# Safety Validation for Deep Reinforcement Learning Based Aircraft Separation Assurance with Adaptive Stress Testing

Wei Guo
Department of Computer Science
George Washington University
Washington, DC 20052, USA
weiguo@gwu.edu

Marc Brittain
Surveillance Systems
MIT Lincoln Laboratory
Lexington, MA USA
marc.brittain@ll.mit.edu

Peng Wei
Department of Mechanical and Aerospace Engineering
George Washington University
Washington, DC 20052, USA
pwei@gwu.edu

***Abstract*—Ensuring safe separation of aircraft has become a major challenge given the growing demand for air transportation. Recently, deep reinforcement learning (DRL) has been successfully applied to develop aircraft separation assurance systems. Though showing promising performance, the DRL systems are trained as black boxes with no safety guarantees. Given the catastrophic impact of potential failures, validations of these systems become critical. However, existing validation approaches focus on system failures caused by single factors and only work on the cases with one validated system, which greatly limits their use in real-world high-density air traffic where numerous validated systems exist and multiple factors cause the failures together. In this paper, we introduce a multi-agent adaptive stress testing formulation for the system validation task and propose a Multi-Agent Hybrid Adaptive Stress Testing (MAHAST) framework to comprehensively validate DRL-based aircraft separation assurance systems given high-density air traffic by detecting system failures caused by unsafe system dynamics and improper environment properties. Moreover, we introduce a novel hybrid adaptive stress testing approach with a hierarchical policy, detecting the failures caused by combinations of multiple factors. We conduct extensive numerical experiments in a real-time air traffic environment. Results empirically show that MAHAST can effectively detect diverse failures in DRL-based aircraft separation assurance systems with high-density air traffic.**

## I. Introduction

With the rapid growth of global air traffic, ensuring the safe separation among aircraft becomes a key challenge. Deep Reinforcement Learning (DRL) has been recently applied to the aircraft separation assurance systems, ensuring aircraft safety in complex and dense traffic [1]–[4]. Despite showing promising performance, these DRL models are trained with no safety guarantee. Since the failures of the separation assurance system lead to catastrophic accidents, the validation of the DRL-based separation assurance systems is of great importance. However, these DRL models have complex structures and interact with external systems in a complicated manner, which makes the safety validation challenging; severely restricting their use in real-world scenarios.

Various approaches on safety validation have been proposed and can be categorized into two types: formal methods and simulation methods. *Formal methods* prove the safety properties of a system via reachability analysis [5]–[7]. However, these methods rely on a precise mathematical formulation, which is difficult for a complex system like the DRL model. *Simulation methods* evaluate the system performance by sampling a series of scenarios with a simulation model [8], [9]. Though simulation methods have fewer constraints on the validated system than formal methods, the sampling process makes the validation inefficient given the large sample space and rarity of failure events in the validated system [10].

To address the sampling inefficiency in validation, Adaptive Stress Testing (AST) [11] is proposed to first formulate the validation as a sequential decision-making problem and then develop a stress testing policy, aiming to find scenarios that lead to failures of the validated system. The key point of AST is the sample selection based on the stress testing policy. Different methods can be applied to solve the problem. Reinforcement Learning (RL) methods [11], [12] design specific reward functions to encourage the detection of failures. Optimization approaches [10], [13] guide the selection with a constructed objective function based on the failure information.

For AST of DRL-based aircraft separation assurance system, each aircraft in the testing environment observes the information about the surrounding aircraft and the environment. The separation assurance system represents the System Under Test (SUT) of aircraft and the AST solver can be implemented on the surrounding aircraft. The failure is a collision between aircraft, representing that the assurance system fails to maintain sufficient separation. Figure 1 shows an example with 4 aircraft in a test environment. Two of them are equipped with SUTs and the others are equipped with AST solvers. SUT aims to maintain the safe separation of aircraft. On the other hand, AST solver tries to guide the aircraft to move in an unsafe way which leads to a collision.

Though achieving remarkable success in the system validation, prior AST methods focus on cases with only one autonomous system equipped with SUT [11], [14], [15], which fails in the tasks where multiple systems controlled by SUT are present. Moreover, these approaches can only find failures

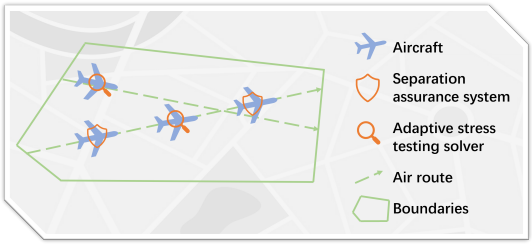



Fig. 1: An illustrative example of Adaptive Stress Testing (AST) on DRL-based aircraft separation assurance system. Four aircraft fly along the routes in a test environment. The separation assurance systems are implemented on two aircraft to maintain safe separation. The AST solvers are installed on the remaining two aircraft and control them to move in an unsafe way, aiming to have conflicts with the aircraft controlled by the assurance systems.

caused by one factor, either the dangerous agent movements [11] or the improper environment settings [10], but cannot detect their combined effects, which can lead to serious and complex failures. All these problems restrict the AST to provide a comprehensive system validation.

To tackle the above challenges, we propose a **M**ulti-**A**gent **H**ybrid **A**daptive **S**tress **T**esting (MAHAST) framework for the safety validation problem of DRL-based aircraft separation assurance system. In contrast to the previous works, our proposed framework can detect the failures caused by different factors and their combinations, given numerous aircraft controlled by SUT and AST solver in the test environment. Specifically, we first introduce a novel multi-agent adaptive stress testing formulation to describe the validation task based on a Multi-Agent Reinforcement Learning (MARL) setting, treating each aircraft in the test as an agent. Then, a *dynamic solver* is proposed to detect the failures caused by unsafe actions from multiple aircraft. For the dynamic solver, a Proximal Policy Optimization (PPO) network is implemented to guide its sample selection. Moreover, we introduce a *static solver* to detect failures due to improper configurations or changes in the environment with high traffic density based on Bayesian Optimization (BO). Afterwards, to address the combined effects of multiple factors in the environment, we develop a *hybrid solver* using a novel bi-level hierarchical policy, detecting the failures caused by the combination of unsafe aircraft actions and improper environment configurations. Extensive experiments in the BlueSky environment [16] demonstrate the effectiveness of our proposed framework.

