EPiC Series in Computing

Volume 96, 2023, Pages 1–8

Proceedings of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH23)


# Closed-Loop ACAS Xu Neural Network Verification

Sanaz Sheikhi[1] and Stanley Bak[1]

Stony Brook University, Stony Brook, NY, USA
ssheikhi,sbak@cs.stonybrook.edu

**Abstract**

Benchmark Proposal: Neural Network Control Systems (NNCS) play critical roles in autonomy. However, verifying their correctness is a substantial challenge. In this paper, we consider the neural network compression of ACAS Xu, a popular benchmark usually considered for open-loop neural network verification. ACAS Xu is an air-to-air collision avoidance system for unmanned aircraft issuing horizontal turn advisories to avoid collision with an intruder aircraft. We propose specific properties and different system assumptions to use this system as a closed-loop NNCS benchmark. We present experimental results for our properties based on randomly generated test cases and provide simulation code.

## 1 Introduction

The autonomous and hybrid system community is interested in verifying safety-critical continuous dynamical systems, including NNCS where the controller is a neural network. This paper describes a popular benchmark, the neural network compression of ACAS Xu, an air-to-air collision avoidance system designed for unmanned aircraft that issues horizontal turn advisories to avoid an intruder aircraft [8, 7]. The original system was designed based on dynamic programming and Markov Decision Processes, having large lookup tables of size 2 GB [8]. The large lookup tables were compressed down to 5 MB used to store the weights of the neural network [5].

The neural network version of ACAS Xu [7, 4, 6] is one of the most widely used benchmarks for *open-loop* neural network verification, where properties are proven between the inputs and outputs of a single execution of the neural network. However, the system's safety is associated with the physical dynamics. Therefore, verifying only open-loop properties does not guarantee the system works correctly, which requires *closed-loop* NNCS verification methods.

While the full closed-loop safety of the system under expected operating conditions may be stated, current NNCS analysis tools are unlikely to be able to check the system's safety fully at this point. In earlier work on closed-loop safety [9], only a few hundred feet of position uncertainty around specific test cases was considered. Other works also did not deal with the full possible system uncertainty, and assumptions like fixed velocities are common [2]. Modifying the semantics of the controller using quantization can also be considered and has better scalability for verification [1], but may not be applicable for other NNCS that have more inputs.

This paper proposes a set of closed-loop benchmarks, building up from properties that only involve a small amount of uncertainty to the full verification problem. Furthermore, through random analysis we are able to gauge how close properties are to the boundary of safety, which are most interesting for verification and falsification approaches that may make use of the benchmark.

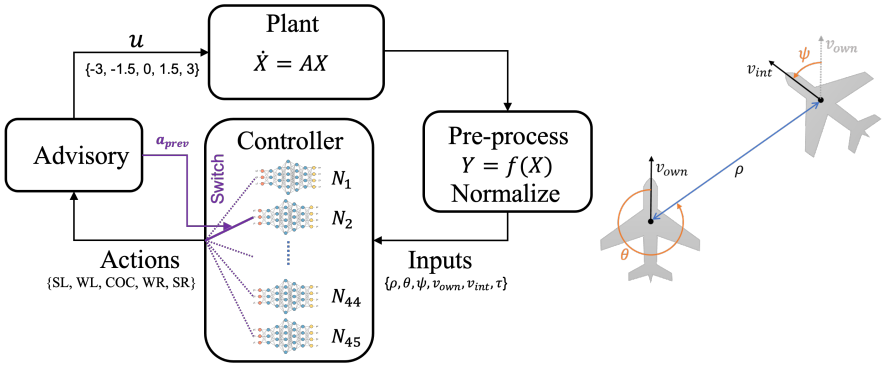


Figure 1: The closed-loop air-to-air collision avoidance system design (image from [1]).

In the following sections, we briefly describe the design and dynamics of closed-loop ACAS Xu and propose specific benchmarks. We also present evaluation results with the proposed benchmarks using random simulations. Finally, we provide simulation code for ACAS Xu with Dubins and F-16 aircraft dynamics in Python.

# 2 ACAS Xu

In the rest of the paper we refer to the neural network compression of the ACAS Xu simply as the ACAS Xu system. We first review the ACAS Xu design, NN controller and briefly discuss two possible plant models: a Dubins aircraft and simulated F-16 aircraft dynamics that can be combined with the decision-making ACAS Xu system.

