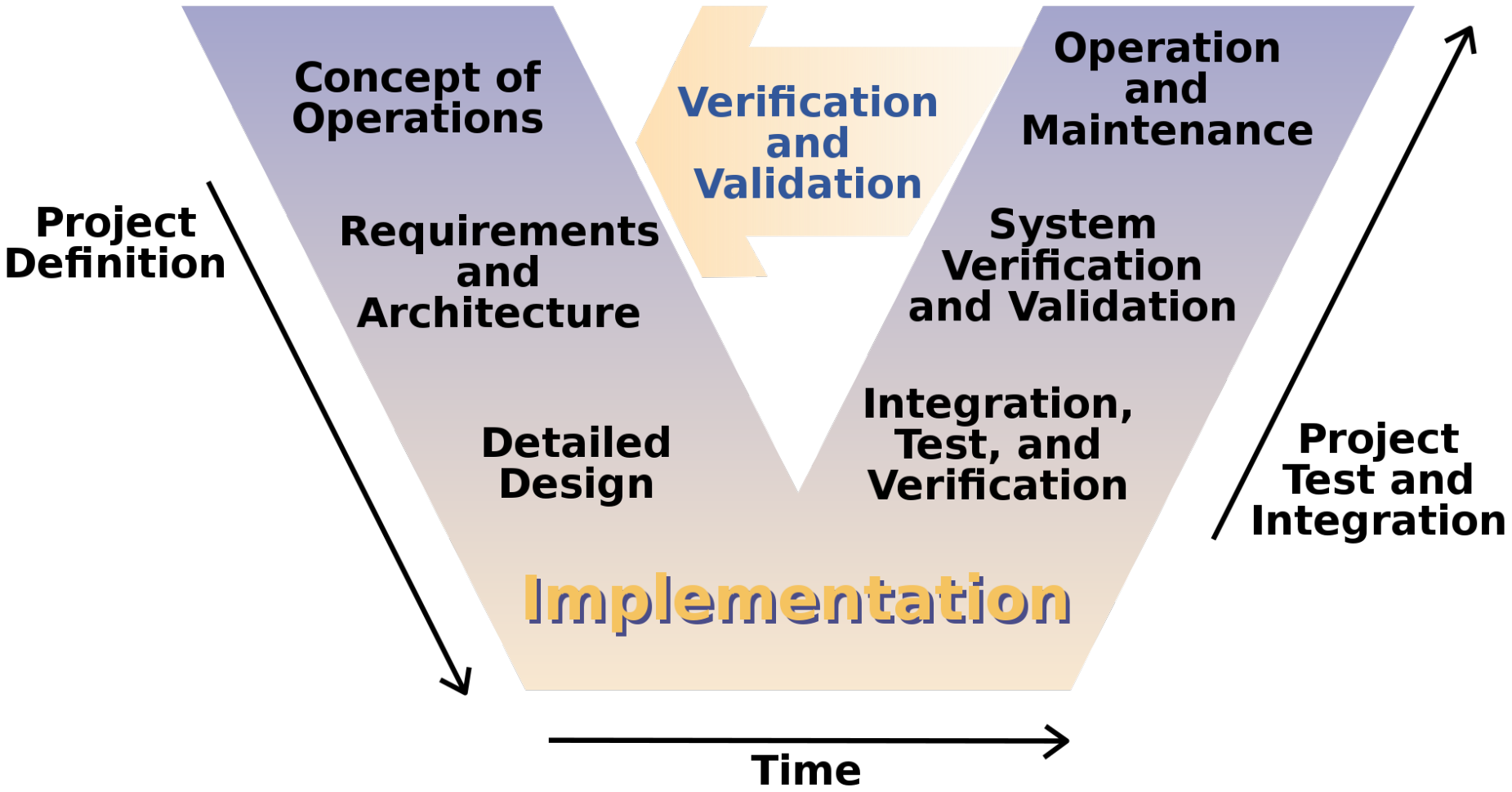
Adrian Agogino and Rory Lipkis

Introduction

- ***AdaStress*** is an open-source software package written in Julia that determines the likeliest failures in a simulated system under test
- The package provides three primary services:
  - Interfaces between user simulations and the AST framework
  - Statistical and learning-based solvers
  - Analysis and visualization tools
- Designed with an emphasis on:
  - Runtime speed
  - Modularity / extensibility
  - Collaboration (two-stream contribution framework)
- Available at `https://github.com/NASA-SW-VnV/AdaStress.jl`

Motivation: verification and validation



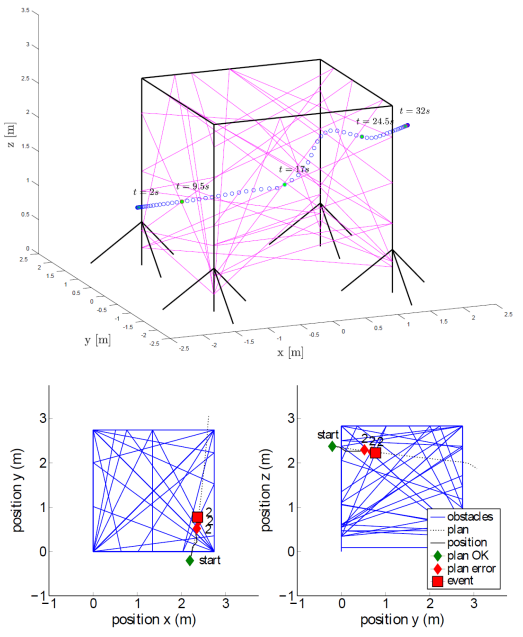
Clarus Concept of Operations, Federal Highway Administration (FHWA), 2005