The main contributions of our work are summarized as follows:

- We frame the safety validation of the aircraft separation assurance system as a multi-agent AST problem based on the MARL setting.
- We propose a framework for the safety validation of DRL-based aircraft separation assurance systems including three AST solvers, which can detect system failures due to diverse reasons in high-density air traffic.
- We conduct extensive experiments to validate a state-of-the-art DRL-based aircraft separation assurance model. The results empirically show that the developed framework can effectively validate its safety by detecting failures due to various reasons.

The rest of this paper is organized as follows. Section II provides a literature review on the related work. Section III presents our formulation of the validation problem. Section IV introduces the proposed framework in detail. Section V shows the experiment results and our findings. Finally, Section VI summarizes our conclusions.

## II. Related Works

### A. Deep Reinforcement Learning for Aircraft Separation Assurance

The application of DRL algorithms has been widely investigated in aircraft separation assurance. Variants of Deep Q-networks [17], [18] are adapted to resolve potential airspace conflicts [19]–[23]. Because PPO [24] can output a stochastic policy that performs better in multi-agent environments, PPO is another popular model for separation assurance in a discrete action space setting [2], [25]. For continuous action space settings, the Deep Deterministic Policy Gradient (DDPG) [26] algorithm has been applied to address conflicts between aircraft [27]–[30]. However, these DRL models have complex internal structures and are trained as black boxes, making them difficult to be validated and further limiting their use in real-world safety-critical systems.

### B. Validation of Autonomous System

Many approaches have been applied to validate the safety of autonomous systems. They can be divided into two categories: *formal methods* and *simulation methods*. Formal methods construct well-designed mathematical models to determine whether certain properties hold [5], [6], [31]. Probabilistic Model Checking (PMC) formally verifies specific characteristics over a stochastic model with discrete states [7], [32], [33]. Other formal methods like Automated Theorem Proving (ATP) methods take the advantage of computer algorithms to automatically generate mathematical proofs based on formal logic models [34]. However, the complex mathematical models strictly limit the scale of the validated models, which cannot work with DRL models. On the other hand, simulation methods assess the system properties with a simulation model based on the results from a fixed number of samples. The simulation model can be manually designed by experts with domain knowledge [8]. Stochastic models can also be implemented to approximate the system environment for simulation [9], [35], but the sampling process in simulation methods makes it inefficient given the large sample space and rarity of failure events.

### C. Adaptive Stress Testing for Validation

AST has been proposed to successfully tackle the sample inefficiency of simulation-based validation methods, which aims

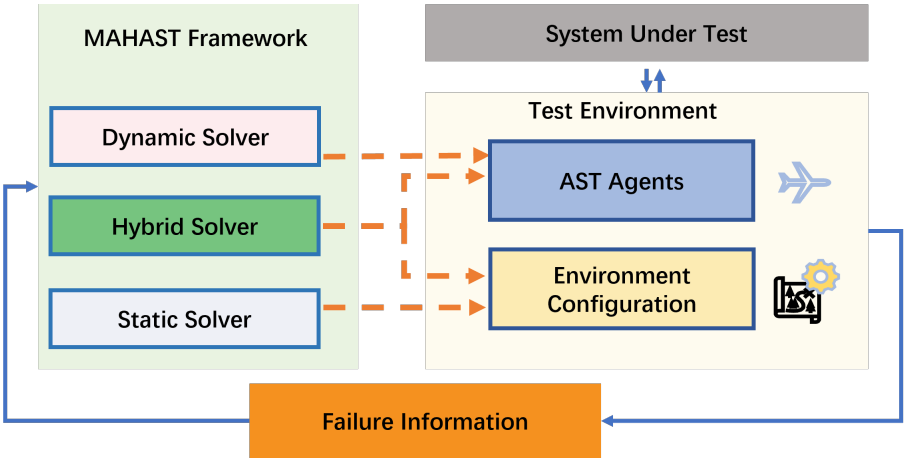



Fig. 2: The architecture of the MAHAST framework, which includes three AST solvers: dynamic solver, static solver, and hybrid solver. These solvers control different components of the environment and learn stress testing policies based on the failure information. The learned policies guide the solvers to efficiently find more failures.

to find the system failures with a stress testing policy. AST was first implemented to find failures of the autonomous vehicle in [11]. [36] integrated AST and neural network verification methods together to validate the image-based controllers. AST was further extended to a partially observable Markov decision process through integration with a modified Monte Carlo tree search method [12]. [10] proposed an optimization-based AST framework for unmanned aircraft systems and improved system safety by revising the environment settings leading to most failures. Go-explore method [10] and reward augmentation [15] are also implemented to build more efficient reward functions. Though the above articles showed promising performance on the failures caused by unsafe actions of autonomous systems and improper environment properties, their simple structures fail to handle the combined effects of multiple factors, which can lead to complex and serious failures. Moreover, these articles concentrate on the environment where only one AST solver and one SUT exist. However, this setting can be too ideal because the real-world environment may always have high-density traffic, which greatly increases the difficulty of validation. In this work, we aim to comprehensively validate the system by considering the failures due to either single factors or their combinations when multiple autonomous systems controlled by SUTs and AST solvers are present in the environment, which is a more complex problem that has not been previously studied.

## III. Problem Formulation

In this paper, the safety validation of the DRL-based aircraft separation assurance system is formulated as an AST problem with MARL settings, where each aircraft is treated as an agent in the test environment. Specifically, the aircraft can either be controlled by the SUT (the DRL-based aircraft separation assurance system to be validated) or the AST solver, which are then called SUT agent and AST agent respectively. In this section, we first introduce the formulation of the AST problem

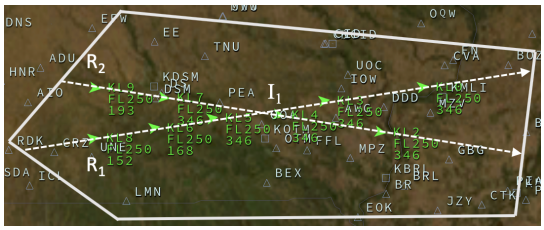



Fig. 3: Structure of the test environment. The green triangles are the aircraft flying in the test environment. The dotted lines and the solid lines represent the routes and environment boundaries. $R_i$ and $I_j$ stands for the $i$th route and $j$th intersection in the sector.

where only one SUT agent and one AST agent are in the test. Afterwards, based on the MARL setting, we expand the AST problem into a multi-agent case where multiple SUT agents and multiple AST agents exist, which makes the validation more difficult.

