Exploring Novel Approaches to the TACE Watchdog

March 2017

Presented by: Bill D'Amico

Authors: Corey Lowman & Bill Van Besien

JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

Contents

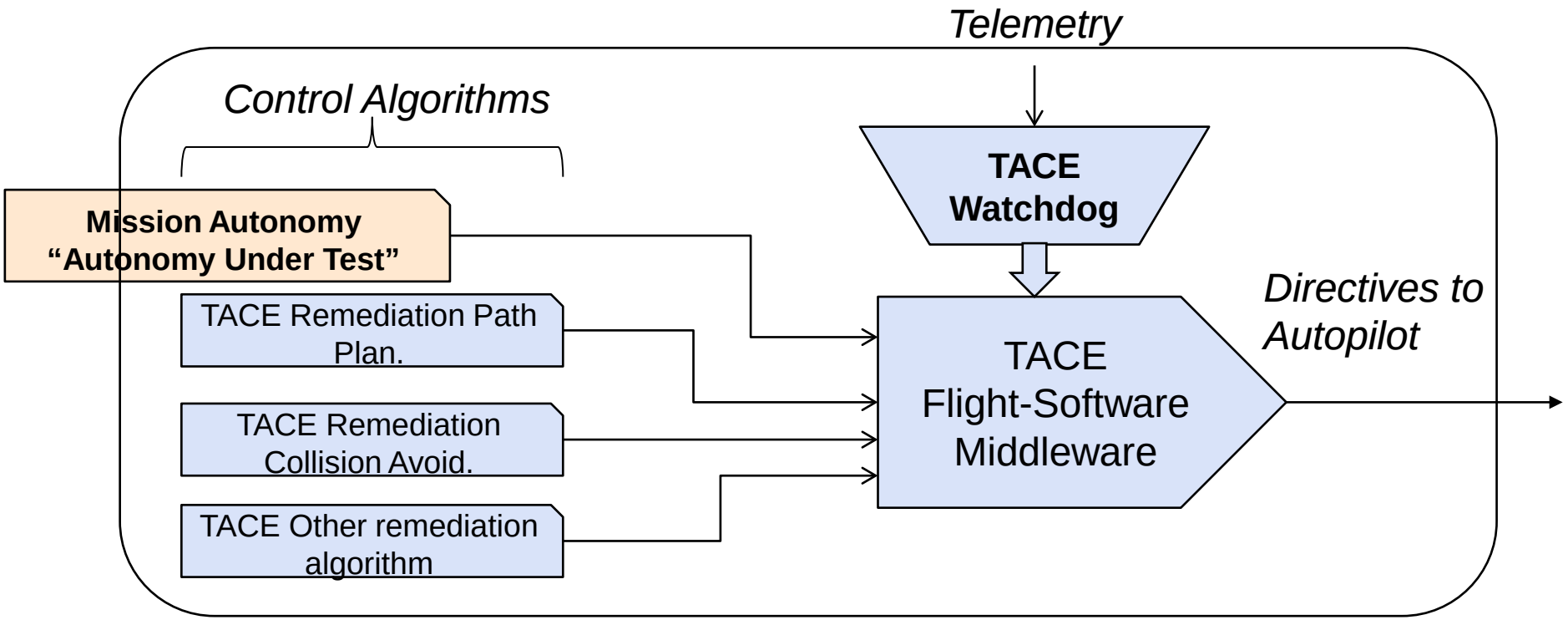
- TACE Background and Motivation
- Watchdog Concept and Design
- Overview of Spaceflight-Heritage *Watchdog* Model
- Overview of Novel *ModelGraph*-Based Approach to *Watchdog*
- Example Videos of *Watchdog* in Simulation
- Approaches to Formal Verification

TACE – Overview and Motivation

- TACE – The *Safe Testing of Autonomy in Complex, Interactive Environments.*
- Capabilities demonstrations at Aberdeen Proving Ground (2014/2015) and Atlantic Test Range (2016).
- Contains full live, virtual, constructive (LVC) infrastructure, integration with range radars.
- The *Watchdog* as the key component of TACE safety infrastructure.
  - Resides on the UAV, supplements onboard avionics.
  - Alerts pilots, safety officers to unsafe operating conditions.
  - Overrides onboard mission autonomy when safety boundaries violated.
  - Remediates aircraft from unsafe states by controlling autopilot directly.

System Context – Watchdog as a Multiplexer

- The *Watchdog* overrides onboard mission autonomy when safety constraints are violated.
  - The *Watchdog* becomes the gatekeeper to the autopilot until the vehicle is safe and test directors resume the test.
  - Watchdog is shown in TACE onboard architecture (Simplified).



Initial Approach – Watchdog core as a VFSM

- TACE *Watchdog* base lined APL-Developed technology used for NASA Solar Probe autonomy.
  - Developed 2006-12 in the Space Exploration Sector at JHUAPL.
  - Designed to manage and minimize complexity for spacecraft fault management/recovery.
  - Allows granular, nuanced responses to faults rather than clunky, pre-programmed sequences.

- Uses a *Virtual Finite State Machine (VFSM)* as the underlying formal computational structure.
  - Generalization of a classical Finite State Machine, suitable for asynchronous events and input.
  - Reduction to well-defined computational structure amenable to formal verification methods.

- Safety/Autonomy models are represented as VFSMs.
  - Models encode instrument/component states (e.g., nominal, overheating, unpowered, etc.), transitions among states, and actions to be taken to bring the UAV into “safe mode” in a fault event.
  - Software provides drag-and-drop utility to create models.

- *Watchdog* software operates as an interpreter of VFSM models (three phases).
  - ***Sense*** any changes in onboard telemetry, ***evaluate*** the given safety model (perform state transitions, if necessary), **act** on any commands as defined by the transition between states.

Motivation: Limitations to the VFSM Model

- In traditional FSMs/VFSMs states are “context free”.
  - Current state defined exclusively by prior state and recent inputs.
  - States themselves do not contain explicit invariants.
  - Can increase design complexity and testing burden, exposes more possibilities for discrepancy between state of FSM and *actual* state of the System-Under-Test.
- Original Space Department VFSM not intended to handle concurrent faults.
  - Space CONOPS different from that of testing DoD Autonomous Systems.
  - Space applications calls for intricate, nuanced responses to at most on fault at a time.
  - TACE requires juggling multiple faults concurrently.
- **Motivating Question*:*** *Can we extend or generalize the VFSM model to…*
  - *… Minimize design complexity?*
  - *… Simplify testing and validation of TACE remediation models?*
  - *… Better handle concurrent safety violations?*

Experimental Approach – The ModelGraph

- Propose novel data structure based on *VFSM*s for remediation models, the *ModelGraph.*
  - FSM-like graph-based data structure.
  - Transitions encode actions trigger a change in state.
  - Supplemented states with explicit invariant expression (are only considered active if-and-only-if Boolean expression evaluates to true).
  - Multiple states can be ‘active’ at one time. If no state is active an integrity error is thrown.
- Algorithm Overview: Three-Phase Evaluation Loop
  - ***Sensing Phase:*** Sense the current state. Performs triage when multiple unsafe states are active based on evaluation of various supplementary metrics.
  - ***Targeting Phase:*** Identify path toward safest *reachable* state in the model graph (sometimes picking a *locally* safe states when the *globally* safest state is not reachable). This is called the *goal* state.
  - ***Commanding Phase:*** Issue commands pertaining to the transition to the next state in the path toward the goal state.
- Lifecycle of remediation defined by path through state-space.
  - As new violations occur while remediating the path may change.

