Exploring Model Quality for ACAS X

Dimitra Giannakopoulou[1], Dennis Guck*[2], and Johann Schumann[1]

[1] NASA Ames Research Center, Moffett Field, CA, USA,
[2] Formal Methods and Tools, University of Twente, the Netherlands

Abstract. The next generation airborne collision avoidance system, ACAS X, aims to provide robustness through a probabilistic model that represents sources of uncertainty. From this model, dynamic programming produces a look-up table that is used to give advisories to the pilot in real time. The model is not present in the final system and is therefore not included in the standard certification processes. Rather, the model is checked indirectly, by ensuring that ACAS X performs as well as, or better than, the state-of-the-art, TCAS. We claim that to build confidence in such systems, it is important to target model quality directly. We investigate this issue of model quality as part of our research on informing certification standards for autonomy. Using ACAS X as our driving example, we study the relationship between the probabilistic model and the real world, in an attempt to characterize the quality of the model for the purpose of building ACAS X. This paper presents model conformance metrics, their application to ACAS X, and the results that we obtained from our study.

1 Introduction

Advanced algorithms for decision-making often rely on the use of models, i.e., abstract representations of knowledge that the algorithms need in order to operate correctly. Such algorithms appear increasingly in safety critical systems with the introduction of autonomy, and the need to make decisions, initiate mitigation actions, and adapt to changing environments and unanticipated situations.

Our encounter with such algorithms has been in the context of ACAS X, the next generation Airborne Collision Avoidance System [7,8]. The current collision avoidance standard, TCAS [9], is required on all large passenger and cargo aircraft worldwide, and has been successful in preventing mid-air collisions. However, its deterministic logic limits robustness in the presence of unanticipated circumstances.

To increase robustness, ACAS X uses a probabilistic model to represent uncertainty. Simulation studies with recorded radar data have confirmed that this novel approach leads to a significant improvement in safety and operational performance. ACAS X has also been the target of several formal verification efforts [6,15,16]. The FAA has formed a team of organizations to mature ACAS X

* Author performed this work while employed by SGT, Inc. as an intern at the NASA Ames Research Center.

into a new international standard for collision avoidance for manned and unmanned aircraft.

In this work, we focus on the problem of identifying quality metrics for models used in decision-making. In particular, since a model is an abstraction of information, how can we determine whether this model is satisfactory? In the context of ACAS X, the ultimate goal is to avoid aircraft collisions. However, to establish trust in such a safety-critical system, we claim that one must additionally show that each decision is supported by accurate model information.

To address these questions, our research aims at establishing criteria for model quality. These criteria should be measurable, and should be helpful in developing certification standards for algorithms that use models. Moreover, model quality should be directly associated with overall system quality. In other words, poor model quality should indicate or predict violations of system requirements.

The work presented in this paper focuses on ACAS X and defines relations between the probabilistic model and the real world. We discuss model conformance relations that we developed and the (sometimes unanticipated) results of their application to ACAS X. Moreover, we have developed techniques that stress-test ACAS X by generating test cases that may exhibit poor model quality with respect to our defined relations.

The rest of the paper is organized as follows. Section 2 describes the ACAS X system and motivates this work, while Section 3 discusses the probabilistic model used by ACAS X in more detail. Section 4 discusses the relations that we have defined for evaluating model quality. The application of these relations to ACAS X is presented in Section 5. In Section 6 we focus on the generation of data for stress-testing ACAS X, and present the results of our analysis. Section 7 lists related work and Section 8 closes the paper with conclusions and future work.

2 Background and Motivation

The aim of aircraft collision avoidance systems is to reliably prevent midair collisions while minimizing unnecessary pilot alerting and evasive maneuvers. Uncertainties such as sensor noise or errors, aircraft intent, and pilot behavior make it challenging to design such systems.

In the context of collision avoidance algorithms, we use the term loss of horizontal separation (LHS) to describe the situation where two aircraft are within 500 ft from each other if their altitude difference is ignored. A Near Mid-Air Collision (NMAC) occurs when the altitude difference between the two aircraft is at most 100 ft when LHS occurs. We refer to the aircraft equipped with a collision avoidance system as the *ownship*, and the other aircraft as the *intruder*.

The current collision avoidance standard, TCAS [9], uses several sources to estimate the current state of the aircraft on which it is deployed and other aircraft in its vicinity. If another aircraft is a potential threat, then TCAS issues a traffic advisory, which gives the pilots of the ownship an audio announcement "Traffic, Traffic" and highlights the intruder on a traffic display. This serves as a warning to raise the pilot's awareness for the potential need to maneuver.

If a maneuver becomes necessary the system will issue a resolution advisory (RA) instructing the pilot to climb or descend in order to maintain a safe distance. After the encounter is resolved, TCAS issues a "Clear of Conflict" (`COC`). Only advisories for vertical maneuvers (climb, descend, and maintain altitude) are given, together with the target rate. For example, advisory `DES1500` stands for descend with rate 1500 ft/min. Preventive advisories to avoid climbing or descending may also be provided, as described in [7].

The TCAS system has been implemented as a traditional piece of software with conditional branches that model all the different situations. Several years of research have resulted in the development of the ACAS X system. Although the interface of ACAS X to the pilot is the same as TCAS, the underlying collision avoidance algorithm is dramatically different [7,8].

In ACAS X, a probabilistic Markov Decision Process (MDP) model provides a coarse abstraction of how an encounter between two aircraft progresses as a result of applying resolution advisories and of time passing. Based on a reward function and this MDP, dynamic programming generates a look-up table (LUT), which associates each encounter state in the MDP with a cost for each possible ACAS X advisory. The LUT is deployed onboard the aircraft. Every second, ACAS X uses sensors and other information to compute a probability distribution of the states in which an encounter may be at the current time t. ACAS X interpolates the state estimate within the discrete states of the LUT, and calculates the advisory that bears the lowest cost.