## 2.1 Overall Design

Fig. 1 illustrates an overview of ACAS Xu design. The system controls the ownship aircraft by issuing advisory commands to avoid collision with an intruder aircraft. The input to the system is the set of the ownship and intruder aircraft states, $\mathcal{I} = \{\rho, \theta, \psi, v_{own}, v_{int}, \tau, a_{prev}\}$, and the output of the system is an advisory command for the ownship from set $\mathcal{A} = \{\text{COC}, \text{WL}, \text{WR}, \text{SL}, \text{SR}\}$. The meaning of these state variables and actions is shown in Tables 1 and 2. The controller in the NNCS runs once per second and produces an output commend. The command is fixed for the next one second and the plant start executing. In the figure, Dubins dynamics is shown which can be modeled as a linear system, $\dot{X} = AX$.

## 2.2 Neural Network Controller

Implementing the initial ACAS Xu lookup state-action tables would require hundreds of gigabytes of floating point storage [5], which may hinder applicability. Using downsampling techniques, the size of the tables were reduced to 2 GB, which may still be too large, especially if dealing with certified avionics hardware that may run on UASs [4]. The motivation to use neural networks was to perform a second compression round. This was done using 45 separate neural networks, each with *six* layers, ReLU activation functions and 50 neurons per layer [5, 7]. Each network is indexed by a pair $\lambda, \beta$ where $\lambda$ is an integer denoting previous advisory $a_{prev} \in \{COC, WL, WR, SL, SR\}$ as presented by Table 2 and $\beta$ is an index representing the value of time to loss of vertical separation $\tau \in \{0, 1, 5, 10, 20, 40, 60, 80, 100\}$ seconds. For instance, $N_{2,3}$ is a neural network where $a_{prev} = WL$ and $\tau = 5$. The inputs to each neural network are the state variables $(\rho, \theta, \psi, v_{own}, v_{int})$ listed in Table 1, and the output of the network is a value of turning advisories listed in Table 2. Note that the network used can switch at

| Inputs | Unit | Description |
|---|---|---|
| $\rho$ | ft | distance between ownship and intruder |
| $\theta$ | rad | angle to intruder w.r.t ownship heading |
| $\psi$ | rad | heading of intruder w.r.t ownship |
| $v_{own}$ | ft/s | ownship velocity |
| $v_{int}$ | ft/s | intruder velocity |
| $\tau$ | s | time until loss of vertical separation |
| $a_{prev}$ | - | previous advisory |

Table 1: Input variables to the neural network used to generate a turn advisory.

| Action | Description |
|---|---|
| $SL$ | strong left turn at 3.0 deg/s |
| $WL$ | weak left turn at 1.5 deg/s |
| $COC$ | clear of conflict (do nothing) |
| $WR$ | weak right turn at 1.5 deg/s |
| $SR$ | strong right turn at 3.0 deg/s |

Table 2: Output of the neural network used as turn advisories

different iterations, since the selection criteria is based on the time to loss of vertical separation and the previous command. This helps prevent chattering at decision boundaries as neural networks can bias the decision towards the previous command. Note that if aircraft are flying at the same altitude, the time to loss of vertical separation is always zero, so the switching is done among five neural networks based solely on the previous command. Otherwise, the switching is among all 45 neural networks.

## 2.3 Aircraft Dynamics

We propose benchmarks with two different plant models, a simple model that uses Dubins aircraft dynamics, and more a complex model that uses simulated F-16 dynamics. In this section, we outline these two models.

**Dubins Aircraft Dynamics:** Directly modeling each aircraft with positions, velocities and headings results in a nonlinear model for each aircraft, as shown in Equation 1.

$$\begin{aligned} \dot{x_1} &= \dot{x} = v\cos(\omega), \\ \dot{x_2} &= \dot{y} = v\sin(\omega), \\ \dot{x_3} &= \dot{\omega} = u \end{aligned} \tag{1}$$

Note that when $\omega$ is fixed, these dynamics can instead be expressed as a linear system with four variables, by replacing the heading angle instead with an $x$ and $y$ velocity variable [1, 9]. This enables efficient linear systems reachability methods to be applicable.