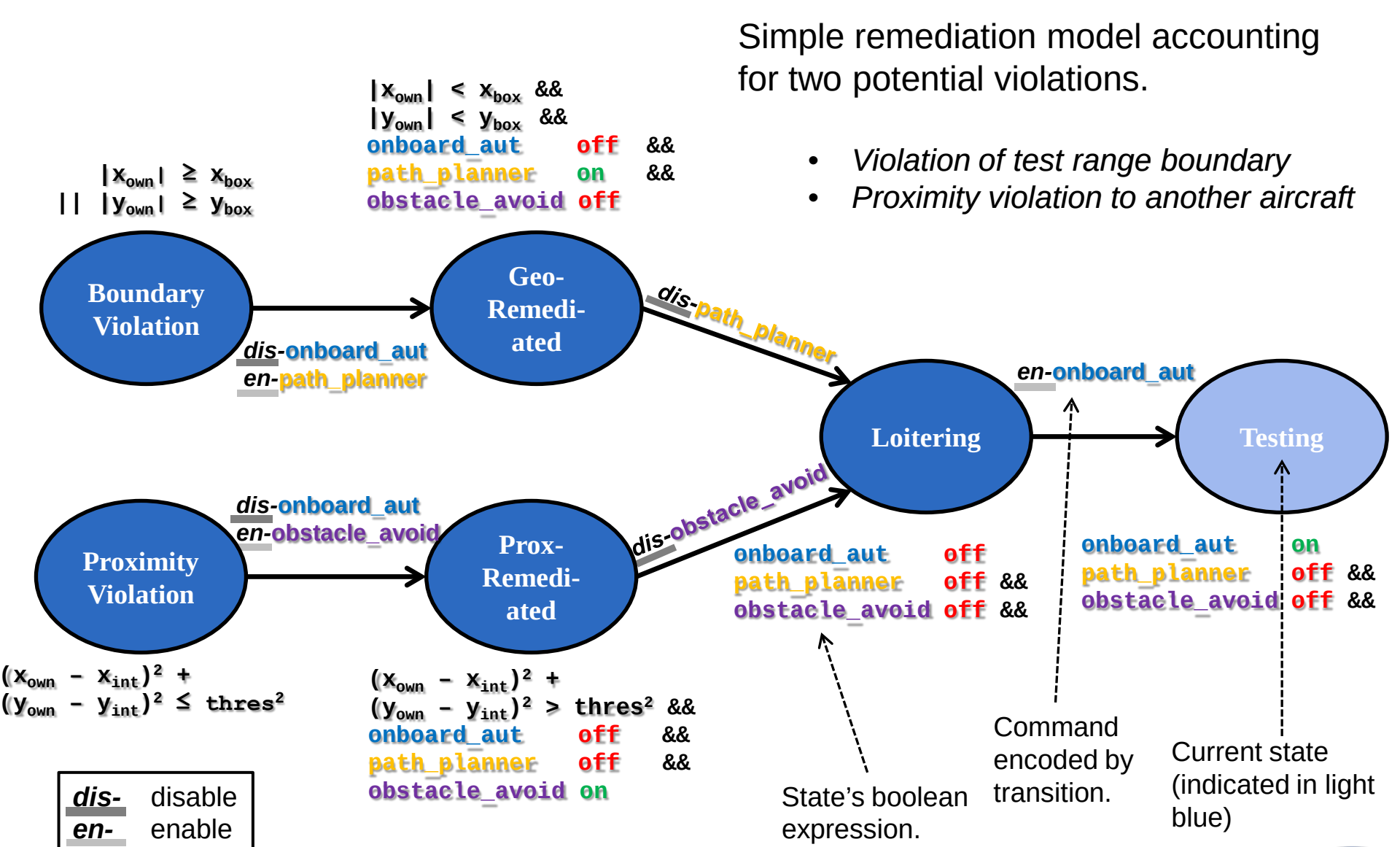
Experimental Approach – Summary

- If the Watchdog determines there is a discrepancy between the current state and the desired safest state, it finds a path between them and attempts to move along the path by executing the corresponding actions
- If a higher priority state becomes active, the path through the state space may change
  - The priority – or "urgency" – is dynamic so it changes based on the state of the system, which allows the highest priority state to change even if the active states don't change.
  - Allows more urgent and/or transient risks to be addressed while already remediating from other safety faults.