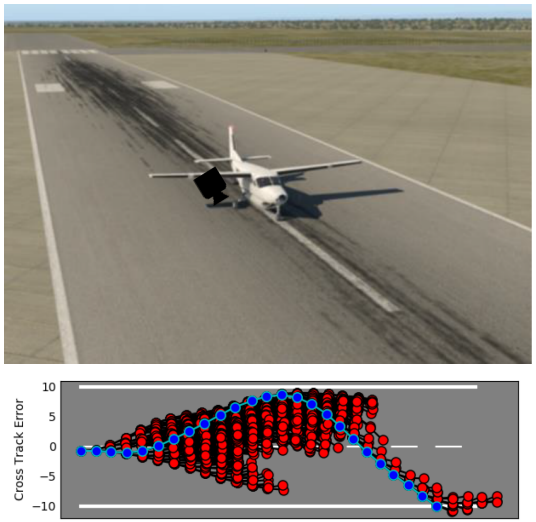
Verification

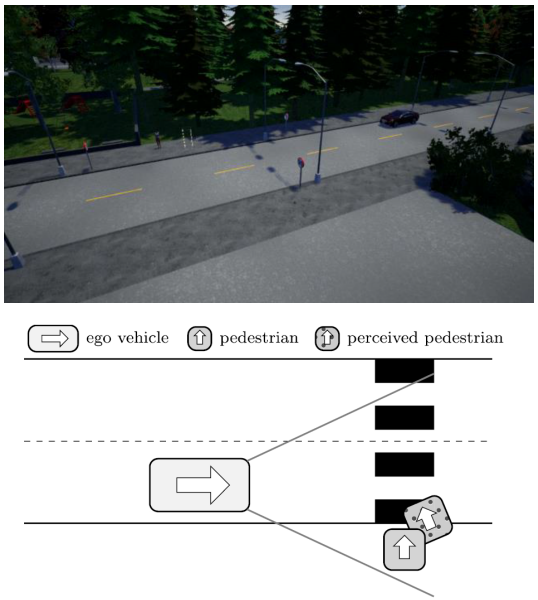
- Formal verification
  - Difficult due to system size and complexity
  - Often requires making strong assumptions about system behavior
  - Produces mathematical guarantees
- Sample-based approach
  - One-sided "proof" (falsifiability)
  - Requires simulation
  - Contingencies are explored in proportion to their likelihood
  - Expensive at scale
  - Often requires domain knowledge to inform exploration
  - Scenario "engineering" can violate principles of IV&V

Adaptive Stress Testing (AST)

- Accelerated validation framework
- Requires no knowledge of system under test
- Generates adversarial environments to find **likeliest failures**
- Better performance at scale
- Flexible and general

