isr institute for SOFTWARE RESEARCH

Carnegie Mellon University

Safe Composition through Dynamic Feature Interaction Resolution




Ben Gafford, Tobias Dürschmid, Eunsuk Kang
CMU

Gabriel Moreno
CMU/SEI

High Confidence Software and Systems (HCSS) May 4, 2021

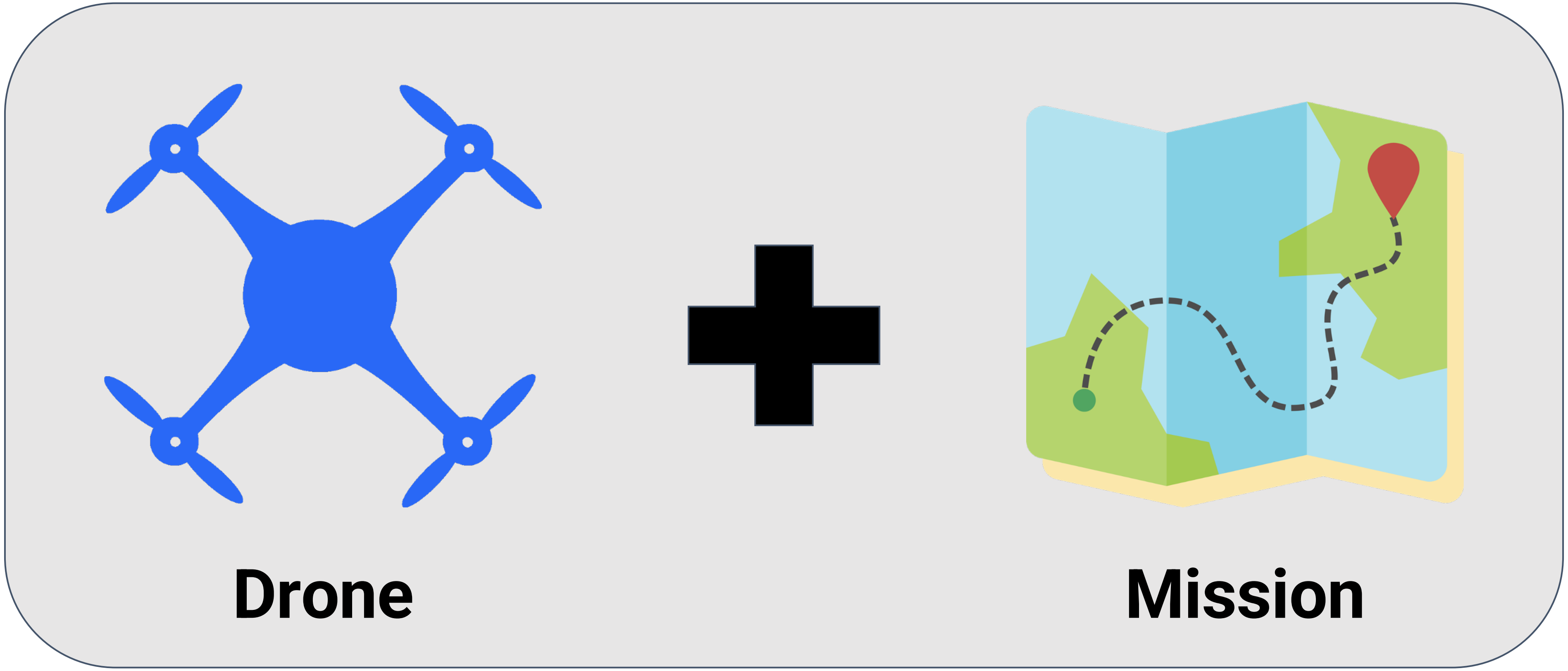
Feature Interaction Problem

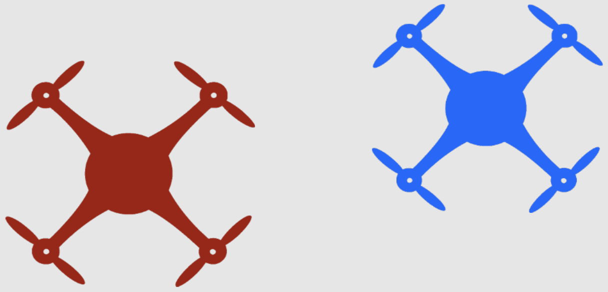
- Two or more features, developed independently, result in undesirable system behavior when composed together

$$F_1 \models G_1 \qquad\qquad F_2 \models G_2$$

$$F_1 \oplus F_2 \not\models G_1 \wedge G_2$$

Example: Autonomous Drones





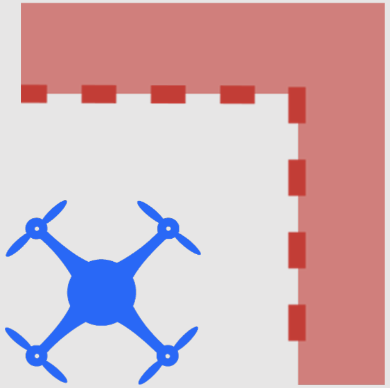
Runaway

Requirement:

Stay > 4 meters from a follower drone

Feature action:

Adjust the direction & velocity to move away from follower



Boundary

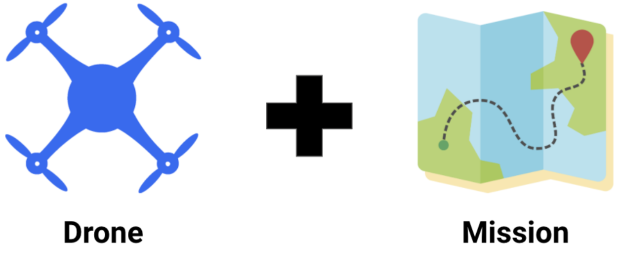
Requirement:

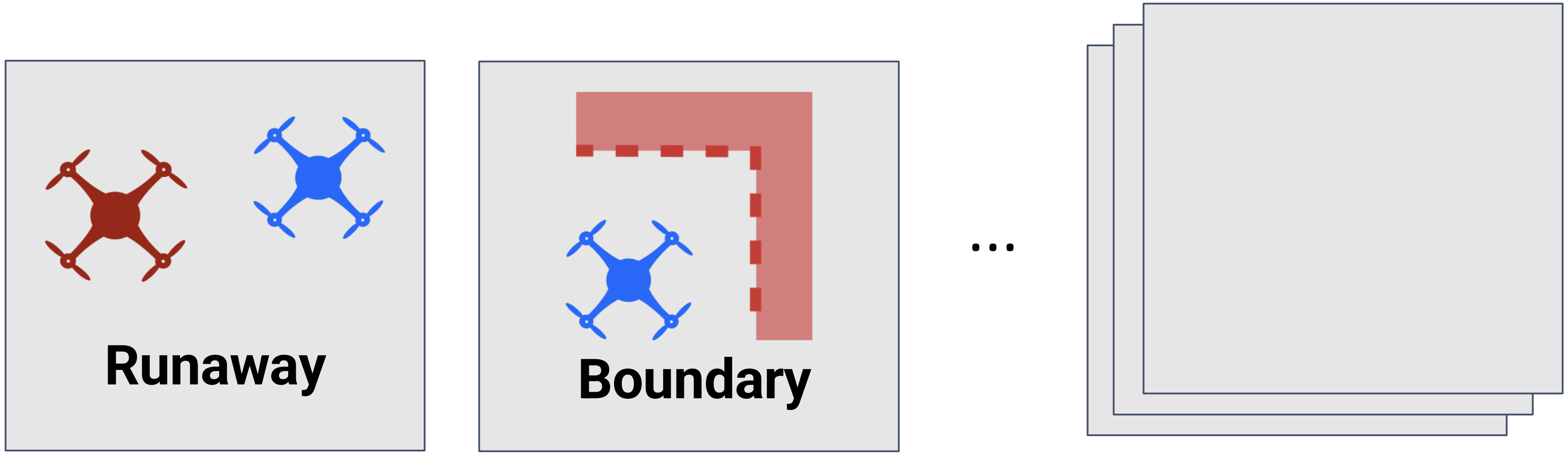
Maintain a time-to-collision of > 3.0 seconds to boundary

Feature action:

Adjust the direction & velocity to move away from the boundary

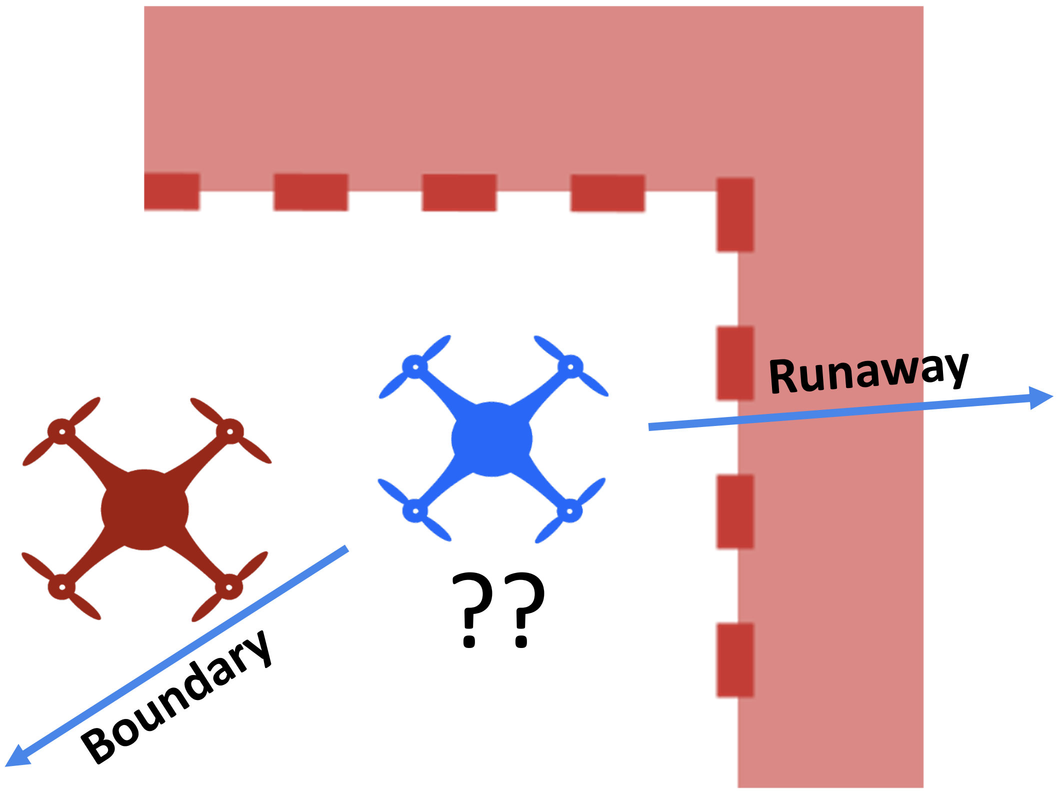
Feature-Oriented Design

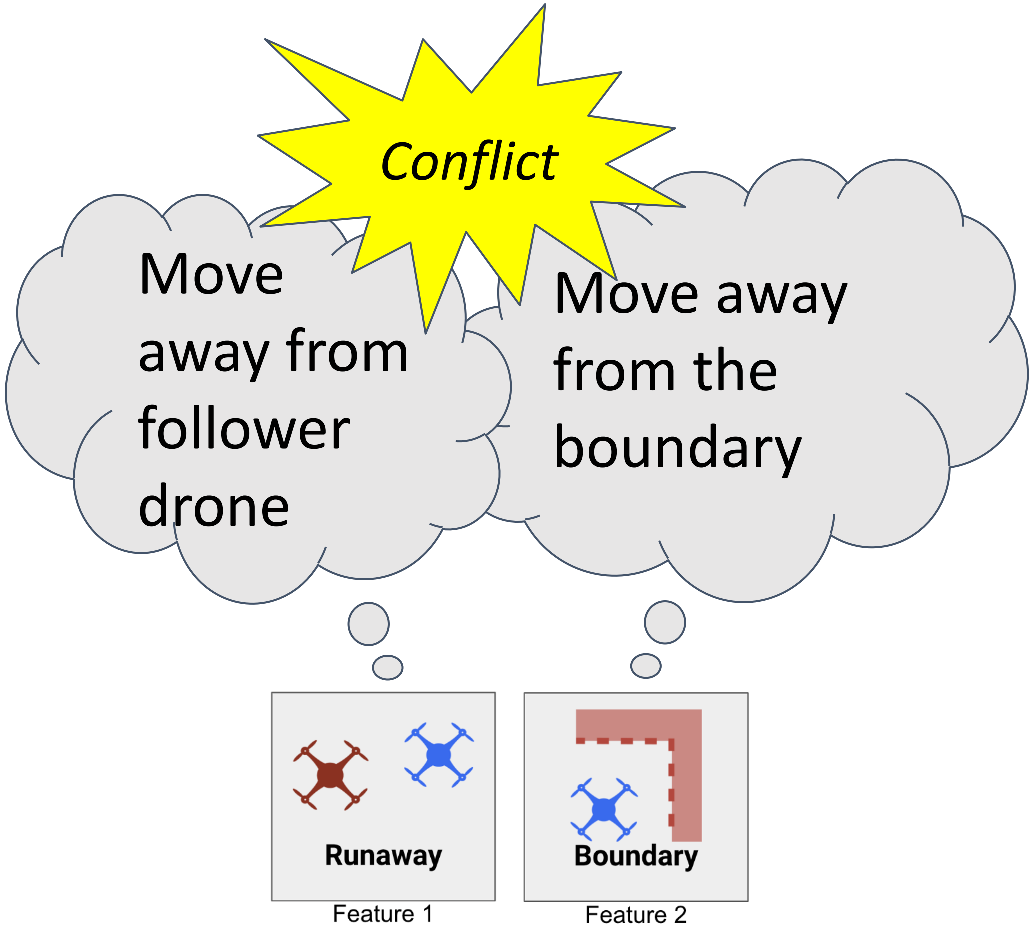




Feature 1 Feature 2 Many other features

What do we do when features conflict?





Feature Interaction Problem

- Well-studied problem in certain domains
  - Telecommunications
  - Software product lines
- But increasingly important in emerging domains
  - Autonomous systems, IoT
  - Open systems with dynamically evolving features
    - More possibilities for unanticipated interactions!
  - Possible safety failures due to undesirable interactions
- A major obstacle to safe system composition!

Research Questions

- Detection: How do we detect undesirable interactions among a possibly large number of features?
- Resolution: How to resolve undesirable interactions when they occur?
  - Our focus today!

Existing Approaches

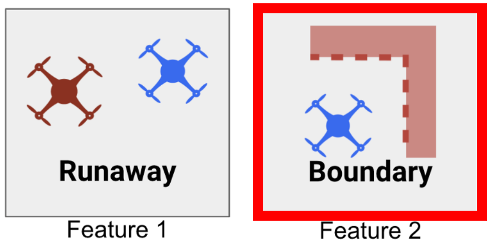
- Priority-based resolution
  - Rank features based on priorities & select the highest one during conflicts
  - Not robust to feature changes: Must update priority list when features added
- Variable-specific resolution
  - Design a resolution strategy for controlled variables in conflict
    - Conflicting actions for velocity: Select one with lowest velocity, since it's likely to be safer
  - Robust against feature changes, but may not produce desirable outcome in unanticipated contexts
- Challenge: What does it mean for a feature to be "desirable"?

Towards Context-Driven Resolution

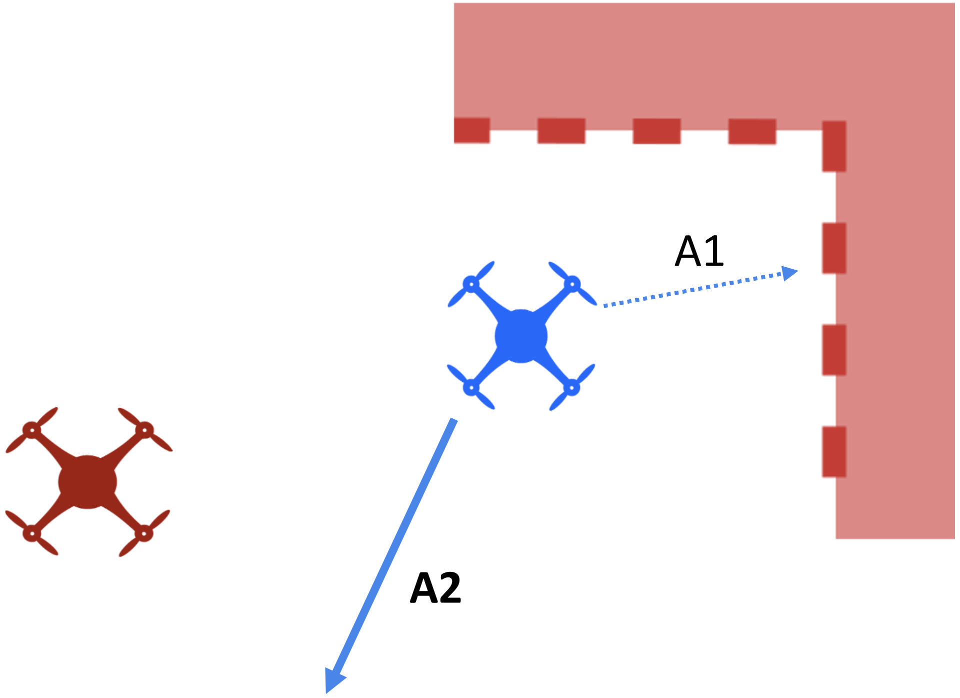
- Desirability of a feature is context-dependent
  - *"How well does this feature satisfy a system requirement in the current environmental context?"*

Towards Context-Driven Resolution

- Desirability of a feature is context-dependent
  - *"How well does this feature satisfy a system requirement in the current environmental context?"*

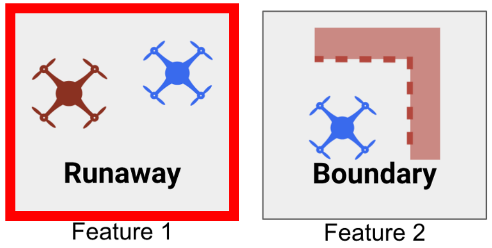


Higher risk of running into the boundary; choose A1

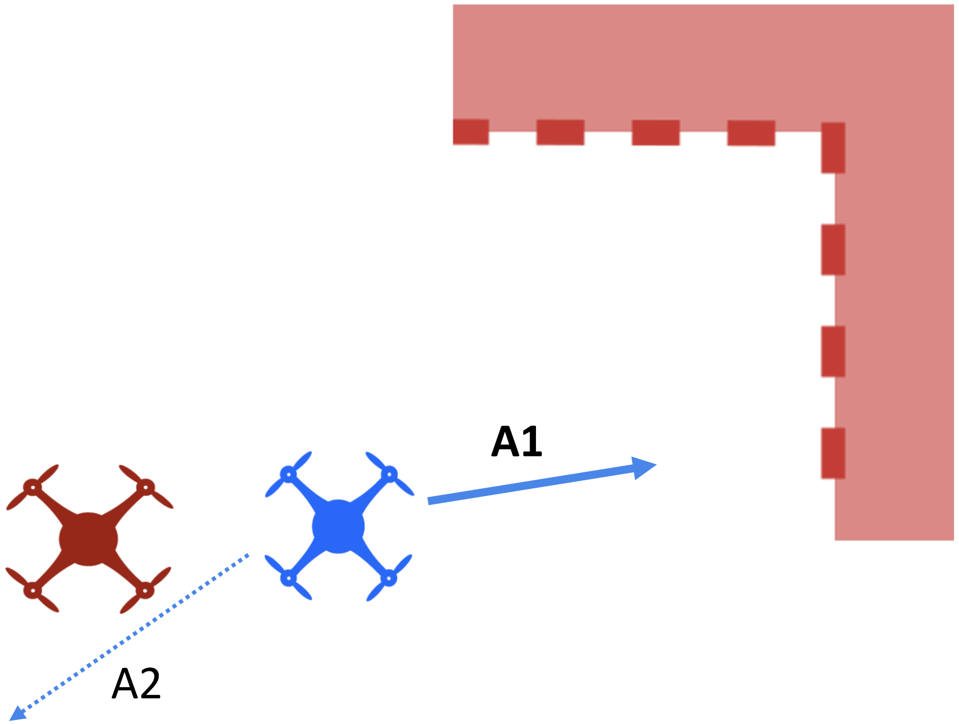


Towards Context-Driven Resolution

- Desirability of a feature is context-dependent
  - *"How well does this feature satisfy a system requirement in the current environmental context?"*

