Verification of Neural-Network Control Systems by Integrating Taylor Models and Zonotopes

Christian Schilling,[1] **Marcelo Forets,**[2] **Sebastián Guadalupe**[2]
[1] Aalborg University, Denmark
[2] Universidad de la República, Uruguay
christianms@cs.aau.dk, mforets@gmail.com, sebastianguadalupe00@gmail.com

Abstract

We study the verification problem for closed-loop dynamical systems with neural-network controllers (NNCS). This problem is commonly reduced to computing the set of reachable states. When considering dynamical systems and neural networks in isolation, there exist precise approaches for that task based on set representations respectively called Taylor models and zonotopes. However, the combination of these approaches to NNCS is non-trivial because, when converting between the set representations, dependency information gets lost in each control cycle and the accumulated approximation error quickly renders the result useless. We present an algorithm to chain approaches based on Taylor models and zonotopes, yielding a precise reachability algorithm for NNCS. Because the algorithm only acts at the interface of the isolated approaches, it is applicable to general dynamical systems and neural networks and can benefit from future advances in these areas. Our implementation delivers state-of-the-art performance and is the first to successfully analyze all benchmark problems of an annual reachability competition for NNCS.

Introduction

In this work we consider controlled dynamical systems where the plant model is given as a nonlinear ordinary differential equation (ODE) and the controller is implemented by a neural network. We call such systems *neural-network control systems* (NNCS). We are interested in reachability properties of NNCS: guaranteed reachability of target states or non-reachability of error states. These questions can be verified by computing a set that overapproximates the reachable states, which is the subject of reachability analysis, with a large body of works for ODEs (Althoff, Frehse, and Girard 2021) and neural networks (Liu et al. 2021).

In principle, reachability analysis for NNCS can be implemented by chaining two off-the-shelf tools for analyzing the ODE and the neural network. The output set of one tool is the input set to the other, and this process is repeated for each control cycle. This idea is indeed applied by several approaches (Tran et al. 2020b; Clavière et al. 2021). While correct, such an approach often yields sets that are too conservative to be useful in practice. The reason is that with each switch to the other tool, a conversion between set representations is required because the tools use different techniques. Thus some of the dependency information encoded in the sets is lost when the tools are treated as black boxes. This incurs an approximation error that quickly accumulates over time, also known as the wrapping effect (Neumaier 1993).

Reachability algorithms at a sweet spot between precision and performance in the literature are based on Taylor models for ODEs (Makino and Berz 2003; Chen, Ábrahám, and Sankaranarayanan 2012) and on set propagation via abstract interpretation (Cousot and Cousot 1977) for neural networks, particularly using zonotopes (Gehr et al. 2018; Singh et al. 2018). In this work we propose a new reachability algorithm for NNCS that combines Taylor models and zonotopes. In general, Taylor models and zonotopes are incomparable and cannot be converted exactly. We describe how to tame the approximation error when converting between these two set representations with two main insights. First, we identify a special structured zonotope, which can be exactly converted to a Taylor model by encoding the additional structure in the so-called remainder. Second, the structure of the Taylor model from the previous cycle can be retained by only updating the control inputs, which allows to preserve the dependencies encoded in the Taylor model.

Our approach only acts at the set interface and does not require access to the internals of the reachability tools. They only need to expose the complete set information, which is only a minor modification of the black-box algorithms. Thus our approach makes no assumptions about the ODE or the neural network, as long as there are sound algorithms for their (almost black-box) reachability analysis available. This makes the approach a universal tool. While our approach is conceptually simple, we demonstrate in our evaluation that it is effective and scalable in practice. We successfully analyzed all benchmark problems from an annual NNCS competition (Johnson et al. 2021) for the first time.

In summary, this paper makes the following contributions:

- We propose structured zonotopes and show how to soundly convert them to Taylor models and back.
- We design a sound reachability algorithm for NNCS based on Taylor models and zonotopes.
- We demonstrate the precision and scalability of the algorithm on benchmarks from a reachability competition.