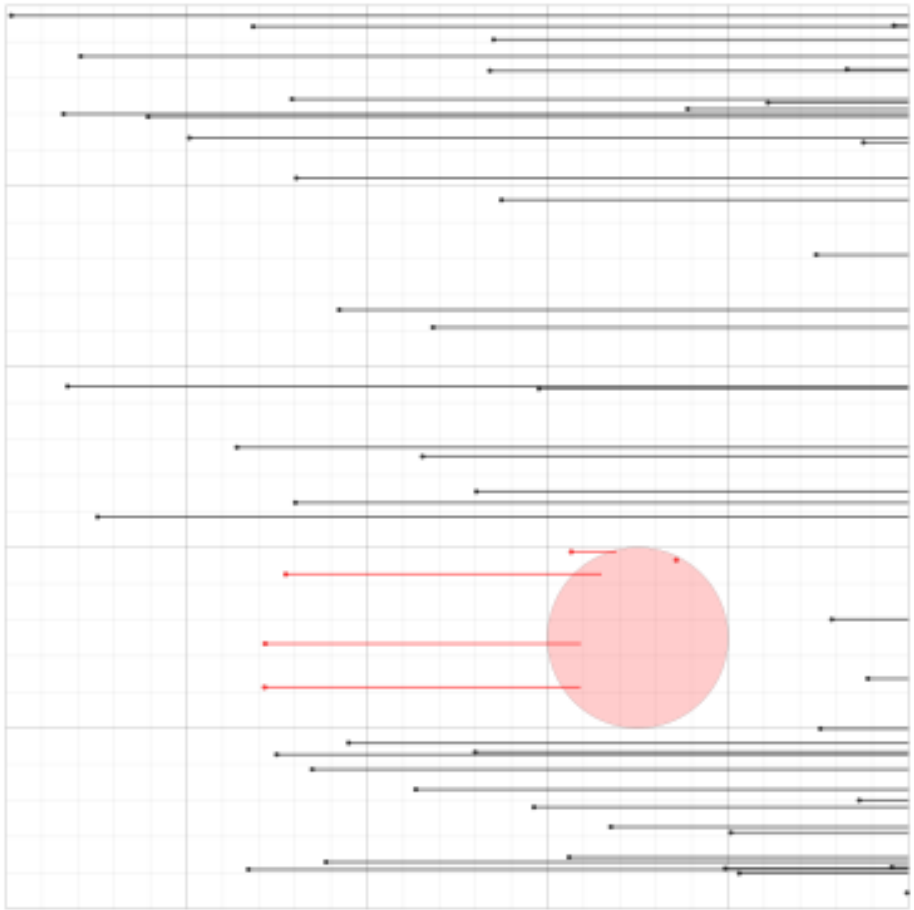






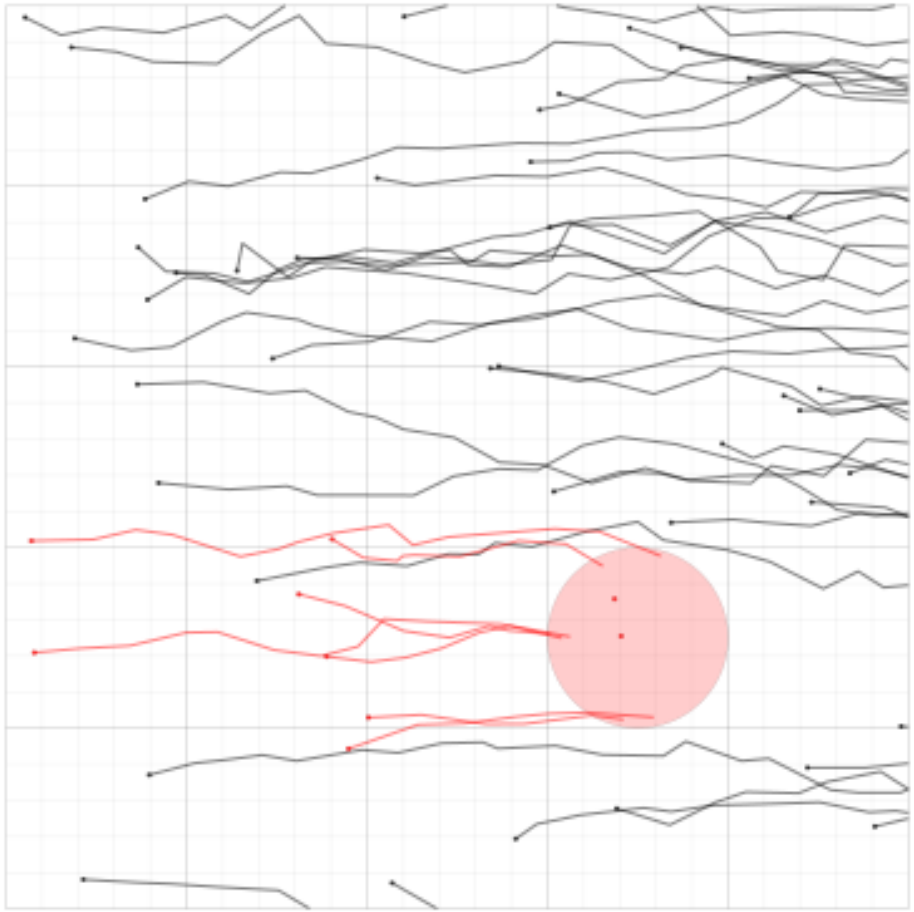
- Implemented by **AdaStress**

Example: sampling without disturbances



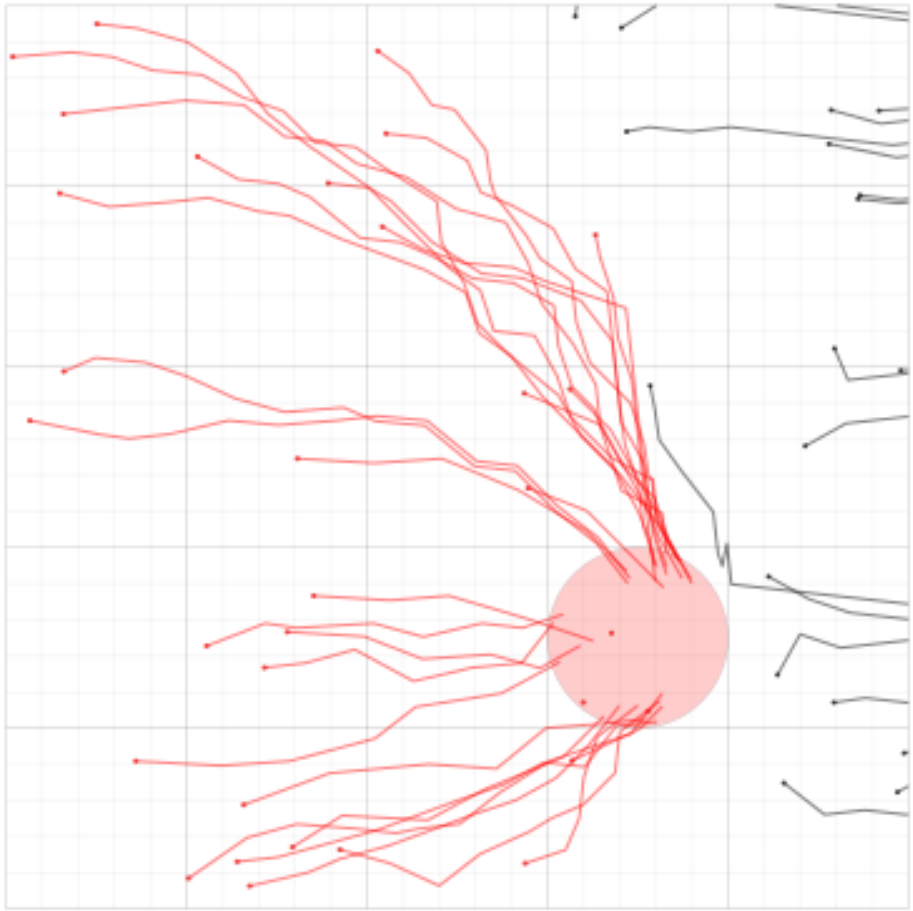
Natural system behavior moves state to right at constant rate

Example: sampling with disturbances (Monte Carlo)