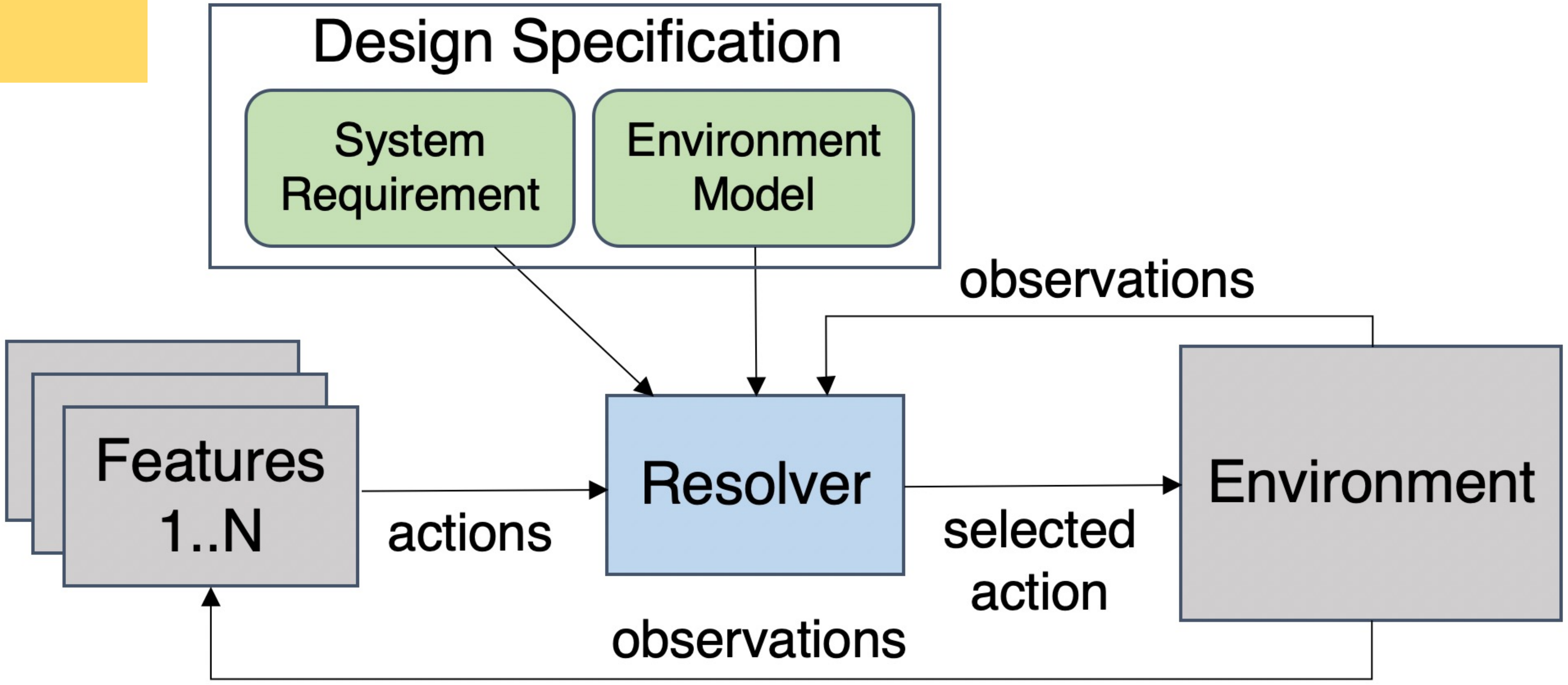


Higher risk of colliding with the follower; choose A2



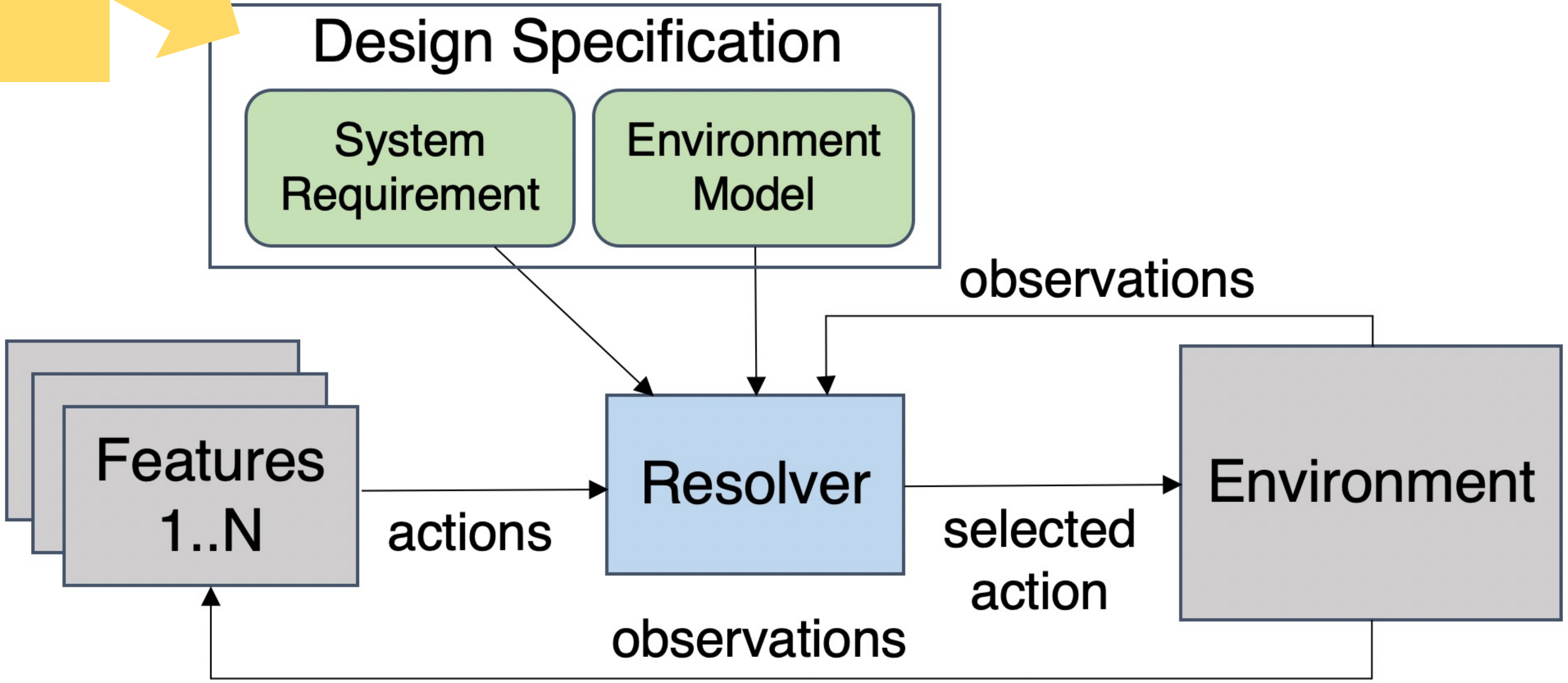
Idea #1: Requirement-Based Feature Evaluation

Evaluate given actions w.r.t. satisfaction of a requirement in the given environment



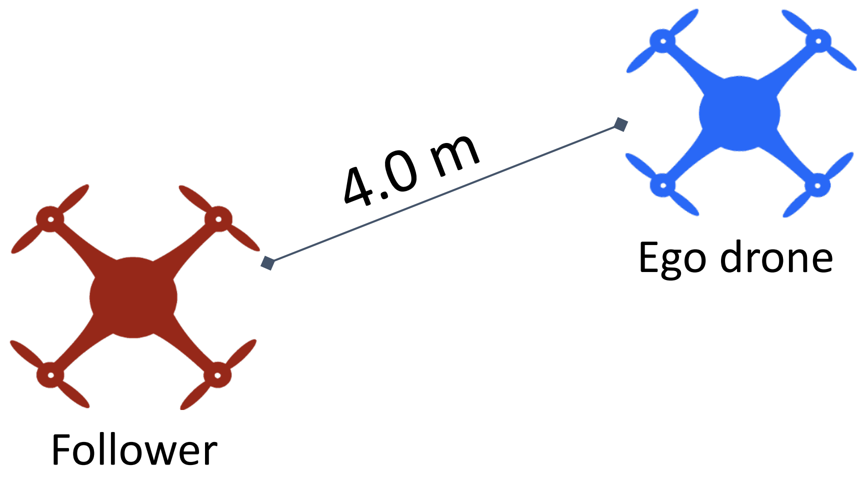
Idea #1: Requirement-Based Feature Evaluation

System requirement & environmental model as explicit parameters



Requirement-Based Feature Evaluation

- System requirement as Signal Temporal Logic (STL)
  - An extension of linear temporal logic w/ time intervals & continuous variables
  - Well-suited for specifying requirements in CPS



"The distance to a nearby drone must be at least 4.0 meters for the next 1 seconds"

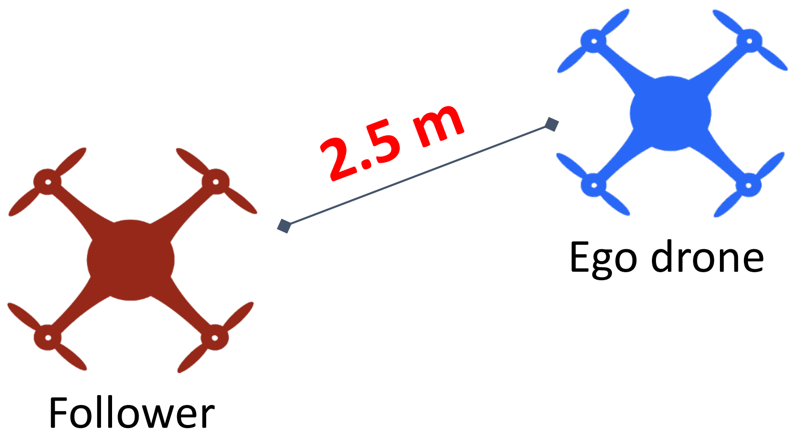
$$\text{Req} \equiv \mathbf{G}_{[0,1]}(\text{distToFollower}(s,t) - 4.0 \geq 0)$$

Signal (sequence of states)

Time

Robustness of Satisfaction

- A quantitative metric for the degree of satisfaction in STL
  - i.e., How much does the system satisfy or violate a property?



