EPiC Series in Computing
Volume 96, 2023, Pages 89–125
Proceedings of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH23)

ARCH-COMP23 Category Report: Artificial Intelligence and Neural Network Control Systems (AINNCS) for Continuous and Hybrid Systems Plants

Diego Manzanas Lopez[1], Matthias Althoff[2], Marcelo Forets[3], Taylor T. Johnson[1], Tobias Ladner[2], Christian Schilling[4]

[1] Vanderbilt University
Nashville, TN
`{diego.manzanas.lopez, taylor.johnson}@vanderbilt.edu`
[2] Technische Universität München (TUM), Munich, Germany
`{althoff, tobias.ladner}@tum.de`
[3] Universidad de la República, Montevideo, Uruguay
`mforets@gmail.com`
[4] Aalborg University, Aalborg, Denmark
`christianms@cs.aau.dk`

Abstract

This report presents the results of a friendly competition for formal verification of continuous and hybrid systems with artificial intelligence (AI) components. Specifically, machine learning (ML) components in cyber-physical systems (CPS), such as feedforward neural networks used as feedback controllers in closed-loop systems, are considered, which is a class of systems classically known as intelligent control systems, or in more modern and specific terms, neural network control systems (NNCS). We broadly refer to this category as AI and NNCS (AINNCS). The friendly competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in 2023. In the fifth edition of this AINNCS category at ARCH-COMP, three tools have been applied to solve ten different benchmark problems, which are CORA, JuliaReach and NNV. In reusing the benchmarks from the last iteration, we demonstrate the continuous progress in developing these tools: Two out of three tools can verify more instances than in the 2022 iteration. A novelty of this year's iteration is the shared computation hardware that allows for a fairer comparison among the participants.

1 Introduction

Neural Networks (NNs) have demonstrated an impressive ability to solve complex problems in numerous application domains [67]. The success of these models in contexts such as adaptive control, non-linear system identification [48], image and pattern recognition, function approximation, and machine translation has stimulated the creation of technologies that are directly impacting our everyday lives [59], and has led researchers to believe that these models possess the power to revolutionize a diverse set of arenas [54].

Despite these achievements, there have been reservations about utilizing them within high-assurance systems for various reasons, such as their susceptibility to unexpected and errant behavior caused by slight perturbations in their inputs [37]. In a study by Szegedy et al. [60], the authors demonstrated that carefully applying a hardly perceptible modification to an input image could cause a successfully trained neural network to produce an incorrect classification. These inputs are known as *adversarial examples*, and their discovery has caused concern over the safety, reliability, and security of neural network applications [67]. As a result, some research has been directed toward obtaining an explicit understanding of neural network behavior.

Neural networks are often viewed as "black boxes" whose underlying operation is often incomprehensible. Still, the last several years have witnessed numerous promising white-box verification methods proposed for reasoning about the correctness of their behavior. However, it has been demonstrated that neural network verification is an NP-complete problem [33]. Despite many recent efforts and significant advances in the past decade [47, 64, 65, 66, 7, 22, 23, 34, 39, 58], there are remaining challenges that prevent these approaches from being successfully applied to very large neural networks used in many real-world applications such as [53]. Most of this work also focuses on verifying pre-/post-conditions for neural networks in isolation. Reasoning about their usage behavior in cyber-physical systems, such as in neural network control systems, remains a key challenge.

The following report aims to provide a survey of the landscape of the current capabilities of verification tools for *closed-loop systems with neural network controllers*, as these systems have displayed great utility as a means for learning control laws through techniques such as reinforcement learning and data-driven predictive control [18, 62]. Furthermore, this report aims to provide readers with a perspective on the intellectual progression of this rapidly growing field and stimulate the development of efficient and effective methods capable of use in real-life applications. Since the first iteration, there have been several publications investigating the formal verification of AINNCS, out of which several of them have participated in one or more of the previous competitions such as Verisig [28], VenMas [1] and ReachNN [21], among others [66, 15, 25, 2, 36, 20, 44, 18, 16, 27, 57].

Disclaimer The presented report of the ARCH-COMP friendly competition for *closed-loop systems with neural network controllers*, termed in short AINNCS (Artificial Intelligence / Neural Network Control Systems), aims to provide the landscape of the current capabilities of verification tools for analyzing these systems that are classically known as *intelligent control systems*. This AINNCS ARCH-COMP category is complementary to the ongoing Verification of Neural Networks Competition (VNN-COMP) [9, 47], the latter of which focuses on open-loop specifications of neural networks. In contrast, the AINNCS category focuses on closed-loop behaviors of dynamical systems incorporating neural networks. We want to stress that each tool has unique strengths and not all of the specificities can be highlighted within a single report. To reach a consensus on what benchmarks are used, some compromises had to be made so that some tools may benefit more from the presented choice than others. To establish further trustworthiness of the results, the code with which the results have been obtained is publicly available at gitlab.com/goranf/ARCH-COMP, and the submitted results are available at arch.repeatability.cps.cit.tum.de/frontend/submissions.

Specifically, this report summarizes results obtained in the 2023 friendly competition of the ARCH workshop[1] for verifying systems of the form

$$\dot{x}(t) = f(x(t), u(x,t)),$$

where $x(t)$ and $u(x,t)$ correspond to the states and inputs of the plant at time t, respectively, and where $u(x,t)$ is the output of a feedforward neural network provided an input of the plant state x at time t. The architecture of the closed-loop systems we consider is depicted in Figure 1, where the input to the neural network controller is additionally sampled.

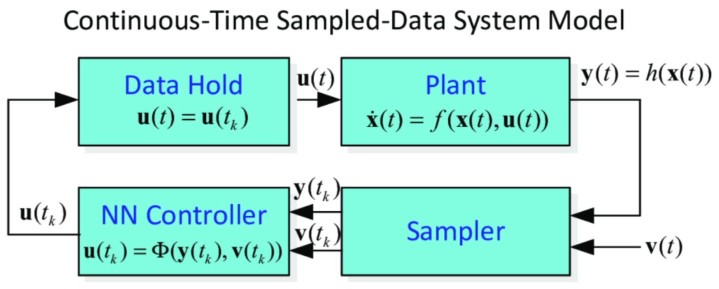

Continuous-Time Sampled-Data System Model
Data Hold
$\mathbf{u}(t) = \mathbf{u}(t_k)$
$\mathbf{u}(t)$
Plant
$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$
$\mathbf{y}(t) = h(\mathbf{x}(t))$
$\mathbf{u}(t_k)$
NN Controller
$\mathbf{u}(t_k) = \Phi(\mathbf{y}(t_k), \mathbf{v}(t_k))$
$\mathbf{y}(t_k)$
$\mathbf{v}(t_k)$
Sampler
$\mathbf{v}(t)$

Figure 1: Closed-loop architecture of the benchmarks to be verified.

This year is the fifth iteration of the AINNCS category at ARCH-COMP and builds on the previous iterations and reports [42, 31, 30, 40]. Participating tools are summarized in Sec. 2. Please, see [67] for further details on these and additional tools. The results of our selected benchmark problems are shown in Sec. 3. This year, we run all tools on the same hardware using docker images for further comparison. The submission server specifications are given in Appendix A. However, one has to factor in the efficiency of the programming language of the tools.

The goal of the friendly competition is not to rank the results but rather to present the landscape of existing solutions in a breadth that is impossible with scientific publications in classical venues. Such publications would typically require the presentation of novel techniques, while this report showcases the current state-of-the-art tools. The selection of the benchmarks has been conducted within the forum of the ARCH website (cps-vo.org/group/ARCH), which is visible for registered users, and registration is open for anyone.

2 Participating Tools

We present a brief overview of all the participating tools in this friendly competition. The tools are CORA, JuliaReach, and NNV. The tools participating in the *Artificial Intelligence / Neural Network Control Systems in Continuous and Hybrid Systems Plants* (AINNCS) category are introduced subsequently in alphabetical order.

[1]Workshop on Applied Verification for Continuous and Hybrid Systems (ARCH), cps-vo.org/group/ARCH

CORA *(Tobias Ladner, Matthias Althoff).* CORA [2] is a COntinuous Reachability Analyzer for the formal verification of cyber-physical systems using reachability analysis. It is written in MATLAB and is available at `https://cora.in.tum.de`. CORA integrates various set representations and operations on them as well as reachability algorithms of various dynamic system classes. For this competition, we used the approach described in [36, 38] for open-loop and closed-loop neural network verification based on polynomial zonotopes [35]. Polynomial zonotopes are particularly well suited for verifying neural networks due to their polynomial time complexity on many operations. The approach described in [36] realizes a fast layer-based computation of an over-approximation of the output set of networks with various activation functions, including ReLU, sigmoid, and tanh, while [38] extends this approach by automatically refining the neuron abstractions in neural networks. Our neural network verification approaches are naturally integrated into our reachability analysis methods for linear and nonlinear plant dynamics. For most benchmarks, we can deploy a fully automatic verification process using CORA: We first simulate random runs to find potential violations of the specifications. If one run violates the specifications, we verify the violating run using reachability analysis, as simulations are not formally sound. Otherwise, we try to verify the benchmark for the given specifications. We will provide the computation times for each step.

JuliaReach *(Marcelo Forets, Christian Schilling).* JuliaReach [15] is an open-source software suite for reachability computations of dynamical systems, written in the Julia language and available at `http://github.com/JuliaReach`. The package ClosedLoopReachability.jl handles the closed-loop analysis and queries sub-problems to our other library ReachabilityAnalysis.jl for continuous-time analysis of plant models. Additional set computations are performed with LazySets.jl [24]. The algorithm we use is described in [55]. For the plant analysis, we use the sound algorithm `TMJets` based on interval arithmetic and Taylor models, which is implemented in TaylorModels.jl [11, 14], which itself integrates TaylorSeries.jl [12, 13] and TaylorIntegration.jl [49]. The algorithm uses a jet transportation of a Taylor polynomial with interval coefficients. It has the following main parameters for tweaking: the absolute tolerance `abstol` and two parameters to define the order at which the Taylor expansion is cut in time (`orderT`) resp. in space (`orderQ`). For neural-network analysis, we use an abstract interpretation based on zonotopes [58]. For falsification, JuliaReach chooses an initial point but uses set-based analysis since, although most models are deterministic, non-validated simulations may yield wrong results.

NNV *(Diego Manzanas Lopez, Taylor Johnson).* The Neural Network Verification (NNV) Tool [66, 41] is a formal verification software tool for deep learning models and cyber-physical systems with neural network components written in MATLAB and available at `https://github.com/verivital/nnv`. NNV uses a star-set state-space representation and reachability algorithm that allows for a layer-by-layer computation of exact or overapproximate reachable sets for feed-forward [64], convolutional [61], semantic segmentation (SSNN) [65], and recurrent (RNN)[63] neural networks, as well as neural network control systems (NNCS) [62, 66] and neural ordinary differential equations (Neural ODEs) [45]. The star-set based algorithm is naturally parallelizable, which allows NNV to be designed to perform efficiently on multi-core platforms. Additionally, if a particular safety property is violated, NNV can be used to construct and visualize the complete set of counterexample inputs for a neural network (exact-analysis). Using NNV in combination with HyST [8, 6] and CORA [2, 3, 4] allows for the verification of closed-loop neural network control systems with nonlinear plant dynamics.

3 Benchmarks

We have selected ten benchmarks for this year's competition – the same benchmarks and specifications as last year [40]. A few of them, such as the TORA benchmark, are presented with several different controllers to be analyzed. We now describe these benchmarks in no particular order and have made them readily available online.[2] All benchmarks are derived for continuous time. Given the continuous dynamics $\dot{x} = f(x)$, where $x \in \mathbb{R}^n$ is the state vector, the discrete-time versions for a time increment of Δt are obtained in this competition using forward Euler integration:

$$x(k+1) = x(k) + f(x)\Delta t.$$

3.1 Adaptive Cruise Controller (ACC)

The Adaptive Cruise Control (ACC) benchmark is a system that tracks a set velocity and maintains a safe distance from a lead vehicle by adjusting the longitudinal acceleration of an ego vehicle [46]. The neural network computes optimal control actions while satisfying safe distance, velocity, and acceleration constraints using model predictive control (MPC) [51]. For this case study, the ego car is set to travel at a set speed $v_{set} = 30$ and maintains a safe distance D_{safe} from the lead car. The car's dynamics are described by the following equations [62, p. 17]:

$$\begin{aligned} \dot{x}_{\text{lead}}(t) = v_{\text{lead}}(t), \dot{v}_{\text{lead}}(t) = a_{\text{lead}}(t), \dot{a}_{\text{lead}}(t) = -2a_{\text{lead}}(t) + 2a_{c,\text{lead}} - uv_{\text{lead}}^2(t), \\ \dot{x}_{\text{ego}}(t) = v_{\text{ego}}(t), \dot{v}_{\text{ego}}(t) = a_{\text{ego}}(t), \dot{a}_{\text{ego}}(t) = -2a_{\text{ego}}(t) + 2a_{c,\text{ego}} - uv_{\text{ego}}^2(t), \end{aligned} \tag{1}$$

where x_i is the position, v_i is the velocity, a_i is the acceleration of the car, $a_{c,i}$ is the acceleration control input applied to the car, and $u = 0.0001$ is a coefficient for air drag, where $i \in \{\text{ego}, \text{lead}\}$. We evaluate a neural network controller with five layers and 20 neurons each for this benchmark. The inputs of the controller are the set speed v_{set}, the desired time gap T_{gap}, the ego velocity v_{ego}, the distance $D_{\text{rel}} = x_{\text{lead}} - x_{\text{ego}}$, as well as the relative velocity v_{rel}, and the output is $a_{\text{c,ego}}$.

Specifications The verification objective of this system is that given a scenario where both cars are driving safely, the lead car suddenly slows down with $a_{c,\text{lead}}$ = -2. We want to check whether there is a collision in the following 5 s. Formally, this safety specification of the system can be expressed as $D_{\text{rel}} \geq D_{\text{safe}}$, where $D_{\text{safe}} = D_{\text{default}} + T_{\text{gap}} \cdot v_{\text{ego}}$, and T_{gap} = 1.4 s and D_{default} = 10. The initial conditions are: $x_{\text{lead}}(0) \in$ [90,110], $v_{\text{lead}}(0) \in$ [32,32.2], $a_{\text{lead}}(0) = a_{\text{ego}}(0) = 0$, $v_{\text{ego}}(0) \in$ [30, 30.2], $x_{\text{ego}} \in$ [10,11]. A control period of 0.1 s is used.