In essence, the cost of an advisory computed by dynamic programming is based on how the MDP expects an encounter to evolve from the current state. Therefore, there is a trade-off between the MDP model being accurate enough for the advisories to be appropriate for collision avoidance in reality, but also abstract enough for enabling a compact LUT that can be deployed onboard the aircraft. The ACAS X system is based on a relatively simple dynamic model of how aircraft behave. As explained in [7], a simple model is easier to understand and validate, makes the dynamic programming problem more tractable, and results in a smaller controller, which is easier to fit into memory onboard an aircraft. How can we establish that this simple model is acceptable for ACAS X? Even though the resulting system is tested extensively with independent high fidelity simulations and flight data, it is hard to establish whether the behavior of such optimization algorithms is as expected.

The aim of our work is therefore to develop measurable criteria that directly address model quality. The approaches that we present in this paper are all based on establishing relationships between the evolution of encounters 1) as expected by the MDP model and 2) as recorded by high fidelity simulations and actual flight data.

3 The ACAS X Model

The formulation of the collision avoidance problem used for ACAS X consists of two aircraft, the ownship and the intruder, on a collision course[3]. The ACAS X model keeps record of the altitude of the intruder relative to the ownship, the aircraft climb rates, the produced advisory and the pilot response.

To keep the model tractable, ranges are set for each variable. The MDP discretizes each state variable with a resolution that depends on the proximity between the aircraft. Previous work [16] has shown that the discretization resolution constitutes an important trade-off between accuracy and the size of the LUT, which is important for implementation on-board the aircraft.

For a state variable x, we denote its discretized value by $\hat{x}$. The discretized MDP model state $\hat{s} = \langle \hat{z}_{\mathrm{rel}}, \hat{dz}_{\mathrm{o}}, \hat{dz}_{\mathrm{i}}, \mathrm{sRA} \rangle$ encodes

1. $z_{\mathrm{rel}} \in [-1000, 1000]$ ft, the altitude difference between the two aircraft;
2. $dz_{\mathrm{o}} \in [-2500, 2500]$ ft/min, the ownship's climb rate;
3. $dz_{\mathrm{i}} \in [-2500, 2500]$ ft/min, the intruder's climb rate;
4. sRA, encoding the ACAS X advisory produced one second earlier and the advisory the pilot is following, thus modeling pilot delay.

Given a discrete state $\hat{s} = \langle \hat{z}_{\mathrm{rel}}, \hat{dz}_{\mathrm{o}}, \hat{dz}_{\mathrm{i}}, \mathrm{sRA} \rangle$ and an advisory adv, the MDP provides probabilistic state transitions into new states $\hat{s}'$, i.e., $MDP : (\hat{s}, \mathsf{adv}) \xrightarrow{p} \hat{s}'$ with state transition probability p.

In order to obtain an optimal controller, the individual advisories are associated with a cost. For example, a clear-of-conflict (`COC`) carries a reward, whereas alerting the pilot through climb and descend advisories carry a small cost to avoid unnecessary pilot alerting. An NMAC situation has an extremely high cost.

Although the MDP in itself is not aware of timing throughout the encounter, the ACAS X system uses a temporal distribution (τ-distribution) to model the temporal sequence of the encounter and avoidance strategy, since it is expected to operate when NMAC situations may occur within the next 40–50 seconds.

Figure 1 shows the overall architecture of the ACAS X system and its development process. The top of the panel shows the development of the probabilistic MDP model and the generation of the look-up table using dynamic programming. This LUT comprises the core of the ACAS X software that is running on-board the aircraft (bottom part of Figure 1). During each 1-second update, new estimates of the aircraft involved and uncertainties are calculated based upon sensor measurements and transponder responses. This information is then used to update the aircraft state and τ distribution, and the look-up table is consulted to come up with an appropriate advisory. In addition to the LUT costs discussed previously, on-line corrections are also taken into account in selecting advisories (see [7] for details).

[3] ACAS X handles cases with multiple aircraft but this paper focuses on scenarios involving two aircraft.

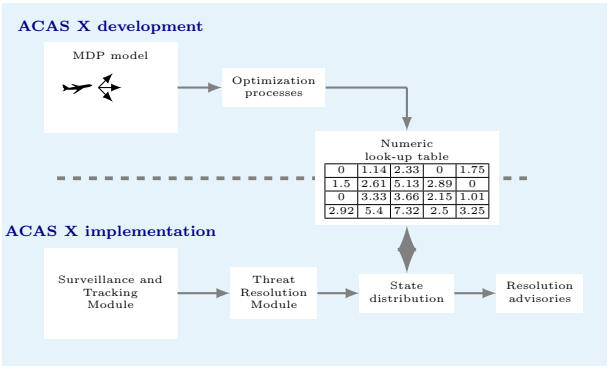

Fig. 1. Principled architecture of the ACAS X modeling process (top) and on-board software architecture (bottom).

4 Model Conformance

As discussed, the MDP model used by ACAS X captures the expected evolution of flight encounters as a result of time passing and of the application of resolution advisories. To ensure safe operation, the model and the actual system behavior must in some way "match up". In other words, we need to be able to justify that the model is appropriate for the decision making that it is used for.

To this aim, we introduce several notions of *conformance* to characterize model quality. All of these notions capture desired relations between the behavior or states of a model $\mathcal{M}$ and the actual system $\mathcal{A}$ that uses this model. In ACAS X for example, conformance might require that whenever an encounter results in an NMAC in the actual ACAS X system, then the model of this encounter also results in an NMAC. Or when the actual system $\mathcal{A}$ produces an advisory adv, then the model will produce an identical or compatible advisory.