However, with either of these models, the state of the aircraft needs to be converted into the input space of the neural network, which was trained using polar coordinate inputs. The ACAS Xu neural networks, for example, take in the distance ($\rho$), angle to intruder w.r.t ownship heading ($\theta$), and heading of the intruder w.r.t. ownship ($\psi$). The conversion between the Dubins model parameters and the NNCS inputs can be performed using Equation 2. Finally, before running the neural network, each input is scaled by normalizing and offsetting it to a range of [-1, 1]. The networks have five outputs, and the output with the minimum value is taken as the advisory command.

$$\begin{aligned}\rho &= \sqrt{(x_{int} - x_{own})^2 + (y_{int} - y_{own})^2},\\ \theta &= \arctan\left(\frac{y_{int} - y_{own}}{x_{int} - x_{own} + \rho}\right) - \psi_{own}\\ \psi &= \psi_{int} - \psi_{own}\end{aligned} \tag{2}$$

**F-16 aircraft dynamics:** The F-16 aircraft model uses a full six-degrees-of-freedom. The low-level model is based on sample code provided by the well-known Stevens & Lewis Aerospace Engineering textbook [11], with mid-level control given as part of an earlier benchmark [3]. Each aircraft has a 16-dimensional state composed of positional states, positional velocities, rotational states, rotational velocities, engine trust lag term, and integrator states for the low-level controllers. The system is controlled using common aircraft control surfaces (the ailerons, elevators, and rudder) and by configuring the engine trust, continuously evolving with piece-wise nonlinear differential equations. We implement fixed-degree turns by having a target bank angle command found experimentally to achieve the desired steady-state response in terms of degrees per second.

# 3 Benchmarks

The above-described dynamics and the benchmarks described in this section are available online[1]. We provide specific benchmarks to verify and test the closed-loop ACAS Xu system. These benchmarks are designed to cover a wide variety of complexity for NNCS test and verification methods. To define all possible state where the system should operate correctly, we extract the variables ranges used for normalization when training the neural networks [8, 7]. The system is active when the distance between the aircraft $\rho \in [0, 60760]$ ft, otherwise clear-of-conflict is recommended, which we implement as a do-not-turn command for the ownship. The ownship velocity range is $v_{own} \in [100, 1200]$ ft/sec, the intruder velocity range is $v_{int} \in [0, 1200]$ ft/sec, and the angular inputs $\theta$ and $\psi$ are in the range $[-\pi, \pi]$.

## 3.1 Enumerating State Variables

Table 3 lists a set of proposed closed-loop benchmarks for ACAS Xu focusing on the neural network inputs (state variables). Our high-level policy for benchmark generation is sampling one state variable at a time while all other variables hold fixed values to see the effect of each state variable on the neural network output. We iteratively increase the number of sampled state variables to investigate the impact of their combination and increase the complexity of the problem. Initially, we assume a minimum distance of 60760 ft between the aircraft, to ensure the initial state is possible and the neural network will not prevent it at the initial step. Also, we exploit symmetry and position the ownship aircraft at the origin, position $(0, 0)$, and initially flying straight up ($\theta = \frac{\pi}{2}$).

## 3.2 Possible Variants with Intruder Maneuvers

The ownship's control input is determined by the ACAS Xu neural network (turn advisory) at each time step. However, we have freedom of action for choosing the intruder's control inputs. Therefore, we suggest two benchmark-generation policies: 1) the intruder is not allowed to take turn maneuvers and flies a fixed-turn, 2) the intruder can turn and maneuver by taking any of the turn control commands. Clearly, the second case is unsafe if the intruder is faster than the ownship and turns adversarially towards the ownship at each step, but safety is less clear if the speed of the intruder is below the ownship. We do not assume aircraft are changing speed, although with the F-16 model small variations are possible due to the turn commands (we keep the velocity setpoint fixed). Allowing aircraft to change speed (for example, speed can increase or decrease up to 5 ft/sec per second) adds another degree of complexity to the verification problem.

[1] https://github.com/stanleybak/ARCH2023_Benchmark_ACASXU

| Test ID | variables to sample | Description |
|---|---|---|
| T1 | $\rho \in [60760, 60760 + 2400]$ | Sample the intruder's initial position $(x_{int}, y_{int})$ to set $\rho$ |
| T2 | $v_{int} \in [0, 1200]$, $v_{own} \in [100, 1200]$ | Sample the velocity variables $v_{own}$ and $v_{int}$ |
| T3 | $\psi \in [-\pi, \pi]$ | Sample the intruder's heading $(\psi)$ |
| T4 | $\rho \in [60760, 60760 + 2400]$, $\psi \in [-\pi, \pi]$ | Sample the intruder's initial position to set $\rho$ and intruder's heading |
| T5 | $\rho \in [60760, 60760 + 2400]$, $v_{int} \in [0, 1200]$ | Sample the intruder's initial position to set $\rho$ and velocity |
| T6 | $v_{int} \in [0, 1200]$, $v_{own} \in [100, 1200]$, $\psi \in [-\pi, \pi]$ | Sample the intruder's velocity, heading and ownship velocity |
| T7 | $\rho \in [60760, 60760 + 2400]$, $v_{int} \in [0, 1200]$, $v_{own} \in [100, 1200]$ | Sample the intruder's initial position, velocity and ownship velocity |
| T8 | $\rho \in [60760, 60760 + 2400]$, $v_{int} \in [0, 1200]$, $v_{own} \in [100, 1200]$, $\psi \in [-\pi, \pi]$ | Sample all intruder's state variables and ownship velocity |