3.2 Sherlock-Benchmark-9 (TORA)

This benchmark considers translational oscillations by a rotational actuator (TORA) [18, 29], where a cart is attached to a wall with a spring and is free to move on a friction-less surface. The cart has a weight attached to an arm inside it, which is free to rotate about an axis. This serves as the control input to stabilize the cart at $\mathbf{x} = \mathbf{0}$. The model is a four-dimensional system, given by the following equations [29, eq. (4)]:

$$\dot{x_1} = x_2, \quad \dot{x_2} = -x_1 + 0.1\sin(x_3), \quad \dot{x_3} = x_4, \quad \dot{x_4} = u. \tag{2}$$

[2]https://github.com/verivital/ARCH-COMP2023

This benchmark has three neural network controllers: the first has three ReLU hidden layers and a linear output layer. This controller was trained using a data-driven model predictive controller proposed in [19]. Note that the output of the neural network $f(\mathbf{x})$ needs to be normalized to obtain u, namely $u = f(\mathbf{x}) - 10$. The sampling time for this controller is 1 s, and we verify it against specification 1 below. The other two controllers have three hidden layers of 20 neurons each and one output layer. In contrast to the first controller, we use sigmoid activation functions for the hidden layers and a tanh output layer. The sampling time of these controllers is 0.5 s, the output of the neural network $f(\mathbf{x})$ needs to be post-processed as $u = 11 \cdot f(\mathbf{x})$, and we verify them against specification 2 below.

Specification 1. This is a safety specification. For an initial set of $x_1 \in [0.6, 0.7]$, $x_2 \in [-0.7, -0.6]$, $x_3 \in [-0.4, -0.3]$, and $x_4 \in [0.5, 0.6]$, the system states have to stay within the box $\mathbf{x} \in [-2, 2]^4$ for a time window of 20 s.

Specification 2. For an initial set of $x_1 \in$ [-0.77, -0.75], $x_2 \in$ [-0.45, -0.43], $x_3 \in$ [0.51, 0.54], and $x_4 \in [-0.3, -0.28]$, it is required that the system reaches the set $x_1 \in [-0.1, 0.2]$, $x_2 \in [-0.9, -0.6]$ within a time window of 5 s.

3.3 Sherlock-Benchmark-10 (Unicycle Car Model)

This benchmark considers a unicycle model of a car [18] with the x and y coordinates on a two-dimensional plane, the velocity magnitude (speed), and steering angle as state variables. The dynamic equations are (see [5, Sec. III.B]; a different input is used here):

$$\dot{x}_1 = x_4 \cos(x_3), \quad \dot{x}_2 = x_4 \sin(x_3), \quad \dot{x}_3 = u_2, \quad \dot{x}_4 = u_1 + w, \tag{3}$$

where w is a bounded error in the range $10^{-4}[-1, 1]$. A neural network controller was trained for this system using a model predictive controller as a “demonstrator” or “teacher”. The trained network has one hidden layer with 500 neurons. Note that the output of the neural network $f(\mathbf{x})$ needs to be normalized in order to obtain (u_1, u_2), namely $u_i = f(\mathbf{x})_i - 20$. The sampling time for this controller is 0.2 s.

Specification This is a reachability specification. For an initial set of $x_1 \in [9.5, 9.55]$, $x_2 \in [-4.5, -4.45]$, $x_3 \in [2.1, 2.11]$, and $x_4 \in [1.5, 1.51]$, the system has to reach the set $x_1 \in [-0.6, 0.6], x_2 \in [-0.2, 0.2], x_3 \in [-0.06, 0.06], x_4 \in [-0.3, 0.3]$ within a time window of 10 s.

3.4 VCAS Benchmark

This benchmark is a closed-loop variant of the *aircraft collision avoidance system ACAS X*. The scenario involves two aircraft, the ownship and the intruder, where the ownship is equipped with a collision avoidance system referred to as VerticalCAS [32]. VerticalCAS issues vertical climb rate advisories every second to the ownship pilot to avoid a near mid-air collision (NMAC). Near mid-air collisions are regions where the ownship and the intruder are separated by less than 100ft vertically and 500ft horizontally. The ownship (black) is assumed to have a constant horizontal speed, and the intruder (red) is assumed to follow a constant horizontal trajectory towards ownship, see Figure 2. The current geometry of the system is described by

- h, intruder altitude relative to ownship,
- $\dot{h}_0$, ownship vertical climb rate, and

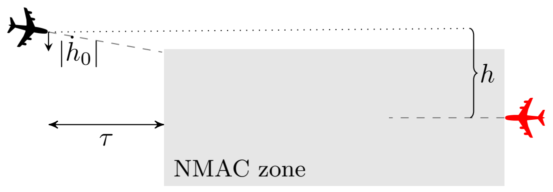


Figure 2: VerticalCAS encounter geometry

- τ, the time until the ownship (black) and intruder (red) are no longer horizontally separated.

We can, therefore, assume that the intruder is static and the horizontal separation τ decreases by one each second. There are nine advisories and each of them instructs the pilot to accelerate until the vertical climb rate of the ownship complies with the advisory:

1. COC: Clear Of Conflict;
2. DNC: Do Not Climb;
3. DND: Do Not Descend;
4. DES1500: Descend at least 1500 ft/s;
5. CL1500: Climb at least 1500 ft/s;
6. SDES1500: Strengthen Descent to at least 1500 ft/s;
7. SCL1500: Strengthen Climb to at least 1500 ft/s;
8. SDES2500: Strengthen Descent to at least 2500 ft/s;
9. SCL2500: Strengthen Climb to at least 2500 ft/s.

In addition to the parameters describing the geometry of the encounter, the current state of the system stores the advisory $\mathrm{adv} \in \{1, \ldots, 9\}$ (numbers correspond to the above list) issued to the ownship at the previous time step. VerticalCAS is implemented as nine ReLU networks N_i, one for each (previous) advisory, with three inputs $(h, \dot{h}_0, \tau)$, five fully-connected hidden layers of 20 units each, and nine outputs representing the score of each possible advisory. Therefore, given a current state $(h, \dot{h}_0, \tau, \mathrm{adv})$, the new advisory adv' is obtained by computing the argmax of the output of N_{adv} on $(h, \dot{h}_0, \tau)$.

Given the new advisory, the pilot can choose acceleration $\ddot{h}_0$ as follows. If the new advisory is COC, then it can be any acceleration from the set $\{-\frac{g}{8}, 0, \frac{g}{8}\}$. For all remaining advisories, if the previous advisory coincides with the new one and the current climb rate complies with the new advisory (e.g., $\dot{h}_0$ is non-positive for DNC and $\dot{h}_0 \geq 1500$ for CL1500), the acceleration $\ddot{h}_0$ is 0; otherwise, the pilot can choose any acceleration $\ddot{h}_0$ from the given sets:

- DNC: $\{-\frac{g}{3}, -\frac{7g}{24}, -\frac{g}{4}\}$;
- DND: $\{\frac{g}{4}, \frac{7g}{24}, \frac{g}{3}\}$;
- DES1500: $\{-\frac{g}{3}, -\frac{7g}{24}, -\frac{g}{4}\}$;
- CL1500: $\{\frac{g}{4}, \frac{7g}{24}, \frac{g}{3}\}$;

- SDES1500: $\{-\frac{g}{3}\}$;
- SCL1500: $\{\frac{g}{3}\}$;
- SDES2500: $\{-\frac{g}{3}\}$;
- SCL2500: $\{\frac{g}{3}\}$,

where g represents the gravitational constant 32.2 ft/s^2.

It was proposed to tweak the benchmark for the tools that cannot efficiently account for all possible acceleration choices. Those tools can consider two strategies for picking a single acceleration at each time step:

- a worst-case scenario selection, where we choose the acceleration to take the ownship closer to the intruder.
- always select the acceleration in the middle.

Given the current system state $(h, \dot{h}_0, \tau, \mathrm{adv})$, the new advisory adv' and the acceleration $\ddot{h}_0$, the new state of the system can be computed by the following equations [32, eq. (15)]:

$$\begin{aligned} h(k+1) &= h(k) - \dot{h}_0(k)\Delta\tau - 0.5\ddot{h}_0(k)\Delta\tau^2 \\ \dot{h}_0(k+1) &= \dot{h}_0(k) + \ddot{h}_0(k)\Delta\tau \\ \tau(k+1) &= \tau(k) + \Delta\tau \\ \mathrm{adv}(k+1) &= \mathrm{adv}' \end{aligned}$$

where $\Delta\tau = 1$.

Specification The ownship has to be outside of the NMAC zone after $k \in \{1, \dots, 10\}$ time steps, i.e., $h(k) > 100$ or $h(k) < -100$, for all possible choices of acceleration by the pilot. The set of initial states considered is: $h(0) \in [-133, -129]$, $\dot{h}_0(0) \in \{-19.5, -22.5, -25.5, -28.5\}$, $\tau(0) = 25$ and $\mathrm{adv}(0) = \mathrm{COC}$.

3.5 Single-Pendulum Benchmark

We consider a classical inverted pendulum. A ball of mass m is attached to a massless beam of length L. The beam is actuated with a torque T and we assume viscous friction with a friction coefficient of c. The governing equation of motion can be obtained as [43, eq. (1)]:

$$\ddot{\theta} = \frac{g}{L}\sin\theta + \frac{1}{mL^2}\left(T - c\,\dot{\theta}\right), \tag{4}$$

where θ is the angle of the link concerning the upward vertical axis and $\dot{\theta}$ is the angular velocity. After defining the state variables $x_1 = \theta$ and $x_2 = \dot{\theta}$, the dynamics in state-space form is

$$\dot{x}_1 = x_2, \tag{5a}$$

$$\dot{x}_2 = \frac{g}{L}\sin x_1 + \frac{1}{mL^2}\left(T - c\,x_2\right). \tag{5b}$$

Controllers are trained using *behavior cloning*, a supervised learning approach for training controllers. The code, as well as training procedures, are provided. The model parameters are chosen as

$$m = 0.5, L = 0.5, c = 0, g = 1, \tag{6}$$

and the time step for the controller and the discrete-time model is $\Delta t = 0.05$. The initial set is

$$x \in [1.0, 1.2] \times [0.0, 0.2].$$

Specification $\forall t \in [0.5, 1] : \theta \in [0, 1]$ (analogously for $k \in [10, 20]$ in discrete time).

3.6 Double-Pendulum Benchmark

The double pendulum is an inverted two-link pendulum with equal point masses m at the end of connected mass-less links of length L. The links are actuated with torques T_1 and T_2, and we assume viscous friction exists with a coefficient of c. The governing equations of motion are described by the following equations [43, eq. (3a-b)]:

$$2\ddot{\theta}_1 + \ddot{\theta}_2 \cos(\theta_2 - \theta_1) - \dot{\theta}_2^2 \sin(\theta_2 - \theta_1) - 2\frac{g}{L}\sin\theta_1 + \frac{c}{mL^2}\dot{\theta}_1 = \frac{1}{mL^2}T_1, \tag{7a}$$

$$\ddot{\theta}_1 \cos(\theta_2 - \theta_1) + \ddot{\theta}_2 + \dot{\theta}_1^2 \sin(\theta_2 - \theta_1) - \frac{g}{L}\sin\theta_2 + \frac{c}{mL^2}\dot{\theta}_2 = \frac{1}{mL^2}T_2, \tag{7b}$$

where θ_1 and θ_2 are the angles of the links concerning the upward vertical axis (see Figure 3) and g is the gravitational acceleration. After defining the state vector as $x = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^T$, the dynamics in state-space form is

$$\dot{x}_1 = x_3, \tag{8a}$$

$$\dot{x}_2 = x_4, \tag{8b}$$

$$\dot{x}_3 = \frac{1}{2\left(\frac{\cos^2(x_1 - x_2)}{2} - 1\right)} \cos(x_1 - x_2)\left(x_3{}^2 \sin(x_1 - x_2) - \cos(x_1 - x_2)\left(\frac{g\sin(x_1)}{L}\right.\right. \tag{8c}$$

$$\left.\left.- \frac{x_4{}^2 \sin(x_1 - x_2)}{2} + \frac{T_1 - c\,x_3}{2\,L^2\,m}\right) + \frac{g\sin(x_2)}{L} + \frac{T_2 - c\,x_4}{L^2\,m}\right) \tag{8d}$$

$$- \frac{x_4{}^2 \sin(x_1 - x_2)}{2} + \frac{g\sin(x_1)}{L} + \frac{T_1 - c\,x_3}{2\,L^2\,m}, \tag{8e}$$

$$\dot{x}_4 = \frac{-1}{\frac{\cos^2(x_1 - x_2)}{2} - 1}\left(x_3{}^2 \sin(x_1 - x_2) - \cos(x_1 - x_2)\left(\frac{g\sin(x_1)}{L} - \frac{x_4{}^2 \sin(x_1 - x_2)}{2}\right.\right. \tag{8f}$$

$$\left.\left.+ \frac{T_1 - c\,x_3}{2\,L^2\,m}\right) + \frac{g\sin(x_2)}{L} + \frac{T_2 - c\,x_4}{L^2\,m}\right). \tag{8g}$$

The controllers for the double pendulum benchmark are obtained using the same methods as the controllers for the single pendulum benchmark; also here, the code as well as training procedures are provided. The model parameters are chosen as in (6) The initial set is

$$x \in [1.0, 1.3]^4.$$

Specification 1 $\forall t \in [0, 1] : x \in [-1.0, 1.7]^4$ (analogously for $k \in [0, 20]$ in discrete time) for $\Delta t = 0.05$.

Specification 2 $\forall t \in [0, 0.4] : x \in [-0.5, 1.5]^4$ (analogously for $k \in [0, 20]$ in discrete time) for $\Delta t = 0.02$.

We verify *controller double pendulum less robust* against specification 1 and *controller double pendulum more robust* against specification 2.