"The distance to a nearby drone must be at least 4.0 meters for the next 1 seconds"

$$\text{Req} \equiv \mathbf{G}_{[0,1]}(\text{distToFollower}(s,t) - 4.0 \geq 0)$$

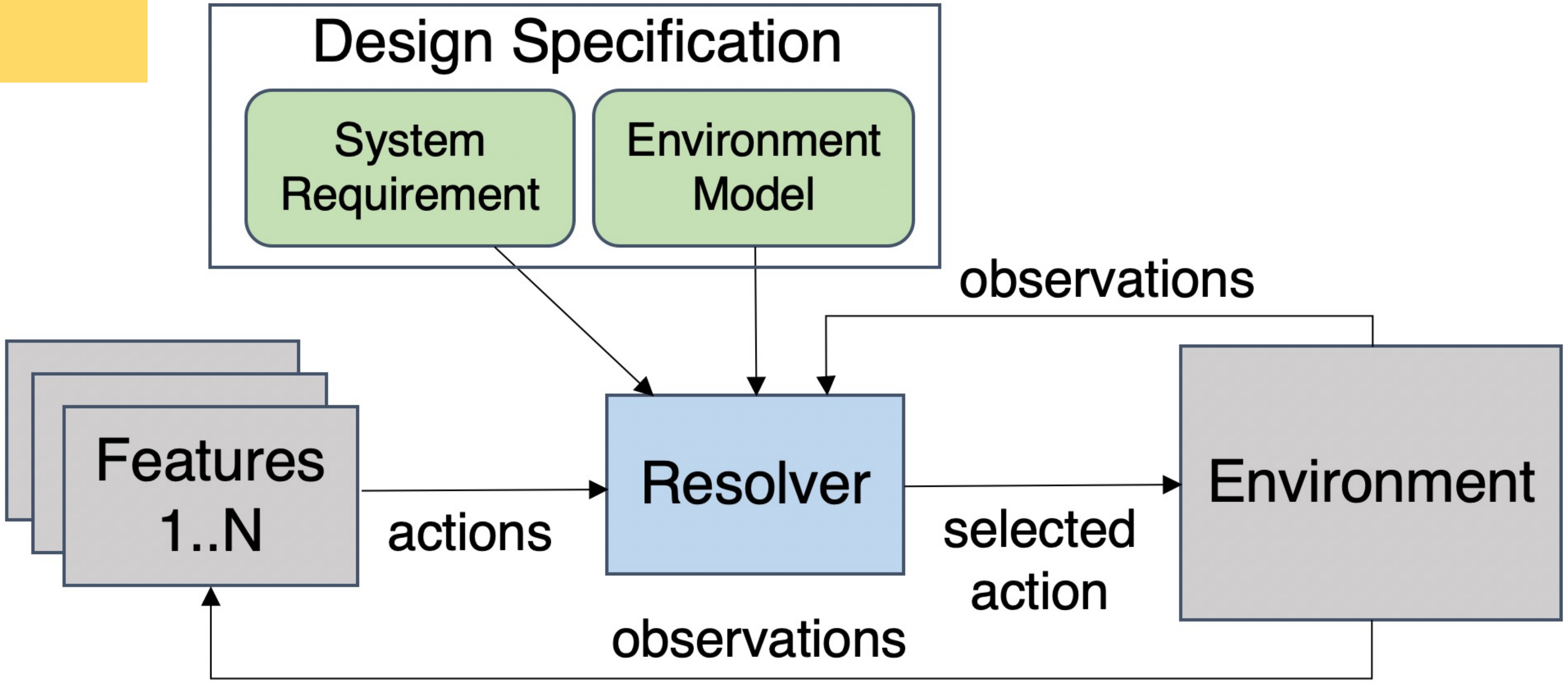
$$\rho(\text{Req}, s, t) = -1.5$$

Robustness of satisfying Req

A. Donze and O Maler. Robust satisfaction of temporal logic over real-valued signals. In Formal Modeling and Analysis of Timed Systems (FORMATS), 2010.

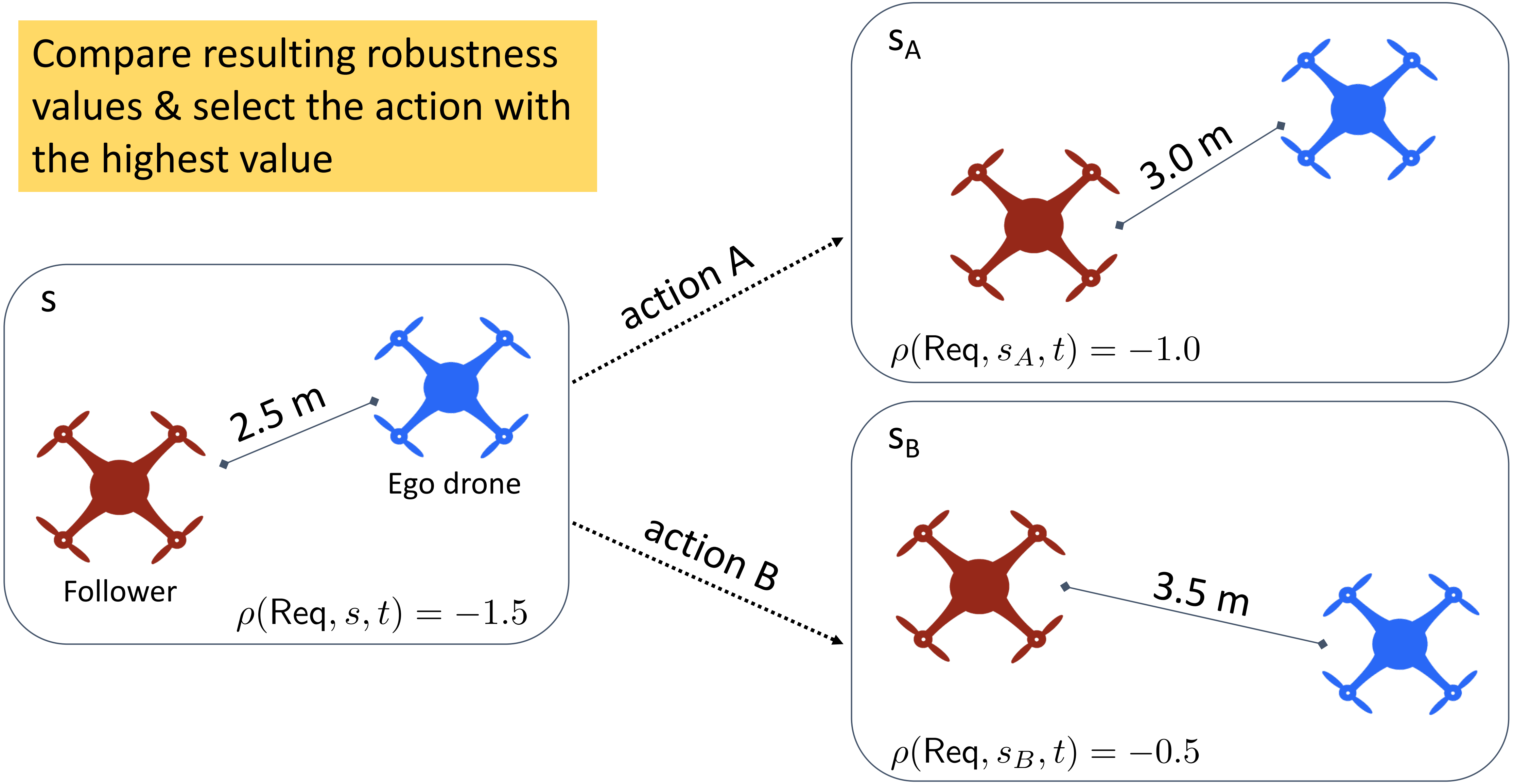
Idea #1: Requirement-Based Feature Evaluation

Evaluate given actions w.r.t. satisfaction of a requirement in the given environment