Table 3: Each row represents a benchmark w.r.t. sampling some state variables. The benchmarks are designed based on these assumptions: at timestep 0, the minimum distance between intruder and ownship is 60760 ft, the ownship is initially at the origin $(0, 0)$ with heading $\theta = \frac{\pi}{2}$, the intruder flies straight (no turns), there is no sensor noise and advisories are followed immediately with no delay. For setting the initial distance between aircraft $(\rho)$, we sample positional state variables of the intruder $(x_{int}, y_{int})$.

## 3.3 Possible Variants with Uncertainty and Delay

Real-world systems always experience a variety of uncertainty, noise, and delay that is not usually reflected by clean simulations. For each of the aforementioned conditions, we can consider variants with observation noise (for example, the position of each aircraft has a disturbance of 100 ft). Another source of uncertainty can be imperfect turns such that weak turns are not exactly 1.5 degrees and strong turns are not exactly 3.0 degrees (for example, allow these to vary in a range of $[1.3, 1.7]$ or $[2.8, 3.2]$ degrees per second). Finally, another set of benchmarks can be generated by assuming an actuation delay between issuing the command and when it is acted upon (for example, each command may be delayed between 0 to 0.25 seconds).

| Benchmark | $\rho$,$ft$ | $v_{int}$ ,$ft/s$ | $\psi$,$rad$ | $v_{own}$,$ft/s$ | Collision (%) | |
|---|---|---|---|---|---|---|
| | | | | | $\tau = 0$ | $\tau > 25$ |
| T1 | * | 1018.5684 | 3.9321 | 110.5440 | 0.2333 | 0.0 |
| T2 | 62052.8753 | * | 1.1109 | * | 0.5963 | 0.0 |
| T3 | 62137.4697 | 695.1384 | * | 109.0707 | 0.2258 | 0.0 |
| T4 | * | 1193.9516 | * | 133.2332 | 0.0687 | 0.0003 |
| T5 | * | * | 4.2875 | 113.3399 | 0.0258 | 0.0005 |
| T6 | 62559.3512 | * | * | * | 0.0022 | 0.0 |
| T7 | * | * | 4.9786 | * | 0.0004 | 0.0 |
| T8 | * | * | * | * | 0.0008 | 0.0 |

Table 4: Experiments based on the benchmarks in Table 3. The first column shows the benchmark ID. Columns 2 to 5 present the state variables (input to the neural network). Each variable has a fixed value or is uniformly sampled from the state space (marked with an asterisk symbol). The last columns show the collision rates (number of collisions per one million simulations) for two different configurations: 1) time until loss of vertical separation ($\tau$) is 0, and the system only uses 5 out of the 45 neural networks 2) $\tau$ is non-zero, and any of the other 40 neural networks may be used.

# 4 Experiments

We conducted experiments for the benchmarks in Table 3 discussed in subsection 3.1 . Our main goal was to find test cases that violate the safety specification. An unsafe state is where the horizontal separation $\rho$ is less than 500 ft and the time to loss of vertical separation $\tau$ is zero seconds [10].

We used ACAS Xu with Dubins dynamics in our experiments with no intruder maneuvers. We ran one million simulations for each benchmark by sampling the associated state variable(s) while keeping all other variables fixed. We sampled the intruder's initial positional state variables $(x_{int}, y_{int})$ from circular areas. The circles are centered at the origin, having random radius $\in [60760, 60760 + 2400]$. All other state variables to be sampled are generated randomly from a uniform distribution. As mentioned before, the ownship is initially positioned at the origin $(0, 0)$ with $\theta = \frac{\pi}{2}$, taking turn advisory from the neural network, and the intruder flies straight ith no turn. Table 4 shows the results where the last two columns present collision rates per million simulations for two different configurations of time until loss of vertical separation $\tau$.

