theorem prover



Focus on Numerical Software

- In the theorem prover PVS we shall investigate
 - Numerical integration
 - Interval arithmetic with floating point numbers
 - Symbolic evaluation
- Target great circle code and similar libraries
 - The algorithms have been used for decades and in some case centuries
 - Are our implementations correct and robust?
 - Mine areas such as computational geometry
- Investigate verified floating point error mitigation strategies

Edge 540T UAS Test Bed



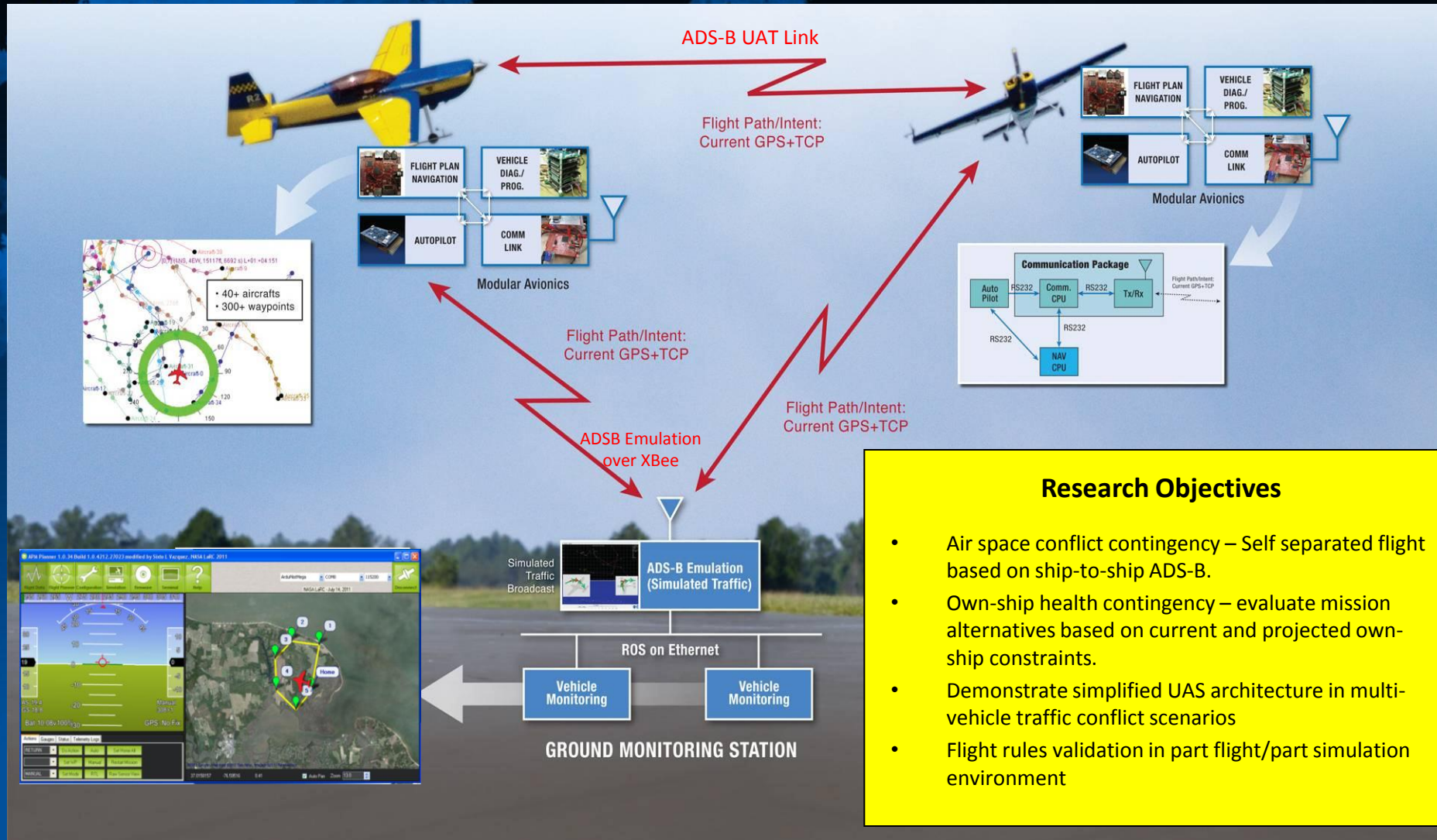
- Initial development funded by ARMD/IVHM now ARMD/SSAT
 - Battery health prognosis
 - Software health
 - Autonomous decision system algorithms
- Current Capability:
 - Autopilot – flight plan following
 - Battery prognostics algorithms
 - ADS-B IN/OUT
 - Air traffic conflict detection and resolution
- Available Data:
 - Airframe parameters
 - Power plant parameters
 - INS/GPS
 - ADSB IN traffic table
- Advantages:
 - Low cost vehicle suitable for high risk avionics algorithm and hardware testing.
 - Low cost air traffic scenario flight testing.
 - Minimal ground support hardware requirements.
 - Environment for testing decision algorithms for space applications.