Related Work

The verification of continuous-time NNCS has recently received attention. The tool *Verisig* (Ivanov et al. 2019) transforms a neural network with sigmoid activation functions into a hybrid automaton (Alur et al. 1992) and then uses *Flow** (Chen, Ábrahám, and Sankaranarayanan 2013), a reachability tool for nonlinear ODEs based on Taylor models, to analyze the transformed control system as a chain of hybrid automata. While that approach allows to preserve dependencies in the Taylor model, it is not applicable to the common ReLU activation functions and the automaton's dimension scales with the number of neurons. The tool *NNV* (Tran et al. 2020b) combines *CORA* (Althoff 2015), a reachability tool for nonlinear ODEs based on (variants of) zonotopes, and an algorithm based on star sets (Tran et al. 2020a) for propagating through a ReLU neural network. That approach suffers from the loss of dependencies when switching between the set representations. The tool *Sherlock* (Dutta, Chen, and Sankaranarayanan 2019; Dutta et al. 2019) combines *Flow** with an output-range analysis for ReLU neural networks (Dutta et al. 2018b). That approach abstracts the neural network by a polynomial, which has the advantage that dependencies can in principle be preserved. While the approach requires hyperrectangular input sets and the abstraction comes with its own error, this approach can be precise in practice for small input sets. The input to the neural network as well as the polynomial order must be small in practice for scalability reasons. Further, the approach only works well for neural networks with a single output; for multiple outputs, the analysis has to be applied iteratively. The tool *ReachNN** (Huang et al. 2019; Fan et al. 2020) approximates Lipschitz-continuous neural networks with Bernstein polynomials and then analyzes the resulting polynomial system with *Flow**; estimates of the Lipschitz constant tend to be conservative. Clavière et al. (2021) combine validated simulations and abstract interpretation.

A number of approaches consider discrete-time systems, which are considerably easier to handle. Xiang et al. (2018) study the simple case of discrete-time piecewise-linear (PWL) systems and controllers with ReLU activation functions, for which one can represent the exact reachable states as a union of convex polytopes. However, the number of polytopes may grow exponentially in the dimension of the neural network. *VenMAS* (Akintunde et al. 2020) also assumes PWL dynamics and considers a multi-agent setting with a temporal-logic specification. Dutta et al. (2018a) consider nonlinear dynamics and compute a template polyhedron that overapproximates the output of the neural network based on range analysis. *OVERT* (Sidrane et al. 2021) approximates nonlinear dynamics by PWL bounds. Bacci, Giacobbe, and Parker (2021) consider unbounded time.

Outline In the next section we continue with the background on NNCS and set representations. Afterward we describe our approach and evaluate it, and finally we conclude.

Preliminaries

We formally introduce NNCS and the core set representations used in our approach: Taylor models and zonotopes.

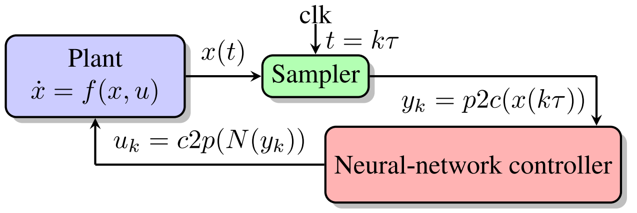


Figure 1: Neural-network control system.

Neural-Network Control Systems

We consider plants modeled by ODEs $\dot{x} = f(x, u)$ where $x \in \mathbb{R}^n$ is the state vector and $u \in \mathbb{R}^m$ is the vector of control inputs. Given an initial state $x(0) = x_0$ and a (constant) control input u_0, we assume that the solution of the corresponding initial-value problem at time $t \geq 0$, denoted by $\xi(t, x_0, u_0, f)$, exists and is unique. (We only discuss deterministic plants here to simplify the presentation. The extension to nondeterministic systems is straightforward; handling such systems is common in the reachability literature (Singer and Barton 2006; Althoff, Frehse, and Girard 2021) and orthogonal to the problem described in this work.)

A neural-network control system (NNCS) is a tuple $(f, N, p2c, c2p, \tau)$ with a plant $f(x, u)$ over $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$, a controller given as a neural network $N : \mathbb{R}^i \to \mathbb{R}^o$, a function $p2c : \mathbb{R}^n \to \mathbb{R}^i$ that takes the current plant state x and turns it into the input to the controller, a function $c2p : \mathbb{R}^o \to \mathbb{R}^m$ that takes the controller output and turns it into the new control input u, and a control period $\tau \in \mathbb{R}_{>0}$.