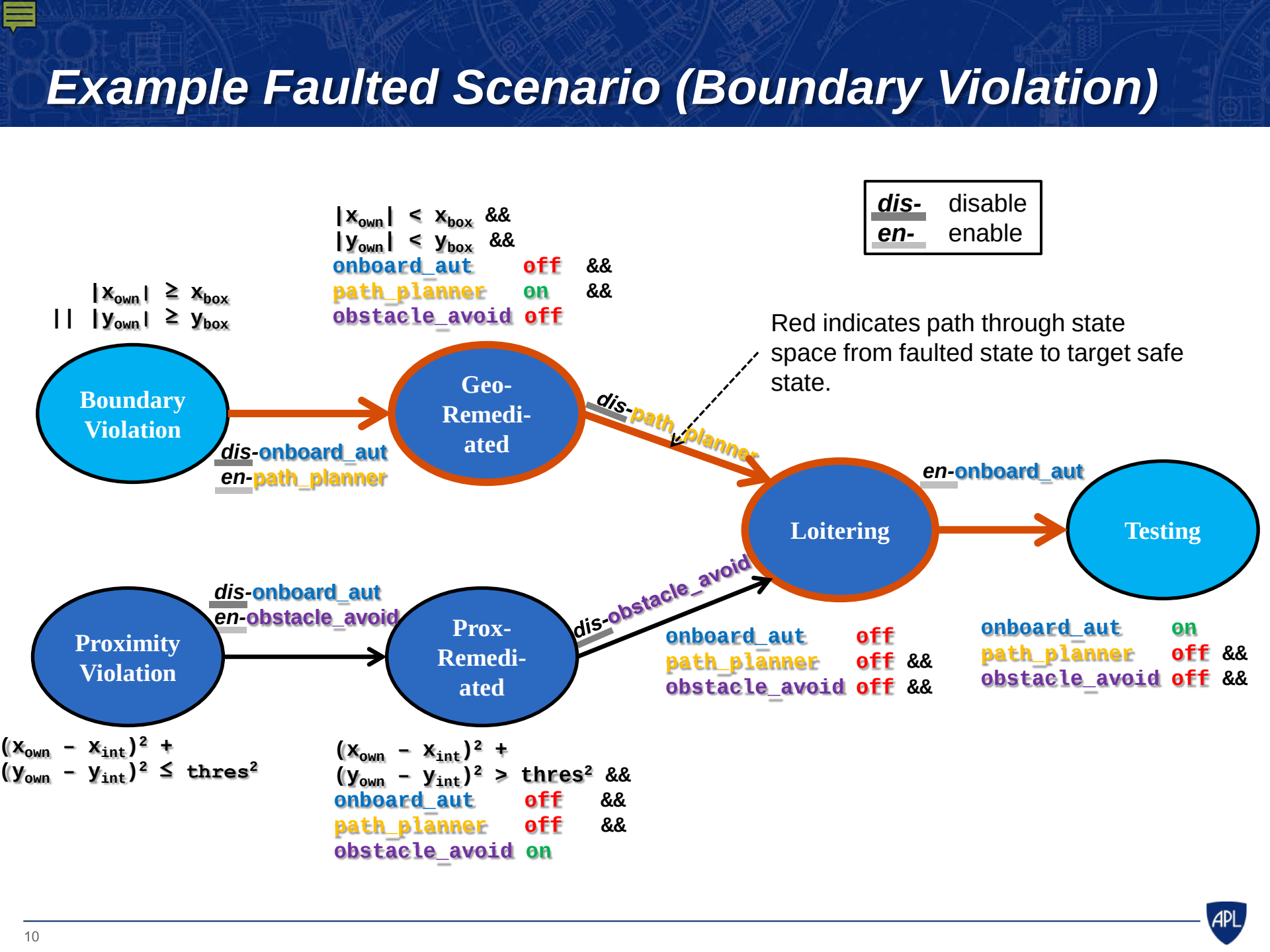
Simple ModelGraph Example

Simple remediation model accounting for two potential violations.

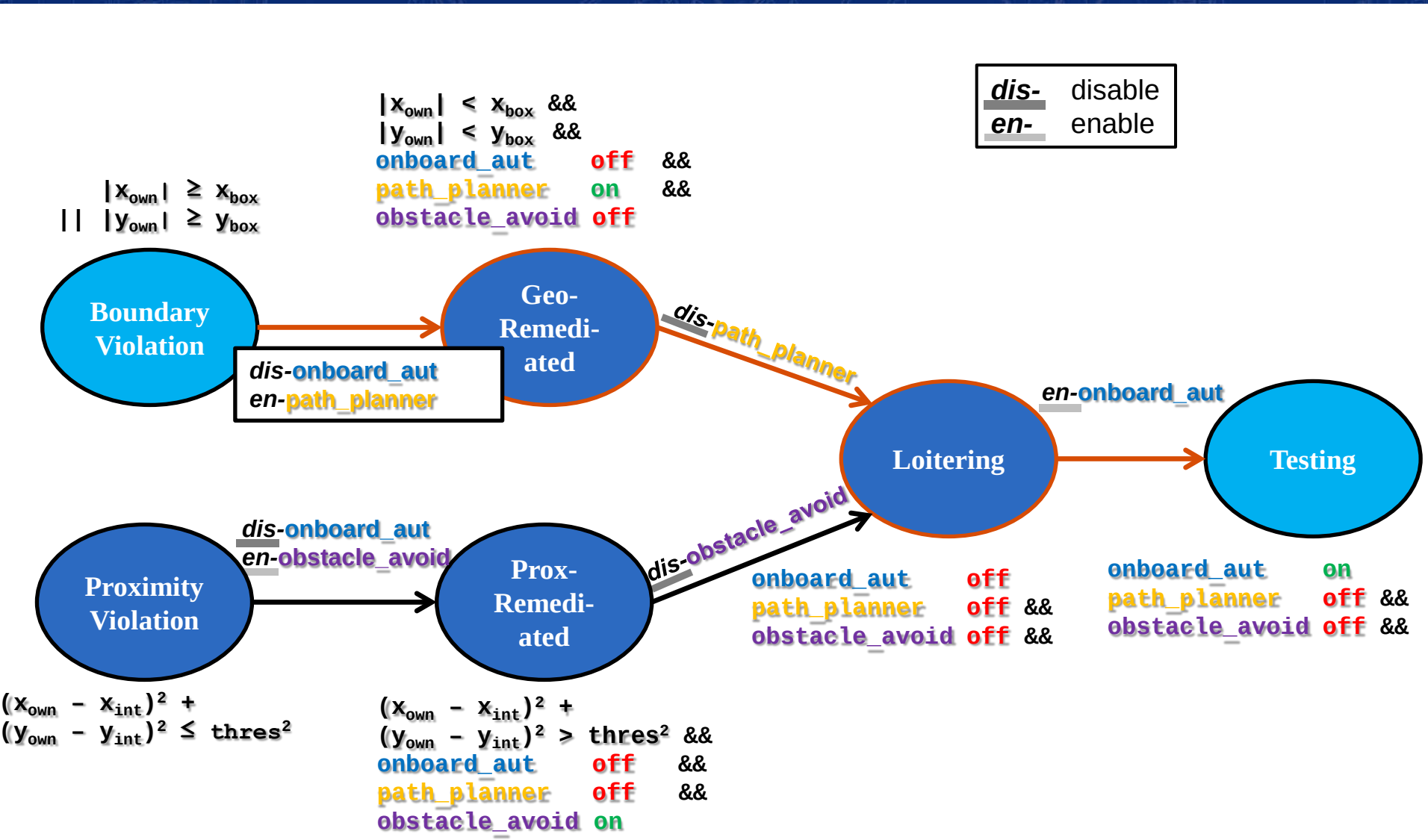
- *Violation of test range boundary*
- *Proximity violation to another aircraft*