FAA registered LaRC 8' UAS:
"N802RE"

Soon to be registered: "N803RE"
~ \$20k per copy

Test Concept



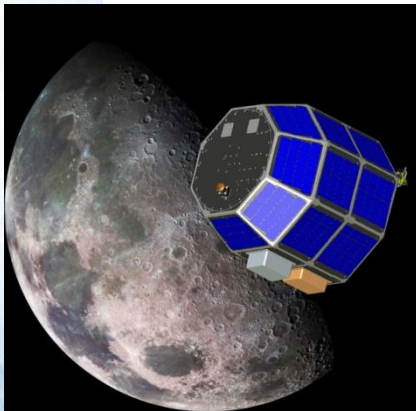
Research Objectives

- Air space conflict contingency – Self separated flight based on ship-to-ship ADS-B.
- Own-ship health contingency – evaluate mission alternatives based on current and projected own-ship constraints.
- Demonstrate simplified UAS architecture in multi-vehicle traffic conflict scenarios
- Flight rules validation in part flight/part simulation environment



Software Change Management Research

RQ: How can we safely and efficiently validate and verify complex evolving software systems?



Develop novel tools and techniques that leverage change information to preserve and improve the integrity of software as it changes over time



Software Change Management Research

- Change Impact Analysis
 - Directed Incremental Symbolic Execution (DiSE)
 - Java and C programs
 - Differential Symbolic Execution (DSE)
 - Java programs
- Techniques to support software evolution and maintenance tasks
 - Software testing
 - Equivalence checking
 - Debugging
- Interns have had a big impact on this work
 - Collaboration with LaRC and ARC
 - Several current employees started out as interns



NASA LARSS Internships

- Additional Information, application, and deadlines at <http://www.nianet.org/larss>
 - Application Deadline for Summer 2013 internships is February 1, 2013
- Extremely well organized program
- Eligibility Requirements:
 - US Citizenship
 - Basic background check for center access
 - Full Time Student Status at an Accredited US University
 - Rising Undergraduate Junior or Senior or Graduate Student (masters or doctoral)
 - Minimum 3.0 GPA or a 4.0 Scale



Questions?







Some PAST FM Projects (1985—2003)

Hardware	Software
Boeing PIU Project (1991) † Draper FFTP Scoreboard (1991) † Rockwell Collins AAMP5 (1994) † Rockwell Collins AAMP-FV (1995) † Derivation Reasoning System (1998) Honeywell/TTTech FADEC (2000+)	Space Shuttle Jet-Select Project (1993) † Honeywell Air Transport (Tablewise) (1995) † Space Shuttle GPS & 3EO upgrades (1995) † Collins Mode Confusion Project (1998) ORA/Aonix Ravenscar Project (1998) Formal Analysis of UML Models (1999) Collins AvSP Project (2000+)
Fault-tolerance	Systems / Algorithms for ATM
Reliable Computing Platform (1990) Allied Signal Hybrid Fault Models (1992) Union Switch and Signal (1994) † SPIDER (2000+)	Aircraft Info for Lateral Spacing (AILS) (1999) Barron--Analysis of Neural Nets (2002) Conflict Detection and Resolution (2002) Small Aircraft Trans System (SATS) (2003) Self-Spacing Terminal Area (2003)
Software Certification	Tools and Techniques
Streamlining SW Aspects Cert (2001) MC/DC studies and tutorial (2002) Object Oriented Software and Cert (2003)	PVS Theorem Prover Development Accident Report Analysis Zeus: Natural Lang + FM



Notable Formal Methods Successes 2000 - 2008

Concept of Operations

Small Aircraft Transportation System (SATS)

Enhanced Oceanic Operations (EOO)

System-level Requirements

Rockwell Collins: Asynchronous Models of Flight Deck (GALS)

Airborne Information for Lateral Spacing (AILS)

Spacecraft Autonomy and AI Planning

Software/Hardware Requirements

Rockwell Collins: Flight Guidance Systems/Flight Management Systems

Mode Confusion Elimination

Software/Hardware Designs

Scalable Processor-Independent Design for Extended Reliability (SPIDER)