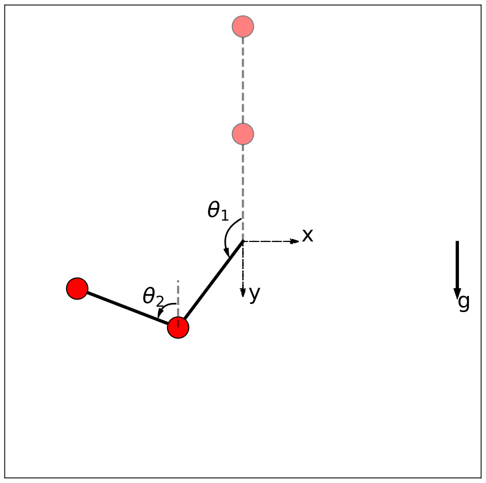


Figure 3: Inverted double pendulum. The goal is to keep the pendulum upright (dashed schematics)

3.7 Airplane Benchmark

The airplane example consists of a dynamical system that is a simple model of a flying airplane as shown in Figure 4. The state is

$$x = [s_x, s_y, s_z, v_x, v_y, v_z, \phi, \theta, \psi, r, p, q]^T, \tag{9}$$

where (s_x, s_y, s_z) is the position of the center of gravity, (v_x, v_y, v_z) are the components of velocity in (x, y, z) directions, (p, q, r) are body rotation rates, and (ϕ, θ, ψ) are the Euler angles. The equations of motion are reduced to [43, eq. (7)]:

$$\dot{v}_x = -g \sin\theta + \frac{F_x}{m} - qv_z + rv_y, \tag{10a}$$

$$\dot{v}_y = g \cos\theta \sin\phi + \frac{F_y}{m} - rv_x + pv_z, \tag{10b}$$

$$\dot{v}_z = g \cos\theta \cos\phi + \frac{F_z}{m} - pv_y + qv_x, \tag{10c}$$

$$I_x\dot{p} + I_{xz}\dot{r} = M_x - (I_z - I_y)qr - I_{xz}pq, \tag{10d}$$

$$I_y\dot{q} = M_y - I_{xz}\left(r^2 - p^2\right) - (I_x - I_z)pr, \tag{10e}$$

$$I_{xz}\dot{p} + I_z\dot{r} = M_z - (I_y - I_x)qp - I_{xz}rq. \tag{10f}$$

The mass of the airplane is denoted with m and I_x, I_y, I_z and I_{xz} are the moment of inertia with respect to the indicated axis; see Figure 4. The control parameters include three force components F_x, F_y and F_z and three moment components M_x, M_y, M_z. Note that for simplicity, we assume that the aerodynamic forces are absorbed in the force vector F. In addition to these six equations, we have six additional kinematic equations [43, eq. (8,9)]:

$$\begin{bmatrix} \dot{s}_x \\ \dot{s}_y \\ \dot{s}_z \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{11}$$

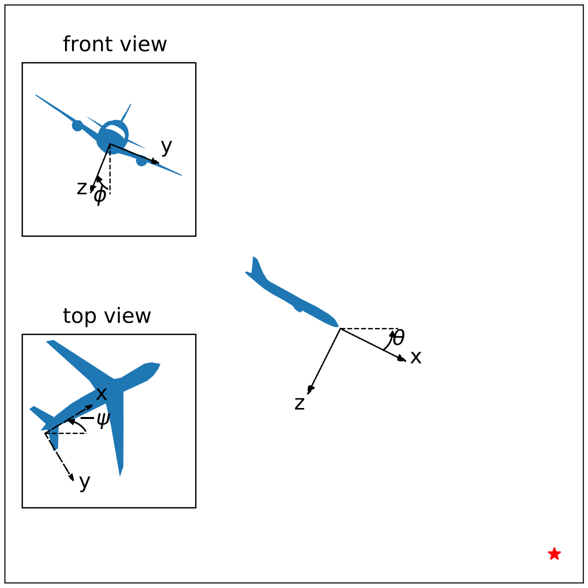


Figure 4: The airplane example.

and

$$\begin{bmatrix}\phi\\\theta\\\psi\end{bmatrix} = \begin{bmatrix}1 & \tan\theta\sin\phi & \tan\theta\cos\phi\\0 & \cos\phi & -\sin\phi\\0 & \sec\theta\sin\phi & \sec\theta\cos\phi\end{bmatrix}\begin{bmatrix}p\\q\\r\end{bmatrix}. \tag{12}$$

As in the pendulum benchmarks, controllers are trained for the airplane problem using behavior cloning. The system involves the model parameters

$$m = 1, I_x = I_y = I_z = 1, I_{xz} = 0, g = 1,$$

and the time step for the controller and the discrete-time model is $\Delta t = 0.1$. The initial set is

$$x = y = z = r = p = q = 0, \quad [v_x, v_y, v_z, \phi, \theta, \psi] \in [0.0, 1.0]^6.$$

Specification $\forall t \in [0, 2] : s_y \in [-0.5, 0.5]$, $[\phi, \theta, \psi] \in [-1.0, 1.0]^3$. Analogously for $k \in [0, 20]$ in discrete time.

We verify the airplane controller against the specification above.

3.8 Benchmark: Attitude Control

We consider the attitude control of a rigid body with six states and three inputs [50, 56]. The system dynamics is given by [50, Sec. V]:

$$
\begin{aligned}
&\dot\omega_1 = 0.25(u_0 + \omega_2\omega_3), \qquad \dot\omega_2 = 0.5(u_1 - 3\omega_1\omega_3), \qquad \dot\omega_3 = u_2 + 2\omega_1\omega_2,\\
&\dot\psi_1 = 0.5\left(\omega_2(\psi_1^2+\psi_2^2+\psi_3^2-\psi_3)+\omega_3(\psi_1^2+\psi_2^2+\psi_2+\psi_3^2)+\omega_1(\psi_1^2+\psi_2^2+\psi_3^2+1)\right),\\
&\dot\psi_2 = 0.5\left(\omega_1(\psi_1^2+\psi_2^2+\psi_3^2+\psi_3)+\omega_3(\psi_1^2-\psi_1+\psi_2^2+\psi_3^2)+\omega_2(\psi_1^2+\psi_2^2+\psi_3^2+1)\right),\\
&\dot\psi_3 = 0.5\left(\omega_1(\psi_1^2+\psi_2^2-\psi_2+\psi_3^2)+\omega_2(\psi_1^2+\psi_1+\psi_2^2+\psi_3^2)+\omega_3(\psi_1^2+\psi_2^2+\psi_3^2+1)\right),
\end{aligned}
$$

wherein the state $x = (\omega^T, \psi^T)^T$ consists of the angular velocity vector in a body-fixed frame $\omega \in \mathbb{R}^3$ and the Rodrigues parameter vector $\psi \in \mathbb{R}^3$.

The control torque $u \in \mathbb{R}^3$ is updated every 0.1 s by a neural network with three hidden layers, each of which has 64 neurons. The activations of the hidden layers are sigmoid and identity, respectively. We train the neural-network controller using supervised learning methods to learn from a known nonlinear controller [50]. The initial state set is:

$$
\begin{aligned}
&\omega_1 \in [-0.45, -0.44], \omega_2 \in [-0.55, -0.54], \omega_3 \in [0.65, 0.66],\\
&\psi_1 \in [-0.75, -0.74], \psi_2 \in [0.85, 0.86], \psi_3 \in [-0.65, -0.64].
\end{aligned}
$$

Specification The system should not reach the following unsafe set in 3 s (30 time steps):

$$
\begin{aligned}
&\omega_1 \in [-0.2, 0], \omega_2 \in [-0.5, -0.4], \omega_3 \in [0, 0.2],\\
&\psi_1 \in [-0.7, -0.6], \psi_2 \in [0.7, 0.8], \psi_3 \in [-0.4, -0.2].
\end{aligned}
$$

We want to show that the above specification does not hold.

3.9 Benchmark: QUAD

This benchmark studies a neural-network controlled quadrotor (QUAD) with twelve state variables [10]. We have the inertial (north) position x_1, the inertial (east) position x_2, the altitude x_3, the longitudinal velocity x_4, the lateral velocity x_5, the vertical velocity x_6, the roll angle x_7, the pitch angle x_8, the yaw angle x_9, the roll rate x_{10}, the pitch rate x_{11}, and the yaw rate x_{12}. The control torque $u \in \mathbb{R}^3$ is updated every 0.1 s by a neural network with 3 hidden layers, each of which has 64 neurons. The activations of the hidden layers and the output layer are sigmoid and identity, respectively. The dynamics are given by the following equations [10, eq. (12-16)]:

$$
\begin{aligned}
\dot{x}_1 =& \cos(x_8)\cos(x_9)x_4 + (\sin(x_7)\sin(x_8)\cos(x_9) - \cos(x_7)\sin(x_9))\,x_5 \\
&+ (\cos(x_7)\sin(x_8)\cos(x_9) + \sin(x_7)\sin(x_9))\,x_6, \\
\dot{x}_2 =& \cos(x_8)\sin(x_9)x_4 + (\sin(x_7)\sin(x_8)\sin(x_9) + \cos(x_7)\cos(x_9))\,x_5 \\
&+ (\cos(x_7)\sin(x_8)\sin(x_9) - \sin(x_7)\cos(x_9))\,x_6, \\
\dot{x}_3 =& \sin(x_8)x_4 - \sin(x_7)\cos(x_8)x_5 - \cos(x_7)\cos(x_8)x_6, \\
\dot{x}_4 =& x_{12}x_5 - x_{11}x_6 - g\sin(x_8,) \\
\dot{x}_5 =& x_{10}x_6 - x_{12}x_4 + g\cos(x_8)\sin(x_7), \\
\dot{x}_6 =& x_{11}x_4 - x_{10}x_5 + g\cos(x_8)\cos(x_7) - g - u_1/m, \\
\dot{x}_7 =& x_{10} + \sin(x_7)\tan(x_8)x_{11} + \cos(x_7)\tan(x_8)x_{12}, \\
\dot{x}_8 =& \cos(x_7)x_{11} - \sin(x_7)x_{12}, \\
\dot{x}_9 =& \frac{\sin(x_7)}{\cos(x_8)}x_{11} - \frac{\cos(x_7)}{\cos(x_8)}x_{12}, \\
\dot{x}_{10} =& \frac{J_y - J_z}{J_x}x_{11}x_{12} + \frac{1}{J_x}u_2, \\
\dot{x}_{11} =& \frac{J_z - J_x}{J_y}x_{10}x_{12} + \frac{1}{J_y}u_3, \\
\dot{x}_{12} =& \frac{J_x - J_y}{J_z}x_{10}x_{11} + \frac{1}{J_z}\tau_\psi,
\end{aligned}
$$

where

$$
\begin{aligned}
&g = 9.81, \quad m = 1.4, \quad J_x = 0.054, \\
&J_y = 0.054, \quad J_z = 0.104, \quad \tau_\psi = 0.
\end{aligned}
$$

The initial set is:

$$
\begin{aligned}
&x_1 \in [-0.4, 0.4], x_2 \in [-0.4, 0.4], x_3 \in [-0.4, 0.4], x_4 \in [-0.4, 0.4], \\
&x_5 \in [-0.4, 0.4], x_6 \in [-0.4, 0.4], x_7 = 0, x_8 = 0, x_9 = 0, x_{10} = 0, x_{11} = 0, x_{12} = 0.
\end{aligned}
$$

Specification The control goal is to stabilize the attitude x_3 to a goal region $[0.94, 1.06]$ and remain within these bounds with a time horizon of 5 s (50 time steps).

3.10 2D Spacecraft Docking

In the 2D spacecraft docking environment, the state of an active deputy spacecraft is expressed relative to the passive chief spacecraft in Hill's reference frame [26]. The dynamics are given by a first-order approximation of the relative motion dynamics between the deputy and chief spacecraft, which is given by Clohessy-Wiltshire [17] equations [52, eq. (12)],

$$
\begin{bmatrix} \dot{s}_x \\ \dot{s}_y \\ \ddot{s}_x \\ \ddot{s}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 2n \\ 0 & 0 & -2n & 0 \end{bmatrix} \begin{bmatrix} s_x \\ s_y \\ \dot{s}_x \\ \dot{s}_y \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & \frac{1}{m} \end{bmatrix} u, \tag{13}
$$

where $m = 12$ (kg), $n = 0.001027$ (rad/s), and $u \in \mathbb{R}^2$.

The neural network controller was trained on the Docking 2D environment with reinforcement learning using the training procedure described in [52]. However, the training procedure differed in providing only the full state (position and velocity) as input and with hard clipping of output actions replaced with soft tanh clipping. The neural network architecture was a shallow multilayer perceptron with 2 hidden layers of 256 neurons and tanh activation functions, and a linear output layer. The pre-processing and post-processing of the controller have been incorporated into the model as linear layers. The controller was trained with a sampling time of 1 s.

Specification The spacecraft should satisfy the following safety constraints for 40 s:

$$(\dot{s}_x^2 + \dot{s}_y^2)^{\frac{1}{2}} \leq 0.2 + 2n(s_x^2 + s_y^2)^{\frac{1}{2}}, \tag{14}$$

given the initial set is

$$s_x \in [70, 106], s_y \in [70, 106], \dot{s}_x \in [-0.28, 0.28], \dot{s}_y \in [-0.28, 0.28].$$

4 Verification Results

For each of the participating tools, we obtained verification results for some or all of the benchmarks. This year's competition included the submission of the tools for repeatability prior to the writing of the report to ensure a fairer competition. Reachable sets are shown for those methods that are able to construct them. The published results (no plots) are available at https://arch.repeatability.cps.cit.tum.de/frontend/submissions.

4.1 CORA

We present the results utilizing *CORA* on each of the benchmarks. CORA is able to show verification/violation of the specifications in all benchmarks except one using the entire input set. For details about the reachability parameters used, such as the step size of our algorithm for continuous-time benchmarks and the parameters for the propagation through the neural network, we refer to the submission code available at https://gitlab.com/goranf/ARCH-COMP/-/tree/master/2023/AINNCS/cora. We show the results of all benchmarks in Tab. 1. In addition, we give details for each benchmark and provide the accompanying plots below.