A conceptual sketch of an NNCS is given in Figure 1. The NNCS periodically queries the controller for new control inputs. At time points $k\tau$, $k \in \mathbb{N}$, the state $x(k\tau)$ is passed to $p2c$, to the controller N, and to $c2p$, which yields the new control inputs u_k. Here we use the common assumption that the computation of u_k is instantaneous. The sequence of the u_k induces a continuous piecewise input signal $u(t)$. Formally, given an initial state x_0 at $t = 0$, we recursively define the sequence of input vectors u_k, $k \in \mathbb{N}$, and the evolution of the state $x(t)$, $t \geq 0$, which is a trajectory of the NNCS:

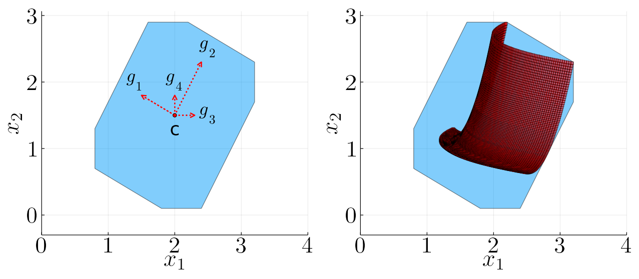
$$u_k = c2p(N(p2c(x(k\tau)))) \tag{1}$$

$$x(t) = \begin{cases} x_0 & t = 0 \\ \xi(t - k\tau, x(k\tau), u_k, f) & t \in (k\tau, (k+1)\tau] \end{cases} \tag{2}$$

We may also write $x(t, x_0)$ resp. $u_k(x_0)$ for the trajectory $x(t)$ resp. for the vector u_k to clarify the dependency on x_0.

Taylor Models

A d-dimensional Taylor model of order k is a tuple $\mathcal{T} = (p, \Delta, \mathcal{D})$ where $p = (p_1, \dots, p_d)^T$ is a vector of multivariate polynomials $p_i : \mathcal{D} \to \mathbb{R}$ of degree at most k, $i = 1, \dots, d$, the remainder $\Delta = \Delta_1 \times \cdots \times \Delta_d$ is a hyperrectangle containing the origin, and $\mathcal{D} \subseteq \mathbb{R}^d$ is the domain (Makino and Berz 2003). Thus $\mathcal{T}$ represents the vector-valued function $p(x) + \Delta$: an interval tube around the polynomial p. We often use the common normalization $\mathcal{D} = [-1, 1]^d$, which can be established algorithmically.



(a) Zonotope. (b) Taylor model.

Figure 2: A structured zonotope (left) and a Taylor model covered with one zonotope and with a union of boxes (right).

Example The tuple $(p, \Delta, \mathcal{D})$ with $p_1(x) = -0.5x^2 + 3x$ over domain $\mathcal{D} = [-1, 1]$ and remainder $\Delta = [-0.1, 0.1]$ is a one-dimensional Taylor model of order 2.

Taylor-Model Reach Sets (TMRS)

A Taylor-model reach set (TMRS) $\mathcal{R}$ is a structure used in reachability algorithms when propagating Taylor models through an ODE in time. For a d-dimensional system, a TMRS is a d-vector of Taylor models in one variable representing time with shared domain. The coefficients of these Taylor models are themselves multivariate polynomials in the d state variables, whose domain is assumed to be the symmetric box $[-1, 1]^d$. (The time domain of a TMRS is not normalized to $[-1, 1]$.) Evaluating a TMRS over a time point (or a time interval) yields a d-dimensional Taylor model.

Example Continuing the previous example, consider the one-dimensional TMRS consisting of the Taylor model $(q, \Delta, [0, 1])$ where $q_1(t) = (-0.5x^2 + 2x)t + x$. Evaluation at $t = 1$ yields the Taylor model from the previous example.

Zonotopes

A zonotope is the image of a hypercube under an affine transformation and hence a convex centrally-symmetric polytope (Ziegler 1995). Zonotopes are usually characterized in generator representation: An n-dimensional zonotope $\mathcal{Z} \subseteq \mathbb{R}^n$ with center $c \in \mathbb{R}^n$ and p generators $g_j \in \mathbb{R}^n$ $(j = 1, \ldots, p)$ is defined as

$$\mathcal{Z} = \left\{ x \in \mathbb{R}^n : x = c + \sum_{j=1}^{p} \zeta_j g_j \ , \ \zeta_j \in [-1, 1] \right\}.$$

The generators are commonly aligned as columns in a matrix $G_\mathcal{Z} = [g_1 \quad g_2 \quad \cdots \quad g_p]$. The order of $\mathcal{Z}$ is the ratio p/n of generators per dimension. In the special case that $G_\mathcal{Z}$ is a diagonal matrix, the zonotope represents a hyperrectangle.

Given two sets $\mathcal{X}_1, \mathcal{X}_2 \subseteq \mathbb{R}^n$, their Minkowski sum is

$$\mathcal{X}_1 \oplus \mathcal{X}_2 = \{x_1 + x_2 : x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2\}$$

We say that a zonotope $\mathcal{Z}$ with center c and generator matrix $G_\mathcal{Z}$ is *structured* if it has order 2 and $G_\mathcal{Z}$ has the block structure $[M \quad D]$, where D is a diagonal matrix. A structured zonotope corresponds to the Minkowski sum of 1) the zonotope centered in c with generator matrix M and 2) the hyperrectangle centered in the origin whose radius corresponds to the diagonal of D.