In another set of experiments, we combined benchmarks proposed in Sections 3.1 and 3.2, where we sampled state variables and the intruder applied a constant turn command, not necessarily flying straight. Table 5 shows the results in terms of collision rates for each type of control input (turn) the intruder takes.

# 5 Conclusion

We have presented a set of closed-loop benchmarks for the ACAS Xu neural network control system. We believe the full verification (case T8) is currently out of reach of verification tools, and so have presented a series of intermediate verification goals to gauge the strength of NNCS verification methods. All proposed properties are difficult to falsify using random analysis. We have also discussed additional variants that further complicate verification for this system such as sensor noise or actuation delay.

| | $SL$ Collision (%) | $WL$ Collision (%) | $SR$ Collision (%) | $WR$ Collision (%) | $COC$ Collision (%) |
|---|---|---|---|---|---|
| T1 | 0.3685 | 1.0607 | 0.5135 | 0.0000 | 0.2333 |
| T2 | 0.5203 | 0.1924 | 0.9920 | 0.2210 | 0.5963 |
| T3 | 2.1898 | 0.5347 | 1.8466 | 0.4785 | 0.2258 |
| T4 | 0.2125 | 0.4414 | 0.1725 | 0.2794 | 0.0687 |
| T5 | 0.2597 | 0.1038 | 0.1357 | 0.1404 | 0.0258 |
| T6 | 0.1680 | 0.1067 | 0.1333 | 0.1064 | 0.0022 |
| T7 | 0.0766 | 0.0940 | 0.00009 | 0.0764 | 0.0004 |
| T8 | 0.0411 | 0.0583 | 0.0450 | 0.0599 | 0.0008 |

Table 5: Experiments according to the Table 3 benchmarks where intruder aircraft can make turns and perform maneuvers. Each column shows the collision rate (number of collisions per one million simulations) where the intruder is controlled by the associated control input (turn advisory) from $\{COC, WL, WR, SL, SR\}$. The ownship is initially positioned at origin $(0, 0)$, with $\theta = \frac{\pi}{2}$. The time to lose vertical separation is $\tau = 0$.

## Acknowledgement

This material is based upon work supported by the Air Force Office of Scientific Research and the Office of Naval Research under award numbers FA9550-19-1-0288, FA9550-21-1-0121, FA9550-23-1-0066 and N00014-22-1-2156. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force or the United States Navy.

## References

[1] Stanley Bak and Hoang-Dung Tran. Neural network compression of acas xu early prototype is unsafe: Closed-loop verification through quantized state backreachability. In *NASA Formal Methods Symposium*, 2022.

[2] Arthur Clavière, Eric Asselin, Christophe Garion, and Claire Pagetti. Safety verification of neural network controlled systems. In *51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 47–54. IEEE, 2021.

[3] Kerianne Hobbs, Peter Heidlauf, Alexander Collins, and Stanley Bak. Verification challenges in f-16 ground collision avoidance and other automated maneuvers. In *5th International Workshop on Applied Verification of Continuous and Hybrid Systems*. EasyChair, 2018.

[4] Kyle D Julian and Mykel J Kochenderfer. Guaranteeing safety for neural network-based aircraft collision avoidance systems. In *IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2019.

[5] Kyle D Julian, Jessica Lopez, Jeffrey S Brush, Michael P Owen, and Mykel J Kochenderfer. Policy compression for aircraft collision avoidance systems. In *IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2016.

[6] Kyle D. Julian, Shivam Sharma, Jean-Baptiste Jeannin, and Mykel J. Kochenderfer. Verifying aircraft collision avoidance neural networks through linear approximations of safe regions, 2019.

[7] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[8] Mykel J Kochenderfer and JP Chryssanthacopoulos. Robust airborne collision avoidance through dynamic programming. *MIT, Lincoln Laboratory, Project Report ATC-371*, 130, 2011.

[9] Diego Manzanas Lopez, Taylor T Johnson, Stanley Bak, Hoang-Dung Tran, and Kerianne L Hobbs. Evaluation of neural network verification methods for air-to-air collision avoidance. *Journal of Air Transportation*, pages 1–17, 2022.

[10] Mike Marston and Gabe Baca. ACAS-Xu initial self-separation flight tests. http://hdl.handle.net/2060/20150008347, 2015.

[11] Brian L Stevens, Frank L Lewis, and Eric N Johnson. *Aircraft control and simulation*. John Wiley & Sons, 2015.