4.1.1 ACC

CORA is able to verify all specifications automatically. The computed reachable set along with some simulations are shown in Figure 5.

4.1.2 Sherlock-Benchmark-9 (TORA)

CORA is able to verify all specifications automatically. The computed reachable set along with some simulations are shown in Figure 6 for the ReLU controller (specification 1) and in Figure 7 for the sigmoid and tanh controllers (specification 2).

4.1.3 Sherlock-Benchmark-10 (Unicycle)

CORA is able to verify all specifications automatically. The computed reachable set along with some simulations are shown in Figure 8.

Table 1: ***CORA.*** Results of all benchmarks including the time to compute the simulations (ST), the time to compute and verify the reachable set (VT), and total time (TT). All time values are in seconds.

Benchmark	Instance	ST	VT	TT	Result
ACC	-	1.63	4.05	5.70	`VERIFIED`
Sherlock-Benchmark-9 (TORA)	ReLU	3.05	21.94	25.00	`VERIFIED`
Sherlock-Benchmark-9 (TORA)	ReLU-Tanh	1.43	13.76	15.21	`VERIFIED`
Sherlock-Benchmark-9 (TORA)	Sigmoid	1.29	10.84	12.23	`VERIFIED`
Sherlock-Benchmark-10 (Unicycle)	-	5.73	6.97	12.71	`VERIFIED`
Sherlock-Benchmark-10 (Unicycle)	-	5.73	6.97	12.71	`VERIFIED`
Single pendulum	-	0.50	2.10	2.61	`VIOLATED`
Double pendulum	Normal controller	0.62	5.59	6.23	`VIOLATED`
Double pendulum	Robust controller	1.13	1.98	3.11	`VIOLATED`
Airplane	-	1.13	1.98	3.11	`VIOLATED`
Attitude Control	-	0.78	10.06	10.84	`VERIFIED`
Quadrotor	-	5.11	352.84	357.96	`VERIFIED`
Spacecraft docking	-	11.86	60.98	73.86	`UNKOWN`
VCAS (middle acc.)	$\dot{h}_0(0) = -19.5$	0.18	0.19	0.38	`VERIFIED`
VCAS (middle acc.)	$\dot{h}_0(0) = -22.5$	0.08	0.08	0.16	`VERIFIED`
VCAS (middle acc.)	$\dot{h}_0(0) = -25.5$	0.07	-	0.07	`VIOLATED`
VCAS (middle acc.)	$\dot{h}_0(0) = -28.5$	0.05	-	0.05	`VIOLATED`
VCAS (worst acc.)	$\dot{h}_0(0) = -19.5$	0.09	0.10	0.20	`VERIFIED`
VCAS (worst acc.)	$\dot{h}_0(0) = -22.5$	0.04	-	0.04	`VIOLATED`
VCAS (worst acc.)	$\dot{h}_0(0) = -25.5$	0.03	-	0.04	`VIOLATED`
VCAS (worst acc.)	$\dot{h}_0(0) = -28.5$	0.03	-	0.04	`VIOLATED`

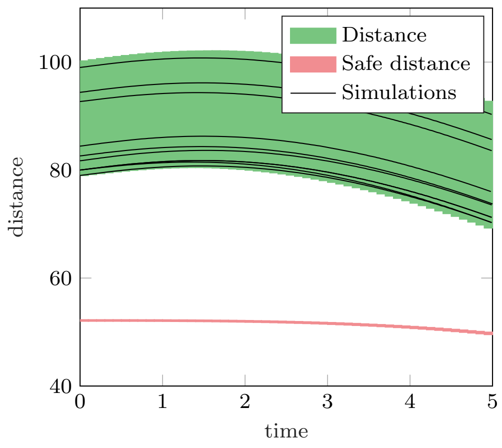
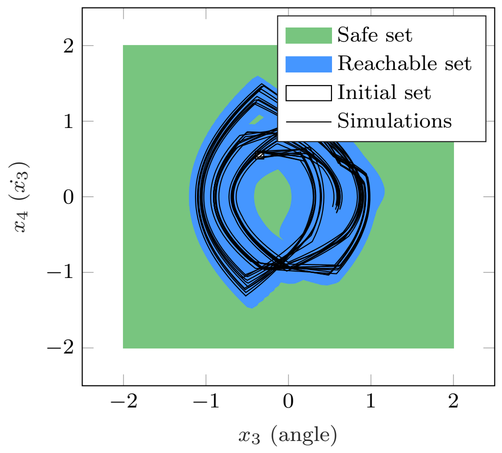
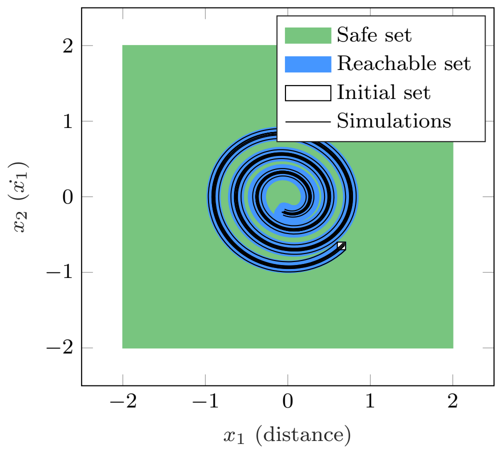


Figure 5: ***CORA.*** Computed reachable set of the ACC benchmark.

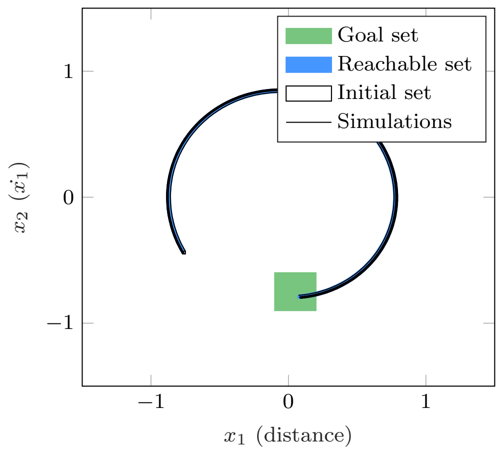


(a) ReLU controller

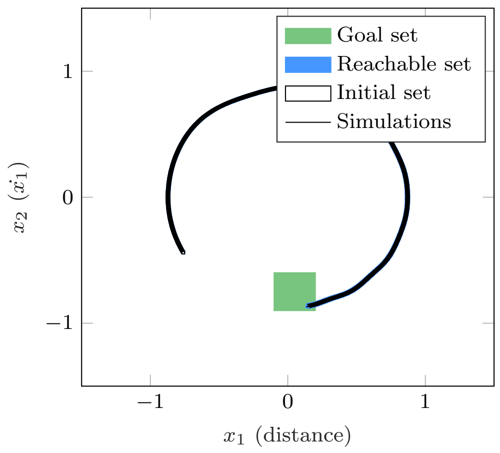


(b) ReLU controller

Figure 6: ***CORA.*** Computed reachable set of the Sherlock-Benchmark-9 (TORA) benchmark using specification 1.



(a) Tanh controller



(b) Sigmoid controller

Figure 7: ***CORA.*** Computed reachable set of the Sherlock-Benchmark-9 (TORA) benchmark using specification 2.

4.1.4 VCAS

The VCAS benchmark has discrete time steps and multiple controllers, which is currently not supported by CORA. Thus, a custom algorithm was built for this benchmark. To deal with the discrete input set $\dot{h}_0(0) \in \{-19.5, -22.5, -25.5, -28.5\}$, we run the algorithm with each element of the input set individually. As proposed in the benchmark specifications, we show the results when always the middle acceleration of the controllers is chosen and the results when always the worst acceleration is chosen.

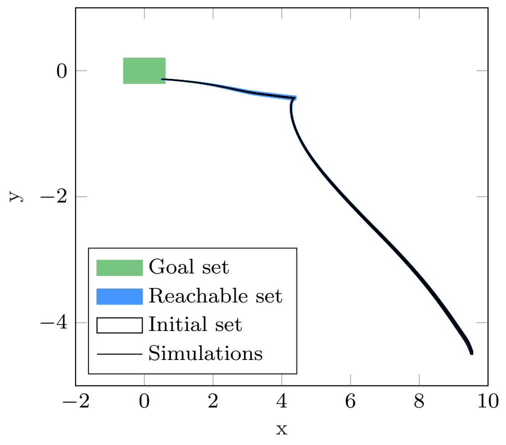
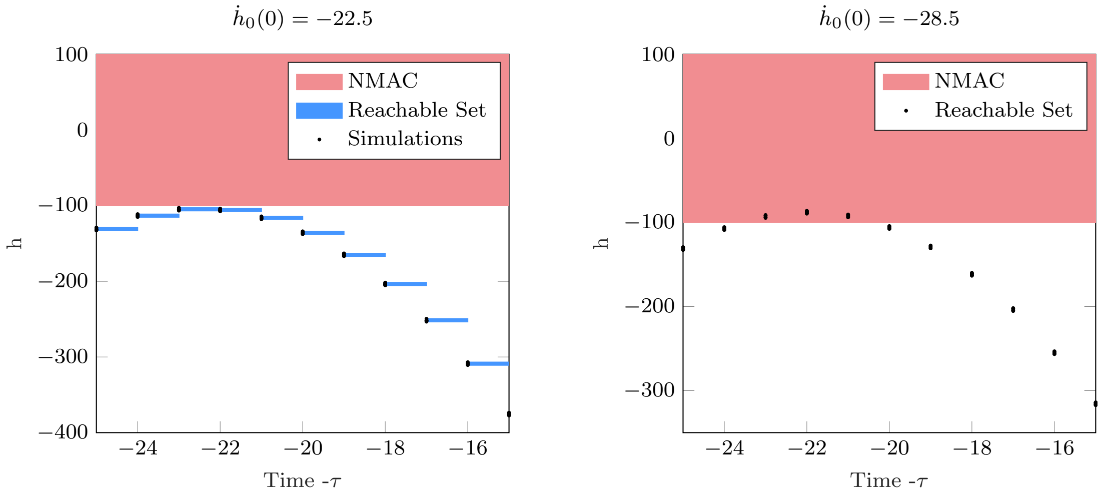


Figure 8: ***CORA.*** Computed reachable set of the Sherlock-Benchmark-10 (Unicycle) benchmark.

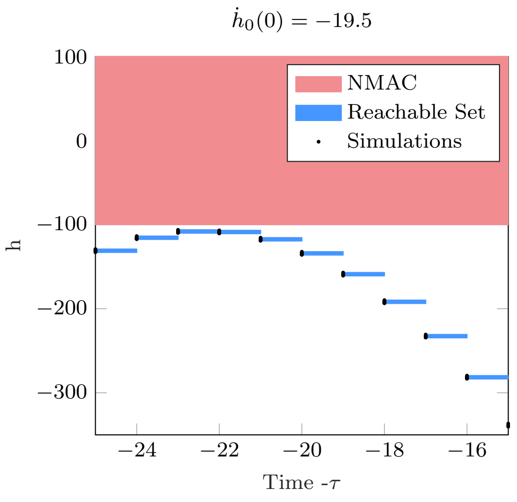


(a) ReLU controller (b) ReLU controller

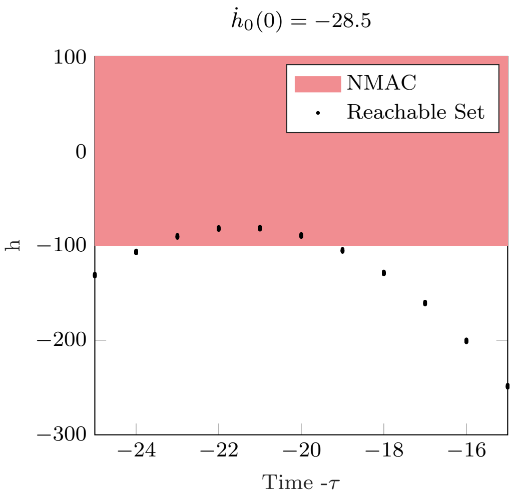
Figure 9: ***CORA.*** Computed reachable set of the VCAS benchmark with middle acceleration.

VCAS (middle acceleration) Here we always use the middle of the possible accelerations. We are able to verify the benchmark for $\dot{h}_0(0) \in \{-19.5, -22.5\}$ and can show violations for $\dot{h}_0(0) \in \{-25.5, -28.5\}$. The computed reachable set along with some simulations are shown in Figure 9.

VCAS (worst acceleration) Here we always use the worst possible acceleration. We are able to verify the benchmark for $\dot{h}_0(0) \in \{-19.5\}$ and can show violations for $\dot{h}_0(0) \in \{-22.5, -25.5, -28.5\}$. The computed reachable set along with some simulations are shown in Figure 10.



(a) ReLU controller



(b) ReLU controller

Figure 10: ***CORA.*** Computed reachable set of the VCAS benchmark with worst acceleration.

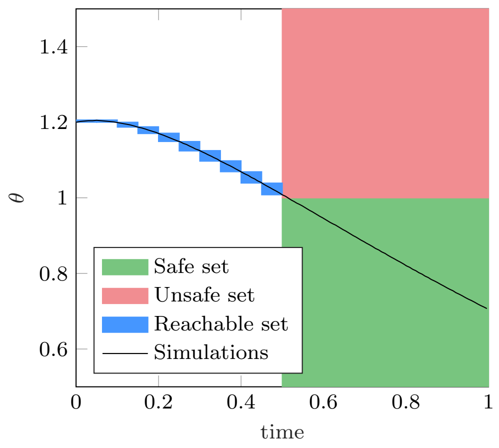


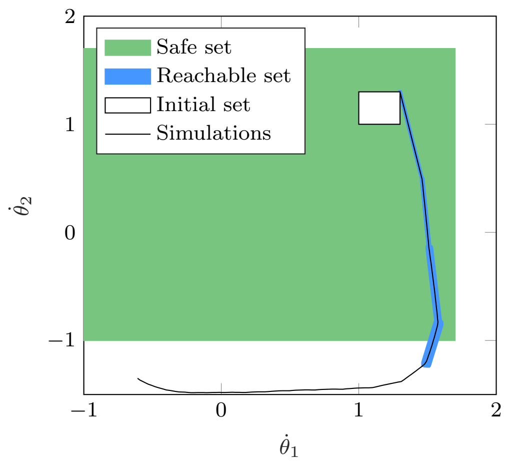
Figure 11: ***CORA.*** Computed reachable set of the single pendulum benchmark.