Example The two-dimensional structured zonotope with the center $c = (2, 1.5)^T$ and the generator matrix $\begin{pmatrix} -0.5 & 0.4 & 0.3 & 0 \\ 0.3 & 0.8 & 0 & 0.3 \end{pmatrix}$ is depicted in Figure 2(a).

Propagating zonotopes through a neural network Abstract interpretation (Cousot and Cousot 1977) is a well-known technique to propagate sets through a system in a sound (i.e., overapproximate) way. The sets are taken from a class called the abstract domain. The idea is to compute the image of the set under the system's successor function; if the image does not fall into the abstract domain, an overapproximation from that domain is chosen. For neural networks the idea is to iteratively propagate a set through each layer. For instance, several algorithms based on the zonotope abstract domain (Ghorbal, Goubault, and Putot 2009) for propagation through neural networks have been proposed (Gehr et al. 2018; Singh et al. 2018). Given an input zonotope, the algorithm outputs a zonotope that overapproximates the exact image of the neural network. The smallest zonotope overapproximation is not unique. Zonotopes are efficient to manipulate and closed under the affine map in each layer (multiplication with the weights and addition of the bias); only the activation function requires an overapproximation.

Reachability Algorithm

In this section we formalize the reachability problem for NNCS, explain how one can convert between (structured) zonotopes and Taylor models in a sound way, and finally integrate these conversions into a reachability algorithm.

Problem Statement

Given an NNCS, a set of initial states $\mathcal{X}_0 \subseteq \mathbb{R}^n$, and another set of states $\mathcal{Y} \subseteq \mathbb{R}^n$, we are interested in answering two types of questions: The *must-not-reach* question asks whether no trajectory reaches any state in $\mathcal{Y}$. The *must-reach* question asks whether each trajectory reaches some state in $\mathcal{Y}$. To answer these questions, we aim at computing the reachable states $\mathcal{S}_t = \{x(t, x_0) : x_0 \in \mathcal{X}_0\}$ at time t. Determining reachability of a state (i.e., membership in $\mathcal{S}_t$) is undecidable, which follows from undecidability of reachability for nonlinear dynamical systems (Hainry 2008).

The common approach in the literature is to consider the reachable states for time intervals $[T_0, T_1]$, $\mathcal{S}_{[T_0,T_1]} = \bigcup_{t\in[T_0,T_1]} \mathcal{S}_t$, and to find a coverage $\overline{\mathcal{S}}_{[T_0,T_1]} \supseteq \mathcal{S}_{[T_0,T_1]}$. This allows to give one-sided guarantees: if $\overline{\mathcal{S}}_{[0,T]} \cap \mathcal{Y} = \emptyset$ for some time horizon T, we can affirmatively answer a *must-not-reach* question, and if $\overline{\mathcal{S}}_{[T_0,T_1]} \subseteq \mathcal{Y}$ for some time interval $[T_0, T_1]$, we can affirmatively answer a *must-reach* question. The task we aim to solve is thus, given a time horizon $T \geq 0$, to compute a covering $\overline{\mathcal{S}}_{[0,T]} \supseteq \mathcal{S}_{[0,T]}$.

Computing Sequences of Covering Sets

Given a set of initial states $\mathcal{X}_0 \subseteq \mathbb{R}^n$ and a bound on the number of control periods $K \in \mathbb{N}$, for each $k = 0, \dots, K$ we want to cover the set of control inputs $\mathcal{U}_k = \{u_k(x_0) : x_0 \in \mathcal{X}_0\}$ by a set $\mathcal{Z}_k \supseteq \mathcal{U}_k$. Similarly, we want to compute sets $\mathcal{R}_k \supseteq \mathcal{S}_{[k\tau,(k+1)\tau]}$. The idea is to represent the sets $\mathcal{Z}_k$ as structured zonotopes and the sets $\mathcal{R}_k$ as TMRS. From the recursive definition in (1) and (2), for each control cycle we shall evaluate the TMRS, obtaining a Taylor model, then convert to a zonotope and back, and finally compute a new TMRS. Exact conversion between Taylor models and zonotopes is generally not possible, so we must overapproximate.

To simplify the presentation, we turn the input variables into new state variables by adding m fresh state variables with zero dynamics. Thus from now on we sometimes assume a state vector of dimension $n + m$.