### A. Adaptive Stress Testing of Aircraft Separation Assurance System

AST formulates the safety validation problem as a sequential decision-making problem and solves it by searching for system failures with a stress testing policy. The system to be validated is called *system under test* (SUT), which is a DRL-based aircraft separation assurance system in our problem. This SUT is installed on the aircraft in the test environment and aims to maintain safe separation from surrounding aircraft by providing speed advisories. The SUT can have complex internal dynamics and is treated as a black box model.

The system failures are defined as a set $\mathcal{E}$, representing the events where the DRL-based separation assurance system fails and the aircraft have conflicts with others in our setting. To detect these failures, an AST solver controls some components in the test environment and receives the information on

whether its actions lead to a failure event in $\mathcal{E}$. Based on this information, the AST solver learns a stress testing policy, which guides the AST solver to modify the environment components and find more failures.

The test environment is also treated as a black box. Given the actions of the AST solver, we can only observe an updated environment state and the indication if a failure happens. Specifically, the test environment in our problem is an en route sector with two routes, one intersection, and high-density air traffic, providing a representative sector structure. The test environment is illustrated in Figure 3.

All aircraft in the test environment are set to be the same type, Airbus A320. The minimum and maximum values of the calibrated airspeed are set to be 156 knots and 346 knots respectively. Aircraft update their action every 12 seconds to represent the worst-case radar surveillance update interval in en route airspace.

Considering that AST and SUT agents are controlled by different systems, communication between them is implemented. We assume the states of all aircraft are available to each other, which allows SUT to obtain all information needed for decision-making and to better maintain separation. Though increasing the difficulty of failure detection, this makes our formulation more applicable to real-world cases where SUT has access to all information.

### B. Formulation of Multi-Agent Adaptive Stress Testing

The real-world test environment may always contain multiple SUT and AST agents, which are beyond the capacity of the previous AST setting. Therefore, in this paper we propose a novel multi-agent AST formulation when multiple SUT and AST agents are in the test environment based on a MARL setting. Since interactions with multiple SUT and AST agents must be taken into account, validation becomes significantly more challenging in our case.

We assume that there are $n$ AST agents and $m$ SUT agents in the test environment. The $i$th AST agent can observe the state $s_i$ $(i = 1, ..., n)$ from its state space $\mathcal{S}_i$. $a_i$ denotes the action of AST agent from its action space $\mathcal{A}_i$. The transition function $\mathcal{T}$ provides the transition probability from the joint state $(s_1, ..., s_n)$ to the joint next state $(s'_1, ..., s'_n)$ given the joint action $(a_1, ..., a_n)$. Based on the transition information, a reward function $r_i$ is provided to determine the immediate reward for each AST agent. Each AST agent aims to learn an optimal policy $\pi_i$ that maximizes the expected value of the cumulative rewards.

Specifically, the state $s_i$ of an AST agent includes the information about itself and 5 closest agents: distance to sector exit, speed, acceleration, route identifier, distance between itself and surrounding agents, and distance between itself and the intersection. The state space follows the definition provided in [2].

Each AST agent has a discrete action space $\mathcal{A}_i$ with three actions: (1) decelerate, (2) maintain the current speed, and (3) accelerate.

The termination of each AST agent happens: (1) when it reaches the failure event by having a conflict with other aircraft or (2) when it reaches the sector exit safely.

To detect failures of the DRL-based separation assurance system, a reward function encouraging small distances between aircraft is provided to each AST agent.

$$r_i = \begin{cases} 1 & \text{if } d_o^c < 3, \\ \alpha + \delta \cdot d_o^c & \text{if } 3 \leq d_o^c < 10, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Parameters $\alpha$ and $\delta$ guarantee that the reward stays between 0 and 1. $d_o^c$ is the distance from the AST agent to its closest aircraft in units of nautical miles.

## IV. Method

To comprehensively validate the safety of the DRL-based aircraft separation assurance system considering the diverse failure causes and multiple aircraft in the test environment, we present a novel Multi-Agent Hybrid Adaptive Stress Testing (MAHAST) framework. Specifically, we propose three AST solvers to detect failures with different reasons, namely, the *dynamic* AST solver, the *static* AST solver, and the *hybrid* AST solver.

The dynamic solver consists of a PPO neural network controlling the actions of the AST agent as a non-cooperative aircraft to generate conflicts with other aircraft. The static solver based on BO searches for improper environment configurations or environment changes. The hybrid solver implements a bi-level hierarchical policy that looks for failures caused by the combination of multiple factors from the test environment. In this section, we first describe the structures of these three solvers separately and then discuss their integration as a comprehensive validation framework.

### A. Dynamic AST Solver

A common reason for separation assurance failures is that some aircraft make unsafe actions, especially when they are non-cooperative [11]. Such failures may be even more prevalent if multiple aircraft exist in the test environment [11]. Therefore, we develop a dynamic AST solver to detect failures caused by dangerous aircraft actions with the existence of multiple aircraft. Dynamic solvers are implemented in some aircraft and control their actions during the test.

*1) Stress Testing Policy:* The core of the dynamic solver is a stress testing policy controlling the actions of AST agents. A PPO neural network is implemented to learn the policy similar to [2] considering complex state space and dense traffic. The input of the PPO network is the states of the AST agent itself and the 5 closest aircraft. The output is a policy guiding the action selection of the AST agent. Because the reward functions encourage small distances among aircraft, the learned policy prefers dangerous actions leading to conflicts. These actions are detected as dangerous actions given the corresponding state. And the failure events can be treated as the worst cases due to the existence of non-cooperative agents.

The information can help revise SUT by avoiding similar behaviors.