State experiences stochastic perturbations

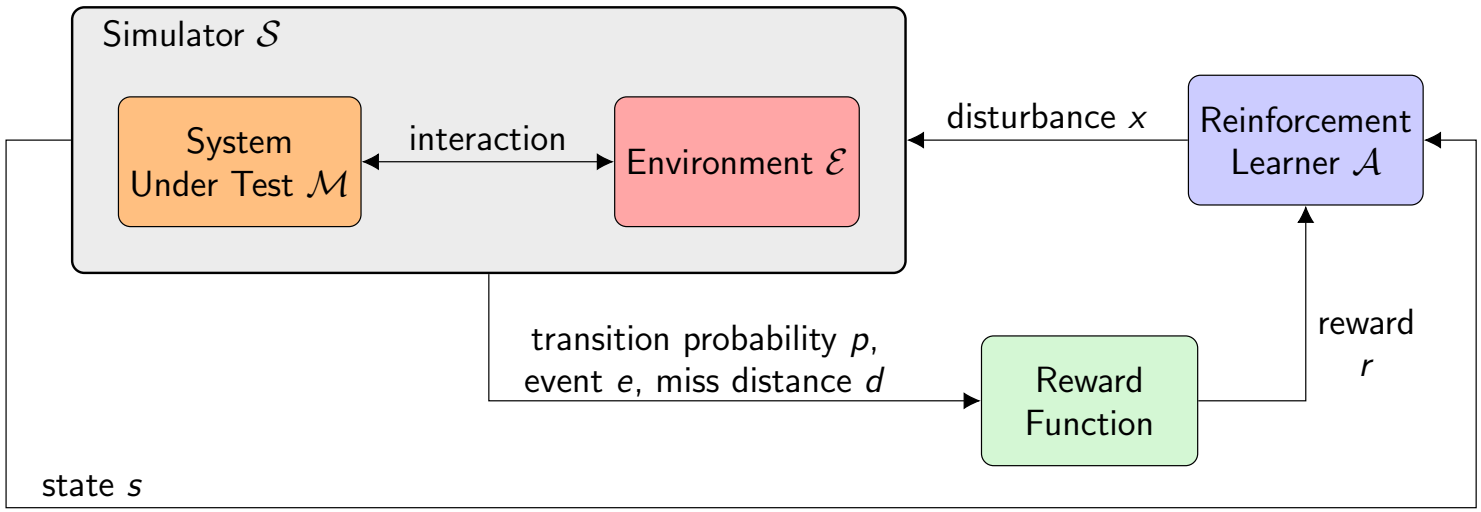
Example: sampling with AST



AST finds failures corresponding to a set likelihood threshold

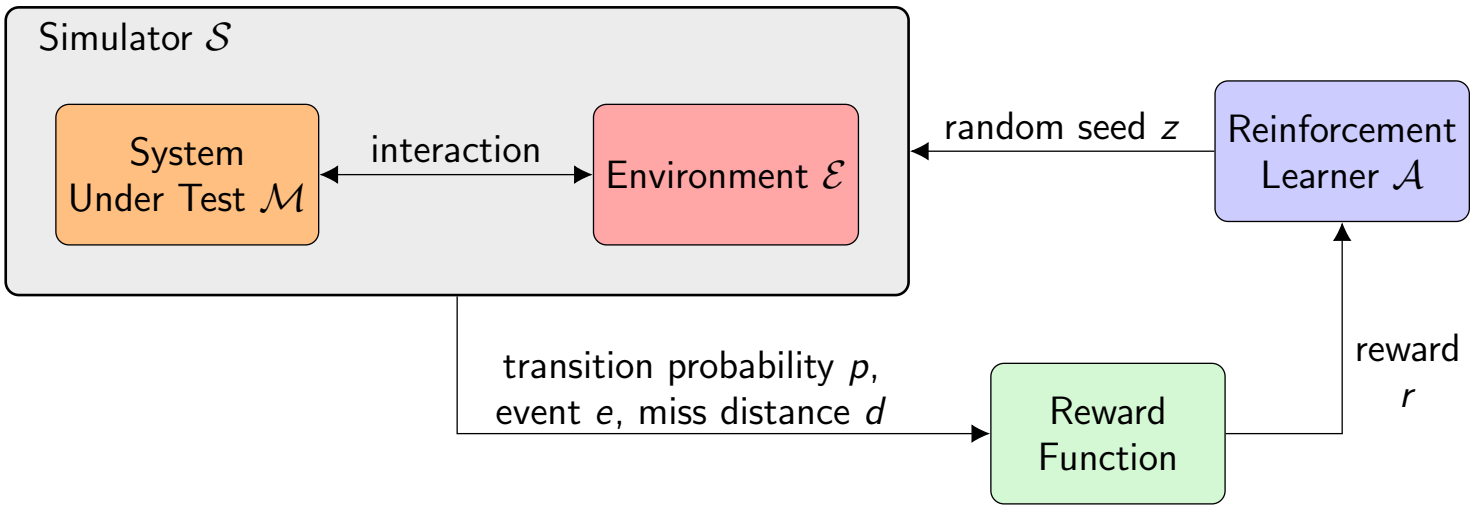
AST overview

- Simulated system under test
- Stochastic system environment (set of disturbances)
- Formulates stress-testing problem as MDP or POMDP
  - Actions: instances of environment or random seeds
- RL agent chooses actions to optimize overall marginal likelihood with the constraint of eventual system failure



AST with blackboxes

- State and environment representations may be inaccessible
  - Proprietary or confidential systems or simulations
  - Difficulty of refactoring and integration
  - Infeasible dimensionality
- AST can still perform reinforcement learning with near-blackboxes by manipulating random seeds underlying the simulation



- ***AdaStress*** can securely perform stress testing on remote and embedded systems through a server abstraction