Similarly, one might expect that all situations that occur in practice in the system will be reflected in the model (within the bounds of the model abstraction). This notion is related to over-approximation in formal methods. Imagine, for example, that during flight, an encounter reaches a state s' from a state s, a transition that the model does not anticipate (i.e., in the model s cannot directly transition to s'). ACAS X may still work appropriately or may produce wrong or unsafe results.

Even though such notions of conformance appear to be relatively simple, defining them in the context of ACAS X is made non-trivial by the presence of probabilistic reasoning and state discretization. In the following sections we describe how we factor in these characteristics into our model quality criteria.

4.1 Conformance framework set up

As discussed in Sections 2 and 3, during ACAS X deployment, the (geometric) state of an encounter is represented by a weighted distribution of states. At

each second, the Cartesian product of the geometric state estimate with the τ-distribution (weighted distribution of estimated time to LHS) forms the current state estimate. This state estimate is then interpolated within the discrete states of the LUT in order to calculate a resolution advisory.

To establish model conformance in this context, our framework needs to set up the model and the actual system appropriately. The model $\mathcal{M}$ is ACAS X as deployed within the MDP model[4]. This means that the geometric state distribution is provided by the transitions of the MDP model, as opposed to sensor information. The actual system $\mathcal{A}$ is ACAS X as deployed and using the LUT within a real flight environment or within a high fidelity simulator. To then establish model conformance we set up a common initial state for $\mathcal{M}$ and $\mathcal{A}$ and compare their evolutions according to conformance criteria. More specifically, the initial state of $\mathcal{M}$ is set to the initial state of $\mathcal{A}$ for each encounter $\mathcal{E}$ to be analyzed for conformance.

There are two options for comparing the evolution of encounters. One is a *stepwise* synchronization, which means that we re-synchronize the state of $\mathcal{M}$ and $\mathcal{A}$ once every second, namely every time ACAS X is invoked. The second is an *initial-state* synchronization, where we start from the same state but then let the two systems evolve independently.

For $\mathcal{A}$, state and τ distributions are obtained by the state estimation components, as discussed in earlier sections. For $\mathcal{M}$, the time to LHS decreases by one with every invocation of ACAS X, which is the way it is also updated for the purpose of calculating the LUT. The state distributions are obtained as follows. Let us assume that at some point in encounter $\mathcal{E}$, the MDP moves from state set $\hat{S}_{\mathcal{M}}$ to state set $\hat{S}'_{\mathcal{M}}$. Then the probability $p_{\mathcal{M}}(s')$ of each state $s' \in \hat{S}'_{\mathcal{M}}$ is obtained as follows. For each $s \in \hat{S}_{\mathcal{M}}$ let $p_{\mathcal{M}}(s, s')$ denote the probability associated with the transition from s to s' in the MDP ($p_{\mathcal{M}}(s, s') = 0$ means that s cannot transition to s' in a single step). Then $p_{\mathcal{M}}(s') = \sum_{s \in \hat{S}_{\mathcal{M}}} p(s, s')$.

Decision making is based on the LUT. The LUT used by $\mathcal{M}$ is identical to that used by $\mathcal{A}$ for ACAS X deployment. For this reason, we compare states of $\mathcal{M}$ and $\mathcal{A}$ in terms of their interpolated states within the LUT. Figure 2 illustrates an example of a stepwise synchronization between $\mathcal{M}$ and $\mathcal{A}$. $\mathcal{M}$ and $\mathcal{A}$ start at the same set $\hat{S}_{\mathcal{A}}$ of geometric states, which has been estimated during operation of $\mathcal{A}$. Note that we only illustrate the possible states in which the system may be, without including the probabilities associated with those states. At the next ACAS X cycle, one second later, the grey state set $\hat{S}'_{\mathcal{M}}$ is computed based on MDP transitions starting from $\hat{S}_{\mathcal{A}}$, whereas the blue set of states $\hat{S}'_{\mathcal{A}}$ is computed based on sensor and transponder information.

4.2 Conformance relations

Based on the above setup, we can define expectations for model and actual system behavior. If $\mathcal{M}(\mathcal{E})$ is the behavior of the model $\mathcal{M}$ on encounter $\mathcal{E}$, and $\mathcal{A}(\mathcal{E})$

[4] We use the MDP rather than its corresponding continuous version for aircraft dynamics, because the generation of the LUT is based on the MDP model.

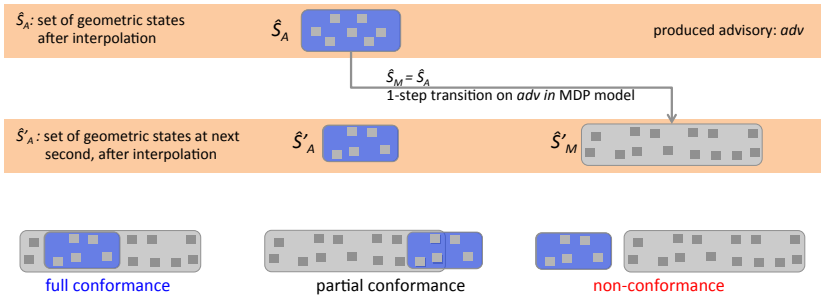

Fig. 2. State conformance between ACAS X and the MDP.

the behavior of ACAS X on $\mathcal{E}$, conformance establishes an expected relation $\mathcal{C}$ between the two behaviors: $\mathcal{C}(\mathcal{M}(\mathcal{E}), \mathcal{A}(\mathcal{E}))$. For the ACAS X system, conformance can be defined at three distinct levels: NMAC conformance, advisory conformance, and state conformance.