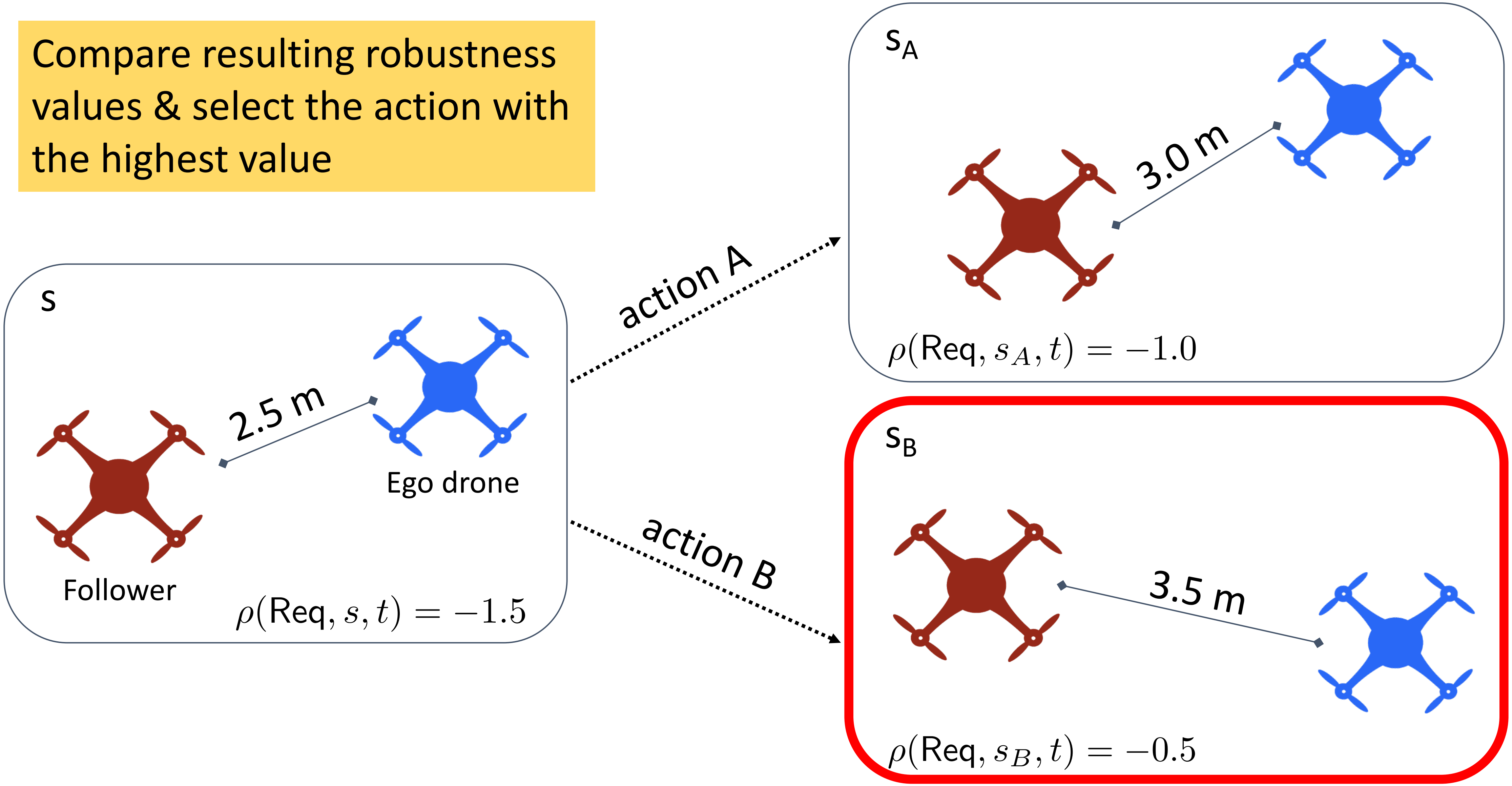
Evaluating Actions using Robustness

Compare resulting robustness values & select the action with the highest value

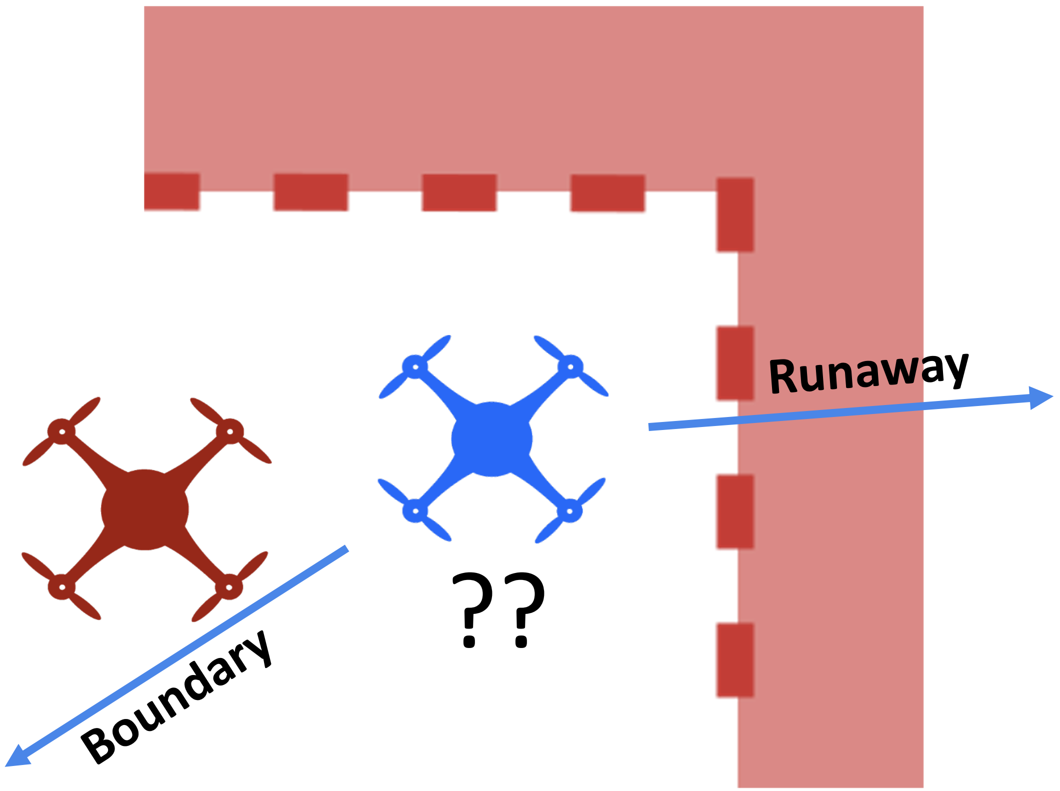


Evaluating Actions using Robustness

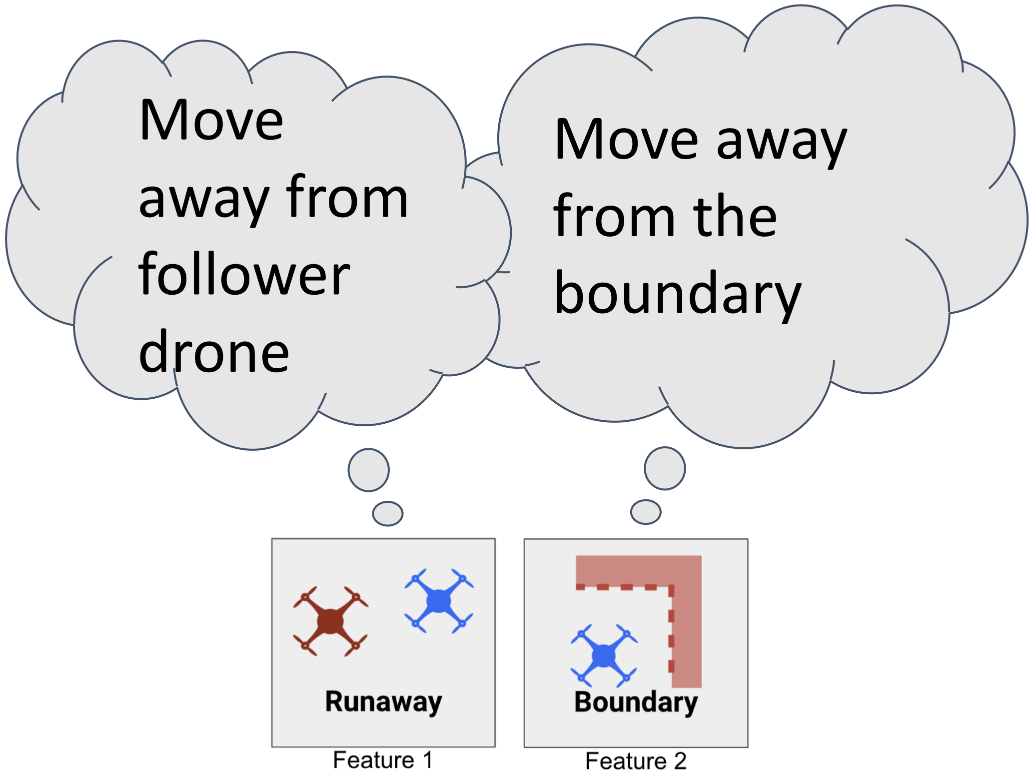
Compare resulting robustness values & select the action with the highest value



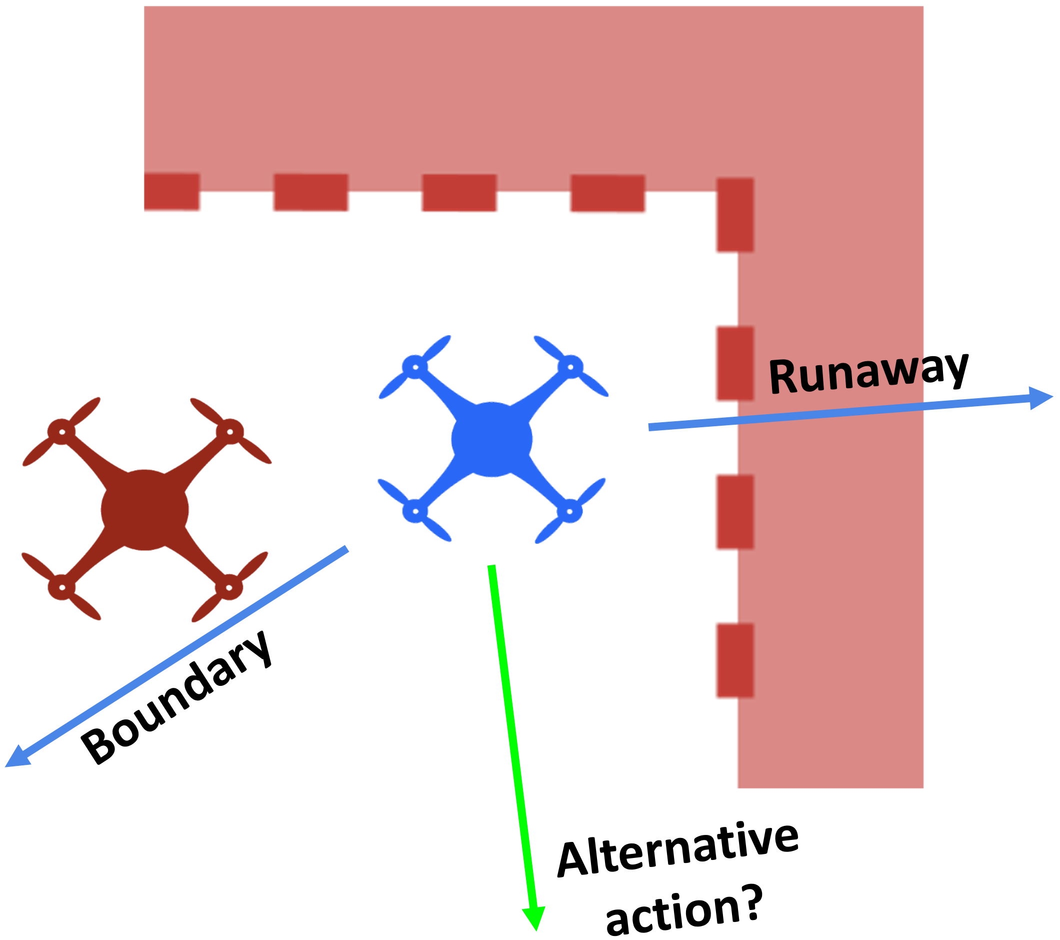
What if none of the given actions is desirable?



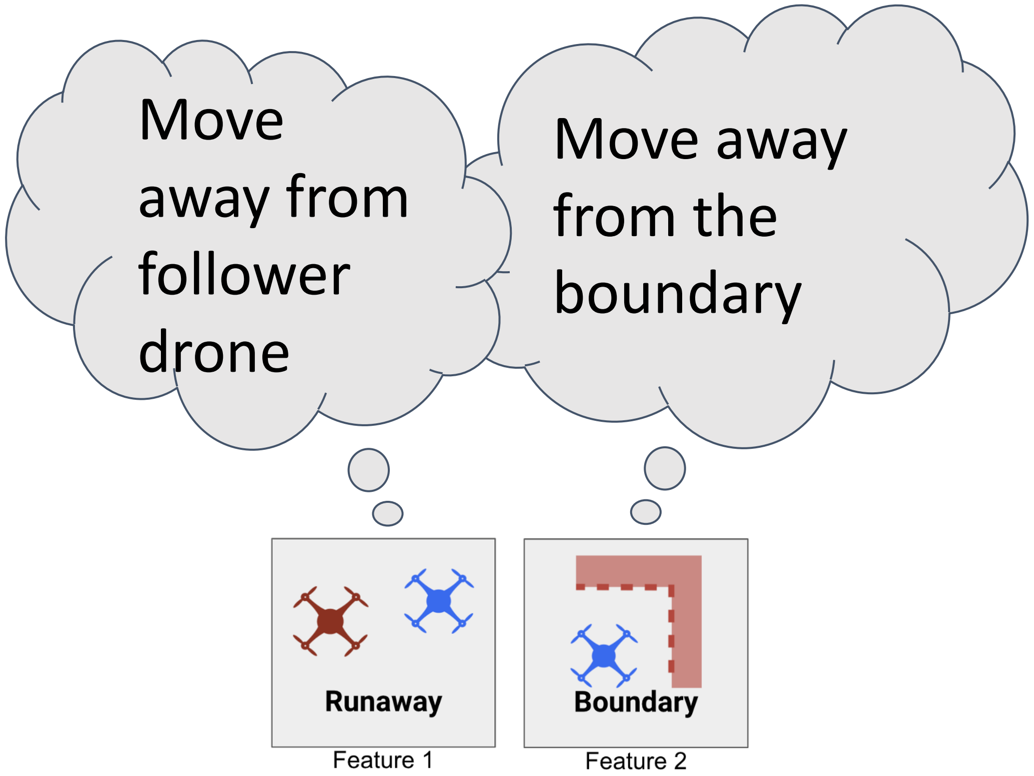
Inherently conflicting goals in this context!