Honeywell/TTTech Full Authority Digital Engine Control (FADEC)

Honeywell DEOS Operating System

Conflict Detection and Resolution (CD&R) Algorithms

Runway Incursion Prevention System (RIPS)

Software/Hardware Implementations

Automatic Code Generation for KB3D

Fixed Structure Neural Networks

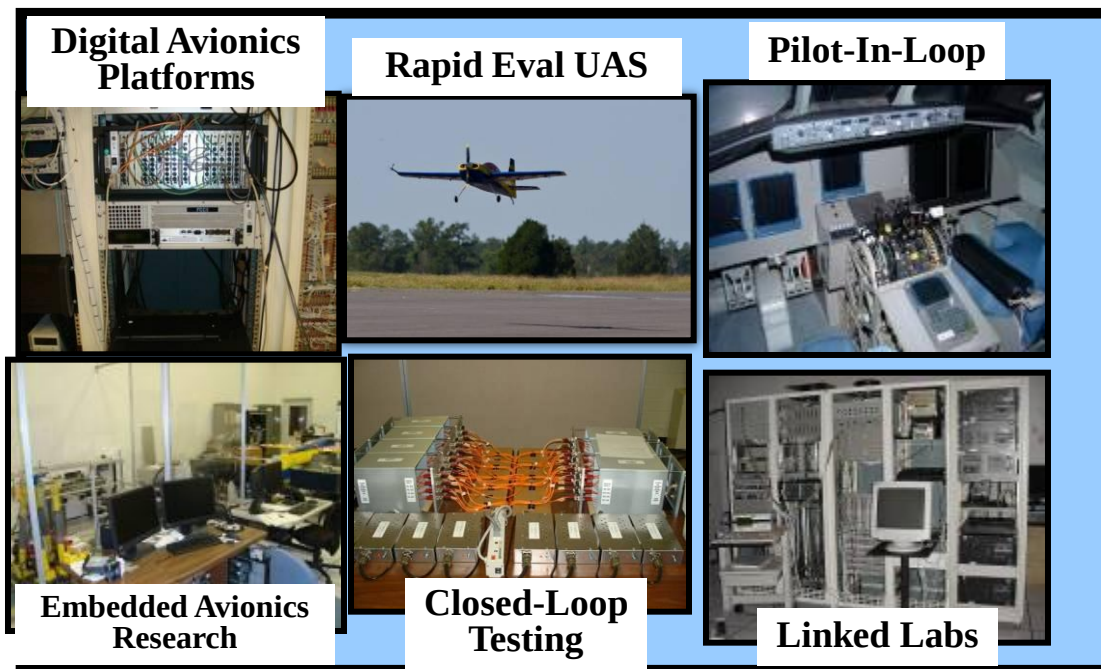


Systems and AirLaboratory frame Failure Emulation, Testing and Integration (SAFETI)

Features

- **Reconfigurable Testbeds**
 - Components and Subsystems
 - Data Buses
 - Vehicle Simulations
- **Integrated Multi-Disciplinary Research and Technology Demonstration**
 - Complex Critical Flight Systems
 - Adaptive Reconfigurable Controls
 - Real-Time Fault Det and Function Monitoring
 - Crew/Vehicle Interface Systems
 - Airframe Structures Prognostics
 - Electromagnetic Effects
 - Enhanced System Reliability for UAS in the NAS

Unique Venue for Integration, Validation, and Demonstration of Aviation Safety Systems Research Efforts and New Technology





Many Approaches

- Formal models of IEEE Floating Point Standard in PVS
- New decision procedures for reasoning about interval arithmetic
- C program verification using Frama-C and Gappa and PVS
- A new research area for us and we are still finding out how much we have to learn



Our Approach

- Simple **criteria** for **coordination** in **resolution** algorithms
 - Criteria is sufficient to guarantee safety property
 - Criteria is simple enough so that algorithms can be checked against the criteria in a straight-forward way
 - Criteria only uses information available to the local aircraft

- Prove that **Criteria**  **Safety Property**

algorithm₁ 

Criteria

...