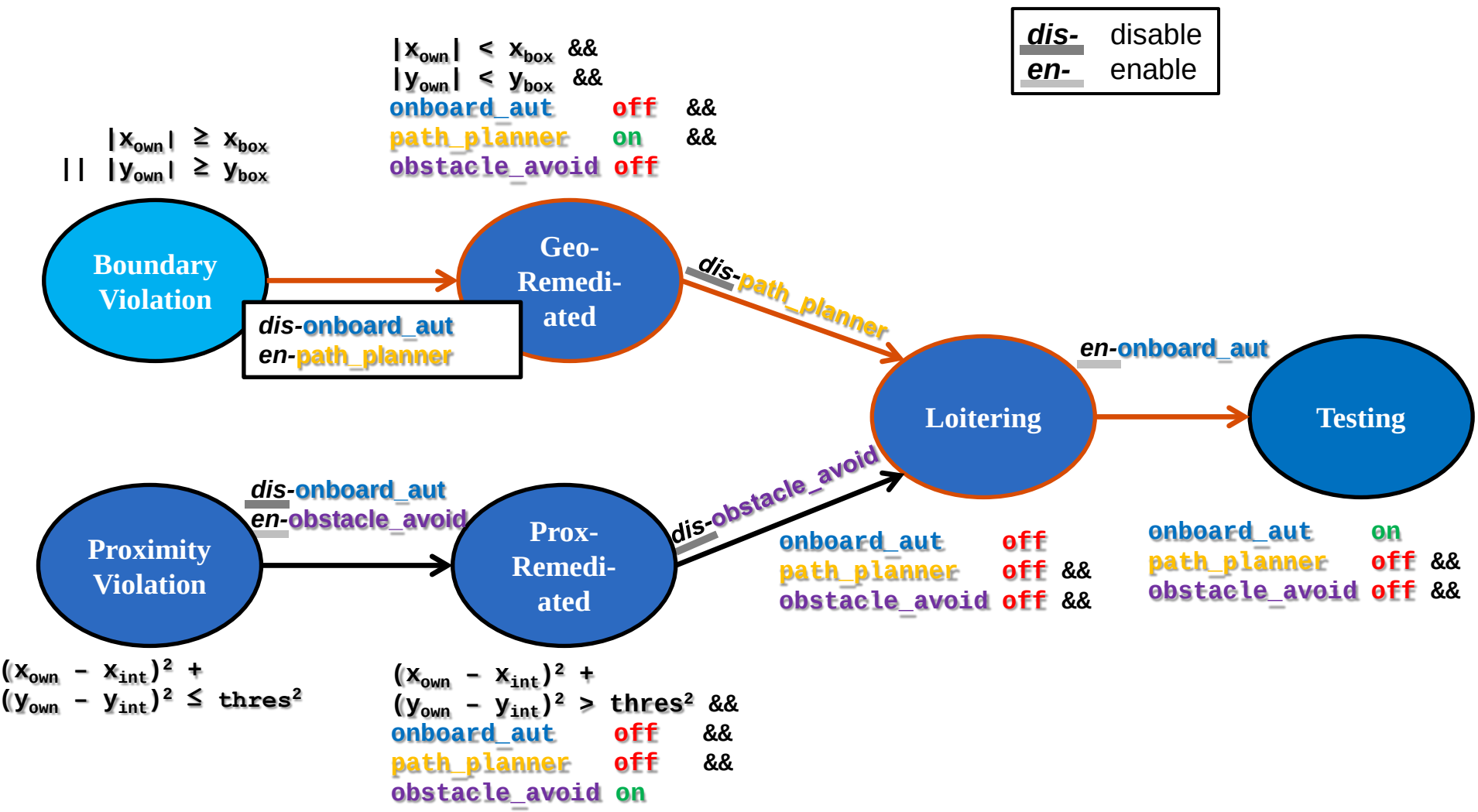
Example Faulted Scenario (Boundary Violation)