Idea #2: Resolution through Action Synthesis

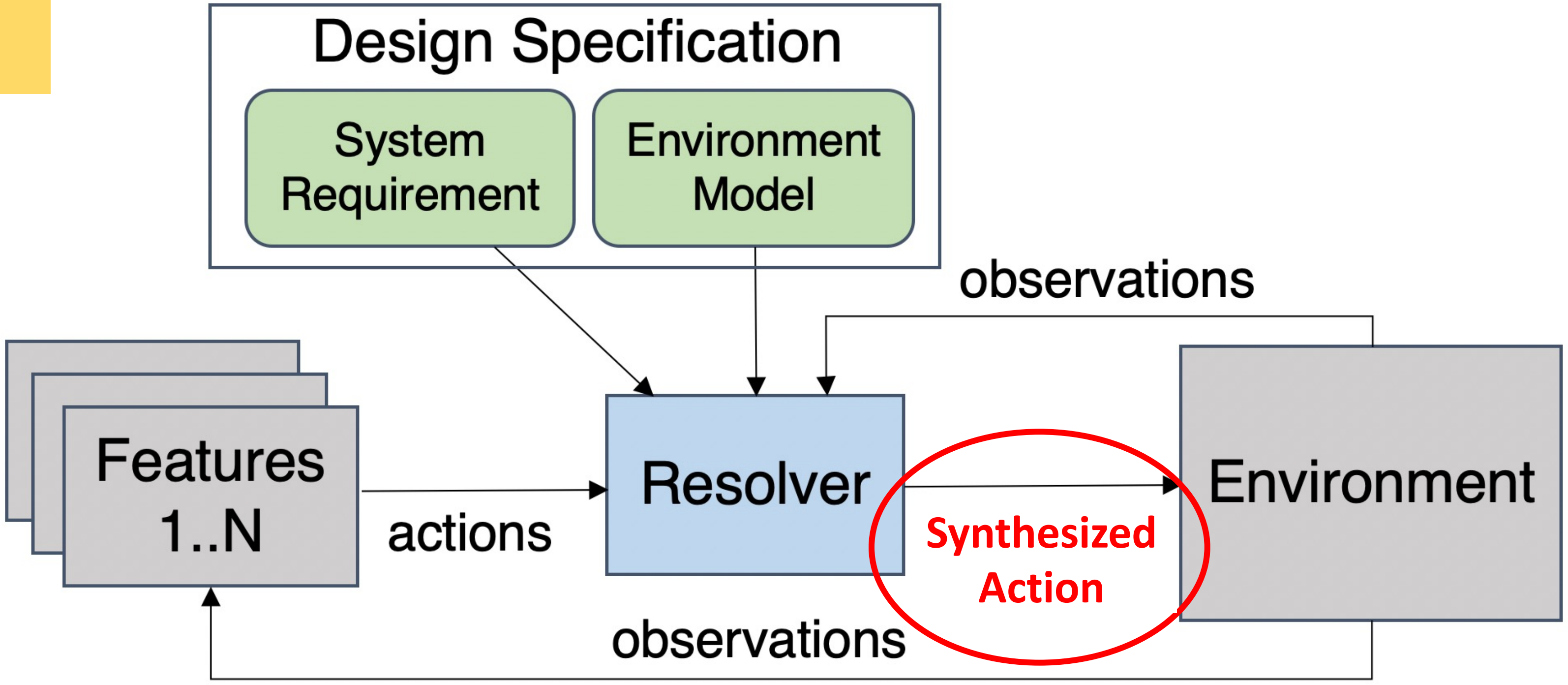


Is there an action that satisfies the goals of both features?

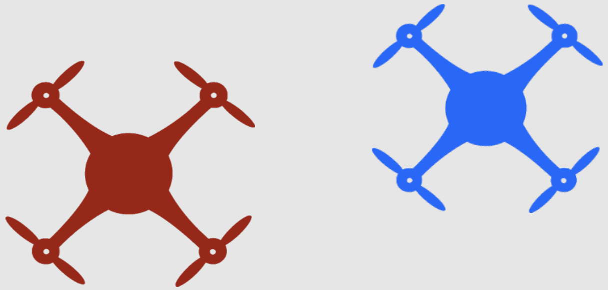


Idea #2: Resolution through Action Synthesis

If none of the actions are satisfactory, *synthesize* an alternative action

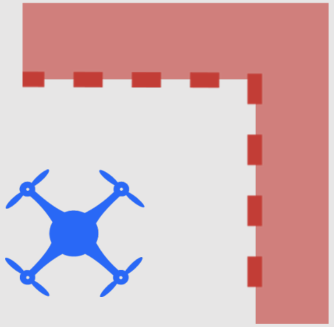


Global System Robustness



Runaway

Requirement: Stay > 4 meters from a follower drone



Boundary

Requirement: Maintain a time-to-collision of > 3.0s to boundary

$$R_{\text{runaway}} \equiv \mathbf{G}_{[0,1]}(\text{distToFollower}(s,t) - 4.0 \geq 0)$$

$$R_{\text{boundary}} \equiv \mathbf{G}_{[0,3]}(\text{timeToObstacle}(s,t) - 3.0 \geq 0)$$

Global robustness:

$$\rho_{sys}(s,t) = \rho(R_{\text{runaway}}, s, t) + \rho(R_{\text{boundary}}, s, t)$$

Global System Robustness

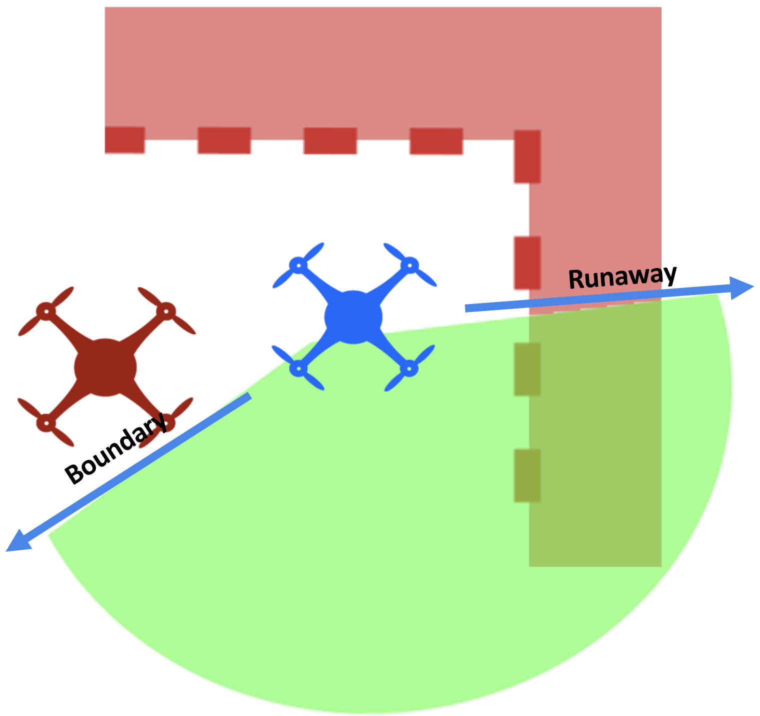
Global robustness:

$$\rho_{sys}(s,t) = w_1\rho(R_1,s,t) + w_2\rho(R_2,s,t) + \ldots + w_n\rho(R_n,s,t)$$

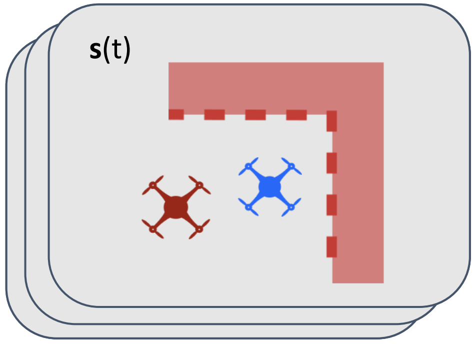
- More generally, a weighted sum of normalized robustness values for individual feature requirements
  - Weights can be used to adjust importance of requirements (e.g., 0.7 for Boundary, 0.3 for Runaway)
- Enables resolution through a trade-off between conflicting requirements
  - vs. “winner-takes-all” in existing approaches
  - Suitable for situations where both features perform critical functions

Synthesis Procedure

Define a space of candidate actions



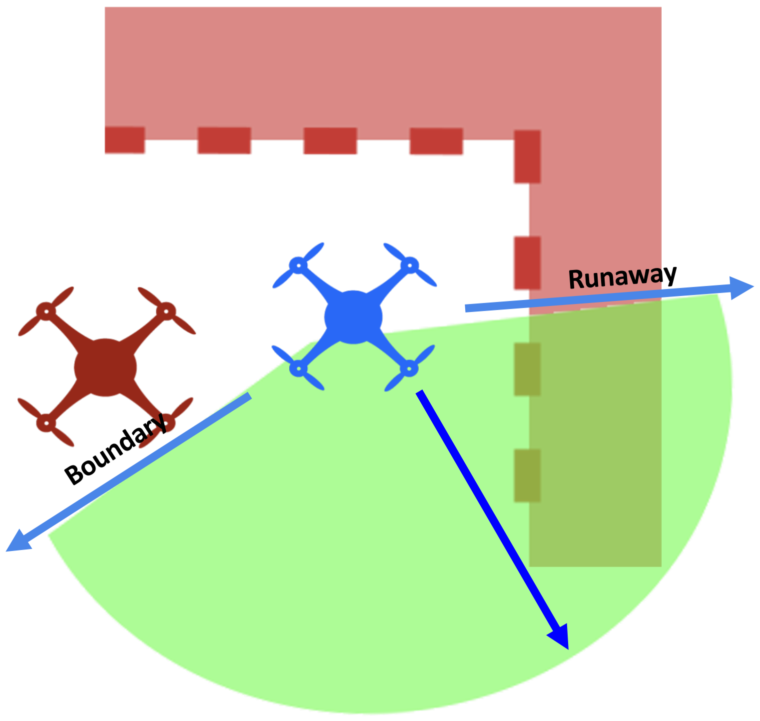
Current state:



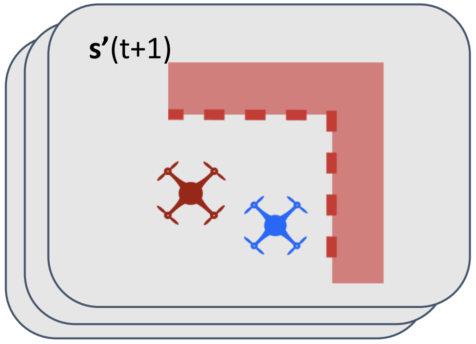
ρ_{sys}= -0.55

Synthesis Procedure

Uniformly sample actions from the search space & evaluate each of them for global robustness