AST formulation

- Given:
  - State space: simulation observations $s \in \mathcal{S}$
  - Environment: set of stochastic disturbances $\mathcal{X} \sim f(x)$
  - Failure criterion: $s \in F \subset \mathcal{S}$
- For particular state trajectory $T = \{s_1, s_2, \ldots s_n\}$,

$$P(T) = P(s_1 \ldots s_n) = \prod_{i=1}^{n} P(s_i \mid s_1 \ldots s_{i-1}) = \prod_{i=1}^{n} P(s_i \mid s_{i-1})$$

$$= \prod_{i=0}^{n-1} P(x_i \mid s_i)$$

- Optimization formulation:

$$\max_{x_1,\ldots,x_T} \quad \sum_{t=1}^{T} \log P(x_t \mid s_t)$$

$$\text{subject to} \quad s_T \in F$$

Reinforcement learning

- State and environment spaces are continuous and high-dimensional
- Environment selection is parameterized as a policy
- Failure constraint is enforced via penalty
- Reward is given by

$$r(s_t, x_t) = \log p(x_t \mid s_t) - \Delta d_F + R_F \cdot \{s_t \in F\}$$

Case study: validation of ACAS sXu

- Airborne Collision Avoidance System (ACAS X)
  - Replaces Traffic Alert and Collision Avoidance System (TCAS)
  - Models aircraft encounters as POMDPs
  - Stores precomputed solution as large lookup table
  - Detects potential collisions between individual aircraft
  - Issues directive guidance in the form of resolution advisories (RAs)
- Variants:
  - Xa (commercial aircraft)
  - Xo (specialized missions)
  - Xu (unmanned aircraft)
  - **sXu** (small unmanned aircraft)

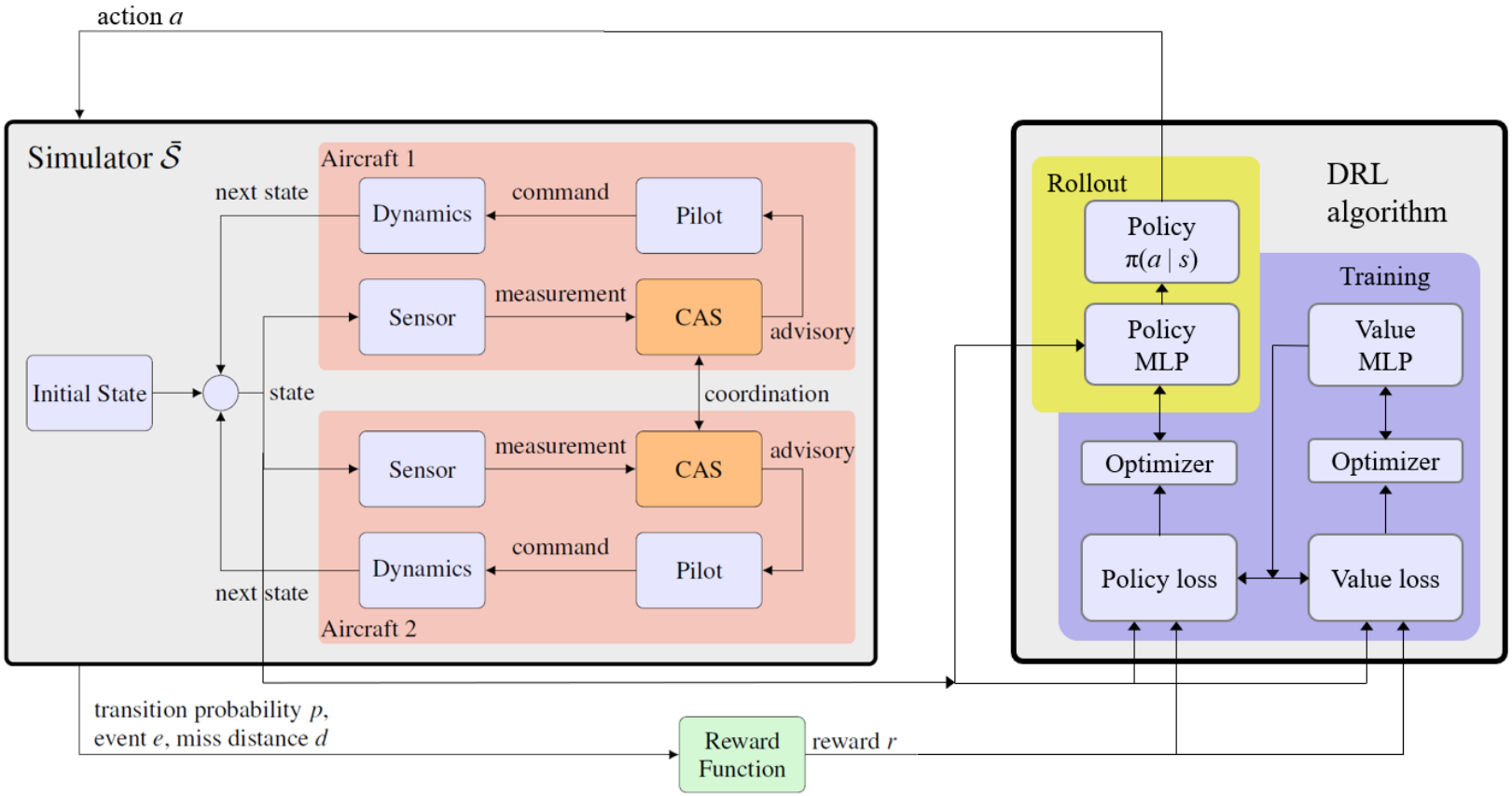
Research goal

Provide ACAS sXu development team with stress-testing tools and infrastructure to inform their ongoing work