algorithm_n 

Criteria



Criteria

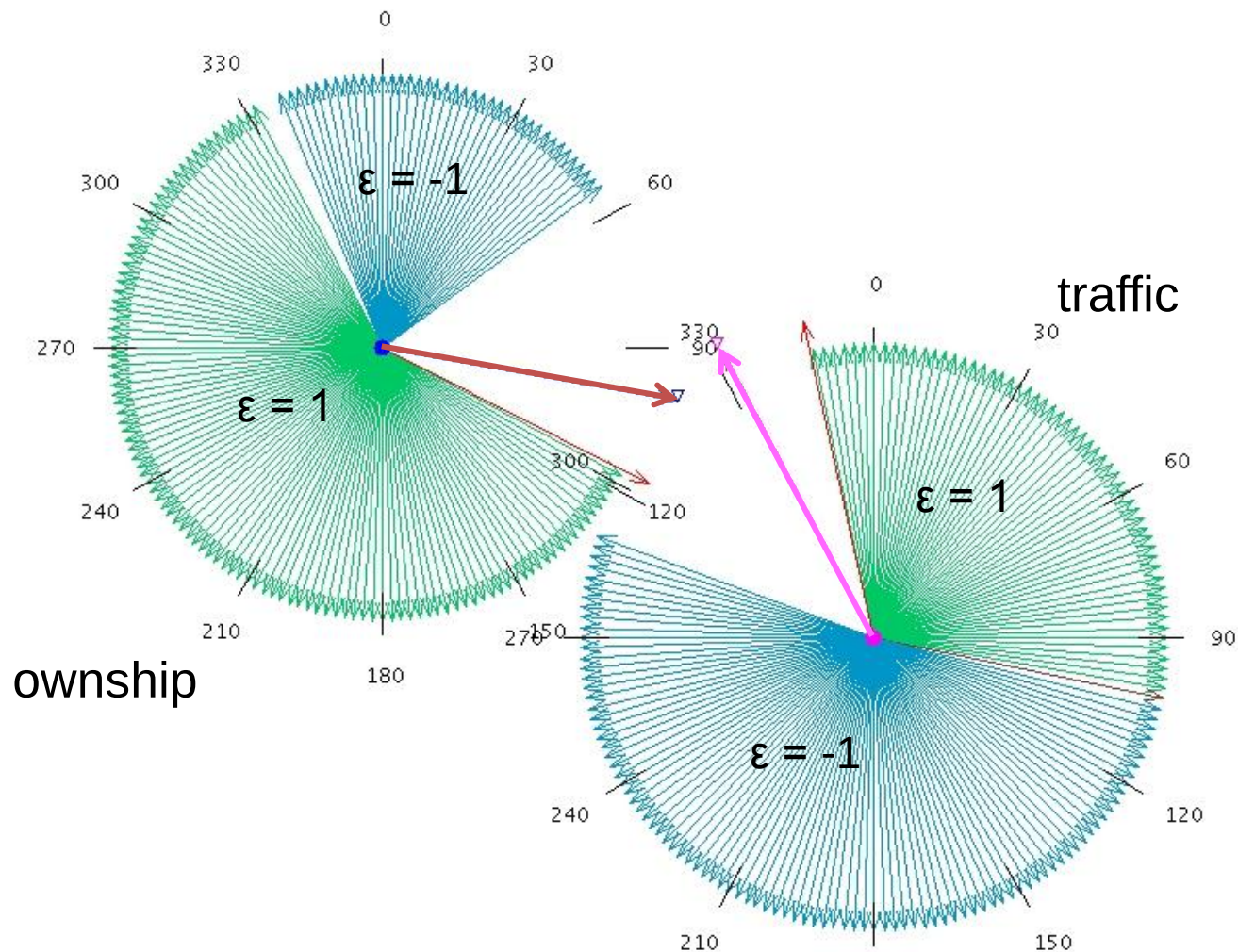
	in Conflict	in Loss of Separation
horizontal	$(s \bullet v') \geq \varepsilon \text{ } \vee (s^\perp \bullet v')$	$s \bullet v' > s \bullet v \text{ } \text{AND}$ $s \bullet v' \geq s (D - S)/T_h$
vertical	$\Delta > 0 \text{ AND } t > 0 \text{ AND}$ $\delta = 1 \text{ AND } s_z v_z \geq 0$ OR $ s_z + t v_z \geq H \text{ AND}$ $\delta (s_z + t v_z) v_z \leq 0$	$v_z' \neq 0 \text{ AND } s_z v_z' \geq 0 \text{ AND}$ $s_z v_z \geq 0 \implies$ if $v_z = 0$ then $\text{break_symm}(s)(v_z') > 0$ else $\text{sign}(v_z) v_z' \geq 0$