*2) Centralized Learning Decentralized Execution:* To help the dynamic solver learn policies efficiently and improve the cooperation among AST agents, we implemented a centralized learning decentralized execution scheme for the dynamic solver. During the training phase, transitions generated by all AST agents are gathered together and sent to an identical PPO neural network as a central learner. The updated model weights from the central PPO network are distributed to the PPO networks on all AST agents. During the evaluation phase, given the diverse states of these AST agents, the PPO networks can provide different decisions accordingly. Since the central PPO network is trained with information from all AST agents, the learned policy will tend to improve the overall expected return among all AST agents, leading to more failures detected. In addition, the decentralized execution design allows users to easily add new AST agents into the test by sending the converged model to the aircraft as a controller, helping users validate the assurance systems under various densities of AST agents.

*3) Interaction with Environment and SUT:* Given the stress testing policy, three functions are designed for AST solvers to interact with the test environment and SUT agents:

- Initialize: An AST solver is attached to one aircraft in the test and the aircraft is initialized as an AST agent.
- Terminate: If the AST agent reaches the termination, its trajectory finishes, and its AST solver detaches.
- Step: AST Solver selects the action for the AST agent and updates its state accordingly.

*4) Generalization Improvement:* Previous articles generally train AST solvers with fixed roles of agents [11], [14]. For example, given two aircraft in the test environment, the leading aircraft is always the SUT agent. While simplifying the problem, this setting deteriorates the model generalization to the real world where the roles of agents can be diverse. This is even more harmful in our multi-agent case because the roles can be more complex.

To address the issue, we introduce a technique called *Agent Role Randomization*. Specifically, the dynamic solvers and SUTs are randomly allocated to aircraft in each test during training, which randomizes the roles of all AST and SUT agents. Therefore, AST solvers cannot simply memorize actions based on a fixed order but need to develop a dynamic strategy, which helps AST solvers generalize better and become more applicable to real world scenarios.

### B. Static AST Solver

Another common reason for failure for aircraft separation assurance is the improper environment configuration or environment changes. Considering the large configuration space, we propose a static AST solver based on BO to detect these failures which influence multiple aircraft in the test environment. Instead of directly guiding the dynamics of some aircraft as the dynamic solver, the BO-based static solver modifies the environment configuration before the test and updates the policy based on the failures after the test iteratively. The detected failures can be used to help experts revise the SUT to perform safely under different settings.

*1) Failure Approximation:* Given the unknown complex relationship between the failures and environment configurations, the static solver approximates the failure numbers in each test with a function $f(x)$ drawn from a Gaussian Process (GP) given the configuration $x$. Specifically, given a collected dataset $X = \{(x_i, f(x_i))|i = 1, ...n\}$, GP can predict the failure number $f_{next}$ with the next configuration value $x_{next}$. The posterior distribution $f_{next}|X, x_{next}$ is a normal distribution where the mean is:

$$K\left(x_{next}, X\right) K(X, X)^{-1} f, \tag{2}$$

and variance is:

$$K\left(x_{next}, x_{next}\right) - K\left(x_{next}, X\right) K(X, X)^{-1} K\left(X, x_{next}\right). \tag{3}$$

Here $f$ is the vector with all function values in the collected set, and $K$ represents the matrix of the covariance functions.

*2) Stress Testing Policy:* Given the approximated failure number $f$ in the test, the static solver develops a policy to modify configurations. An intuitive policy is to select the configuration value that maximizes the expected value of the failure number in the test. However, considering the approximated function $f$ can be inaccurate because of the uncertainty in the large configuration space, we implement a policy that maximizes the Upper Confidence Bound (UCB) of the failure number. Specifically, the policy is defined as follows with the tunable parameter $k$:

$$x_{next} = argmax_x[\mu(x|X) + k\sigma(x|X)]. \tag{4}$$

Here $\mu$ and $\sigma$ represent the mean and standard deviation respectively. This UCB-based policy can effectively minimize regret over the optimization process and balance the trade-off between exploration and exploitation [37].

Moreover, to guarantee the failures detected here are caused by improper configurations instead of unsafe aircraft actions, all aircraft are equipped with SUT to move in a safe way.

### C. Hybrid AST Solver

While the dynamic and static solvers can find failures caused by either unsafe aircraft actions or improper environment configurations respectively, they cannot detect failures caused by the combinations of these two types of factors. However, their combinations interact with SUT complexly, making the potential failures even more challenging to be detected. Therefore, we propose a hybrid AST solver in this subsection to detect failures caused by combination of dangerous aircraft actions and improper environment configurations.

*1) Stress Testing Policy:* A bi-level hierarchical policy is proposed in the hybrid solver. Specifically, the solver consists of a high-level policy that chooses the environment configuration and the number of AST agents before the test, and a low-level policy that controls the actions of the AST agents during the test. The failure information is sent back to the

solver for policy updating. Considering the complexity of the hierarchical structure, we train the low-level policy with a PPO network similar to that in dynamic solver and implement the UCB policy as the high-level policy.

Given the hierarchical policy, the hybrid solver can find the failures caused by combining the aircraft dynamics and environment configurations. The information can help understand the influence of the combination of these factors on SUT.

### D. Integration

To comprehensively validate the safety of DRL-based aircraft separation assurance systems, we integrate the dynamic solver, the static solver, and the hybrid solver together into the MAHAST framework. We illustrate its architecture in Figure 2.

On a high level, the MAHAST framework detects failures of SUT by modifying the configurations and controlling the actions of aircraft in the test environment. The failure information can then be used to refine the SUT and improve its safety.

Given different system failures to be validated, users can select specific AST solvers accordingly. Specifically, the failures from dangerous aircraft actions can be detected by the dynamic solver through guiding the actions of AST agents during the test. Failures resulting from improper environment configurations or environment changes can be found by the static solver, which modifies the environment configuration before each test. Failures due to combinations of multiple factors in the test environment can be sought by the hybrid solver, which controls both the environment configuration and AST agent actions.

## V. Experiments

In this section, we first introduce the experiment setups and then conduct a series of experiments to evaluate the effectiveness of the proposed solvers in detecting the failures with different causes on the BlueSky environment [16].