4.1.5 Single Pendulum

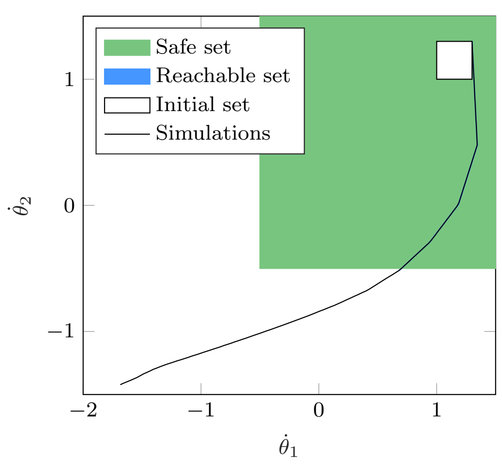
CORA is able to show specification violations of the single-pendulum benchmark. Figure 11 shows the verified simulation run which is violating the specification.

4.1.6 Double Pendulum

CORA is able to show violations for both controllers of the double-pendulum benchmark. Figure 12 shows the verified simulation runs which are violating the specification. Note that the reachable set computation is very tight for the more robust controller and thus barely visible.



(a) Less robust controller



(b) More robust controller

Figure 12: ***CORA.*** Computed reachable set of the double pendulum benchmark.

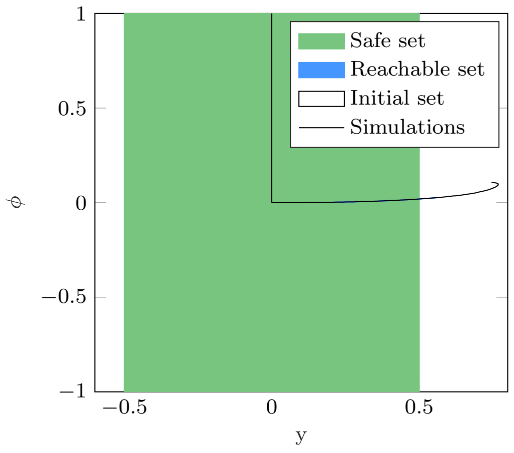


Figure 13: ***CORA.*** Computed reachable set of the airplane benchmark.

4.1.7 Airplane

CORA is able to show specification violations of the airplane benchmark. Figure 13 shows the verified simulation run which is violating the specification. Note that the reachable set computation is very tight and thus barely visible.

4.1.8 Attitude Control

CORA is able to verify all specifications automatically. The computed reachable set along with some simulations are shown in Figure 14.

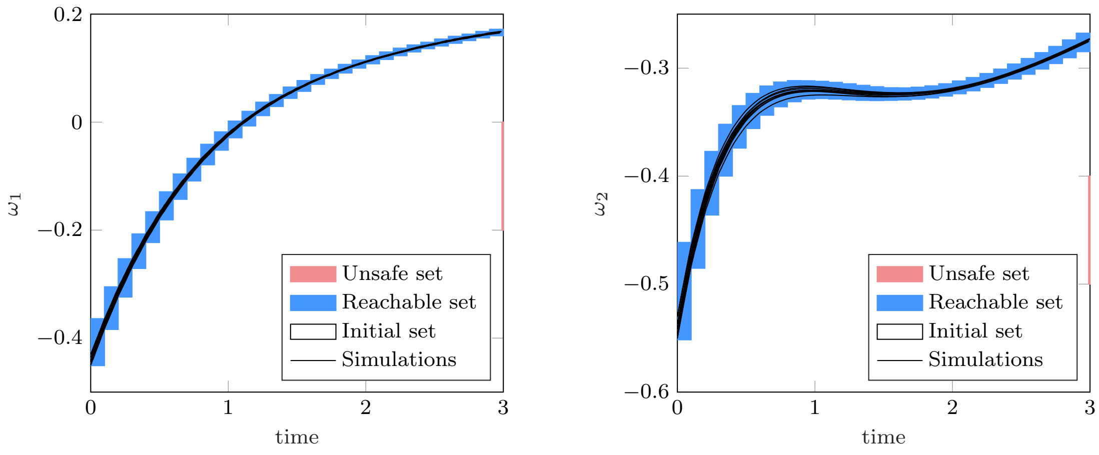


Figure 14: ***CORA.*** Computed reachable set of the attitude control benchmark.

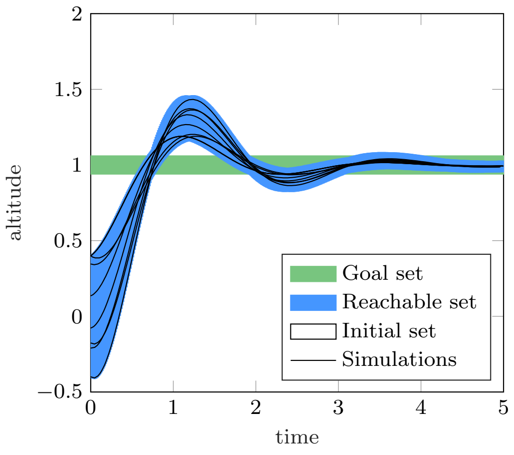


Figure 15: ***CORA.*** Computed reachable set of the quadrator benchmark.

4.1.9 Quadrotor

CORA is able to verify all specifications automatically. While it is not possible to verify the benchmark using linear abstractions of all neurons, we can verify the benchmark by applying the refinement approach described in [38]. The computed reachable set along with some simulations are shown in Figure 15.

4.1.10 2D Spacecraft Docking

The spacecraft docking benchmark appeared difficult to verify for our approach. While the simulations seem to be stable, the reachable set explodes over time and thus we are unable to verify the benchmark. The computed reachable set along with some simulations are shown in Figure 16.

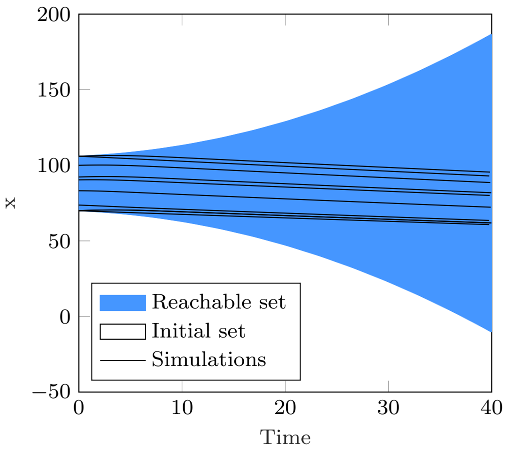


Figure 16: ***CORA.*** Computed reachable set of the spacecraft docking benchmark.

4.2 JuliaReach

This subsection presents the results of *JuliaReach*. JuliaReach was able to analyze eight benchmark problems (four verified, four falsified). For each problem, JuliaReach uses slightly different settings as described below.

4.2.1 ACC

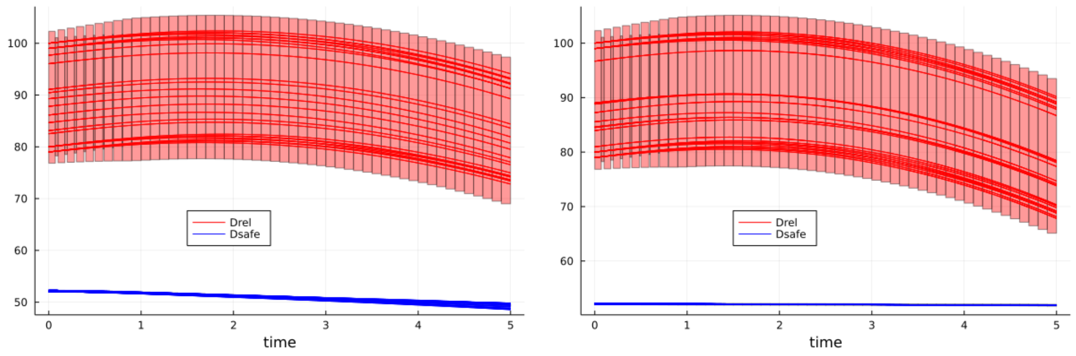


Figure 17: ***JuliaReach.*** Analysis results for the ACC benchmark and the ReLU controller (left) resp. the tanh controller (right). The plot additionally shows simulations.

Using the parameters `abstol=1e-6, orderT=6, orderQ=1`, JuliaReach verifies the property $D_{\text{rel}} \geq D_{\text{safe}}$ for the whole time horizon in 0.5 s. The reach sets of D_{rel} and D_{safe} together with some simulations are shown in Figure 17.

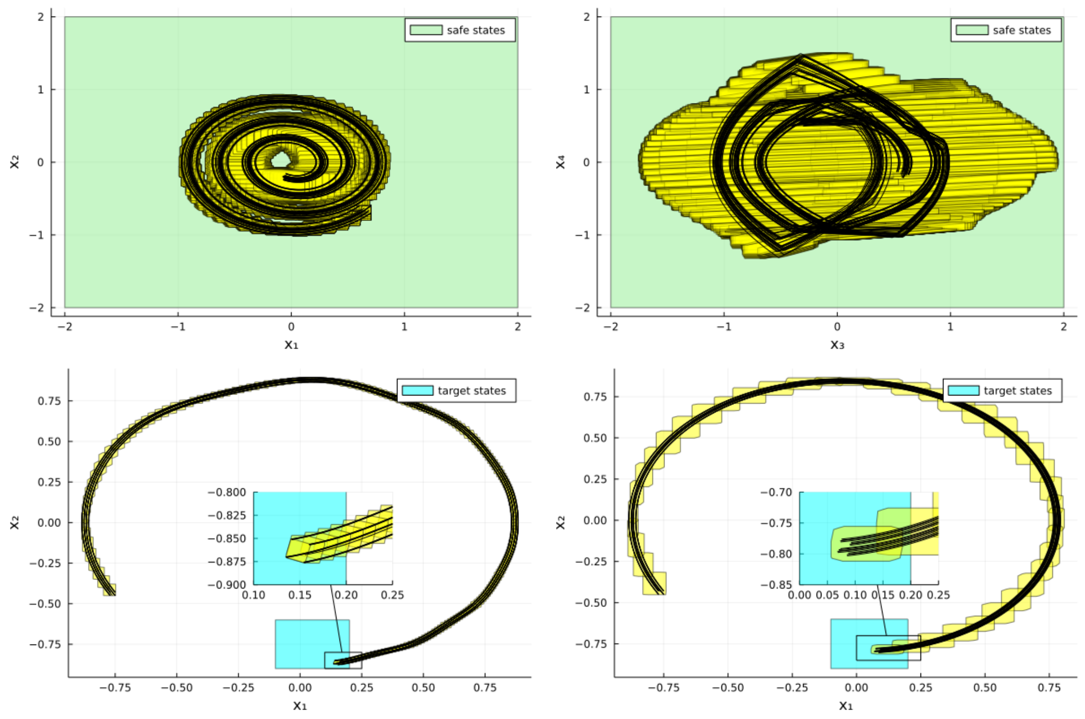


Figure 18: ***JuliaReach.*** Analysis results for the TORA benchmark for the ReLU controller (top), the sigmoid controller (bottom left), resp. the ReLU/tanh controller (bottom right). The plots additionally show simulations.

4.2.2 Sherlock-Benchmark-9 (TORA)

We analyze three different controllers for the TORA benchmark problem. For the ReLU controller, the approximation error is hard to tame for the JuliaReach approach. To maintain enough precision to prove correctness, the initial states are split into $4 \times 4 \times 3 \times 5$ boxes. While each box spawns an independent analysis that could be parallelized, the run time of the sequential execution is 25 minutes. We use the parameters `abstol=1e-10, orderT=8, orderQ=3`. The reach sets of all 240 runs together with some simulations, projected to x_1/x_2 resp. x_3/x_4, are shown in Figure 18. The sigmoid and ReLU/tanh controllers reach the target set within the given time constraints, as shown in the x_1/x_2 projections in Figure 18. The latter analyses take 5 s respectively 1 s.

4.2.3 Sherlock-Benchmark-10 (Unicycle)

The disturbance w is modeled here as a constant with an uncertain initial value. Simulations show that the target set is reached only at the last moment, so the analysis requires high precision to prove containment of the last reach set. Using the parameters `abstol=1e-15, orderT=10, orderQ=1` and splitting the initial states into $3 \times 1 \times 8 \times 1$ boxes, JuliaReach verifies the property in 61 s. The reach sets of all 24 runs together with some simulations, projected to x_1/x_2 resp. x_3/x_4, are shown in Figure 19. JuliaReach can evaluate the Taylor

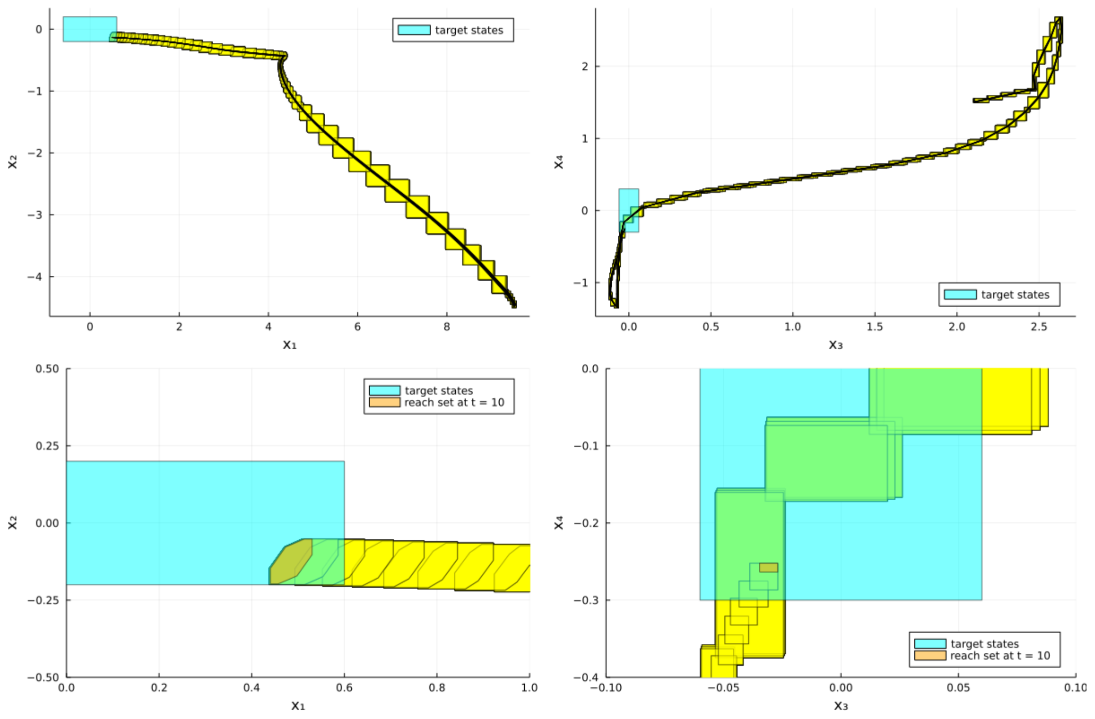


Figure 19: ***JuliaReach.*** Analysis results for the Unicycle benchmark. The first two plots show the overall reach sets and simulations. The other two plots show a close-up of the target set. The orange subset of the last reach set is obtained at time point $t = 10$.

polynomial at the time point $t = 10$ (rather than the last time *interval*), which results in a more precise result (as shown in the plots), although additional precision is not required for this problem, as the reach set for the last time interval is already fully contained in the target set.