Horizontal Criterion

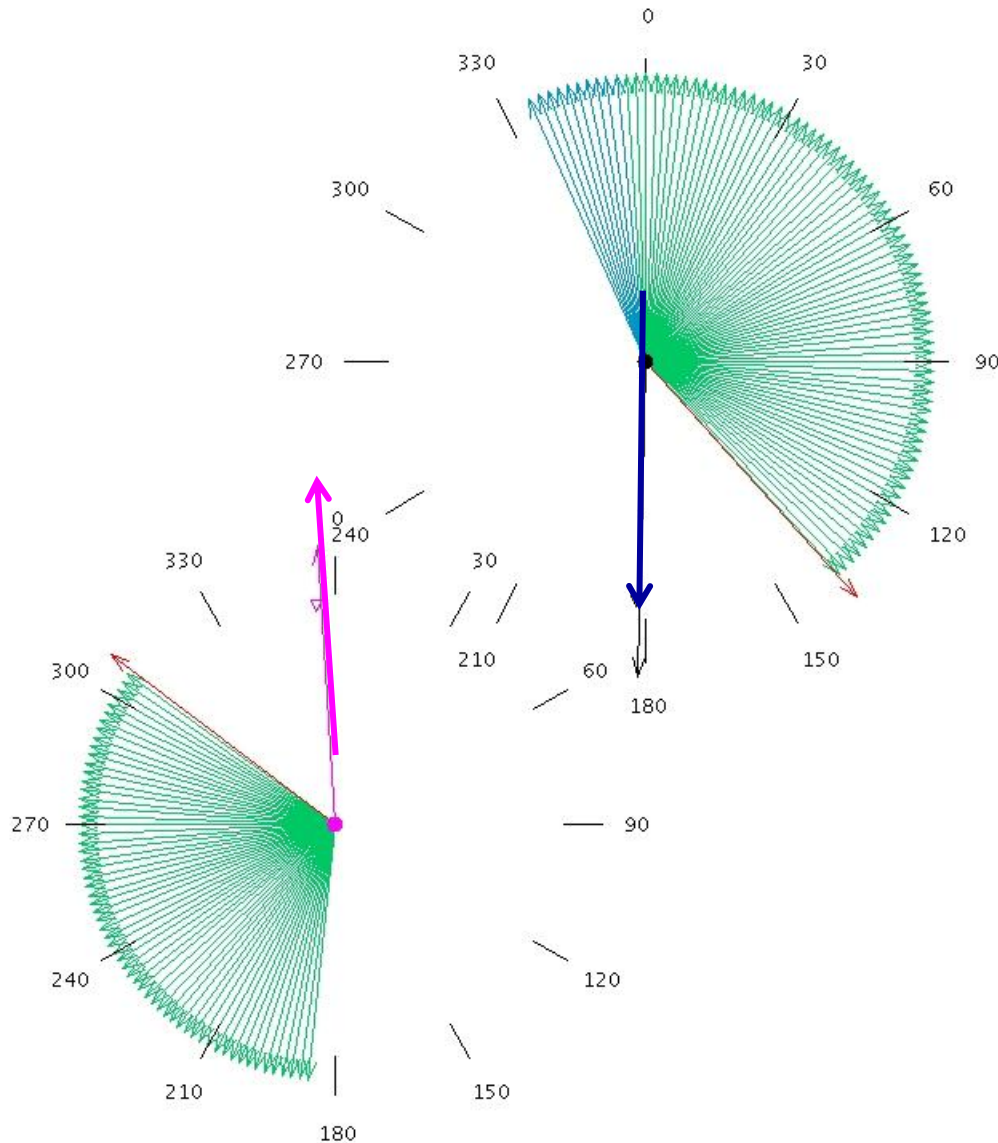
$$(\mathbf{s} \cdot \mathbf{v}) \geq \varepsilon \quad R(\mathbf{s}^\perp \cdot \mathbf{v}) \quad \text{where } \varepsilon = \{-1, 1\}$$

and $R = \frac{\sqrt{s^2 - D^2}}{D}$





Blue and Green Regions



If *both* aircraft produce resolution vectors anywhere in the **green** region, the combined result will be *coordinated*.

The blue *and* green region provides *independent* correctness



How to Choose ϵ ?

- If we want *implicit coordination* universally, there has to be an international agreement

Some possibilities

- $\epsilon = 1$ (always green)
- $\epsilon = -1$ (always blue)
- $\epsilon = (s^\perp \cdot v) \geq 0$ (our recommendation)

Requirement:

$$\text{Policy}(s, v) = \text{Policy}(-s, -v)$$



ownership



traffic