### A. Experiment Setups

**Test Environment.** In this work, we used the Bluesky [16] simulator as our test environment. BlueSky is an open source air traffic simulation environment which allows for realistic real-time air traffic scenarios. For each test episode, 30 aircraft will enter the airspace and the inter-arrival time follows a discrete uniform distribution over 4, 5, and 6 minutes. The test episode ends after all 30 aircraft reach the termination criteria. To further decrease the impact of noise, we impose a setting to make sure all aircraft cannot deviate from their routes during the episode.

**Evaluation metric.** We adopt the *Average Number of Failed Aircraft* (ANFA) as the metric to evaluate the performance. Specifically, ANFA calculates the average number of aircraft that have conflicts with others in a test episode. ANFA measures the frequency of detected failures, and a higher ANFA means that more failures are detected.

**System Under Test.** Deep Distributed Multi-Agent Reinforcement Learning (DD-MARL) framework [2] is used as the SUT in our evaluation as it maintains state-of-the-art performance in DRL-based aircraft separation assurance tasks, which makes the validation challenging. DD-MARL uses a DRL model to maintain safe separation for SUT agents by providing speed advisories.

TABLE I: Average Numbers of Failed Aircraft (ANFA) for dynamic solver with no communication loss in 200 episodes. Each column denotes a specific conflict class. D and B represent the dynamic solver and baseline.

| **Setting** | **AST-AST** | | **AST-SUT** | | **SUT-SUT** | | **Total** | |
|---|---|---|---|---|---|---|---|---|
| | D | B | D | B | D | B | D | B |
| 5 AST | 0.16 | 0.40 | 4.97 | 4.53 | 0.17 | 0.27 | **5.30** | 5.20 |
| 2 AST | 0.03 | 0.03 | 3.52 | 1.88 | 0.34 | 0.21 | **3.89** | 2.12 |
| 1 AST | 0.00 | 0.00 | 1.84 | 1.03 | 0.18 | 0.17 | **2.02** | 1.20 |

### B. Evaluation of Dynamic Solver with No Communication Loss

We first evaluate the effectiveness of the dynamic solver in finding failures caused by unsafe aircraft actions. To make a comprehensive evaluation, we conduct three tests given three different numbers of AST agents in the environment. Each test runs 200 episodes to eliminate the influence of randomness. To provide thorough comparisons, we introduce a baseline solver that uniformly samples the action of the AST agent. We further suppose that there is no communication loss among aircraft in this experiment, so each aircraft can receive information from all the surrounding aircraft in the environment.

The ANFA values are shown in Table I. Each row reports the ANFA of a test given a particular number of AST agents. To gain more insights, we categorize the conflicts into three classes depending on the participants: *AST-AST*, *AST-SUT*, and *SUT-SUT*. For example, *AST-AST* conflict is the conflict that happens between two AST agents. The first three columns show the results of three conflict classes and the last column shows the total number of ANFA. The results of our proposed method and the baseline are shown in two different subcolumns under each column, denoted as $D$ and $B$ respectively.

We notice that the ANFA value is 2.02 with only 1 AST agent in the environment. This implies that the DRL models can be vulnerable to unsafe actions from other agents, which demonstrates the necessity of the safety validation system.

Moreover, we observe that the dynamic solver consistently beats the performances of the baseline by finding more failures given all three numbers of AST agents. This verifies the effectiveness of the dynamic solver in detecting the failures due to unsafe aircraft actions, as the proposed solver can efficiently directly force the AST agents to move in an unsafe way, leading to more failures with other aircraft.

### C. Evaluation of Dynamic Solver with Communication Loss

We have demonstrated the effectiveness of the dynamic solver when there is no communication loss in the test. However, the communication systems may not always work

TABLE II: Average Numbers of Failed Aircraft (ANFA) for dynamic solver with communication loss in 200 episodes. Each column denotes a specific conflict class. D and B represent the dynamic solver and baseline.

| **Setting** | **AST-AST** | | **AST-LC** | | **AST-SUT** | | **LC-LC** | | **LC-SUT** | | **SUT-SUT** | | **Total** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D | B | D | B | D | B | D | B | D | B | D | B | D | B |
| 5 AST,10 LC | 0.20 | 0.39 | 3.45 | 1.64 | 4.81 | 2.42 | 0.59 | 0.71 | 3.83 | 4.02 | 0.24 | 0.09 | **13.12** | 9.27 |
| 5 AST,5 LC | 0.46 | 0.47 | 1.20 | 0.81 | 5.12 | 3.48 | 0.13 | 0.12 | 3.07 | 3.09 | 0.25 | 0.19 | **10.23** | 8.16 |
| 5 AST,2 LC | 0.22 | 0.32 | 0.72 | 0.34 | 7.39 | 4.00 | 0.03 | 0.01 | 1.75 | 1.57 | 0.47 | 0.20 | **10.58** | 6.44 |
| 5 AST,1 LC | 0.45 | 0.48 | 0.20 | 0.17 | 7.41 | 3.94 | 0.00 | 0.00 | 0.78 | 0.78 | 0.27 | 0.19 | **9.11** | 5.56 |
| 2 AST,10 LC | 0.01 | 0.04 | 1.51 | 0.81 | 1.94 | 1.09 | 0.68 | 0.64 | 5.30 | 4.89 | 0.14 | 0.13 | **9.58** | 7.60 |
| 2 AST,5 LC | 0.03 | 0.06 | 0.72 | 0.44 | 2.85 | 1.56 | 0.11 | 0.10 | 3.41 | 3.81 | 0.24 | 0.15 | **7.36** | 6.12 |
| 2 AST,2 LC | 0.04 | 0.05 | 0.32 | 0.12 | 3.24 | 1.67 | 0.00 | 0.01 | 2.01 | 1.77 | 0.42 | 0.19 | **6.03** | 3.81 |
| 2 AST,1 LC | 0.01 | 0.02 | 0.10 | 0.05 | 1.88 | 2.04 | 0.00 | 0.00 | 0.98 | 0.96 | 0.17 | 0.19 | 3.14 | **3.26** |
| 1 AST,10 LC | 0.00 | 0.00 | 0.69 | 0.38 | 1.11 | 0.71 | 0.67 | 0.74 | 5.48 | 5.38 | 0.14 | 0.07 | **8.09** | 7.28 |
| 1 AST,5 LC | 0.00 | 0.00 | 0.30 | 0.20 | 1.50 | 0.92 | 0.15 | 0.15 | 3.90 | 3.91 | 0.23 | 0.24 | **6.08** | 5.42 |
| 1 AST,2 LC | 0.00 | 0.00 | 0.15 | 0.10 | 1.65 | 0.89 | 0.02 | 0.03 | 1.64 | 1.76 | 0.27 | 0.11 | **3.73** | 2.89 |
| 1 AST,1 LC | 0.00 | 0.00 | 0.07 | 0.04 | 1.74 | 1.05 | 0.00 | 0.00 | 0.97 | 1.04 | 0.12 | 0.18 | **2.90** | 2.31 |

perfectly, so some aircraft may be unable sense the surrounding aircraft [25]. This increases the complexity of the validation because it requires the AST solver to consider the behaviors of both the SUT agents and the agents that cannot communicate.