4.2.4 VCAS

The VCAS benchmark problem differs from the other problems in that it uses multiple controllers and discrete time. There is currently no native support for this setting in JuliaReach, so a custom simulation algorithm that always chooses the central acceleration was used. JuliaReach produces ten simulations in 1 s, which indicate satisfaction for the initial values $\dot{h}(0) \in \{-19.5, -22.5\}$ but show a violation of the specification for the other initial values. The simulation results are shown in Figure 20.

4.2.5 Single Pendulum

This system violates the specification; hence it suffices to start the analysis from a subset of the initial states and interrupt when a violation is detected. Here, starting from the highest coordinate in each dimension, a violation occurs within eleven control periods. Using the parameters `abstol=1e-7, orderT=4, orderQ=1`, JuliaReach falsifies the property in 0.5 s. Figure 21 shows the reach sets together with a simulation projected to time and θ.

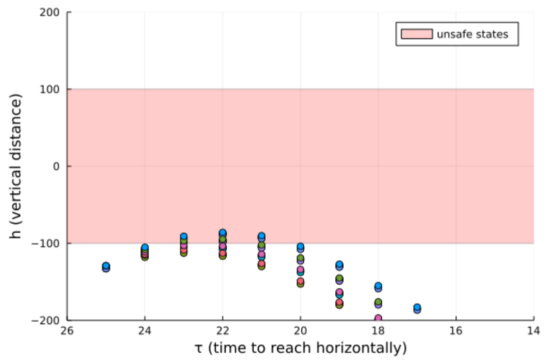


Figure 20: ***JuliaReach.*** Simulations of the VCAS benchmark.

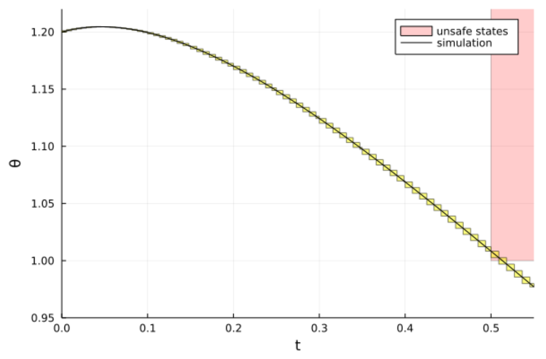


Figure 21: ***JuliaReach.*** Analysis results for the Single-Pendulum benchmark until time $t = 0.55$. The plot additionally shows a simulation.

4.2.6 Double Pendulum

Similarly to the single-pendulum problem, this system violates the specification for both controllers; hence it suffices to start the analysis from a subset of the initial states and interrupt when a violation is detected. Considering the less robust controller, when starting from the highest value in each dimension, a violation occurs within five control periods. Similarly, considering the more robust controller and starting from the lowest value in each dimension, a violation occurs within seven control periods. Using the parameters `abstol=1e-9, orderT=8, orderQ=1` and an older version of the Taylor-model algorithm, JuliaReach falsifies the property in 5 s (less robust controller) resp. 1 s (more robust controller). Figure 22 shows the reach sets together with a simulation projected to $\dot{\theta}_1/\dot{\theta}_2$.

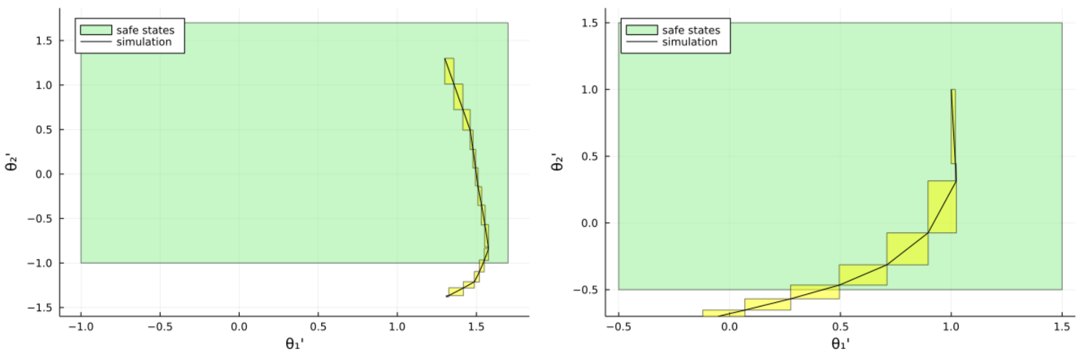


Figure 22: ***JuliaReach.*** Analysis results for the double-pendulum benchmark, including a simulation. The first plot shows the results for the less robust controller until time $t = 0.25$. The second plot shows the results for the more robust controller until time $t = 0.14$.

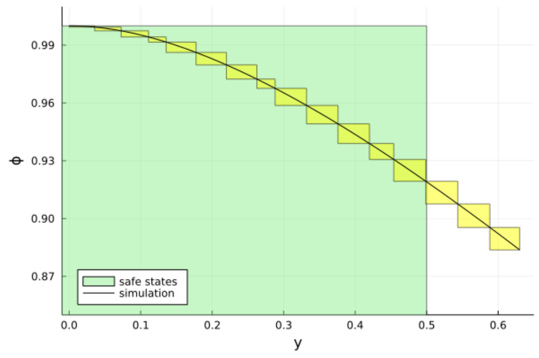


Figure 23: ***JuliaReach.*** Analysis results for the airplane benchmark until time $t = 0.4$, including a simulation.

4.2.7 Airplane

This system violates the specification; hence it suffices to start the analysis from a subset of the initial states and interrupt when a violation is detected. When starting from the highest coordinate in each dimension, a violation occurs immediately in dimension θ and within four control periods in dimension y. To obtain some nontrivial results, JuliaReach ignores the violation in dimension θ. Using the parameters `abstol=1e-10, orderT=7, orderQ=1`, JuliaReach falsifies the property in 5 s. The reach sets together with a simulation, projected to y/ϕ, are shown in Figure 23.

4.2.8 Attitude Control

Using the parameters `abstol=1e-6, orderT=6, orderQ=1`, JuliaReach verifies the property in 1 s. The reach sets together with some simulations are shown in Figure 24.

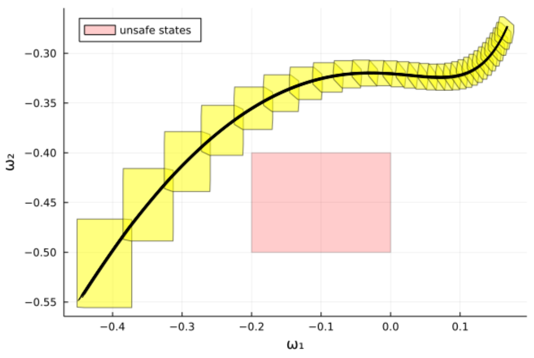


Figure 24: ***JuliaReach.*** Analysis results for the attitude control benchmark, including simulations.

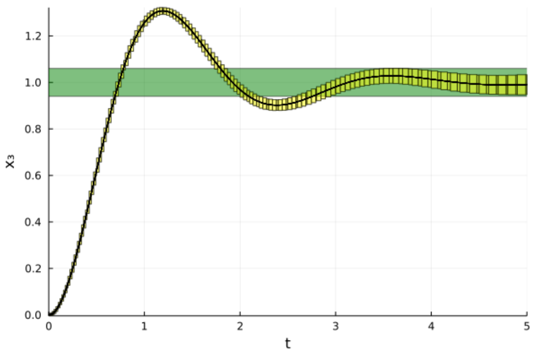


Figure 25: ***JuliaReach.*** Analysis results for the quadrotor benchmark, including simulations.

4.2.9 QUAD

Although simulations indicate that this controller is safe, the precision of JuliaReach is not high enough to prove it. The property can be proven for a smaller initial set $[-0.004, 0.004]^6 \times \{0\}^6$. Using the parameters `abstol=1e-8, orderT=5, orderQ=1`, JuliaReach verifies the property in 11 s. The reach sets together with some simulations are shown in Figure 25.

4.2.10 2D Spacecraft Docking

Although simulations indicate that this controller is safe, the precision of JuliaReach is not high enough to prove it. The property can be proven for a smaller initial set $[87, 89]^2 \times [-0.01, 0.01]^2$. Using the parameters `abstol=1e-10, orderT=5, orderQ=1`, JuliaReach verifies the property in 0.5 s. The reach sets together with some simulations are shown in Figure 26. Since the property is four-dimensional, it cannot be illustrated in the plot.

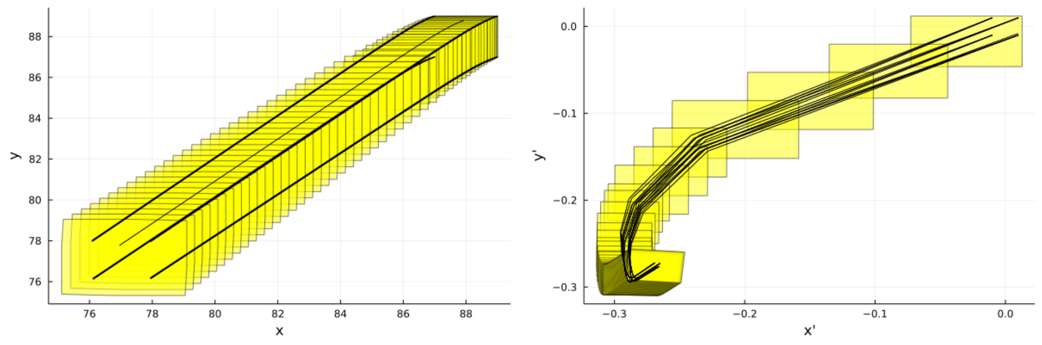


Figure 26: ***JuliaReach.*** Analysis results for the Spacecraft benchmark, including simulations.

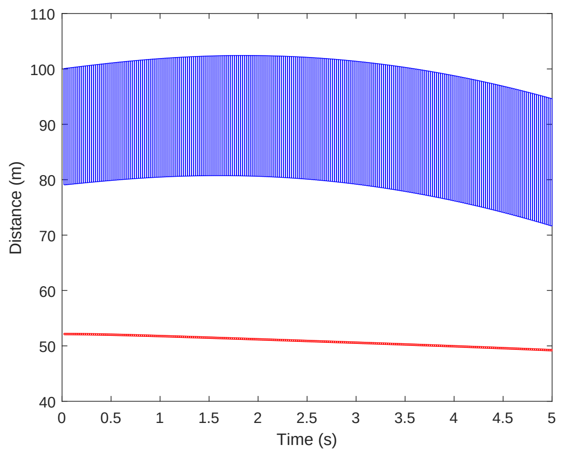


Figure 27: ***NNV.*** Computed reachable set of the ACC benchmark.

4.3 NNV

We present the results utilizing *NNV* on each of the benchmarks. For details about the reachability parameters used, we refer to the submission code available at https://gitlab.com/goranf/ARCH-COMP/tree/master/2023/AINNCS/nnv. We have been able to encode all the benchmarks into NNV and attempted to verify all of them, however, due to the conservativeness of the methods and the complexity of the different benchmarks, NNV was unable to verify the unicycle, quadrotor, attitude, and docking spacecraft benchmarks.

4.3.1 ACC

NNV is able to verify the safety property $D_{\text{rel}} \geq D_{\text{safe}}$ in 24.98 s. The reach sets of D_{rel} and D_{safe} are shown in Figure 27.

4.3.2 Sherlock-Benchmark-9 (TORA)

NNV is able to verify all three controllers for the TORA benchmark. For the ReLU controller, NNV verifies the specification in 21.97 s, while it takes 2148 s and 4074 s to verify the ReLU-tanh and sigmoid controllers respectively. The differences in computation time across controllers are due to the partitioning of the initial state sets. The reach sets are shown in Figure 28.