The NMAC conformance metric is the coarsest one and just focuses on the dangerous NMAC state: whenever $\mathcal{A}(\mathcal{E})$ encounters an NMAC, $\mathcal{M}(\mathcal{E})$ has to encounter an NMAC, and vice versa.

At the next level of detail, we focus on the interactions of ACAS X with the outside world, namely the sequence of advisories. Model and implementation are in conformance if the sequence of advisories issued are the same or compatible, where the notion of compatibility has to be specified. For example, we may tolerate small deviations in the time at which an advisory is issued.

A state conformance requires that, at each point in time, the state of $\mathcal{M}$ and $\mathcal{A}$ are compatible, where compatibility may take into account state distributions. For simplicity, we first consider the geometric states of $\mathcal{M}$ and $\mathcal{A}$, without taking into account their associated probabilities, as illustrated in Figure 2. Each state can be viewed as a point in a high-dimensional state space, where each dimension corresponds to one of the state variables in the geometric state.

One measure of conformance is to expect that the system states are fully contained in the model states, which we name *full state conformance* (Figure 2).

Intuitively, this means that the model anticipates all the possible evolutions of the encounter in a single step. Whereas full conformance represents the ideal situation, a partial overlap might also be acceptable, leading to a notion of *partial state conformance*. For this particular case, and given that states are weighted, partial conformance relations may also take these weights into account in order to evaluate the significance of the overlap.

Finally, if the model and system states are disjoint, we have an undesirable *no state conformance* situation. Decision making in ACAS X is based on how the MDP expects encounters to evolve. Such an extreme mismatch between the MDP and the actual system may result in providing advisories that will not prevent an NMAC from occurring. Figure 2 illustrates these three types of conformance.

Note that "no state conformance" need not directly result in a violation of ACAS X requirements for the encounter that is being studied; it is possible that NMAC just happens to be avoided. However, detecting non-conformance in a

tested encounter is still valuable since it indicates a mismatch that may result in dangerous behavior in other encounters that are not included in the testing. As such, conformance relations might be useful for predictive runtime analysis of systems.

5 Analyzing Conformance Issues

We applied a variety of conformance metrics to ACAS X test data that we obtained from the ACAS X team as part of the ACAS X Run 13 distribution. We ran both initial-state and stepwise variants of these metrics. Overall, we observed several conformance issues that need to be studied more carefully to determine their significance. For example, Figure 3B illustrates three bars of advisories at the bottom. The first bar is for the actual system, the second for the interpolated system states (i.e. ACAS X without on-line advisory corrections) and the third one for the model with a stepwise synchronization of the MDP and the system. We can observe that the same advisories are produced but in the third case the descend advisory is issued much earlier.

Among all the observations we made through our experiments, the cases that we believe need the most immediate investigation are cases of stepwise non-conformance. In fact, we did not anticipate that such mismatches between model and system might be possible within a single step. In this section, we analyze one of the cases of non-conformance.

Let us consider as an example encounter 86 within the official ACAS X distribution referred to as Run 13. Figure 3A shows a 3D representation of the flight path for each of the involved aircraft. The encounter starts when the two aircraft are still safely apart (locations marked by small circles). As soon as the ACAS X system on the ownship detects a potentially dangerous development (at $t = 19\,\mathrm{s}$) into the encounter, an alarm "Descend, Descend!" is annunciated in the cockpit and an advisory to descend with 1500 ft/min (`DES1500`) is issued (Figure 3B, red line). After a short delay, the pilot reacts and follows that advisory and the vertical velocity of the ownship becomes negative. The vertical speed of both aircraft are shown in Figure 3B, middle panel. At time $t = 29\,\mathrm{s}$, ACAS X advises the pilot to not climb (`DNC`, Figure 3B, green line) and the pilot levels off. As soon as the danger of an NMAC has been averted, a `COC` advisory is given ($t = 45\,\mathrm{s}$) and the encounter ends successfully.

We checked state conformance on this encounter. We remind the reader that state conformance synchronizes the geometric model state to that of the system at each step, and then compares the geometric states of the system and model in the next step. Given that the value of sRA is always the same in this encounter, we focus on the remaining 3 state variables: $\mathrm{z}_{\mathrm{rel}}$, dz_{o}, and dz_{i}.

Figure 3D-F illustrates model/system states as gray/blue clouds respectively, at different points in the encounter. Each cloud contains the projection of states on these 3 variables. Due to uncertainties in movement of the intruder and pilot reactions, size and shape of the overlapping parts of the blue and gray clouds vary