From Taylor Model to Structured Zonotope

Given an n-dimensional Taylor model $\mathcal{T}$, we want to compute a covering zonotope. (Since a Taylor model can represent nonlinear dependencies and a zonotope only consists of linear constraints, one cannot hope for an exact conversion.) We construct a structured zonotope $\mathcal{Z}$ as the Minkowski sum $\mathcal{Z}_l \oplus \mathcal{H}_{nl}$ of a zonotope and a hyperrectangle. The intuition is that $\mathcal{Z}_l$ exactly captures the linear part of the polynomial and $\mathcal{H}_{nl}$ overapproximates the nonlinear part and the remainder.

We split the Taylor model's polynomial vector into the linear part $p_l = Ax + b$ (for some $A \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n$) and the nonlinear part p_{nl}. The zonotope $\mathcal{Z}_l$ has the center b corresponding to the constant term of p_l and the generator matrix A corresponding to the linear coefficients. Recall that the domain of p_l is normalized to $[-1, 1]^n$, so it conforms with the zonotope's definition. Then we compute the interval approximation $\mathcal{H}'$ of p_{nl} by evaluating the polynomials over the domain using interval arithmetic. Finally we define $\mathcal{H}_{nl} = \mathcal{H}' \oplus \Delta$, where Δ is the remainder of $\mathcal{T}$.

Example Let the Taylor model $\mathcal{T} = (p, [0, 0]^2, [-1, 1]^2)$ with polynomials $p_1(x) = 0.6x_1^2 - 0.5x_1 + 0.4x_2 + 1.7$ and $p_2(x) = 0.6x_2^2 + 0.3x_1 + 0.8x_2 + 1.2$. We obtain the zonotope $\mathcal{Z}$ from the previous example. Figure 2(b) shows $\mathcal{Z}$ together with a multi-box cover of $\mathcal{T}$, for which we split the domain into 10,000 uniform boxes and evaluate $\mathcal{T}$ using interval arithmetic. Note that $\mathcal{Z}$ is tight at multiple edges.

From Structured Zonotope to Taylor Model

Reachability algorithms for nonlinear dynamical systems based on Taylor models assume that the set of initial states $\mathcal{X}_0$ itself is given as a Taylor model. Converting hyperrectangles to a Taylor model is easy. Hence the typical approach is to first overapproximate $\mathcal{X}_0$ with a hyperrectangle. However, this way we lose all dependencies between variables. If we were to apply this conversion in each control cycle, the approximation error would quickly explode. We describe a better approximation when $\mathcal{X}_0$ is a zonotope. Zonotopes of order > 1 generally cannot be converted exactly to a Taylor model (see (Kochdumper and Althoff 2021, Corollary 1) applied to zonotopes). Here we show that for structured zonotopes (which have order 2) an exact conversion is possible.

Say we are given a structured zonotope $\mathcal{Z} \subseteq \mathbb{R}^m$ (the new control inputs). We construct the Taylor model $\mathcal{T} = (p, \Delta, [-1, 1]^m)$ corresponding to $\mathcal{Z}$, for which it remains to describe how to construct each p_j and Δ_j, $j = 1, \dots, m$.

First we explain the construction in the simpler case that $\mathcal{Z}$ is a hyperrectangle $\mathcal{H}$ with center c and radius r. Then each p_j is the constant polynomial c_j, where c_j is the j-th component of c, with the remainder $\Delta_j = [-r_j, r_j]$ (i.e., Δ is the hyperrectangle $\mathcal{H}$ with the center shifted to the origin).

Now we explain the construction in the case that $\mathcal{Z}$ is a structured zonotope with generator matrix $G_{\mathcal{Z}} = [M \quad D]$. Let $A_{i,j}$ denote the entry of matrix A at row i and column j. We define the polynomial $p_j(x) = c_j + \sum_{k=1}^{m} M_{j,k}\, x_k$ (which is correct because we use the domain $[-1, 1]^m$) and the remainder $\Delta_j = [-d, d]$ where $d = |D_{j,j}|$.

Observe that this conversion is exact and compatible with the other conversion algorithm. Let $\mathcal{Z}$ be a structured zonotope. Then converting to a Taylor model and back using our algorithms yields $\mathcal{Z}$ again.

Example We continue with the structured zonotope $\mathcal{Z}$ from the previous example. The corresponding Taylor model is $p_1(x) = 2.0 - 0.5x_1 + 0.4x_2$ and $p_2(x) = 1.5 + 0.3x_1 + 0.8x_2 + [-0.3, 0.3]$, with $\Delta_1 = \Delta_2 = [-0.3, 0.3]$.