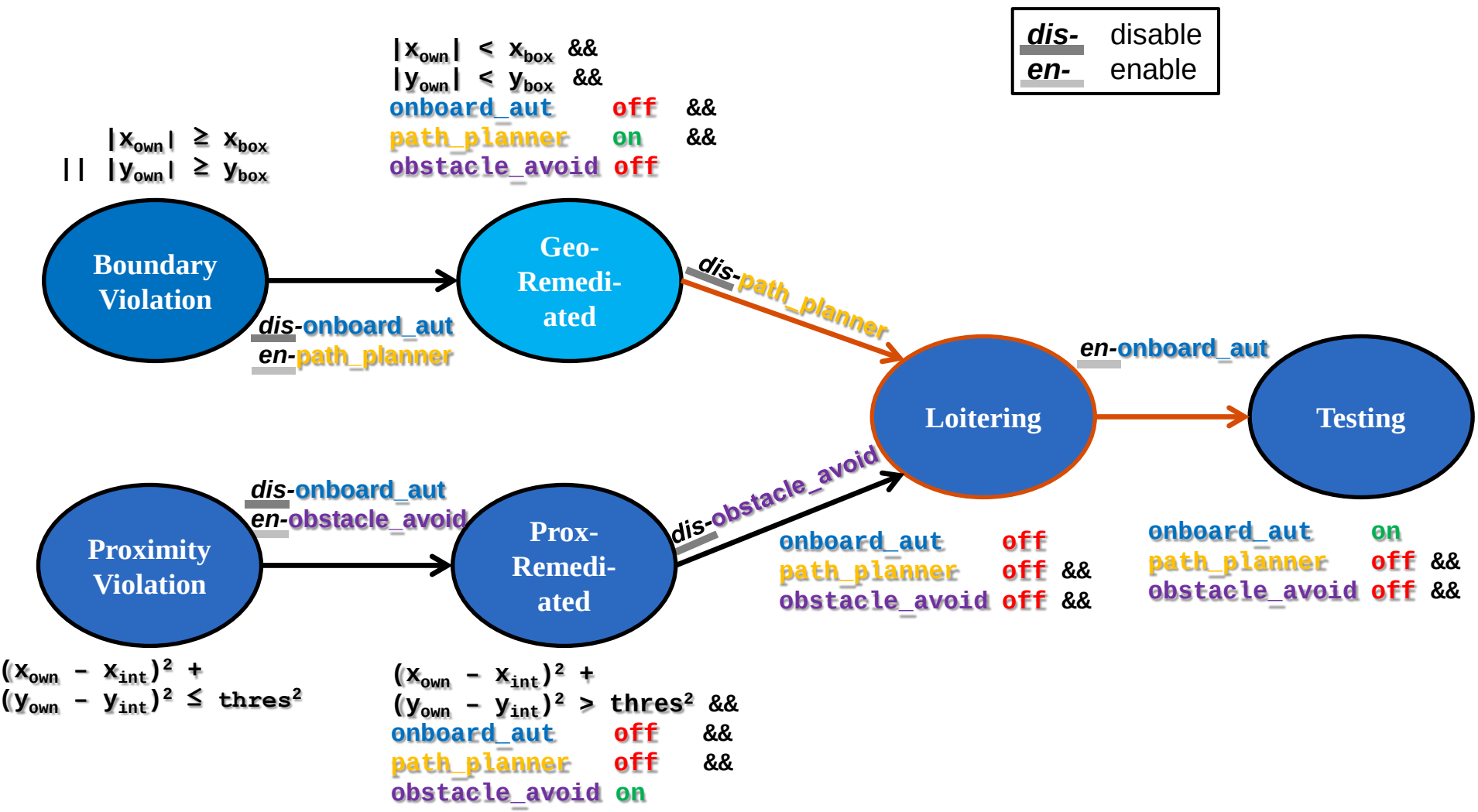
Boundary Violation Example Model



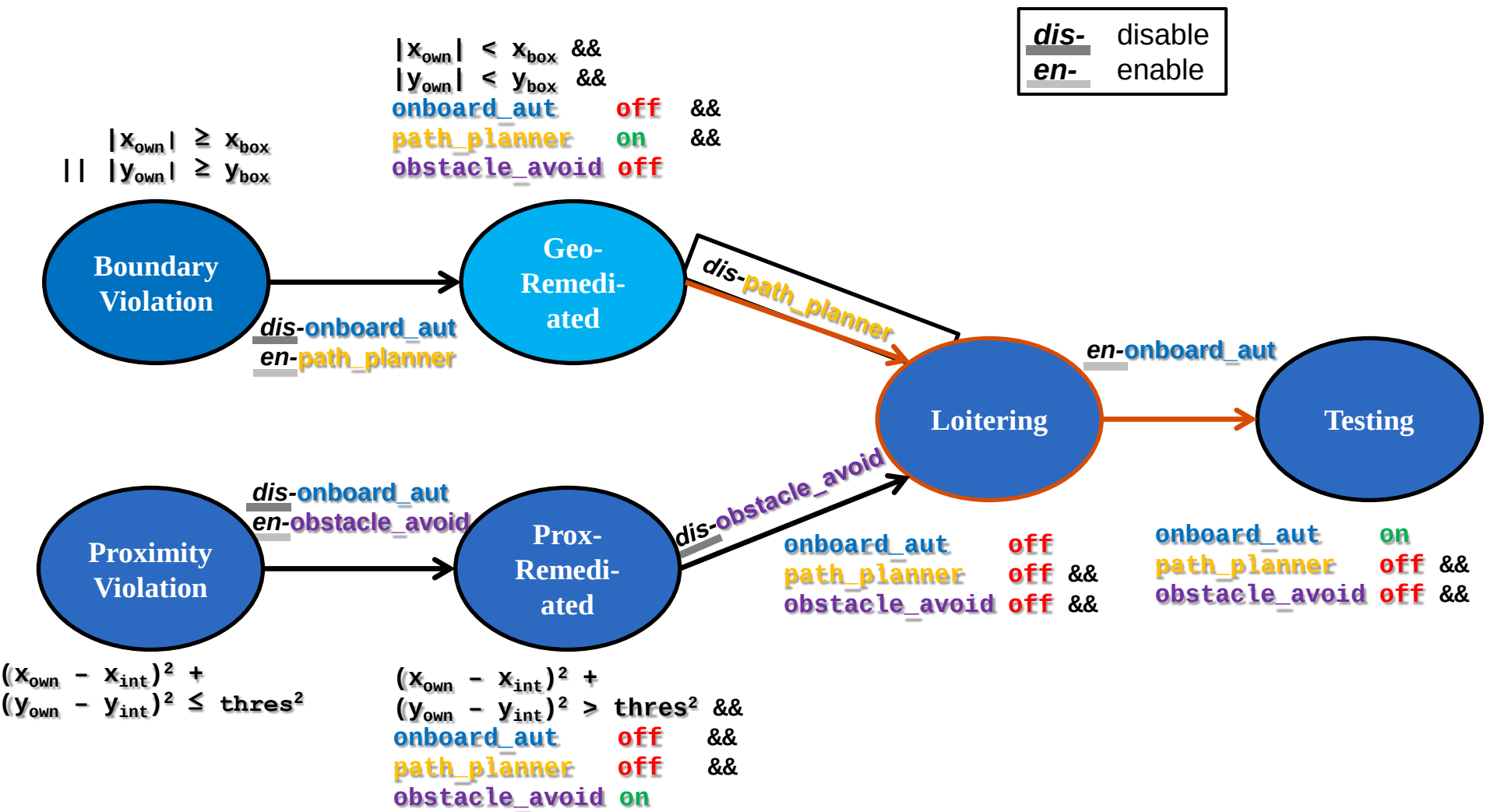
Boundary Violation Example Model



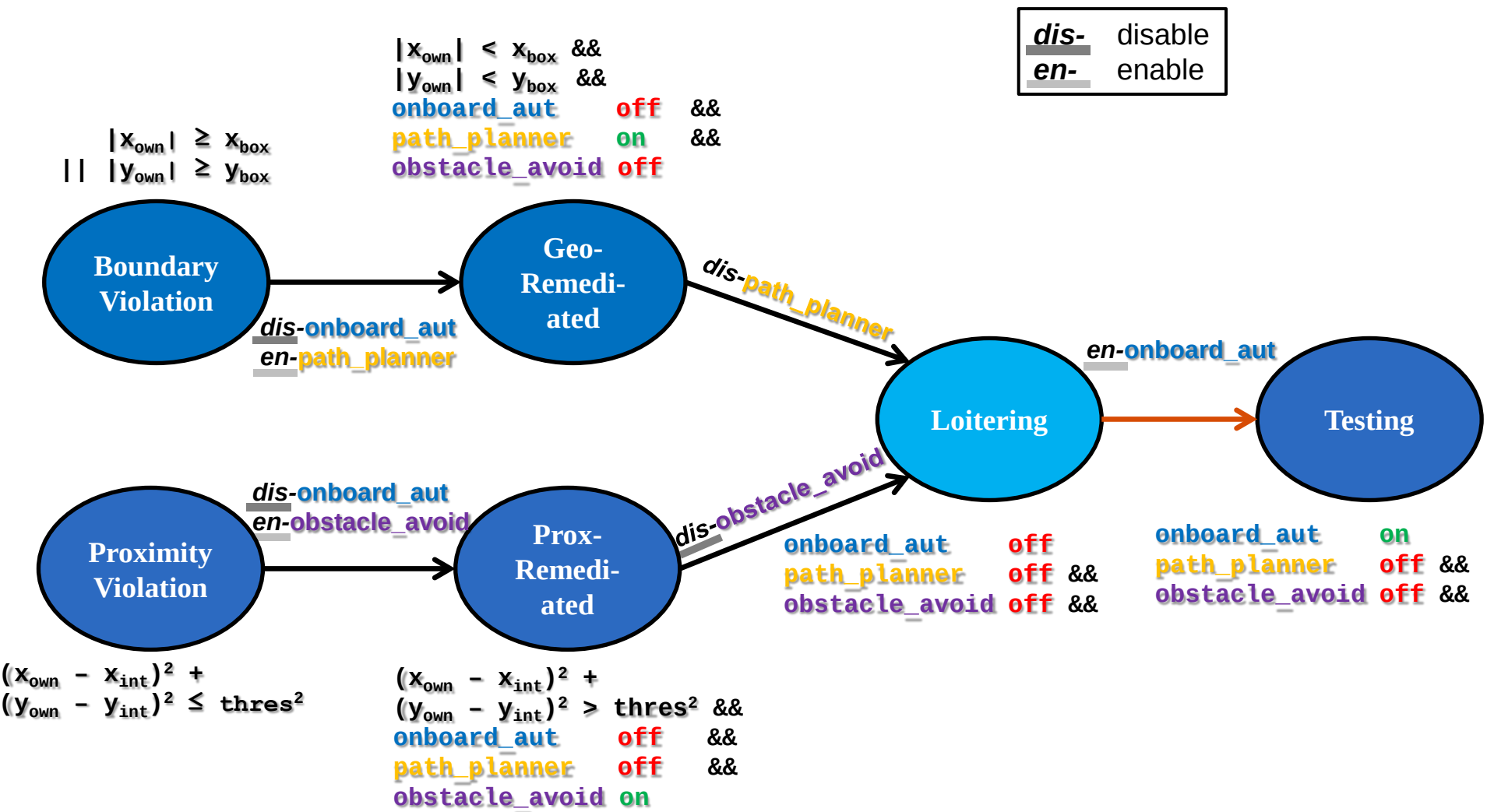
Boundary Violation Example Model



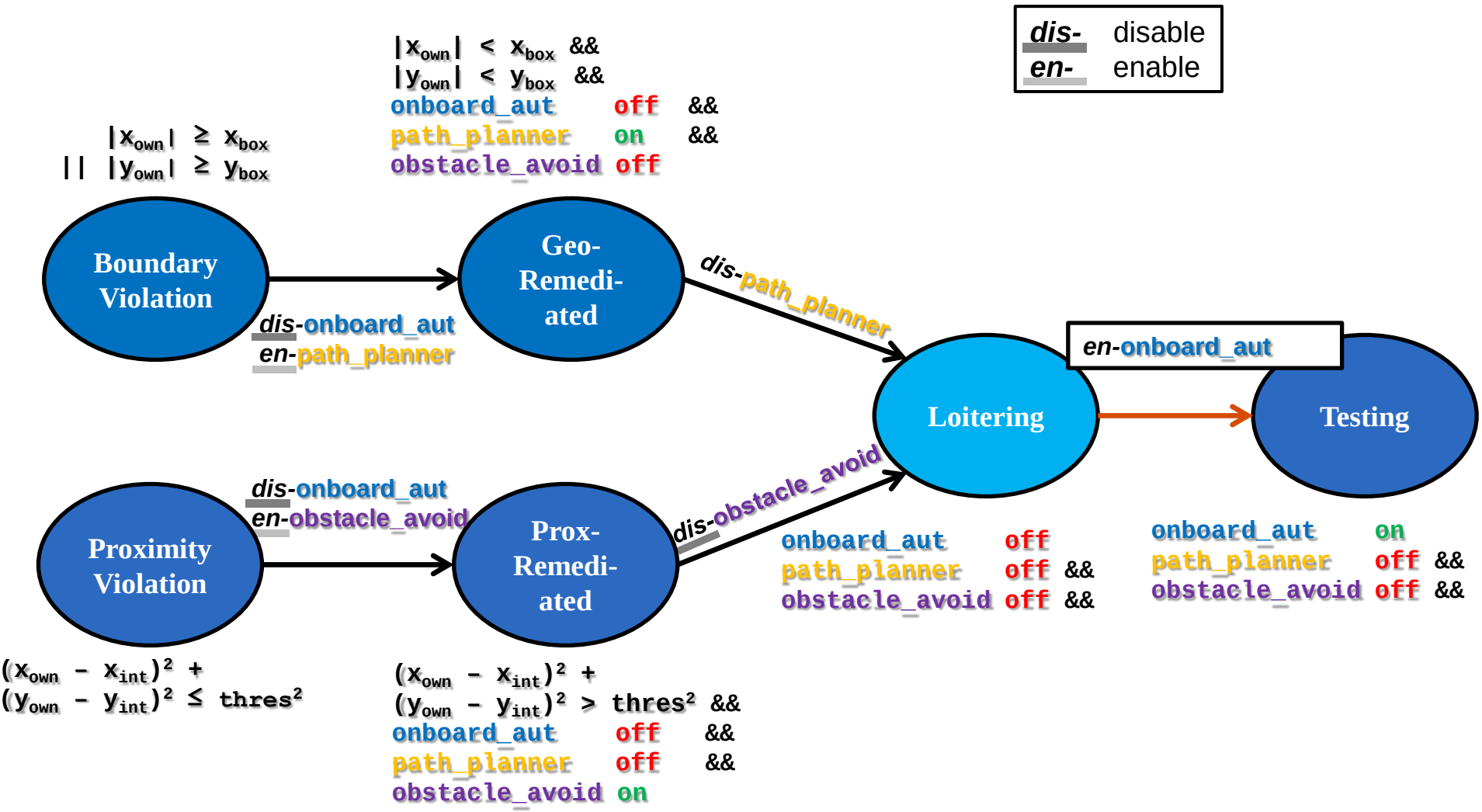
Boundary Violation Example Model



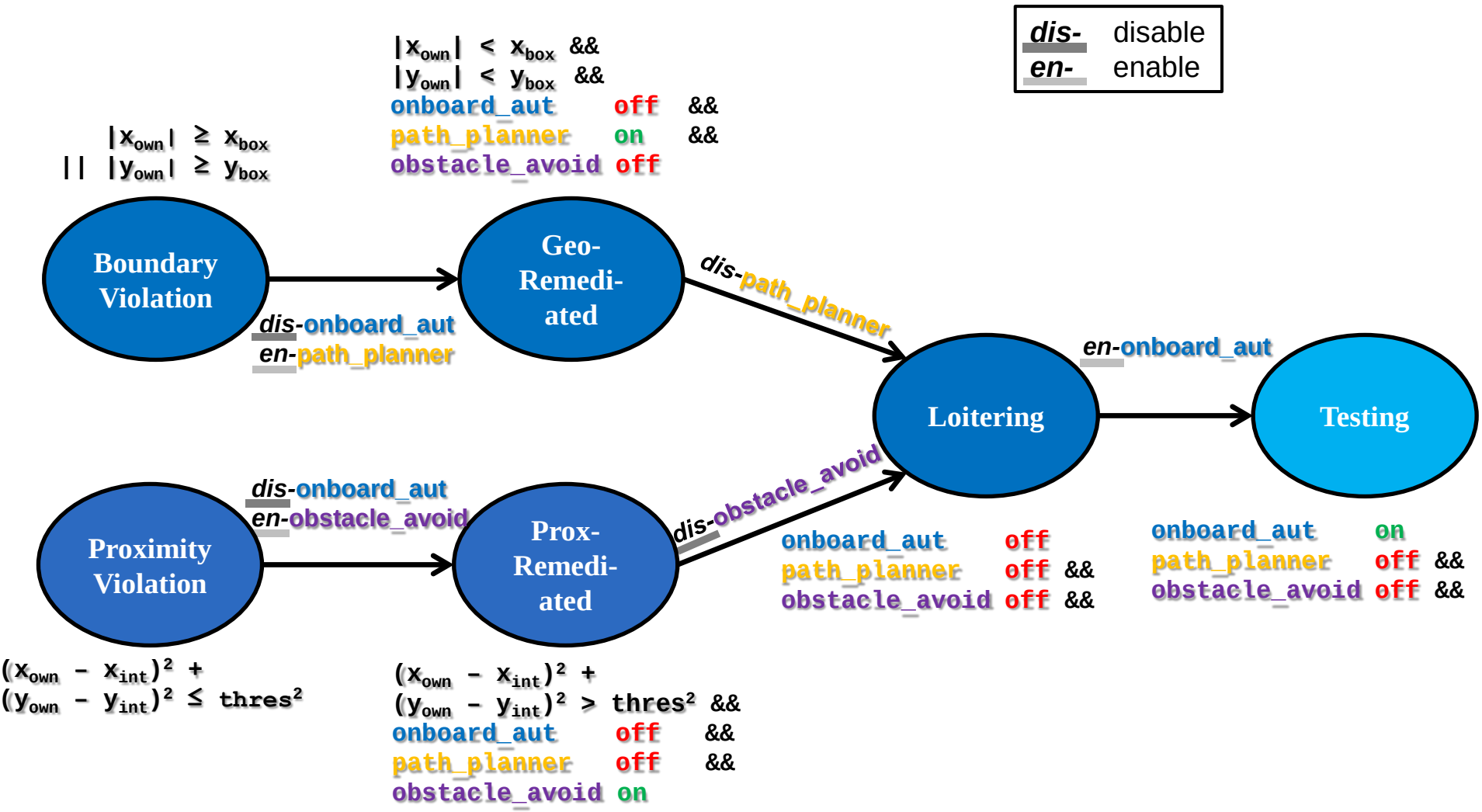
Boundary Violation Example Model



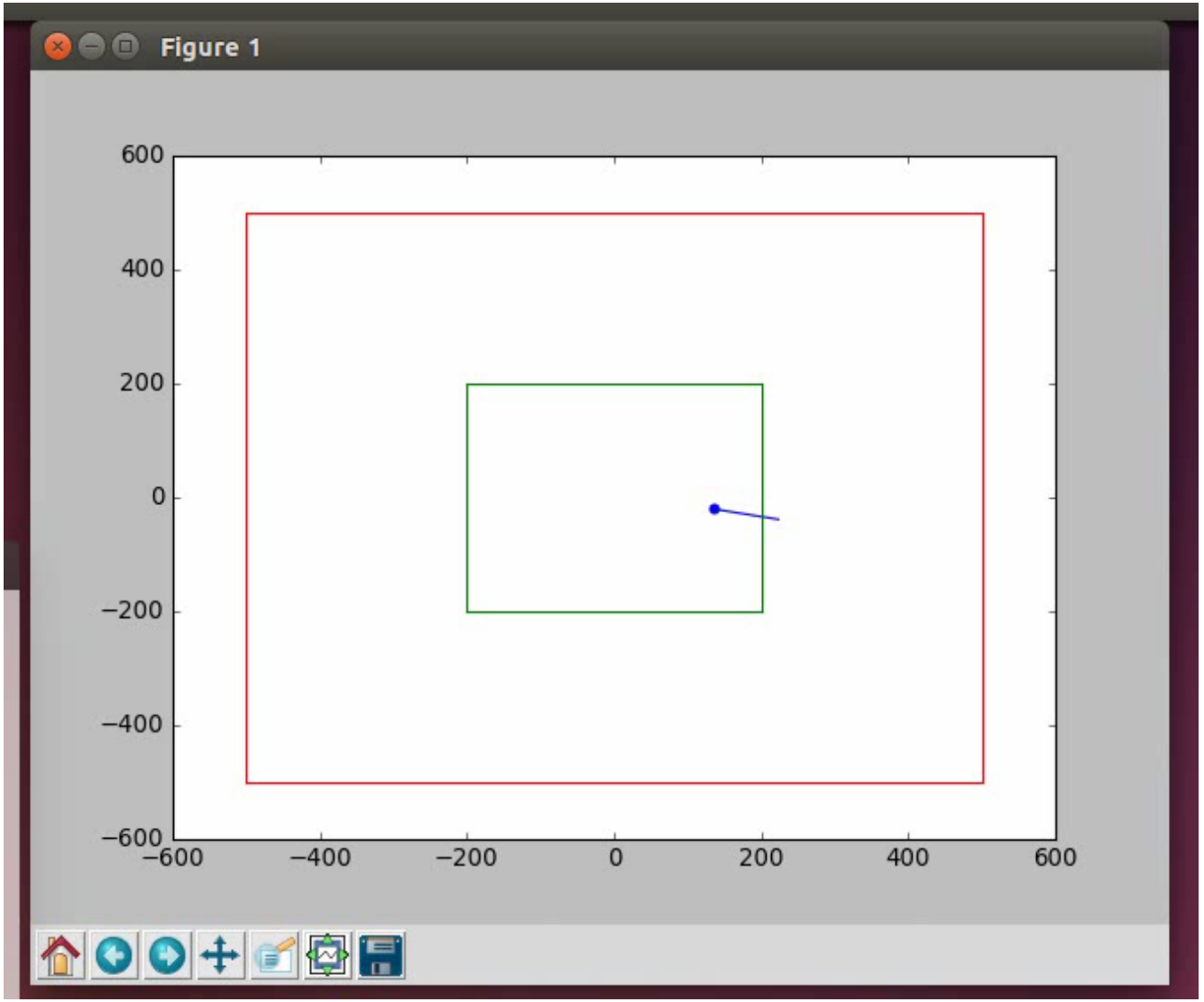
Boundary Violation Example Model



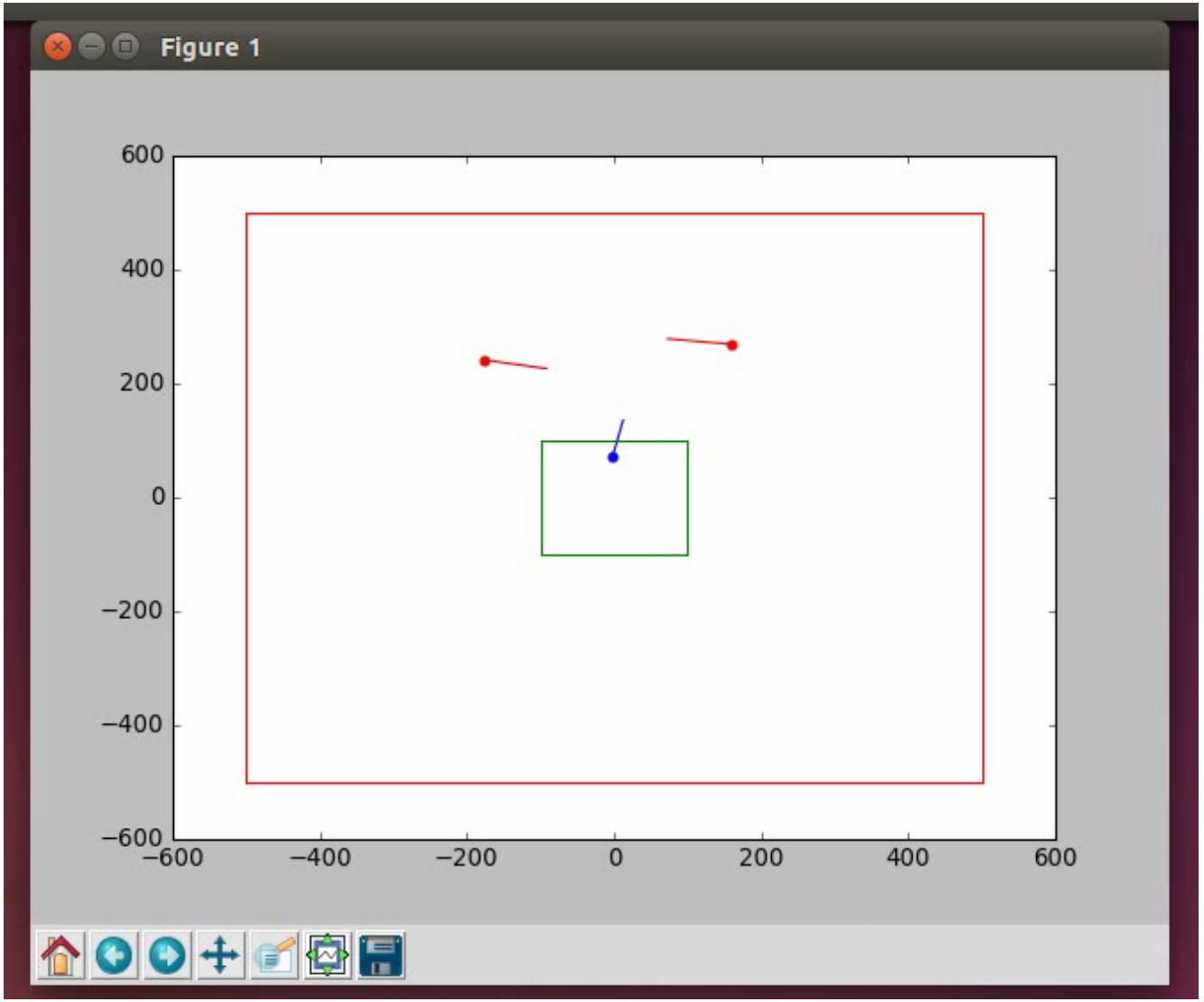
Boundary Violation Example Model



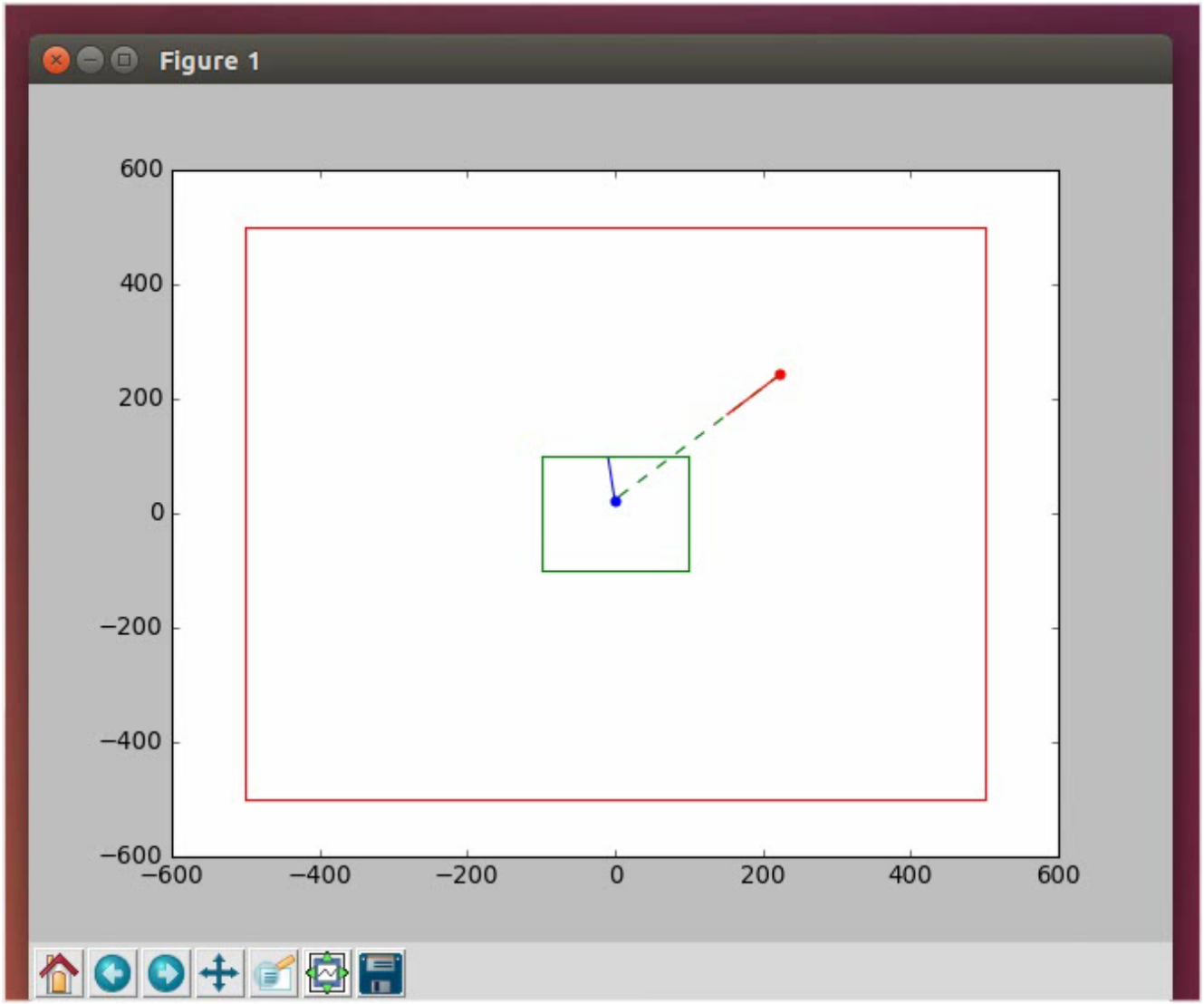
Single Fault Scenario



Multi-Fault Scenario



Stress-Test Adversarial Multi-Fault Scenario