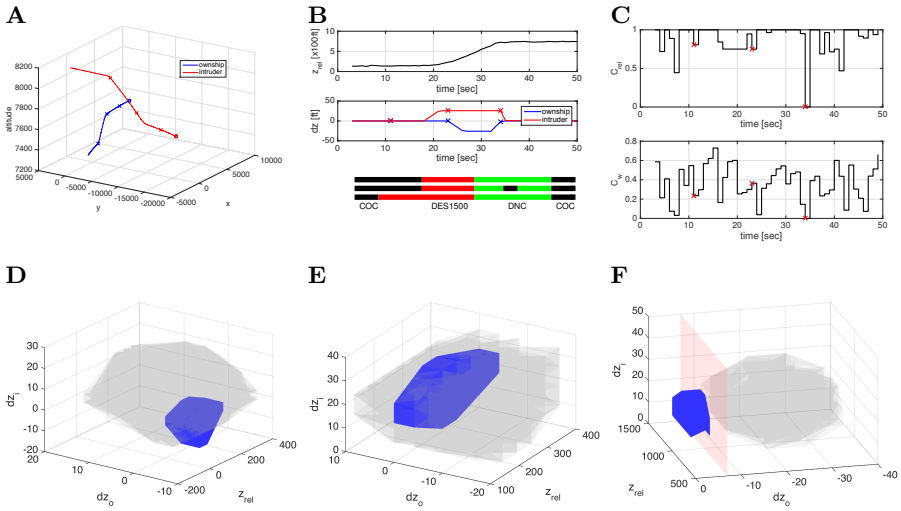

Fig. 3. Visualization of non-conformance encounter. **A:** trajectories of ownship (blue) and intruder (red). Circles mark the starting-points. **B:** ACAS X data showing (top to bottom) relative altitude z_{rel} over time, vertical velocities (dz_o and dz_i) and sequence of produced advisories by ACAS X, interpolated system states, stepwise synchronized MDP. **C:** Conformance metrics C_{rel} (top) and C_w (bottom). Small red x signs indicate time points $t = 11\,\text{s}, 23\,\text{s}, 34\,\text{s}$, associated with panels D-F. **D-F:** 3D projections of $\hat{S}'_{\mathcal{M}}$ (gray) and $\hat{S}'_{\mathcal{A}}$ (blue) at $t = 11\,\text{s}, 23\,\text{s}, 34\,\text{s}$.

during the encounter. Fig. 3F, however, illustrates a non-conformance situation. The red plane in the figure clearly separates the two clouds.

To analyze non-conformance situations such as the one illustrated in Fig. 3F, it is sometimes helpful to analyze the conformance with respect to individual variables of the state. The reason is that non-conformance may be due to the way a particular variable is modeled, which enables us to give precise and helpful information to developers. For example, the non-conformance situation of Fig. 3 is due to variable dz_o. Note that this does not necessarily occur in all non-conformance situations. It may be that variables are covered individually, but their combination is not.

We define a metric of relative state conformance w.r.t. $\hat{S}'_{\mathcal{M}}$ and $\hat{S}'_{\mathcal{A}}$ as follows:

$$C_{\text{rel}} = \frac{|\hat{S}'_{\mathcal{M}} \cap \hat{S}'_{\mathcal{A}}|}{|\hat{S}'_{\mathcal{A}}|}.$$

Hence, $C_{\text{rel}} \in [0, 1]$ where $C_{\text{rel}} = 0$ describes a non-conformance situation and $C_{\text{rel}} = 1$ full conformance. Figure 3C (top) shows C_{rel} for the example shown in Figure 3. The non-conformance at $t = 34\,\text{s}$ is clearly visible.

In order to study more accurately cases of partial conformance i.e., $0 < C_{\text{rel}} < 1$, we want to consider additional information perusing the fact that $\hat{S}'_{\mathcal{A}}$ and $\hat{S}'_{\mathcal{M}}$ are distributions. Informally speaking, the higher $p_{\mathcal{A}}(s)$ of some $s \in \hat{S}'_{\mathcal{A}}$,

the more weight it carries for decision making. If those important states are well represented in the MDP (with a high $p_{\mathcal{M}}$) then conformance is very good and the metric should be high. In other words, we wish to measure the similarity between the MDP and actual system information. In an ideal situation, the model and the system will contain the exact same pairs of states and corresponding weights.

Let us assume that we are comparing $\hat{S}'_{\mathcal{M}}$ and $\hat{S}'_{\mathcal{A}}$ for state conformance, where each state $s_{\mathcal{M}} \in \hat{S}'_{\mathcal{M}}$ and $s_{\mathcal{A}} \in \hat{S}'_{\mathcal{A}}$ is associated with probability $p_{\mathcal{M}}(s)$ and $p_{\mathcal{A}}(s)$, respectively. We then define a weighted state conformance metric $C_w = 1 - C_{\text{diff}}$ in terms of the sum of the absolute differences between the probabilities of the states in $\hat{S}'_{\mathcal{A}}$ and $\hat{S}'_{\mathcal{M}}$:

$$C_{\text{diff}} = \frac{1}{2} \sum_{s \in (\hat{S}'_{\mathcal{A}} \cup \hat{S}'_{\mathcal{M}})} |p_{\mathcal{A}}(s) - p_{\mathcal{M}}(s)|$$

The sum is divided by 2, which represents the maximum possible divergence between the sets. Indeed, in the presence of non-conformance, the probability differences will add up to 1 for the model, and 1 for the system, for a total of 2. Note that when a state does not belong to a set of states, we represent it as its probability being 0 for that set. $C_w = 1$ corresponds to $C_{\text{diff}} = 0$, i.e., the sets have the same states with the same associated probabilities. Full conformance situations only have a high value of C_w if the probability mass of the model lies mostly within the subset that is covered by the actual system, and if the states in that subset are weighted similarly to their corresponding ones in the actual system. Figure 3C (bottom) shows C_w for our example encounter.

Since the case of non-conformance is the most urgent to report to the ACAS X team, the rest of the paper focuses on our study of non-conformance cases aiming at providing useful information for the developers to examine the issue. Since non-conformance is very rare in the test data that we received as part of the ACAS X release, we decided to focus on generating additional encounters exhibiting non-conformance, using machine learning.

6 Automatic Generation of Non-Conformance Encounters

The complexity of the ACAS X input domain makes it hard to explore it systematically. We therefore based our initial experiments on test encounters prepared by the developers of ACAS X that were included with the ACAS X distribution. When measuring state conformance on those, we encountered only a handful of situations where non-conformance exists. However, those situations are a hint that there are discrepancies between the real world evolution based on ACAS X and the corresponding MDP model. For a more thorough analysis of this phenomenon a much larger data set is required.

In this section, we propose techniques for the automated generation of non-conformance scenarios. In addition to generating such scenarios, we want to be able to provide constraints that further filter the encounters to the most interesting and safety-critical cases. For example, we wish to focus on situations where an actual advisory is issued because of the close proximity of the aircraft.