Reachability Algorithm Based on Taylor Models and Zonotopes

Now we have all ingredients to formulate a reachability algorithm. Initially we are given an NNCS $(f, N, p2c, c2p, \tau)$ as defined before, a set of initial states $\mathcal{X}_0 \subseteq \mathbb{R}^n$, and a time horizon $T \in \mathbb{R}_{>0}$. If $\mathcal{X}_0$ is not already given as a Taylor model, we need to convert (e.g., from a structured zonotope) or overapproximate it first. We assume two black-box reachability algorithms ρ_N and ρ_f. Algorithm ρ_N receives a zonotope $\mathcal{Z}$ and produces another zonotope that covers the image of $\mathcal{Z}$ under the controller N, e.g., implementing the algorithm in (Singh et al. 2018). Algorithm ρ_f receives a Taylor model and a time horizon and produces a TMRS covering the reachable states of the plant f, e.g., implementing the algorithm in (Makino and Berz 2003).

Algorithm 1 consists of a loop of four main steps. We assume that the time horizon T is a multiple K of the period τ. (For other time horizons one can just execute the loop body once more with a shortened time frame in line 7.) Say we are in iteration k. The first step is to obtain the control inputs for the next time period from the controller. According to (1), we need to extract the current state information, which is stored as part of a Taylor model $\mathcal{T}$. We obtain $\mathcal{T}$ by evaluating the current TMRS at $t = k\tau$ (lines 8 and 9). Then we convert $\mathcal{T}$ to a structured zonotope $\mathcal{Z}'$ using our algorithm described above (line 3). Finally we apply the function $p2c$ and pass the set to the second step. In typical cases such as affine maps ($p2c(x) = Ax + b$), we can apply $p2c$ directly to $\mathcal{Z}'$. In more complicated cases we could instead apply $p2c$ to the Taylor model before the conversion.

The second step is to propagate $\mathcal{Z}'$ through the controller via ρ_N (line 4). The output is a new zonotope $\mathcal{Z}$. Then we apply the function $c2p$ to it; again this is easy for affine maps, and otherwise we need to overapproximate.

Algorithm 1: Reachability algorithm for NNCS.

```
Input: $(f, N, p2c, c2p, \tau)$: NNCS; $\mathcal{X}_0$: initial states;
       $T = K\tau$: time horizon; $\rho_N$: reachability algorithm for $N$; $\rho_f$: reachability algorithm for $f$
Output: TMRS overapproximating the reachable states until $T$

1  $\mathcal{T}_x \leftarrow TaylorModel(\mathcal{X}_0)$; // construct Taylor model from $\mathcal{X}_0$
2  for $k \leftarrow 0$ to $K - 1$ do
3      $\mathcal{Z}' \leftarrow p2c(\textbf{to\_Zonotope}(\mathcal{T}_x))$;   // convert to zonotope
4      $\mathcal{Z} \leftarrow c2p(\rho_N(\mathcal{Z}'))$;   // zonotope covering $N$'s output
5      $\mathcal{T}_u \leftarrow \textbf{to\_TM}(\mathcal{Z})$;   // convert to Taylor model
6      $\mathcal{T}' \leftarrow merge(\mathcal{T}_x, \mathcal{T}_u)$;   // $n + m$-dimensional Taylor model by merging Taylor models
7      $\mathcal{R}_k \leftarrow \rho_f(\mathcal{T}', \tau)$; // TMRS covering reachable states for one control cycle
8      $\mathcal{T} \leftarrow evaluate(\mathcal{R}_k, k\tau)$;   // Taylor model at next sampling time point
9      $\mathcal{T}_x \leftarrow project(\mathcal{T}, [1, \dots, n])$; // project Taylor model to the state variables
10 end
11 return $(\mathcal{R}_0, \dots, \mathcal{R}_{K-1})$
```

The third step is to construct a Taylor model $\mathcal{T}_u$ from $\mathcal{Z}$, using our algorithm from above (line 5), and then merge with $\mathcal{T}_x$, the first n dimensions of $\mathcal{T}$, to obtain an $n+m$-dimensional Taylor model. This works because $\mathcal{T}_x$ does not depend on the inputs. To obtain a structured zonotope $\mathcal{Z}$, we use the order-reduction algorithm in (Girard 2005).

The fourth step is to propagate the TMRS through the plant via ρ_f for the next τ time frame (lines 6 and 7).