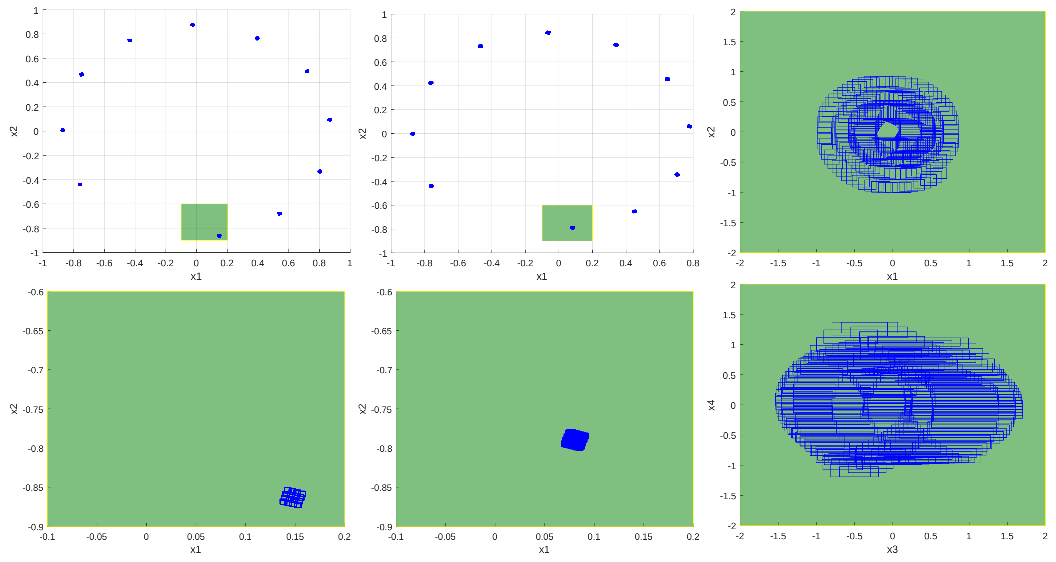


Figure 28: ***NNV.*** Analysis results for the Tora benchmark showing the TORA sets in **blue** and the goal region in **green**. The left figures correspond to the sigmoid controller, the middle two to the ReLU-tanh controller, and the two on the right to the ReLU controller. For the *sigmoid* and *ReLU-tanh* controllers only the reach sets at every control period are shown in the top row. The corresponding zoomed-in pictures of the goal region are depicted in the bottom row.

4.3.3 VCAS

NNV is able to verify the NMAC safety property for the whole time horizon for each of the cases with an average computation time of 3.1 s. There are 5 cases where we prove that the system is unsafe and 3 where the system is safe, which corresponds to [middle, 19.5], [middle, 22.5], and [worst, 19.5]. These results are depicted in Figures 29 and 30.

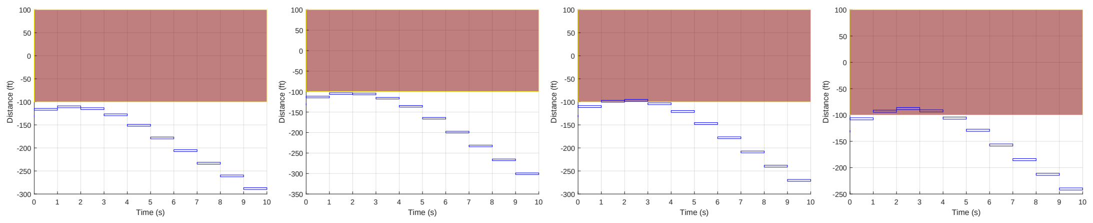


Figure 29: ***NNV.*** Analysis results for the VCAS benchmark showing the aircraft sets in blue and the unsafe region in red, when selecting the middle acceleration value at each control period

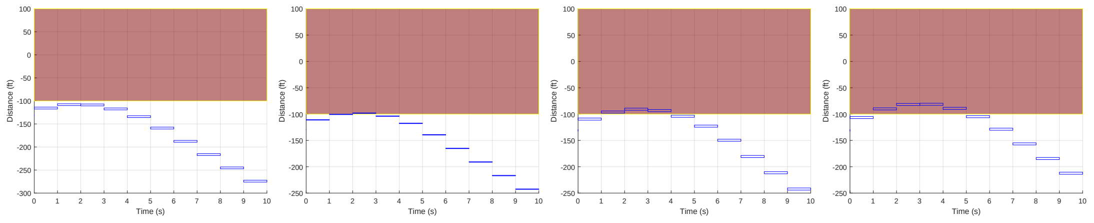


Figure 30: ***NNV.*** Analysis results for the VCAS benchmark showing the aircraft sets in blue and the unsafe region in red, when selecting the worst possible acceleration value at each control period.

4.3.4 Single Pendulum

For the single pendulum, it is sufficient to start with a smaller initial state to prove that the safety property is violated, with a computation time of 2.85 s. The reach sets are depicted in Figure 31.

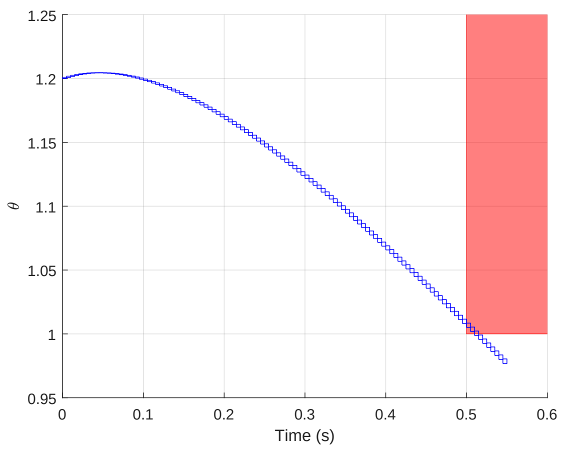


Figure 31: ***NNV.*** Analysis results for the single pendulum benchmark showing the sets in **blue** and the unsafe region in **red**.

4.3.5 Double Pendulum

Similar to the single pendulum, we can demonstrate that the property is violated starting from a smaller initial set, for both the more robust and less robust controllers. The results are depicted in Figure 32, and the computation times are 45 s and 43.25 s for the more robust and less robust controllers respectively.

4.3.6 Airplane

NNV is able to show that the property is violated by computing the reach sets from a smaller initial region, in 41.7 s. The results are depicted in Figure 33.

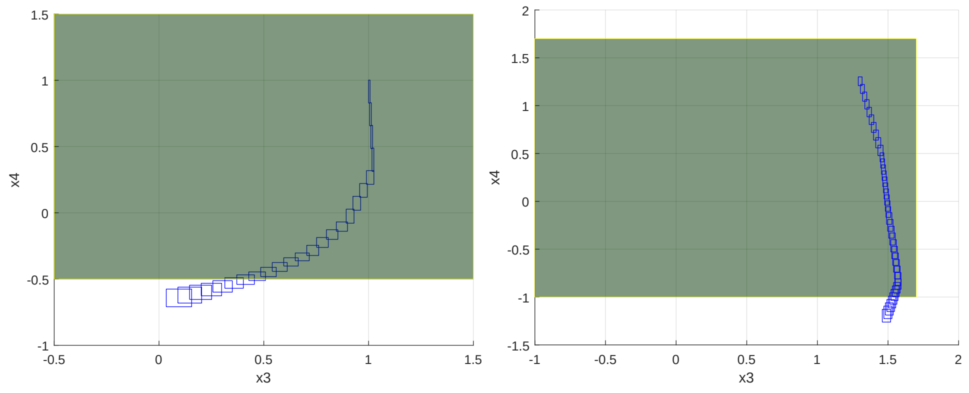


Figure 32: ***NNV.*** Analysis results for the Double Pendulum benchmark showing the reach sets in **blue** and the safe region in **green**, when using the more robust controller (left) and less robust controller (right).

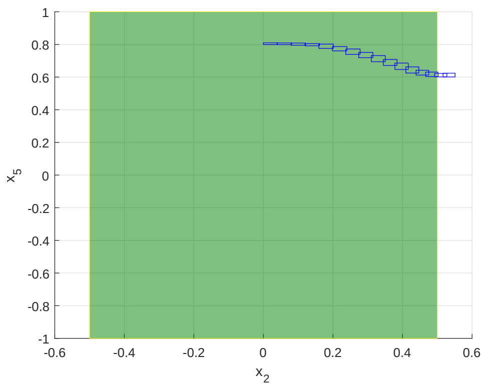


Figure 33: ***NNV.*** Analysis results for the airplane benchmark showing the reach sets in **blue** and the goal region in **green**.

4.4 Summary

Finally, we present a table with the timed results for each of the benchmarks and highlight the fastest tool for each of them in Table 2. We present the average computation time across all VCAS instances. However, JuliaReach only presents the time it took to falsify 2 out of 8 instances (0.95 s), which is the data presented here.

Table 2: Summary of verification results, showing the computation time (s) for verified instances and "-" for those unknown. The fastest computation times are highlighted in **bold**.

	ACC	**TORA** *ReLU*	**TORA** *ReLU-tanh*	**TORA** *sigmoid*	**Unicycle**	**VCAS**	**Pend (S)**	**Pend (D)** *more*	**Pend (D)** *less*	**Airplane**	**Attitude**	**Quad**	**Docking**
CORA	5.70	25.01	15.21	12.23	**12.73**	**0.12**	2.63	3.12	6.24	10.85	8.19	**358.0**	-
JuliaReach	**0.47**	1491	**1.47**	**5.15**	60.86	0.95	**0.53**	**1.07**	**5.27**	**4.75**	**0.98**	-	-
NNV	24.98	**21.97**	2149	4074	-	3.18	2.85	45.07	43.25	41.75	-	-	-

5 Category Status and Challenges

Year-over-year comparison. It has been 5 years of this competition, and we have observed an immense improvement in the area in this last half-decade. In 2019, when the competition began, there were only 5 benchmarks, out of which the 3 participants were only able to verify an average of less than 2 benchmarks per tool – including the 2 instances where the controller was modified (ReLU vs tanh) to be supported by one of the tools. Over the next three years (2020-2022), there were a total of 8 different tools that participated in the competition, increasing the complexity of the benchmarks along with the quality improvement of the participating tools. Ending with this year's competition, we can also observe certain improvements over last year: Overall, the tools successfully verified 78.3% of all instances, compared to 75% from last year (2022) and 60% from the 2021 competition.

Automatic Verification. Some advances have been made to determine the verification results automatically. The current state of the art requires the authors to tune the hyperparameters of the verification manually as well as hard-code counterexamples, which can be time-consuming and requires expert knowledge. As an attempt towards a push-button approach, CORA is able to find counterexamples through simulations and verify the falsifying trajectory automatically. If no counterexample is found, CORA automatically tries to verify the benchmark.

Closed-loop reachability integration. Since the first NNV publication [66], there have been several additions and enhancements to the tool, including the development of new star-based reachability methods [65] and support for several layers and deep learning architectures such as neuralODEs and RNNs [41]. In terms of AINNCS verification, these changes include a reduction in the overapproximation of the plant reachable sets over the control steps, as well as allowing for faster but more conservative reachability methods for the controller (relax-reachability [65]). These changes helped NNV to verify two more benchmark instances (TORA) than last year's competition. The integration of polynomial zonotopes into reachability analysis of neural-network controlled systems [36] as well as an adaptive refinement scheme [38] allowed CORA to verify the seemingly difficult QUAD benchmark.

Activation Function Types (controllers): For this year's set of benchmarks, all neural network controllers contain one or more of the following activation functions: ReLU, linear, sigmoid, and tanh. The tools have support for the types: linear, piecewise linear (ReLU), and nonlinear activation functions. In the future, we will consider adding other variations of the existing activation functions, such as LeakyReLU or PReLU.

Plant Models: This and last year's competition have considered linear and nonlinear plants, both in discrete and continuous time. In future iterations, we plan to add hybrid automata plants as we look to report a more complete analysis of the participating verification tools. Hybrid automata plants will be especially interesting because combined continuous and discrete dynamics are complex, which is very challenging for current AINNCS verification tools. We will also consider adding neural ODEs as in [45], which will also increase the complexity of the benchmarks and may require a more general encoding of NNCS by most participating tools to verify them.

Neural Network Architectures and Parameterization: The neural network architectures presented in this work are fairly simple. Similar to last year, they have no more than a thousand neurons and no more than 5 hidden layers in their architecture, unlike some of the networks that can be analyzed without the plant. Also, the maximum number of inputs and outputs of the controllers are 12 and 6, respectively, both in the airplane benchmark. Considering the VCAS benchmark, these networks have 9 outputs, although these are translated into a single input to the plant model. However, for some benchmarks, there are still state-space explosion and scalability issues to address in both the neural network controllers and plant analysis. These issues could come from the repeated interaction between the network and the states.

Model Formats: Following previous iterations, we have found it more useful and convenient to simply share the plant models in a plain format, such as MATLAB functions, where the participants could easily extract the ODEs. As for the neural network models, we provide them in the ONNX format[3], .mat format[4], and the original format used by the proposer of the benchmark. A few years ago, we began providing the neural network controllers in the ONNX format, as it was very convenient to have a standard exchange format that most of the participating tools supported. However, we have found that there are still discrepancies among the different versions and frameworks these ONNX models were created from (e.g., different input/output transformations are not always supported by every framework as experienced on the *Docking spacecraft* benchmark). Thus, having a standard format easily imported by all participants without local modifications, such as a unified ONNX version, remains a challenge. Initiatives more focused on neural network verification, such as VNN-LIB[5] and VNN-COMP[6], may help toward this goal.

6 Conclusion and Outlook

This report presents the results of the fifth ARCH friendly competition for closed-loop systems with neural network controllers. For this edition, three tools have participated and attempted to solve 10 benchmarks: CORA, JuliaReach, and NNV. The problems elucidated in this paper are challenging and diverse; the presented results probably provide the most complete assessment of current tools for the safety verification in AINNCS. The report provides a good overview of the intellectual progression of this rapidly growing field, and it is our hope to stimulate the development of efficient and effective methods capable of use in real-world applications. In the past five years, the complexity of the benchmarks has consistently increased along with the capabilities of the participant tools, leading to the most challenging competition (equal to 2022 competition) and the best verification results thus far, which is a good indicator for this growing and maturing field. This has been achieved thanks to the continuous development and improvements in existing formal verification frameworks, including CORA, JuliaReach, and NNV. We would also like to encourage other tool developers to consider participating next year, as well as new benchmark proposals are highly welcome. Authors agree that although participation consumes time, we have gained unique insights that have allowed us to improve in each iteration and will allow us to improve in the future. The reports of other categories can be found in the proceedings and on the ARCH website: cps-vo.org/group/ARCH.

[3]Open Neural Network Exchange: `https://github.com/onnx/onnx`

[4]Direct input format used by *NNV* without transformation.