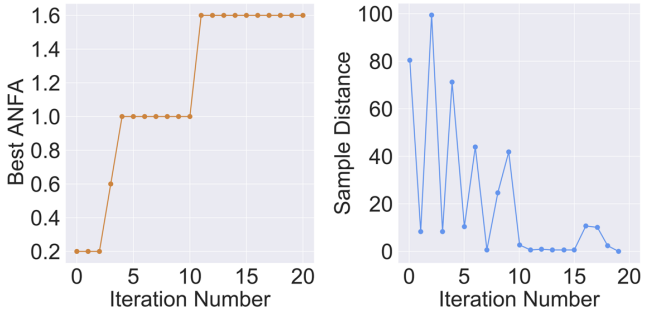
To evaluate whether the dynamic solver can still detect failures under this complex situation, we first introduce a novel Loss of Communication (LC) agent into the problem and then conduct a series of tests given different numbers of LC, SUT, and AST agents in the environment.

Since the LC agent can not receive the information from the surrounding aircraft, a reasonable strategy of LC agent is to maintain the same speed and avoid speed changes. Therefore, we set the LC agents to always maintain the same speed at each time step.

Table II reports the ANFA for 200 episodes given different numbers of SUT and LC agents in the test environment. In this experiment, a baseline AST solver that uniformly samples the action is used for comparison. Similar to the previous experiment, each column displays the results for a specific conflict class based on its participants. Each row shows the results given a particular combination of AST and LC agents.

Overall, the dynamic solver gives a higher total ANFA value than the baseline in most cases under this complex condition, consistent with the results in the previous experiments with no communication loss. This demonstrates the superiority of applying MARL into AST, as the proposed method can overcome the negative influence of communication loss and find more failures.

We also observe that the proposed dynamic solver can detect more *SUT-SUT* failures than the baseline in almost all cases. This shows that although the solver cannot directly control the SUT agents, the non-cooperative behaviors of AST agents can also force some unsafe actions of the SUT agents, leading to more conflicts where only SUT agents are involved. This should benefit from our centralized learning decentralized execution design given the central PPO network learns the diverse failure modes during training and then encourages cooperation among AST agents to push the SUT agents to move unsafely.




(a) Best ANFA (b) Sample Distance

Fig. 4: Evaluation results for static solver on configuration of the minimum calibrated airspeed $v_{min}$. The x-axis of both plots represents the iteration numbers. Figure 4a shows the best Average Number of Failed Aircraft (ANFA) till each iteration, and Figure 4b shows the Euclidean distance between consecutively sampled points in each iteration. A large distance represents that the points are far apart and the policy is exploring; a small distance shows that the points are close and the policy focuses more on exploiting. In this case, the maximum allowed sample distance is 124.

In addition, we have two observations about communication loss. Firstly, given any fixed number of LC agents, increasing the number of AST agents can always lead to more failures. This validates the effectiveness of the dynamic solver as it can detect failures under communication loss. Secondly, controlling the number of AST agents, there is no clear pattern indicating that more LC agents result in more *SUT-SUT* collisions. This fact implies that SUT can still help SUT agents maintain safety separation to some degree even with the existence of some LC agents.

### *D. Evaluation of Static AST Solver with Single Configuration*

Since we have demonstrated the effectiveness of the dynamic solver, in this subsection, we focus on whether the static solver can detect failures caused by the improper value of a single environment configuration. To provide thorough comparisons, we also introduce a baseline solver that uniformly

TABLE III: Best Average Numbers of Failed Aircraft (ANFA) for static solver based on the configuration of the minimum calibrated airspeed $v_{min}$. *Static* and *Baseline* represent the results of static solver and baseline.

| | Static | Baseline |
|---|---|---|
| Best ANFA | **1.60** | 1.00 |

samples the values of environment configuration. Specifically, we concentrate on the minimum calibrated airspeed $v_{min}$, one of the most important configurations of the test environment. The sample space is set to be continuous between 156 and 280 knots. To eliminate the influence of unsafe actions, all aircraft are controlled by SUT to maintain safe separation. We also suppose there is no communication loss in this experiment.

To comprehensively analyze the optimization process, we show the detailed run-time optimization results of the proposed static solver in Figure 4 instead of only providing the final ANFA values. Specifically, Figure 4a shows the best ANFA up to the current iteration. Figure 4b gives the Euclidean distance between the two consecutively sampled points from the sample space for the current iteration. When the distance is large, the points are far apart as the stress policy tends to explore widely; otherwise, the points are close to each other since the policy tends to exploit the information.

Based on Figure 4b, we observe that the static solver explores in the first 10 iterations because the sample distances in the right plot tend to be larger than in the later iterations. At the same time, the best ANFA in Figure 4a keeps increasing during the exploration, showing that new failures are detected in the process. Afterwards, the sample distances become smaller in the next iterations, showing that the static solver tends to focus more on a small region based on the gathered information. More failures are also found in the period as the best ANFA increases again. The changes in sample distances show the trade-off between the two components in the UCB policy, as it does not simply aim to find more failures but also considers the uncertainty in space.

The final results of our proposed static solver and the baseline solver are reported in Table III. Specifically, the best ANFA detected by the static solver is 1.60 while the best ANFA detected by the baseline is only 1.00. The comparison demonstrates that the proposed static solver can effectively detect failures due to the improper value of a single environment configuration with the help of Bayesian Optimization.