Testing setup

- **System under test:** ACAS sXu binary
- **Disturbances:** pilot commands
  - Turning rate
  - Vertical rate
  - Forward acceleration
- **Aircraft dynamics:** simple multirotor with limited acceleration
- **Response to CAS:** pilot compliance with 1-second delay
  - Turning rate of $3^\circ/\text{s}$ complying with advised horizontal maneuvers
  - Acceleration of $g/4$ to $g/3$ complying with advised vertical maneuvers
- **Failure criterion:** small near mid-air collision (sNMAC)
  - 50 ft. horizontal separation
  - 15 ft. vertical separation

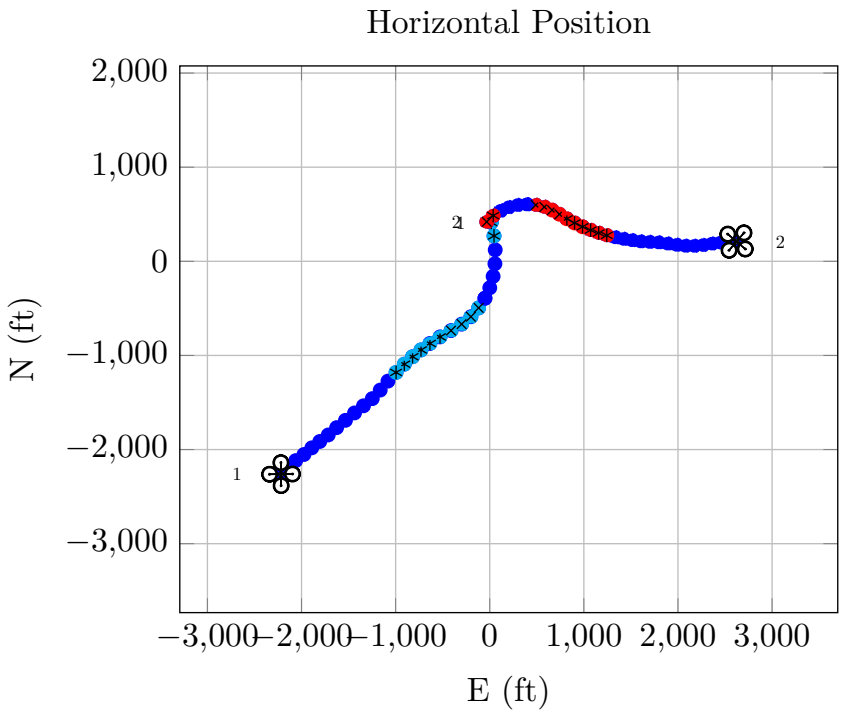
Testing diagram

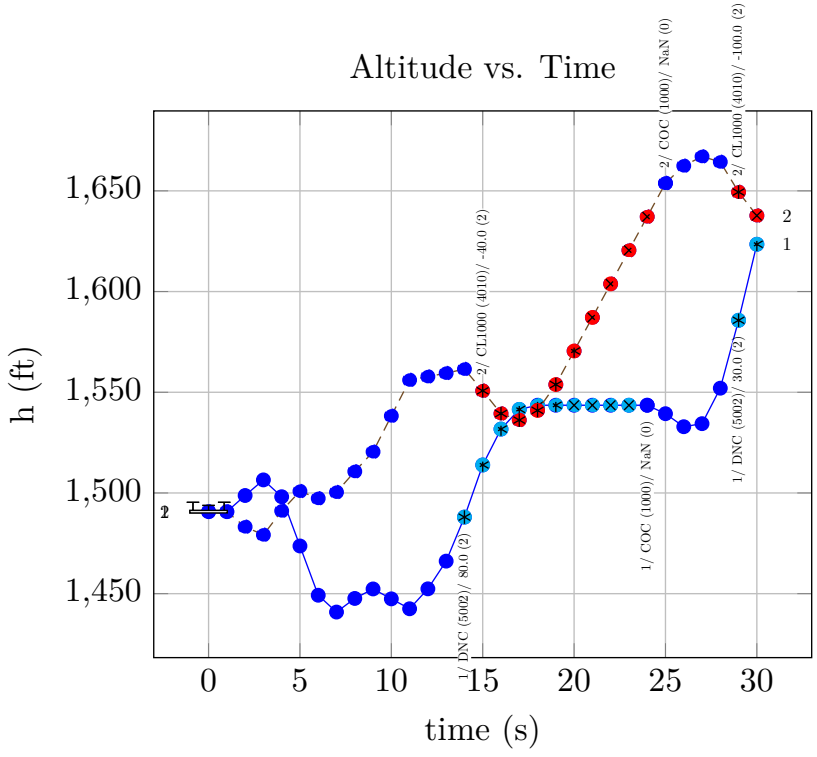


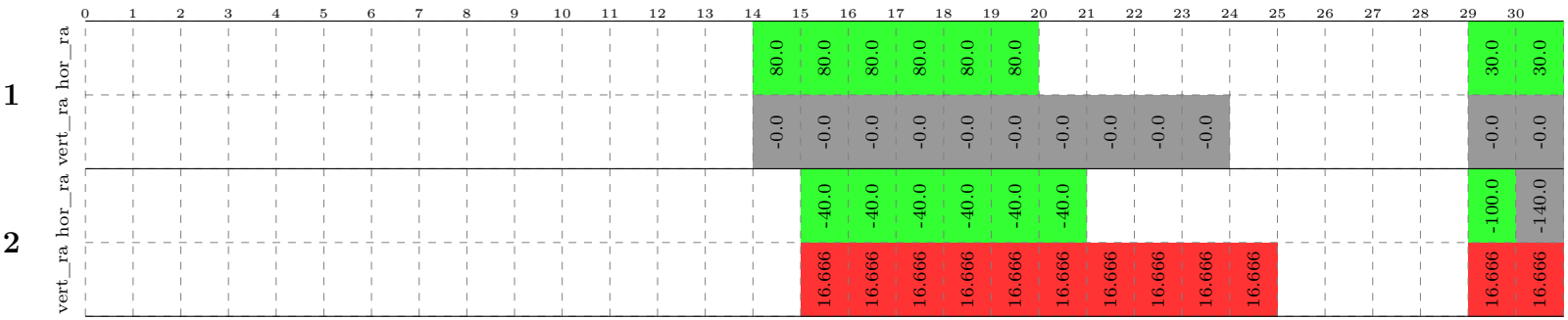
Results

- **97% sNMAC rate** when $a_{\text{max}} = g/2$
- Categorizations of failures
  - Dart failures
  - Freeze failures
  - Miscellaneous failures
- Classifications of CAS involvement
  - Uninduced sNMACs
  - Induced sNMACs