[5]`http://www.vnnlib.org/`

[6]`https://github.com/verivital/vnn-comp/`

7 Acknowledgments

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant numbers 1910017, 2028001, 2220418, 2220426 and 2220401, the Defense Advanced Research Projects Agency (DARPA) under contract numbers FA8750-23-C-0518 and FA8750-18-C-0089, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-22-1-0019 and FA9550-23-1-0135. Christian Schilling acknowledges the support from DIREC - Digital Research Centre Denmark and the Villum Investigator Grant S4OS. Tobias Ladner gratefully acknowledges financial support from the project FAI funded by the German Research Foundation (DFG) under project number 286525601. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of AFOSR, DARPA, DIREC, DFG, or NSF.

A Specification of Used Machines

This year, we run all tools on the same hardware using tool-specific docker images. The specification of the server used for the evaluation is given below:

- Processor: AMD EPYC 7742 64-Core
- Memory: 995 GB
- OS: Ubuntu 22.04
- Docker: 20.10.21

References

[1] Michael E. Akintunde, Elena Botoeva, Panagiotis Kouvaros, and Alessio Lomuscio. Formal verification of neural agents in non-deterministic environments. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pages 25–33, 2020.

[2] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015.

[3] M. Althoff and D. Grebenyuk. Implementation of interval arithmetic in CORA 2016. In *Proc. of the 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 91–105, 2016.

[4] M. Althoff, D. Grebenyuk, and N. Kochdumper. Implementation of Taylor models in CORA 2018. In *Proc. of the 5th International Workshop on Applied Verification for Continuous and Hybrid Systems*, 2018.

[5] M. Althoff, M. Koschi, and S. Manzinger. CommonRoad: Composable benchmarks for motion planning on roads. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 719–726, 2017.

[6] S. Bak, S. Bogomolov, T. A. Henzinger, T. T. Johnson, and P. Prakash. Scalable static hybridization methods for analysis of nonlinear systems. In *Proc. of the 19th ACM International Conference on Hybrid Systems: Computation and Control*, pages 155–164, 2016.

[7] Stanley Bak. nnenum: Verification of relu neural networks with optimized abstraction refinement. In *NASA Formal Methods Symposium*, pages 19–36. Springer, 2021.

[8] Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. Hyst: A source transformation and translation tool for hybrid automaton models. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, pages 128–133, New York, NY, USA, 2015. ACM.

[9] Stanley Bak, Changliu Liu, and Taylor Johnson. The second international verification of neural networks competition (vnn-comp 2021): Summary and results. 2021.

[10] Randal W. Beard. Quadrotor dynamics and control. Technical report, 2008.

[11] Luis Benet, Marcelo Forets, David P. Sanders, and Christian Schilling. TaylorModels.jl: Taylor models in Julia and its application to validated solutions of ODEs. In *SWIM*, 2019.

[12] Luis Benet and David P. Sanders. TaylorSeries.jl: Taylor expansions in one and several variables in Julia. *Journal of Open Source Software*, 4(36):1043, 2019.

[13] Luis Benet and David P. Sanders. JuliaDiff/TaylorSeries.jl. `https://github.com/JuliaDiff/TaylorSeries.jl`, 2021.

[14] Luis Benet and David P. Sanders. JuliaIntervals/TaylorModels.jl. `https://github.com/JuliaIntervals/TaylorModels.jl`, 2021.

[15] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin, and Christian Schilling. JuliaReach: a toolbox for set-based reachability. In *HSCC*, pages 39–44. ACM, 2019.

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

[17] W. H. CLOHESSY and R. S. WILTSHIRE. Terminal guidance system for satellite rendezvous. *Journal of the Aerospace Sciences*, 27(9):653–658, 1960.

[18] Souradeep Dutta, Xin Chen, and Sriram Sankaranarayanan. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019.*, pages 157–168, 2019.

[19] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Learning and verification of feedback control systems using feedforward neural networks. *IFAC-PapersOnLine*, 51(16):151 – 156, 2018. 6th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2018.

[20] Michael Everett, Golnaz Habibi, and Jonathan P. How. Efficient reachability analysis of closed-loop systems with neural network controllers. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4384–4390, 2021.

[21] Jiameng Fan, Chao Huang, Wenchao Li, Xin Chen, and Qi Zhu. ReachNN*: A tool for reachability analysis ofneural-network controlled systems. In *to appear on International Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2020.

[22] James Ferlez, Haitham Khedr, and Yasser Shoukry. Fast BATLLNN: fast box analysis of two-level lattice neural networks. In Ezio Bartocci and Sylvie Putot, editors, *HSCC '22: 25th ACM International Conference on Hybrid Systems: Computation and Control, Milan, Italy, May 4 - 6, 2022*, pages 23:1–23:11. ACM, 2022.

[23] Marc Fischer, Christian Sprecher, Dimitar Iliev Dimitrov, Gagandeep Singh, and Martin Vechev. Shared certificates for neural network verification. In Sharon Shoham and Yakir Vizel, editors, *Computer Aided Verification*, pages 127–148, Cham, 2022. Springer International Publishing.

[24] Marcelo Forets and Christian Schilling. LazySets.jl: Scalable symbolic-numeric set computations. *Proceedings of the JuliaCon Conferences*, 1(1):11, 2021.

[25] Eric Goubault and Sylvie Putot. Rino: Robust inner and outer approximated reachability of neural networks controlled systems. In Sharon Shoham and Yakir Vizel, editors, *Computer Aided Verification*, pages 511–523, Cham, 2022. Springer International Publishing.

[26] G. W. Hill. Researches in the lunar theory. *American Journal of Mathematics*, 1(1):5–26, 1878.

[27] Chao Huang, Jiameng Fan, Xin Chen, Wenchao Li, and Qi Zhu. POLAR: A polynomial arithmetic framework for verifying neural-network controlled systems. In *To appear on International Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2022.

[28] Radoslav Ivanov, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. *CoRR*, abs/1811.01828, 2018.

[29] M. Jankovic, D. Fontaine, and P. V. Kokotovic. Tora example: cascade- and passivity-based control designs. *IEEE Transactions on Control Systems Technology*, 4(3):292–297, May 1996.

[30] Taylor T. Johnson, Diego Manzanas Lopez, Luis Benet, Marcelo Forets, Sebasti\'an Guadalupe, Christian Schilling, Radoslav Ivanov, Taylor J. Carpenter, James Weimer, and Insup Lee. Arch-comp21 category report: Artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. In Goran Frehse and Matthias Althoff, editors, *8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)*, volume 80 of *EPiC Series in Computing*, pages 90–119. EasyChair, 2021.

[31] Taylor T Johnson, Diego Manzanas Lopez, Patrick Musau, Hoang-Dung Tran, Elena Botoeva, Francesco Leofante, Amir Maleki, Chelsea Sidrane, Jiameng Fan, and Chao Huang. Arch-comp20 category report: Artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. In Goran Frehse and Matthias Althoff, editors, *ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*, volume 74 of *EPiC Series in Computing*, pages 107–139. EasyChair, 2020.

[32] K. D. Julian and M. J. Kochenderfer. A reachability method for verifying dynamical systems with deep neural network controllers. *CoRR*, abs/1903.00520, 2019.

[33] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In Rupak Majumdar and Viktor Kunčak, editors, *Computer Aided Verification*, pages 97–117, Cham, 2017. Springer International Publishing.

[34] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. The marabou framework for verification and analysis of deep neural networks. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification*, pages 443–452, Cham, 2019. Springer International Publishing.

[35] Niklas Kochdumper and Matthias Althoff. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *IEEE Transactions on Automatic Control*, 66(9):4043–4058, 2020.

[36] Niklas Kochdumper, Christian Schilling, Matthias Althoff, and Stanley Bak. Open-and closed-loop neural network verification using polynomial zonotopes. In *NASA Formal Methods Symposium*, pages 16–36. Springer, 2023.

[37] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[38] Tobias Ladner and Matthias Althoff. Automatic abstraction refinement in neural network verification using sensitivity analysis. *HSCC'23: Proceedings of the 26th International Conference on Hybrid Systems: Computation and Control*, 2023.

[39] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, and Mykel J. Kochenderfer. Algorithms for verifying deep neural networks. *Foundations and Trends in Optimization*, 4(3-4):244–404, 2021.

[40] Diego Manzanas Lopez, Matthias Althoff, Luis Benet, Xin Chen, Jiameng Fan, Marcelo Forets, Chao Huang, Taylor T Johnson, Tobias Ladner, Wenchao Li, Christian Schilling, and Qi Zhu. Arch-comp22 category report: Artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. In Goran Frehse, Matthias Althoff, Erwin Schoitsch, and Jeremie Guiochet, editors, *Proceedings of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH22)*, volume 90 of *EPiC Series in Computing*, pages 142–184. EasyChair, 2022.

[41] Diego Manzanas Lopez, Sung Woo Choi, Hoang-Dung Tran, and Taylor T. Johnson. NNV 2.0: The neural network verification tool. In *35th International Conference on Computer-Aided Verification (CAV)*, July 2023.

[42] Diego Manzanas Lopez, Patrick Musau, Hoang-Dung Tran, Souradeep Dutta, Taylor J. Carpenter, Radoslav Ivanov, and Taylor T. Johnson. Arch-comp19 category report: Artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. In Goran Frehse and Matthias Althoff, editors, *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 61 of *EPiC Series in Computing*, pages 103–119. EasyChair, 2019.

[43] Amir Maleki and Chelsea Sindrane. Benchmark examples for ainncs-2020, 2020.

[44] 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*, 31(1):1–17, 2023.

[45] Diego Manzanas Lopez, Patrick Musau, Nathaniel Hamilton, and Taylor Johnson. Reachability analysis of a general class of neural ordinary differential equation. In *Proceedings of the 20th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2022), Co-Located with CONCUR, FMICS, and QEST as part of CONFEST 2022.*, Warsaw, Poland, September 2022.

[46] MathWorks. *Adaptive Cruise Control System* block. https://www.mathworks.com/help/mpc/ref/adaptivecruisecontrolsystem.html, 2018.

[47] Mark Niklas Müller, Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T. Johnson. The third international verification of neural networks competition (vnn-comp 2022): Summary and results, 2022.

[48] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, March 1990.

[49] Jorge A. Pérez-Hernández and Luis Benet. PerezHz/TaylorIntegration.jl. https://github.com/PerezHz/TaylorIntegration.jl, 2021.

[50] S. Prajna, P.A. Parrilo, and A. Rantzer. Nonlinear control synthesis by convex optimization. volume 49, pages 310–314, 2004.

[51] S. Joe Qin and Thomas A. Badgwell. An overview of nonlinear model predictive control applications. In Frank Allgöwer and Alex Zheng, editors, *Nonlinear Model Predictive Control*, pages 369–392, Basel, 2000. Birkhäuser Basel.

[52] Umberto J. Ravaioli, James Cunningham, John McCarroll, Vardaan Gangal, Kyle Dunlap, and Kerianne L. Hobbs. Safe reinforcement learning benchmark environments for aerospace control systems. In *2022 IEEE Aerospace Conference (AERO)*, pages 1–20, 2022.

[53] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[54] Stuart Russell, Daniel Dewey, and Max Tegmark. Research Priorities for Robust and Beneficial Artificial Intelligence. *AI Magazine*, 36(4):105, 2015.

[55] Christian Schilling, Marcelo Forets, and Sebastián Guadalupe. Verification of neural-network control systems by integrating Taylor models and zonotopes. In *AAAI*, pages 8169–8177. AAAI Press, 2022.

[56] Malcolm D. Shuster. Survey of attitude representations. *Journal of the Astronautical Sciences*, 41(4):439–517, October 1993.

[57] Chelsea Sidrane and Mykel J. Kochenderfer. OVERT: Verification of nonlinear dynamical systems with neural network controllers via overapproximation. *Safe Machine Learning workshop at ICLR*, 2019.

[58] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. Fast and effective robustness certification. In *NeurIPS*, pages 10825–10836, 2018.

[59] Peter Stone, Rodney Brooks, Erik Brynjolfsson, Ryan Calo, Oren Etzioni, Greg Hager, Julia Hirschberg, Shivaram Kalyanakrishnan, Ece Kamar, Kevin Leyton-Brown, David C. Parkes, William Press, AnnaLee Saxenian, Julie Shah, Milind Tambe, and Astro Teller. "artificial intelligence and life in 2030." one hundred year study on artificial intelligence: Report of the 2015-2016 study panel, 2016.

[60] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[61] Hoang-Dung Tran, Stanley Bak, Weiming Xiang, and Taylor T. Johnson. Verification of deep convolutional neural networks using imagestars. In *32nd International Conference on Computer-Aided Verification (CAV)*. Springer, July 2020.

[62] Hoang-Dung Tran, Feiyang Cai, Manzanas Lopez Diego, Patrick Musau, Taylor T. Johnson, and Xenofon Koutsoukos. Safety verification of cyber-physical systems with reinforcement learning control. *ACM Trans. Embed. Comput. Syst.*, 18(5s), October 2019.

[63] Hoang Dung Tran, SungWoo Choi, Tomoya Yamaguchi, Bardh Hoxha, and Danil Prokhorov. Verification of recurrent neural networks using star reachability. In *The 26th ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, May 2023.

[64] Hoang-Dung Tran, Diago Manzanas Lopez, Patrick Musau, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T. Johnson. Star-based reachability analysis of deep neural networks. In Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira, editors, *Formal Methods – The Next 30 Years*, pages 670–686. Springer International Publishing, 2019.

[65] Hoang-Dung Tran, Neelanjana Pal, Patrick Musau, Xiaodong Yang, Nathaniel P. Hamilton, Diego Manzanas Lopez, Stanley Bak, and Taylor T. Johnson. Robustness verification of semantic segmentation neural networks using relaxed reachability. In *33rd International Conference on Computer-Aided Verification (CAV)*. Springer, July 2021.

[66] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T. Johnson. NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *32nd International Conference on Computer-Aided Verification (CAV)*, July 2020.

[67] Weiming Xiang, Patrick Musau, Ayana A. Wild, Diego Manzanas Lopez, Nathaniel Hamilton, Xiaodong Yang, Joel A. Rosenfeld, and Taylor T. Johnson. Verification for machine learning, autonomy, and neural networks survey. *CoRR*, abs/1810.01989, 2018.