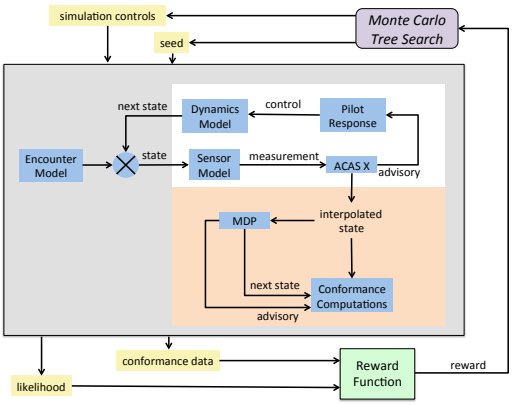

Fig. 4. Encounter generation framework. The white box at the top is RLESCAS for one aircraft equipped with ACAS X, and the orange box is our extension.

6.1 The scenario generation environment

Our approach extends the RLESCAS (Reinforcement Learning Encounter Simulator for Collision Avoidance Systems) package for adaptive stress testing of airborne collision avoidance systems [10]. RLESCAS uses Monte Carlo Tree Search (MCTS) to automatically generate two-aircraft NMAC encounters.

Figure 4 illustrates how our framework extends RLESCAS for the case where a single aircraft is equipped with ACAS X. Our framework implements two main changes to the original framework. First, it introduces the MDP in order to compute information needed for the conformance relations. Second, it modifies the reward function to favor the generation of low-conformance encounters.

The original framework relies on a simulator for aircraft encounters; in our case one of the aircraft is equipped with ACAS X. Inputs to this simulation environment are basic simulator controls, like initialize, and a seed. These are the only variables that the framework is able to manipulate in order to target specific types of encounters.

Our extension intercepts the operation of the ACAS X component within the simulation to obtain interpolated states that can be fed to the MDP. Then the MDP and the LUT are used as described in the previous sections to calculate conformance data. The output of the framework is the likelihood of the current transition, a variable describing the conformance of the current state, and the weighted conformance metric. The outputs are subjected to our modified reward function used by the MCTS algorithm. The result of the MCTS is in turn used to choose the seed and control inputs for the simulator. A detailed description of the original RLESCAS and the MCTS algorithm can be found in [10].

6.2 The reward function

The encounter generation framework aims at maximizing the reward function. Therefore, our reward function $\mathcal{R}$ must be designed to reward the occurrence

of non-conformance, but it must also be able to provide guidance on how to evaluate the current situation.

We start with the reward function given in [10] and gradually modify it to serve the purposes of our framework. The objective of the function in [10] is to find high probability encounters that contain NMAC events:

$$\mathcal{R}(s_t, s_{t+1}) = \begin{cases} 0 & \text{if } s_t \in \text{NMAC}, \\ -\infty & \text{if } s_t \notin \text{NMAC}, t \geq T, \\ log(P(s_{t+1} \mid s_t)) & \text{if } s_t \notin \text{NMAC}, t < T. \end{cases} \tag{1}$$

The reward for going from a state s_t to s_{t+1} depends on two main events: 1) if an NMAC occurs ($s_t \in \text{NMAC}$), and 2) if the maximum simulation time has been reached ($t \geq T$). T is set to the time horizon of ACAS X, which is 50s in the RLESCAS framework. Reaching T therefore indicates a terminal state in our framework; all NMAC situations, if any, have to occur by that time.

The first two conditions of $\mathcal{R}$ represent the NMAC occurrence constraint. If an NMAC occurs a maximal reward is issued, whereas if the time horizon T has been reached and no NMAC has occurred, an infinite penalty is issued. In all other cases a reward based on the probability to be in the current state is issued to maximize the likelihood of the encounter. Note that this function assigns negative rewards, in other words, penalizes undesirable situations to a higher or lower degree, and assigns 0 to the desired outcome.

To adapt the reward function for the generation of non-conformance encounters we investigate variations of Equation (1). As a first attempt, our objective is similar to that of the original reward function, but for non-conformance (NC) instead of NMAC events. Our reward function then infinitely penalizes the learner when no non-conformance event is encountered.

However, we introduce a change for the evaluation of intermediate situations, because we want to generally encourage mismatches between the system and the MDP. We do so by rewarding small intersections between system and model states, i.e., partial conformance (the smaller the intersection the better):

$$\mathcal{R}(s_t, s_{t+1}) = \begin{cases} nc & \text{if } s_t \in \mathsf{NC}, \\ 0 & \text{if } s_t \notin \mathsf{NC}, t \geq T, \\ (1 - C_{\text{rel}}) \cdot pc & \text{if } s_t \notin \mathsf{NC}, t < T. \end{cases} \tag{2}$$

Hence, $\mathcal{R}$ is geared towards finding encounters with a non-conformance event (NC) and with a low conformance metric throughout the encounter. Note that instead of using negative rewards as in Equation (1) this function uses positive rewards. We define two positive reward parameters nc and pc representing the reward for non-conformance events and partial conformance events, respectively. We parameterize the reward function in this fashion in order to be able to experiment with different levels of relative importance to the two aspects of the targeted encounters. Parameter pc is weighted by $(1-C_{\text{rel}})$, which represents the ratio of system states that are not covered by the model. If no non-conformance event occurred, a reward of 0 is issued.

EC #	time point	$C_{\text{rel}\,\vert \text{z}_{\text{rel}}}$	$C_{\text{rel}\,\vert \text{dz}_{\text{o}}}$	$C_{\text{rel}\,\vert \text{dz}_{\text{i}}}$	$C_{\text{rel}\,\vert \text{sRA}}$
1	24	16/16	16/16	0/16	16/16
	49	26/30	30/30	0/30	30/30
6	19	10/12	12/12	0/12	12/12
	33	24/28	28/28	0/28	28/28
	35	28/28	28/28	4/28	28/28
	45	20/20	20/20	0/20	20/20
9995	35	40/40	40/40	0/40	40/40
	40	24/28	28/28	0/28	28/28
	42	24/24	12/24	4/24	24/24

Table 1. Subset of generated non-conformance encounters, including conformance information per state variable.