### *E. Evaluation of Static AST Solver with Multiple Configurations*

We have demonstrated the effectiveness of the static solver given only one environment configuration. However, values of multiple configurations can change in real-world scenarios. This increases the complexity of the validation due to the high dimensional sample space. In this subsection, we evaluate the effectiveness of the static solver given improper values of multiple environment configurations. Specifically, we concentrate on the minimum calibrated airspeed $v_{min}$ and the maximum calibrated airspeed $v_{max}$. The sample space of $v_{min}$ is a continuous space between 156 and 280 knots. And that of $v_{max}$ is a continuous space between 300 and 346 knots.

TABLE IV: Best Average Numbers of Failed Aircraft (ANFA) for static solver based on the configurations of the minimum calibrated airspeed $v_{min}$ and the maximum calibrated airspeed $v_{max}$. *Static* and *Baseline* represent the results of static solver and baseline.

| | Static | Baseline |
|---|---|---|
| Best ANFA | **5.40** | 2.60 |

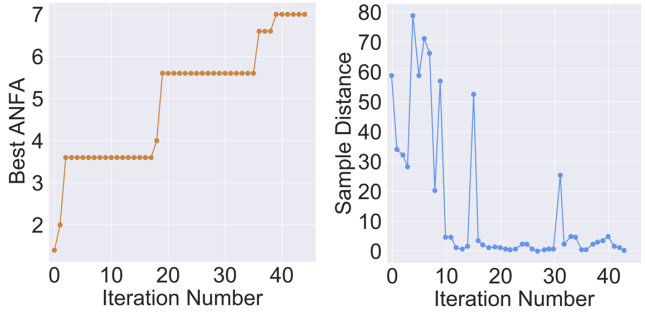
To give a comprehensive comparison, we also use the baseline solver that uniformly samples the environment configuration values. The final results are reported in Table IV. Specifically, the best ANFA detected by the static solver is 5.40 while the best ANFA detected by the baseline is only 2.60. The results demonstrate that the proposed static solver can detect failures due to improper values of multiple environment configurations effectively. In addition, comparing the results with two configurations in Table IV and one configuration in Table III, we find that the performance of the static solver improves more than that of the baseline when the dimension number of sample space increases. The possible reason may be that the 2-dimensional sample space is too complex for the baseline to find the failures.

### *F. Evaluation of Hybrid AST solver*

We evaluate whether our proposed hybrid solver can detect the failures caused by combinations of unsafe actions and improper configurations in this subsection. To make a fair comparison, we here focus on the combined effects of the factors that have been examined in previous experiments. Specifically, we concentrate on the combination of the minimum calibrated airspeed $v_{min}$ and the unsafe aircraft actions. The hybrid solver modifies the values of $v_{min}$ and the number of AST agents before each test and controls the actions of all AST agents during the test. The sample space of $v_{min}$ is set to be between 156 and 280 knots. The number of AST agents can be 1, 2, or 5. A baseline solver is implemented for comparison which uniformly selects the configuration values and actions of AST agents.

The run-time optimization plot is drawn in Figure 5 to provide detailed results with the hybrid solver. Figure 5a shows the best ANFA till each iteration and Figure 5b shows the sample distance in sample space. Based on Figure 5a, we find that the hybrid solver has been trapped in multiple local optima during the optimization. This may relate to the fact that the complex 2-dimensional sample space makes the optimization a challenging problem.

The final results of the hybrid solver and the baseline are reported in Table V. Specifically, the highest ANFA found by the hybrid solver is 7.00 while that by the baseline is 4.80. The comparison demonstrates the effectiveness of our proposed




(a) Best ANFA (b) Sample Distance

Fig. 5: Evaluation results for hybrid solver based on the configurations of the minimum calibrated airspeed $v_{min}$ and unsafe actions of AST agents. The x-axis of both plots represents the iteration numbers. Figure 5a shows the best Average Number of Failed Aircraft (ANFA) till each iteration, and Figure 5b shows the Euclidean distance between consecutively sampled points in each iteration. In this case, the maximum allowed sample distance is 128.

TABLE V: Best Average Numbers of Failed Aircraft (ANFA) for hybrid solver based on the configuration of the minimum calibrated airspeed $v_{min}$ and unsafe actions of AST agents. *Hybrid* and *Baseline* represent the results of hybrid solver and baseline.

| | Hybrid | Baseline |
|---|---|---|
| Best ANFA | **7.00** | 4.80 |

hybrid solver in detecting failures due to combinations of unsafe actions and improper configurations.

## VI. Conclusions

In this article, we propose MAHAST, a multi-agent adaptive stress testing framework, for the safety validation of DRL-based aircraft separation assurance systems. Three AST solvers are included in MAHAST to detect system failures due to different causes, namely, a dynamic AST solver, a static AST solver, and a hybrid AST solver. The dynamic solver incorporates a PPO network to comprehend the movements of the SUT, detecting the failures caused by unsafe aircraft actions during the test. In the static solver, we introduce a Bayesian Optimization approach to iteratively find the improper environment configurations leading to the most failures. Finally, we design a hybrid solver which can detect the failures caused by the combinations of multiple factors in the environment based on a bi-level hierarchical policy. We conduct extensive experiments in the real-time air traffic simulation environment provided by BlueSky. The results empirically demonstrate that MAHAST can effectively detect failures caused by multiple causes and validate the system safety of DRL-based aircraft separation assurance systems in high-density air traffic.

## Acknowledgment

The authors are partially supported by the NASA Grant 80NSSC21M0087 under the NASA System-Wide Safety (SWS) program.

## References

[1] W. Guo, M. Brittain, and P. Wei, "Safety enhancement for deep reinforcement learning in autonomous separation assurance," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 348–354.

[2] M. Brittain and P. Wei, "Autonomous separation assurance in an high-density en route sector: A deep multi-agent reinforcement learning approach," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3256–3262.

[3] M. W. Brittain and P. Wei, "One to any: Distributed conflict resolution with deep multi-agent reinforcement learning and long short-term memory," in *AIAA Scitech 2021 Forum*, 2021, p. 1952.