Evaluation

We implemented the algorithm in JuliaReach, a toolbox for reachability analysis (Bogomolov et al. 2019). Set representation and set conversion is implemented in the library LazySets (Forets and Schilling 2021). For the Taylor-model analysis we use the implementation by Benet and Sanders (2019); Benet et al. (2019). For the zonotope propagation we implemented the algorithm by Singh et al. (2018). To obtain simulations in the visualizations we use the ODE solver by Rackauckas and Nie (2017). All results reported here were obtained on a standard laptop with a quad-core 2.2 GHz CPU and 8 GB RAM running Linux.

We consider the benchmark problems used in the competition on NNCS at ARCH-COMP 2021 (Johnson et al. 2021). In total there are seven problems with various features. The problems have up to 12 continuous states, 5 hidden layers, and 500 hidden units. All problems use ReLU activation functions. We exclude one problem from the presentation because it differs in scope (linear discrete-time behavior and multiple controllers). Next we study one of the problems in detail. Then we report on the results for the other problems. The experiments are available at *https://github.com/JuliaReach/AAAI22_RE/.*

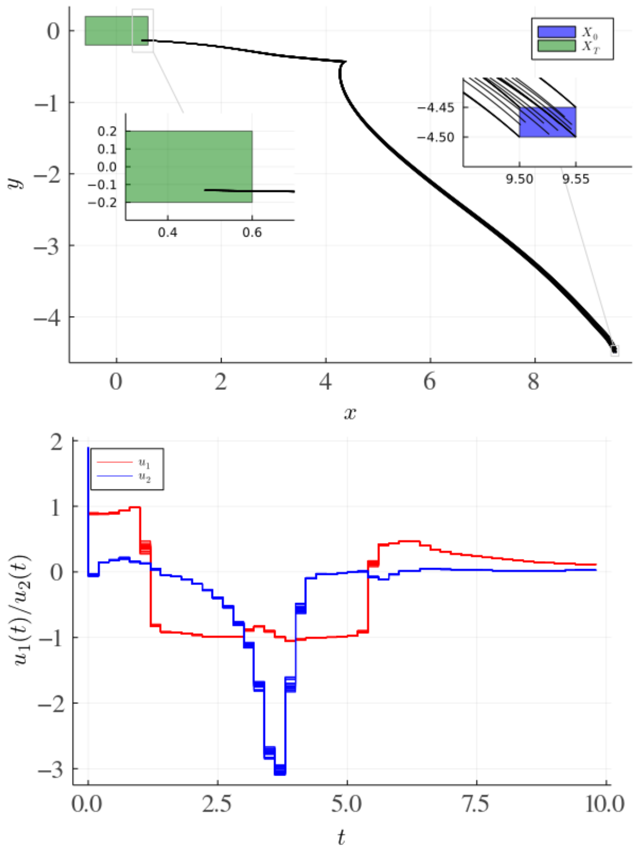


Figure 3: Simulations for the unicycle model: projection in state space (first plot) and control signals (second plot).

Case Study: Unicycle Model

We consider the model of a unicycle, which was originally used in (Dutta, Chen, and Sankaranarayanan 2019). The plant has four state variables (x, y, θ, v), where x and y represent the wheel coordinates in the plane, θ is the yaw angle of the wheel, and v is the velocity. There are two inputs (u_1, u_2), where u_1 controls the acceleration and u_2 controls the wheel direction. Finally, there is a disturbance w. The dynamics are given as the following system of ODEs:

$$\dot{x} = v\cos(\theta) \qquad \dot{y} = v\sin(\theta) \qquad \dot{\theta} = u_2 \qquad \dot{v} = u_1 + w$$

A neural-network controller with one hidden layer (500 neurons) was trained with a model-predictive control scheme as teacher. The function $p2c$ is the identity, while the controller output is post-processed with $(u_1, u_2)^T = c2p((o_1, o_2)^T) = (o_1 - 20, o_2 - 20)^T$. The controller is sampled with a period $\tau = 0.2\,s$. The uncertain set of initial states $\mathcal{X}_0$ is given by $x \in [9.5, 9.55]$, $y \in [-4.5, -4.45]$, $\theta \in [2.1, 2.11]$, $v \in [1.5, 1.51]$, and $w \in [-10^{-4}, 10^{-4}]$. The specification is to reach a target set $\mathcal{X}_T$ given by $x \in [-0.6, 0.6]$, $y \in [-0.2, 0.2]$, $\theta \in [-0.06, 0.06]$, $v \in [-0.3, 0.3]$ within a time horizon of $T = 10\,s$.