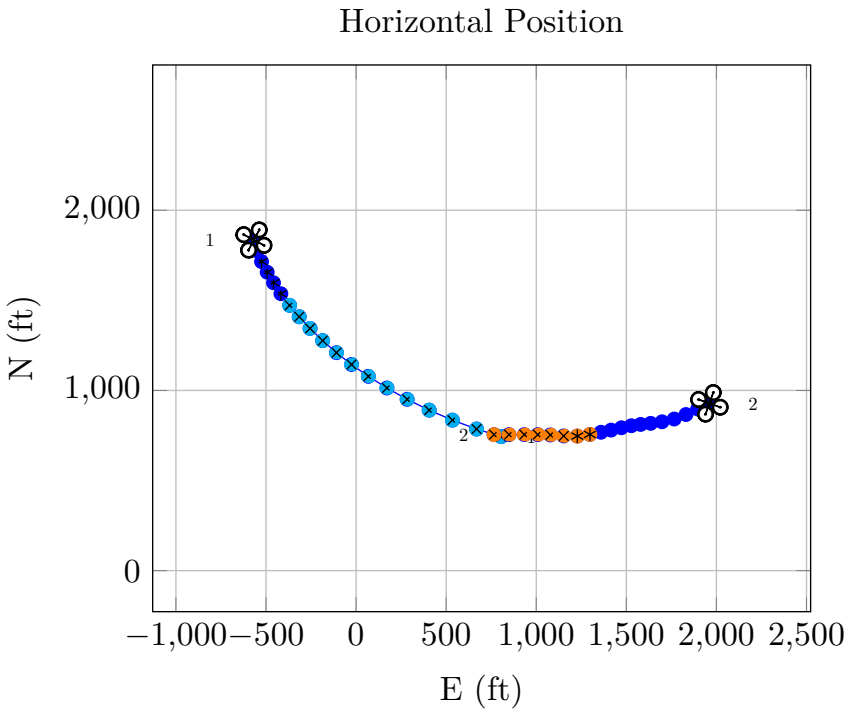
Results: dart failure

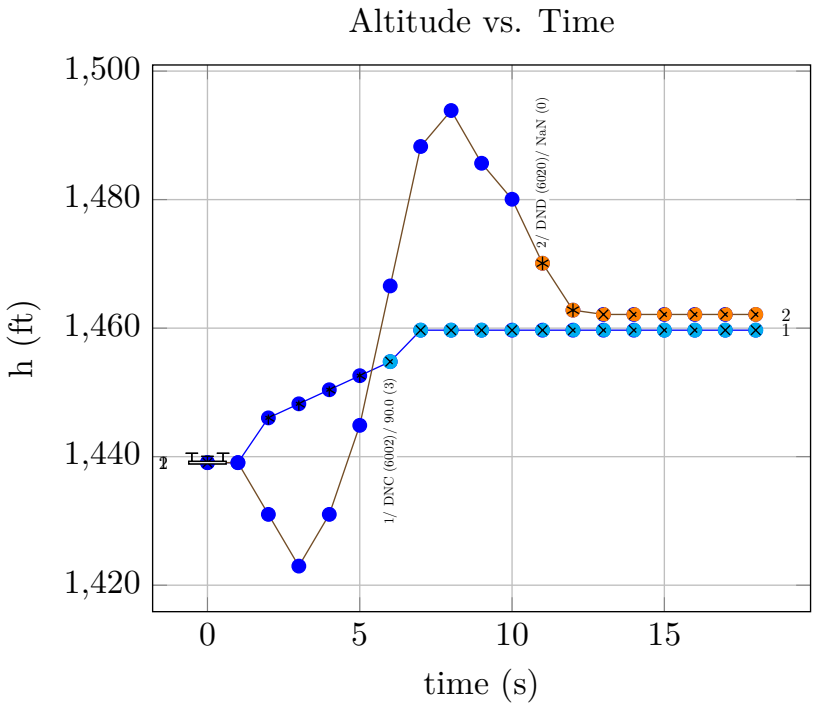


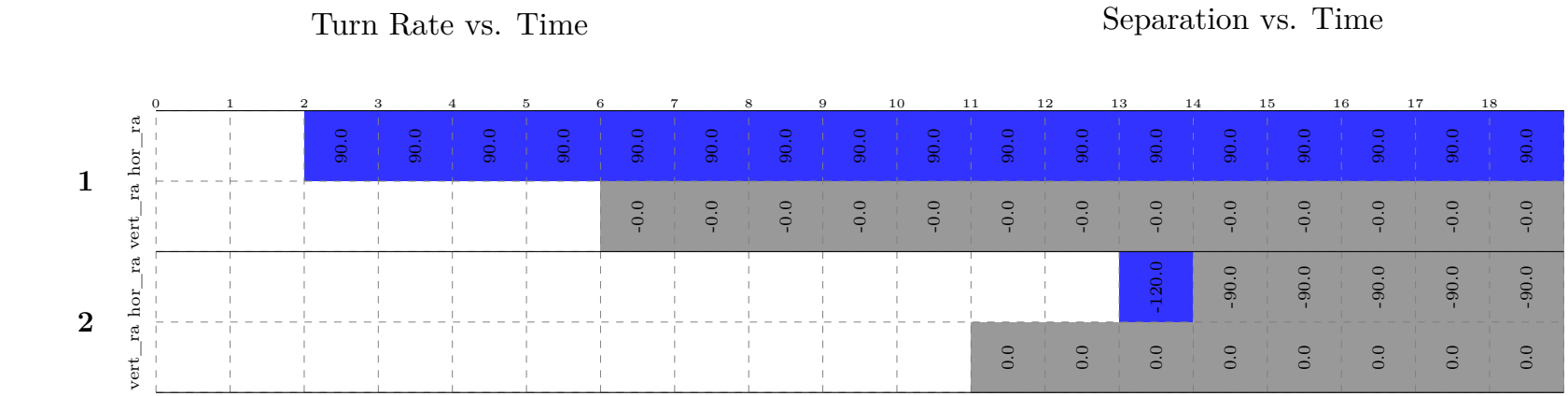




Results: freeze failure

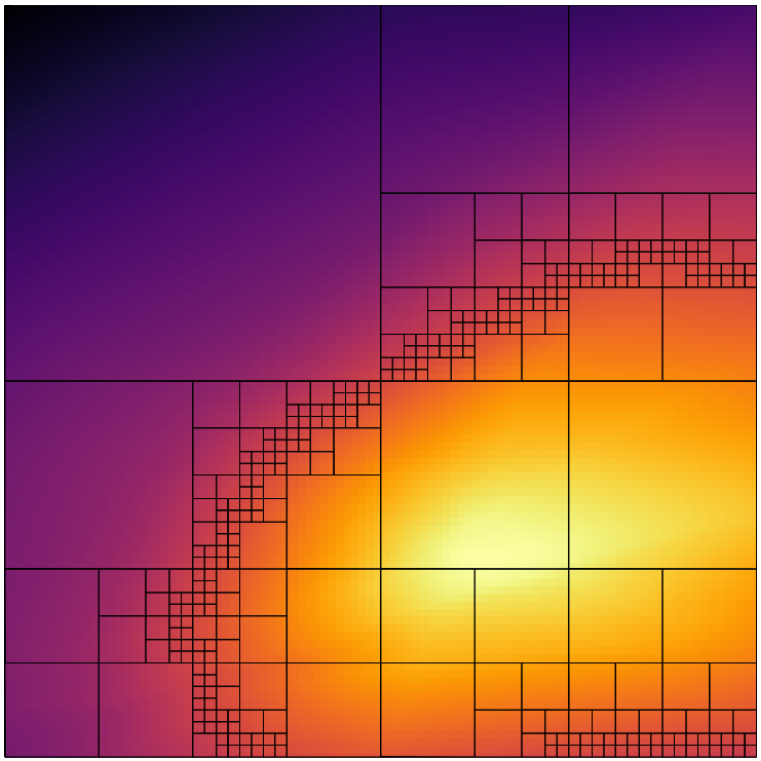


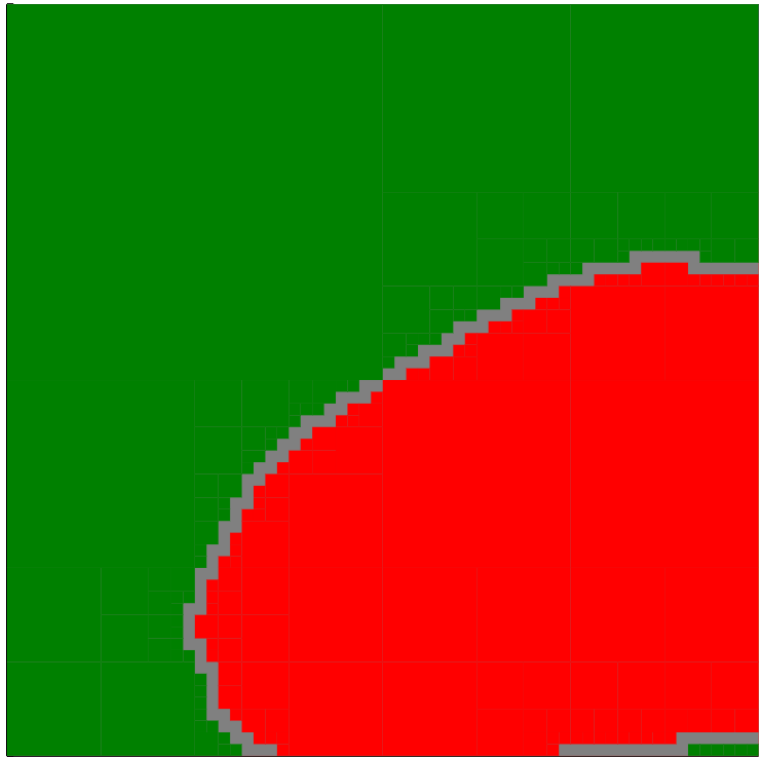




Ongoing research

- AST readily generalizes to learn **global adversarial policies**
- Provides opportunities for analysis of learned networks
- Certain probabilistic properties can be bounded





- Python API (*future*)
- Operation mode as runtime assurance (RTA) tool (*future*)

Acknowledgments

- Ritchie Lee
- JHUAPL ACAS X Team
- FAA TCAS Program Office

This work was supported by the Systems-Wide Safety (SWS) Project under NASA Aeronautics Research Mission Directorate (ARMD) Airspace Operations and Safety Program (AOSP).

Demonstration