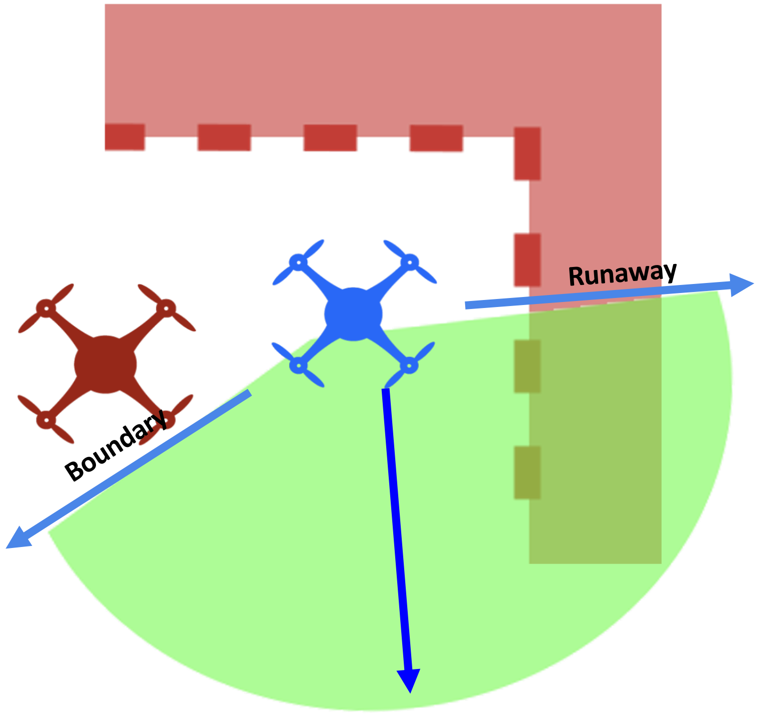
Predicted state:



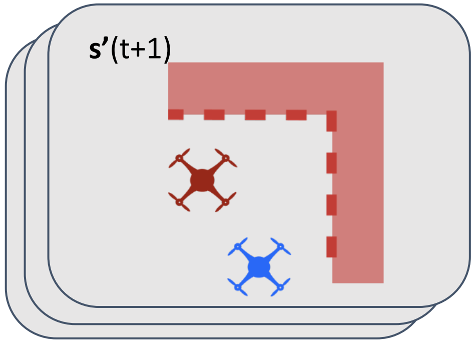
ρ_{sys}= -0.25

Synthesis Procedure

Uniformly sample actions from the search space & evaluate each of them for global robustness



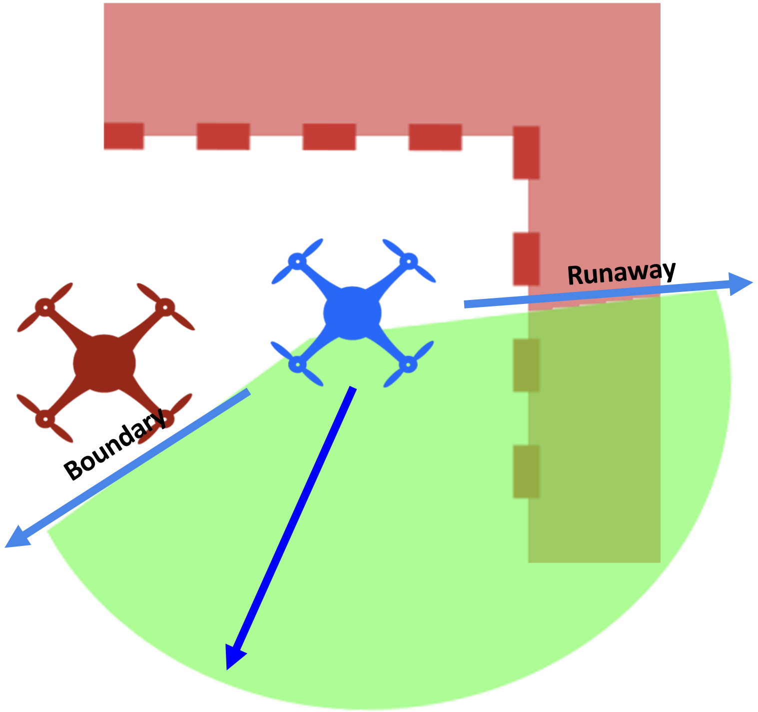
Predicted state:



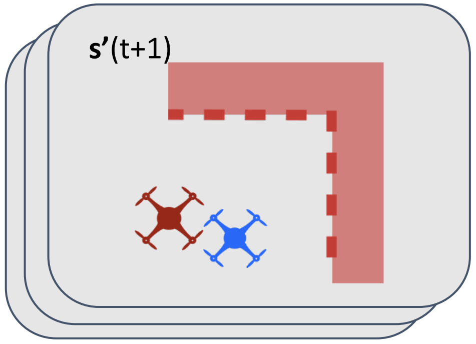
ρ_{sys}= 0.2

Synthesis Procedure

Uniformly sample actions from the search space & evaluate each of them for global robustness



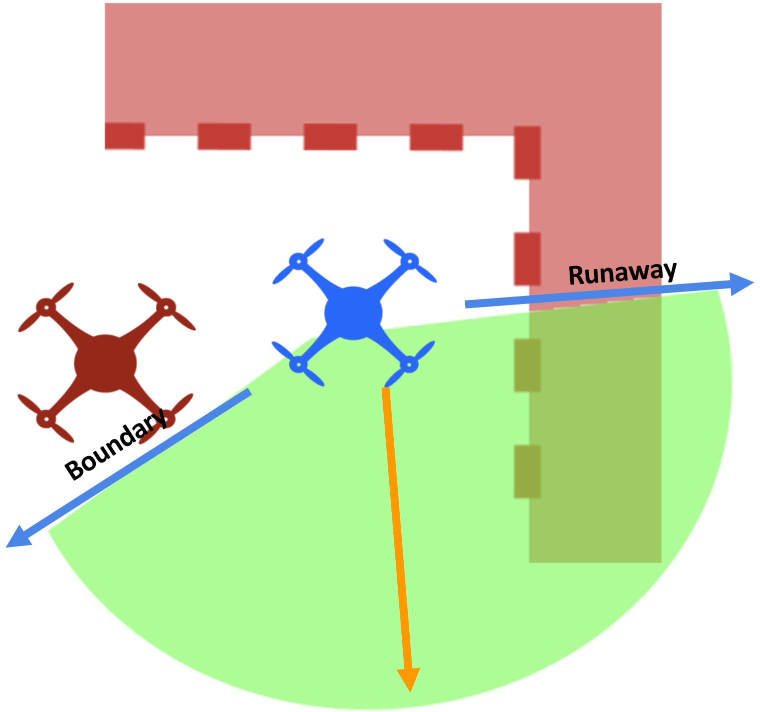
Predicted state:



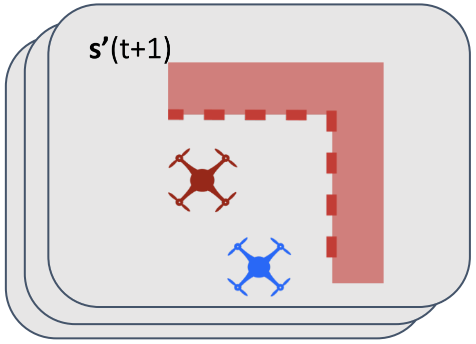
$\rho_{sys} = -0.3$

Synthesis Procedure

Select the most satisfactory action (i.e., one with highest global robustness)



Predicted state:

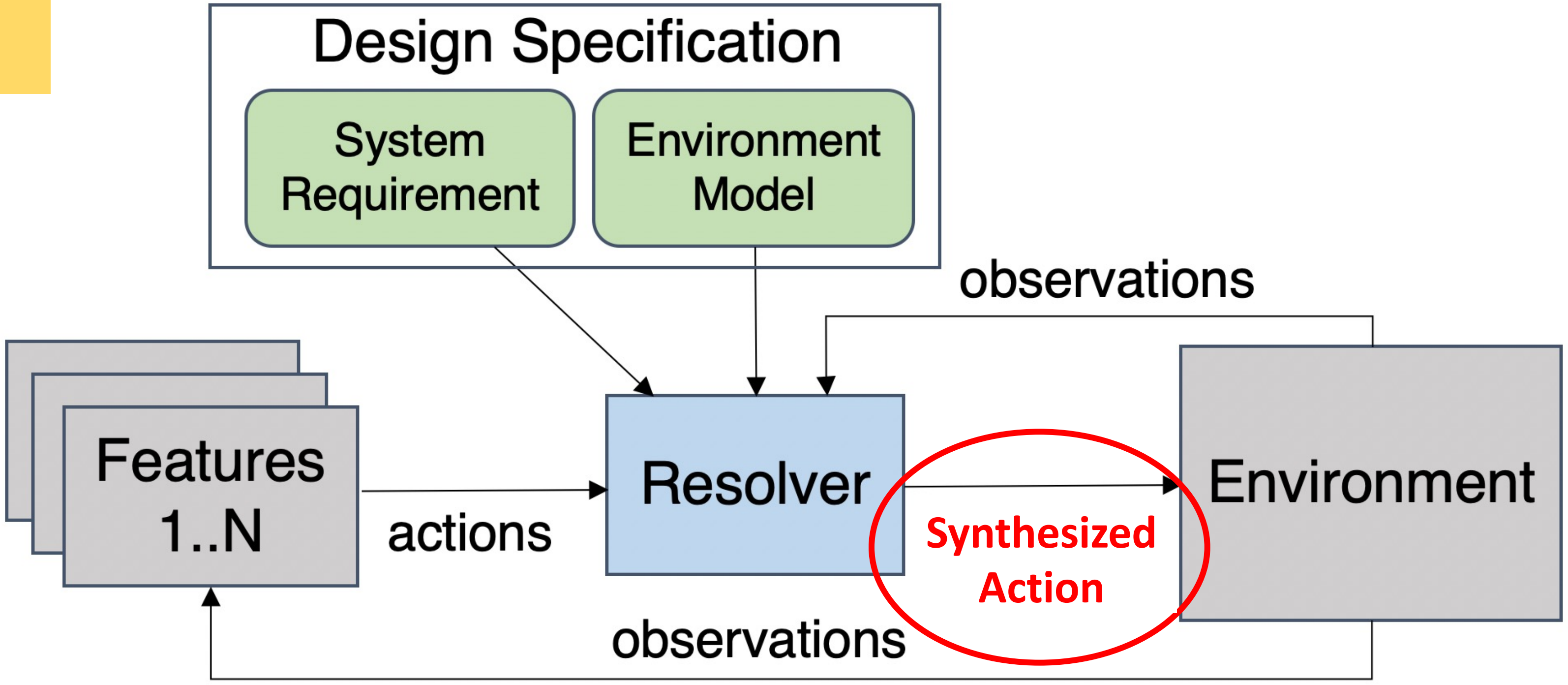


ρ_{sys}= 0.2

***ρ_{sys} maximized by this candidate action**

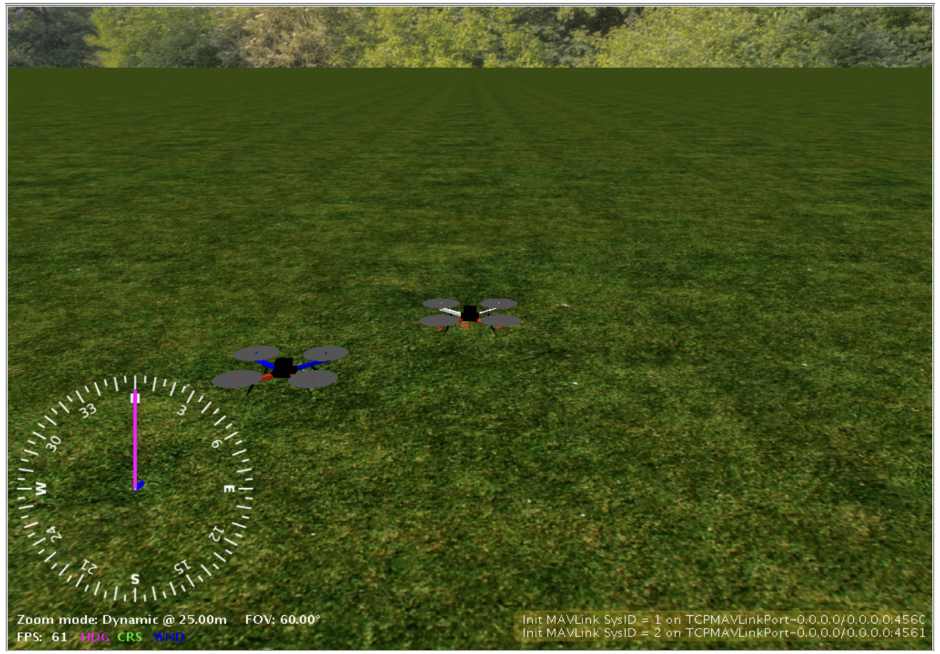
Idea #2: Resolution through Action Synthesis

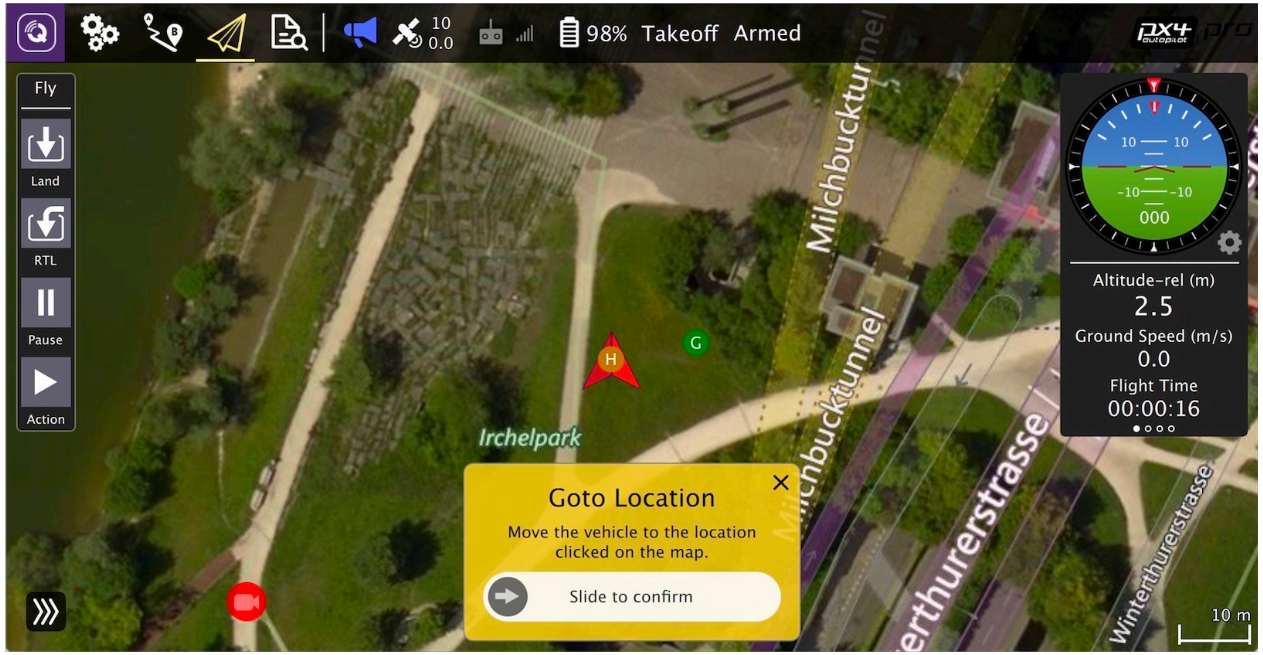
If none of the actions are satisfactory, *synthesize* an alternative action



Evaluation: Drone Case Study

Implemented the resolution framework on flight control software **PX4**
Used **JMAVSim** for Software-In-the-Loop (SIL) testing





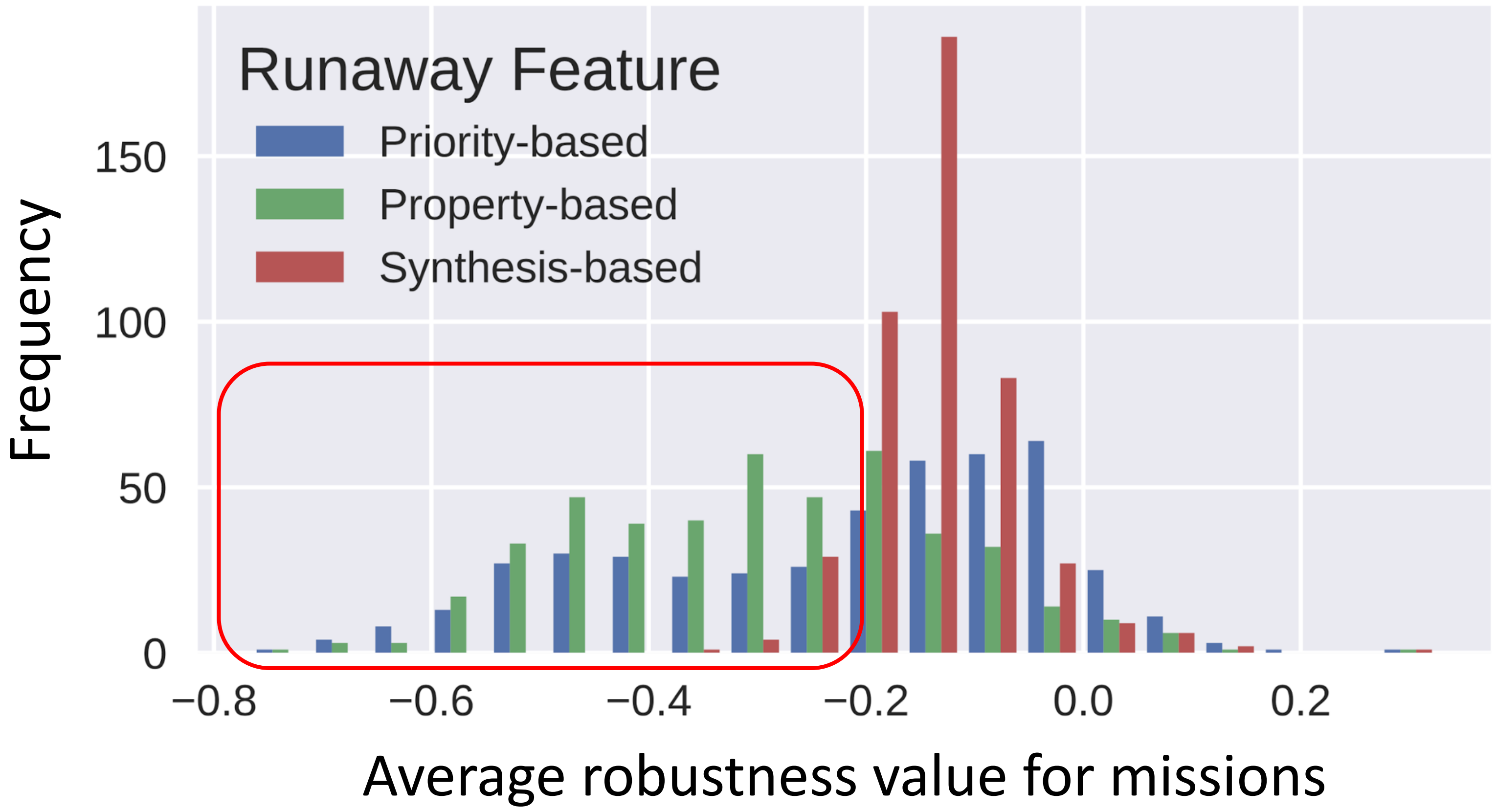
Evaluation: Drone Case Study

- Compared the following resolution strategies
  - Priority-based resolution: Fixed priority list
  - Requirement-based resolution: But without synthesis
  - Synthesis-based resolution: Synthesis of alternative actions
- Four features evaluated
  - Runaway
  - Boundary
  - Reconnaissance: Achieve a low altitude when in certain regions
  - Ground control: Maintain a safe altitude depending on terrain

Evaluation: Drone Case Study

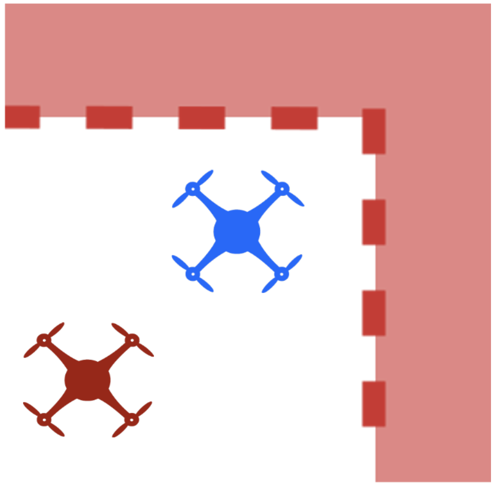
- Generated 500 randomly configured missions
- Evaluated each resolution strategy over these configurations
- Missions consisted of the drone flying to waypoints and performing a recon. maneuver at each waypoint

Synthesis results in fewer extreme violations



Challenges

- Efficient search & evaluation at runtime
  - Search heuristics (e.g., gradient descent)
  - Techniques from optimal control theory
- Uncertainty in the environment
  - Probabilistic models of the environment
- Unresolvable, difficult to resolve conflicts
  - What happens if the drone gets “stuck” in corner?
  - Predictive analysis to identify and avoid such conflicts



Takeaways

- Feature interactions remain a major obstacle to safe system composition in CPS
- Context-driven methods are needed for resolving undesirable interaction in an open, highly dynamic environment
- Requirement-based resolution
  - Desirability of a feature as the degree of satisfaction of STL requirements
- Synthesis of alternative actions
  - Greater system-level satisfaction through a trade-off between conflicting feature requirements

More details:
Synthesis-based resolution of feature interactions in cyber-physical systems (ASE 2020)
Property-driven runtime resolution of feature interactions (RV 2018)