[4] M. Brittain, X. Yang, and P. Wei, "A deep multi-agent reinforcement learning approach to autonomous separation assurance," *CoRR*, vol. abs/2003.08353, 2020. [Online]. Available: https://arxiv.org/abs/2003.08353

[5] V. D'silva, D. Kroening, and G. Weissenbacher, "A survey of automated techniques for formal software verification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1165–1178, 2008.

[6] C. Kern and M. R. Greenstreet, "Formal verification in hardware design: a survey," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 4, no. 2, pp. 123–193, 1999.

[7] R. W. Gardner, D. Genin, R. McDowell, C. Rouff, A. Saksena, and A. Schmidt, "Probabilistic model checking of the next-generation airborne collision avoidance system," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE, 2016, pp. 1–10.

[8] B. Chludzinski, "Evaluation of tcas ii version 7.1 using the faa fast-time encounter generator model-appendix," *Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-346 Volume*, vol. 2, 2009.

[9] M. J. Kochenderfer and J. P. Chryssanthacopoulos, "A decision-theoretic approach to developing robust collision avoidance logic," in *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2010, pp. 1837–1842.

[10] X. Yang, M. Egorov, A. Evans, S. Munn, and P. Wei, "Stress testing of uas traffic management decision making systems," in *AIAA AVIATION 2020 FORUM*, 2020, p. 2868.

[11] R. Lee, M. J. Kochenderfer, O. J. Mengshoel, G. P. Brat, and M. P. Owen, "Adaptive stress testing of airborne collision avoidance systems," in *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*. IEEE, 2015, pp. 6C2–1.

[12] R. Lee, O. J. Mengshoel, A. Saksena, R. W. Gardner, D. Genin, J. Silbermann, M. Owen, and M. J. Kochenderfer, "Adaptive stress testing: Finding likely failure events with reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 69, pp. 1165–1201, 2020.

[13] A. Baheri, H. Ren, B. Johnson, P. Razzaghi, and P. Wei, "A verification framework for certifying learning-based safety-critical aviation systems," *arXiv preprint arXiv:2205.04590*, 2022.

[14] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, "Adaptive stress testing for autonomous vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1–7.

[15] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, "Adaptive stress testing with reward augmentation for autonomous vehicle validatio," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 163–168.

[16] J. M. Hoekstra and J. Ellerbroek, "Bluesky ATC simulator project: an open data and open source approach," in *Proceedings of the 7th International Conference on Research in Air Transportation (ICRAT)*, vol. 131. FAA/Eurocontrol USA/Europe, 2016, p. 132.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[18] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[19] M. Brittain and P. Wei, "Autonomous aircraft sequencing and separation with hierarchical deep reinforcement learning," in *Proceedings of the 8th International Conference on Research in Air Transportation (ICRAT)*. FAA/Eurocontrol USA/Europe, 2018.

[20] M. Ribeiro, J. Ellerbroek, and J. Hoekstra, "Determining optimal conflict avoidance manoeuvres at high densities with reinforcement learning," in *Proceedings of the Tenth SESAR Innovation Days, Virtual Conference*, 2020, pp. 7–10.

[21] R. Isufaj, D. Aranega Sebastia, and M. A. Piera, "Towards conflict resolution with deep multi-agent reinforcement learning," in *Proceedings of the 14th USA/Europe Air Traffic Management Research and Development Seminar (ATM2021), New Orleans, LA, USA*, 2021, pp. 20–24.

[22] Z. Wang, H. Li, J. Wang, and F. Shen, "Deep reinforcement learning based conflict detection and resolution in air traffic control," *IET Intelligent Transport Systems*, vol. 13, no. 6, pp. 1041–1047, 2019.

[23] B. Wulfe, "Uav collision avoidance policy optimization with deep reinforcement learning," 2017.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[25] M. W. Brittain, X. Yang, and P. Wei, "Autonomous separation assurance with deep multi-agent reinforcement learning," *Journal of Aerospace Information Systems*, vol. 18, no. 12, pp. 890–905, 2021.

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[27] D.-T. Pham, N. P. Tran, S. K. Goh, S. Alam, and V. Duong, "Reinforcement learning for two-aircraft conflict resolution in the presence of uncertainty," in *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*. IEEE, 2019, pp. 1–6.

[28] N. P. Tran, D.-T. Pham, S. K. Goh, S. Alam, and V. Duong, "An intelligent interactive conflict solver incorporating air traffic controllers' preferences using reinforcement learning," in *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*. IEEE, 2019, pp. 1–8.

[29] H. Wen, H. Li, Z. Wang, X. Hou, and K. He, "Application of ddpg-based collision avoidance algorithm in air traffic control," in *2019 12th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 1. IEEE, 2019, pp. 130–133.

[30] D.-T. Pham, N. P. Tran, S. Alam, V. Duong, and D. Delahaye, "A machine learning approach for conflict resolution in dense traffic scenarios with uncertainties," in *ATM Seminar 2019, 13th USA/Europe ATM R&D Seminar*, 2019.

[31] A. G. Taye, J. Bertram, C. Fan, and P. Wei, "Reachability based online safety verification for high-density urban air mobility trajectory planning," in *AIAA AVIATION 2022 Forum*, 2022, p. 3542.

[32] J.-P. Katoen, "The probabilistic model checking landscape," in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, 2016, pp. 31–45.

[33] C. von Essen and D. Giannakopoulou, "Probabilistic verification and synthesis of the next generation airborne collision avoidance system," *International Journal on Software Tools for Technology Transfer*, vol. 18, no. 2, pp. 227–243, 2016.

[34] J. H. Gallier, *Logic for computer science: foundations of automatic theorem proving*. Courier Dover Publications, 2015.

[35] J. E. Holland, M. J. Kochenderfer, and W. A. Olson, "Optimizing the next generation collision avoidance system for safe, suitable, and acceptable operational performance," *Air Traffic Control Quarterly*, vol. 21, no. 3, pp. 275–297, 2013.

[36] K. D. Julian, R. Lee, and M. J. Kochenderfer, "Validation of image-based neural network controllers through adaptive stress testing," in *2020 IEEE 23rd international conference on intelligent transportation systems (ITSC)*. IEEE, 2020, pp. 1–7.

[37] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.