Applying the reward function given in Equation (2) in the MCTS algorithm enables us to generate non-conformance encounters. However, we observed that in many cases, the altitude difference between the two aircraft remained high, so ACAS X never issued any advisories other than COC. Such encounters are not very interesting for our study.

The natural next step is then to find a reward function that combines objectives from Equations (1) and (2). The objective of our new reward function is to trigger non-conformance, involve low conformance, and minimize the altitude difference between the aircraft at the time of closest approach.

$$\mathcal{R}(s_t, s_{t+1}) = \begin{cases} nc & \text{if } s_t \in \mathsf{NC}, \\ fc & \text{if } s_t \in \mathsf{FC}, \\ (1 - C_w) \cdot pc & \text{if } s_t \in \mathsf{PC}, t < T, \\ -\text{z}_{\text{rel}} & \text{if } s_t \notin \text{NMAC}, t \geq T. \end{cases} \tag{3}$$

Like before, nc and pc are positive parameters, and we introduce parameter fc, which is negative or 0. Function $\mathcal{R}$ now includes both positive and negative rewards. It penalizes encounters with no NMAC with the relative distance, whereas positively rewards partial and non-conformance. Partial conformance (PC) is weighted by $(1 - C_w)$ to encourage a low probability match between the model and system states. Full conformance (FC) receives a negative or 0 reward. Note that, in order to increase the likelihood of such encounters, one could add to the reward the probability to be in the current state, similarly to Equation (1).

6.3 Analysis of generated non-conformance encounters

We used reward function (3) in our framework to generate 18 encounters with a total of 33 non-conformance events. Table 1 displays three of these encounters and decomposes non-conformance events into conformance of individual state variables, to identify whether mismatches are associated with particular variables, as discussed in Section 5.

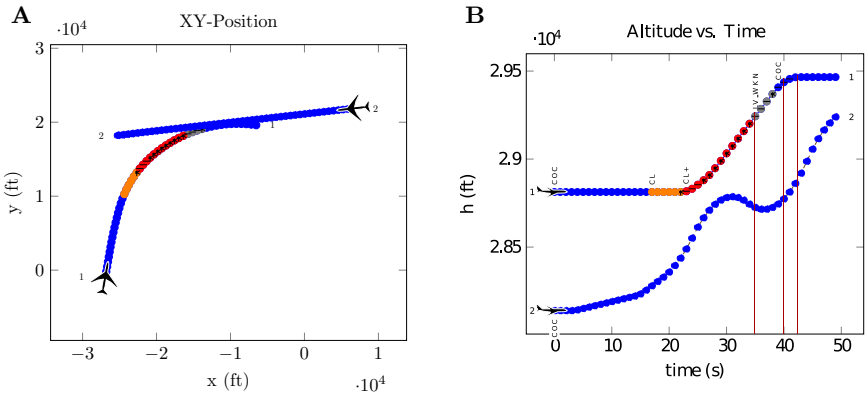

Fig. 5. Encounter 9995 generated with reward function from Equation (3). Thin red lines indicate non-conformance at $t = 35\,\mathrm{s}, 40\,\mathrm{s}, 42\,\mathrm{s}$.

We use $\mathcal{P}_{\mathrm{z_{rel}}}(\mathcal{S})$, $\mathcal{P}_{\mathrm{dz_o}}(\mathcal{S})$, $\mathcal{P}_{\mathrm{dz_i}}(\mathcal{S})$ and $\mathcal{P}_{\mathrm{sRA}}(\mathcal{S})$, to denote the sets obtained by projecting each state $s \in \mathcal{S}$ onto its variable $\mathrm{z_{rel}}$, $\mathrm{dz_o}$, $\mathrm{dz_i}$, and sRA, respectively. Then for system state $\hat{S}_{\mathcal{A}}$ and model state $\hat{S}_{\mathcal{M}}$, conformance relative to each state variable x is defined as:

$$C_{\mathrm{rel}\,|x} = \frac{|\mathcal{P}_x(\hat{S}_{\mathcal{M}}) \cap \mathcal{P}_x(\hat{S}_{\mathcal{A}})|}{|\mathcal{P}_x(\hat{S}_{\mathcal{A}})|}$$

Table 1 indicates that for the wide majority of non-conformance events, the states deviate in variable $\mathrm{dz_i}$. We further analyzed some of these encounters to gain intuition of the types of characteristics that may be causing non-conformance. Consider Figure 5B visualizing the encounter of two aircraft w.r.t. their altitude over time, for example. The corresponding non-conformance time points of this encounter are at 35, 40 and 42 seconds into the encounter. Inspecting the altitude changes of the intruder in the interval of $[35, 42]$ reveals a sudden change from descend to a relatively strong climb. This sudden and steep altitude change is not reflected in the MDP.

Even though in such encounters the behavior of ACAS X appears reasonable, it is important to study non-conformance occurrences closely, in case they trigger problematic decision-making under potentially rare circumstances. We are currently in the process of examining these results together with the ACAS X team in order to determine whether model fine-tuning may be beneficial, and whether the root cause is in the continuous model or its discretization.

7 Related Work

Essen and Giannakopoulou developed the Verica tool and applied probabilistic verification and synthesis to an early version of ACAS X [15,16]. Their aim was to study the impact of design issues such as model discretization and the selection of costs for the dynamic programming.