Formal Verification Approach

- Motivation
  - The precise behavior of the Watchdog may be configured and specified for each test or unique test range conditions.
  - Logical errors in model specifications are likely to occur on occasion and could be dangerous.
  - During early testing we encountered an unexpected fault in which there was no active state, resulting in the *simulated* aircraft being adrift in the air.
  - This is something that could have been caught by applying formal methods to the model.

- Approach
  - Automatically translate each custom watchdog model to logical verification conditions corresponding to desired correctness properties.
    - E.g., The watchdog always has at least one valid active state.
    - E.g., The watchdog can potentially reach any possible goal state from any current state.
  - Use a hybrid satisfiability solver to formally prove the verification conditions hold.

Next Steps

- Determine best-practices for developing remediation ModelGraphs.
  - How specific/broad to be defining states and their invariants.
  - Defining command dictionaries in a way that makes them easily reversible.
  - Identify good design patterns and poor constructs.
- Stress test with more complex remediation models, more agents.
  - Regressions with fairly simple remediations.
  - Increase scenario complexity – Multiple SUTs, intricate range boundaries, full WFN.
- Make final determination to VFSM vs. ModelGraph as Watchdog core algorithm.
- Transition code to a more flight-quality implementation.
  - Embedded ANSI C implementation for integration to flight hardware.

Acknowledgements

- Support was provided by the Test Resource Management Center (TRMC) under contract W900KK-13-C-0036 Unmanned & Autonomous System Test (UAST) Test Technology Area
  - Vernon Panei and Kris Melton
  - Multiple NAVAIR Test Range Personnel
- JHU/APL Team
  - Corey Lowman, Bill Van Besien, Kristi Ramachandran, SW Team
  - Dave Scheidt, *Principal Investigator*
  - Bill D'Amico, *Project Manager*


APL
JOHNS HOPKINS
APPLIED PHYSICS LABORATORY