Jeannin et al. [6] analyzed ACAS X using hybrid approaches. They performed analysis on hybrid models of the system. They then used the KeYmaera tool to compute safe regions for restricted types of encounters and for a single advisory. Safe regions characterize the types of advisories that are safe for the corresponding encounter. ACAS X advisories for specific encounters can then be compared against their corresponding safe regions. The advantage of taking a hybrid approach is that it does not require discretization. However, the entire hybrid model for ACAS X is prohibitively large, which forced the authors to work with a restricted number of scenarios.

Researchers [2,3,11,12,13,14] have investigated hybrid techniques and theorem proving for other collision avoidance systems. Some researchers have developed testing frameworks for automated air-traffic control [1,4,5]. In order to evaluate the performance of ACAS X, the ACAS X team heavily relies on the simulation of a large number of encounters including recorded flight data.

More broadly, our work is related to several disciplines, such as model-based design, runtime monitoring, abstraction, and model validation. Model-based design uses models to describe, analyze, and generate code for a software system. In our work, models are abstractions of the real-world that software algorithms use for decision-making. Our conformance relations can be viewed as requirements that can be monitored at runtime, and be used for predictive analysis. These requirements are aimed particularly at establishing model quality. With respect to abstraction and model validation, our work develops metrics for validation of an abstraction in the context of its use for decision making. In other words, our conformance relations set application-specific requirements for an abstraction.

8 Conclusions

Autonomy requires models for decision making, adaptation and self-healing. For safety-critical autonomous systems, it is imperative that we develop new methods for establishing trust in these models.

In this paper, we explored and applied several model conformance criteria in the context of the ACAS X collision avoidance system, and discovered rare cases of non-conformance. We used machine learning to automatically generate additional encounters that exhibit non-conformance, and were able to identify potential causes for this issue.

In the future, we will work with the ACAS X team on developing approaches that help them prioritize the issues that our techniques report. We will also experiment with additional reward functions for encounter generation. Finally, we plan to use statistical learning techniques for the analysis of conformance issues in order to help developers with debugging their models.

Acknowledgements. We thank Ritchie Lee for his help with RLECAS, Ryan Gardner for helping us interpret the MDP, and Ryan Gardner, Mykel Kochenderfer, Ritchie Lee, and Josh Silbermann for useful discussions and for proof-reading the paper. This work was performed under the Safe Autonomous Systems Operations (SASO) project of the NASA AOSP program.

References

1. M. Dimjasevic and D. Giannakopoulou. Test-case generation for runtime analysis and vice versa: verification of aircraft separation assurance. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis, ISSTA 2015, Baltimore, MD, USA, July 12-17, 2015*, pages 282–292, 2015.
2. A. L. Galdino, C. Muñoz, and M. Ayala-Rincón. Formal verification of an optimal air traffic conflict resolution and recovery algorithm. In *Logic, Language, Information and Computation, 14th International Workshop, WoLLIC 2007, Rio de Janeiro, Brazil, July 2-5, 2007*, pages 177–188, 2007.
3. K. Ghorbal, J. Jeannin, E. Zawadzki, A. Platzer, G. J. Gordon, and P. Capell. Hybrid theorem proving of aerospace systems: Applications and challenges. *J. Aerospace Inf. Sys.*, 11(10):702–713, 2014.
4. D. Giannakopoulou, D. H. Bushnell, J. Schumann, H. Erzberger, and K. Heere. Formal testing for separation assurance. *Ann. Math. Artif. Intell.*, 63(1):5–30, 2011.
5. D. Giannakopoulou, F. Howar, M. Isberner, T. Lauderdale, Z. Rakamaric, and V. Raman. Taming test inputs for separation assurance. In *ACM/IEEE International Conference on Automated Software Engineering, ASE '14, Vasteras, Sweden - September 15 - 19, 2014*, pages 373–384, 2014.
6. J. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, E. Zawadzki, and A. Platzer. A formally verified hybrid system for the next-generation airborne collision avoidance system. In *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015*, pages 21–36, 2015.
7. M. J. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. MIT Press, 2015.
8. M. J. Kochenderfer and J. P. Chryssanthacopoulos. Robust airborne collision avoidance through dynamic programming. Project Report ATC-371, Massachusetts Institute of Technology, Lincoln Laboratory, 2011.
9. J. Kuchar and A. C. Drumm. The traffic alert and collision avoidance system. *Lincoln Laboratory Journal*, 16(2):277, 2007.
10. R. Lee, M. J. Kochenderfer, O. J. Mengshoel, G. P. Brat, and M. P. Owen. Adaptive stress testing of airborne collision avoidance systems. In *Digital Avionics Systems Conference (DASC), 2015 IEEE/AIAA 34th*, pages 6C2–1. IEEE, 2015.
11. S. M. Loos, D. W. Renshaw, and A. Platzer. Formal verification of distributed aircraft controllers. In *Proceedings of the 16th international conference on Hybrid systems: computation and control, HSCC 2013, April 8-11, 2013, Philadelphia, PA, USA*, pages 125–130, 2013.
12. J. Lygeros and N. Lynch. On the formal verification of the TCAS conflict resolution algorithms. In *36th IEEE Conference on Decision and Control*, pages 1829–1834, 1997.
13. A. Platzer and E. M. Clarke. Formal verification of curved flight collision avoidance maneuvers: A case study. In *FM 2009: Formal Methods, Second World Congress, Eindhoven, The Netherlands, November 2-6, 2009*, pages 547–562, 2009.
14. C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, 1998.

15. C. von Essen and D. Giannakopoulou. Analyzing the next generation airborne collision avoidance system. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014*, pages 620–635, 2014.
16. C. von Essen and D. Giannakopoulou. Probabilistic verification and synthesis of the next generation airborne collision avoidance system. *STTT*, 18(2):227–243, 2016.