In Figure 3 we show $\mathcal{X}_0$, $\mathcal{X}_T$, and ten random simulations together with simulations from all 32 extremal points of $\mathcal{X}_0$ and the domain of w. We can see that $\mathcal{X}_T$ is reached only

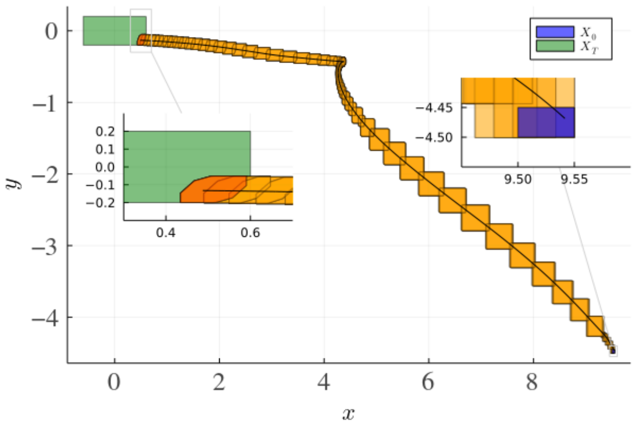


Figure 4: Reachability analysis for the unicycle model.

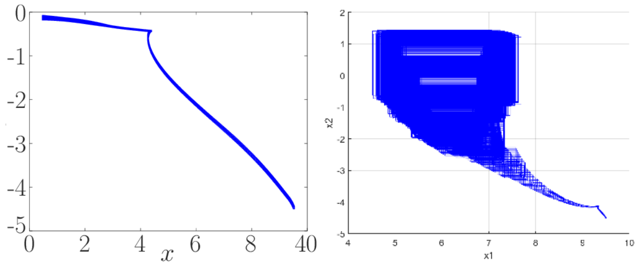


Figure 5: Results from Sherlock (left) and NNV (right; only partial result) on the unicycle model (x/y projection).

in the last moment, so the analysis requires a high precision to prove containment of the reachable states at $t = 10\,s$. Our implementation can prove containment for three state variables, but the lower bound for y slightly exceeds -0.2.

We found that the zonotope approximation is suboptimal for this controller. To improve precision, we reduce the dependency uncertainty in the initial states by splitting $\mathcal{X}_0$ into $3 \times 1 \times 8 \times 1 = 24$ smaller hyperrectangles. Then we have to solve 24 reachability problems, where the final reachable states are the union of the individual results. Mathematically, these sets are equivalent, but set-based analysis generally gains precision from smaller initial states. We note that the analysis is embarrassingly parallelizable, but our current implementation does not make use of that.

Using an adaptive step size with absolute tolerance 10^{-15} and order-10 Taylor models, we can verify the property within 93 seconds (i.e., four seconds for each sub-problem). The reach sets of all 24 runs together with a random simulation are shown in Figure 4. The reach sets for different sub-problems overlay each other soon after the beginning, indicating that the controller quickly steers trajectories from different sources to roughly the same states. We could have evaluated the final TMRS at the time point $t = 10$ for higher precision, but this was not required.

We compare to the results of *Sherlock* (Dutta, Chen, and Sankaranarayanan 2019), which originally proposed the benchmark problem. (We are not aware of any tool that can

Problem	Dimensions	Cyc.	Sherlock	JuliaReach
Unicycle	$\mathbf{4}; 500; \mathbf{2}$	50	526	93
TORA	$\mathbf{4}; 100, 100, 100; \mathbf{1}$	20	30	2040
ACC	$\mathbf{6}; 20, 20, 20, 20, 20; \mathbf{1}$	50	4	1
S. pend.	$\mathbf{2}; 25, 25; \mathbf{1}$	20	1	1
D. pend.	$\mathbf{4}; 25, 25; \mathbf{2}$	20	$6^\dagger$	4
Airplane	$\mathbf{12}; 100, 100, 20; \mathbf{6}$	20	$169^\dagger$	29

Table 1: Benchmarks. The second column shows the number of state variables n, neurons per hidden layer, and control variables m. The other columns show the number of control cycles and the the run time of Sherlock resp. JuliaReach in seconds (averaged over five runs, rounded to integers). A "†" marks measurement until the tool stopped working.

handle all benchmark problems.) As discussed in the related work, that approach can be very precise because it does not have to switch between set representations, and indeed there is no splitting required here. However, controllers with multiple outputs need to be handled by repeating the analysis for each output neuron individually, which is costly. In total the analysis takes 525 seconds and produces the plot in Figure 5.

We also compare to the result of NNV (Tran et al. 2020b), which uses a black-box reachability method for the plant and hence loses many dependencies. In (Johnson et al. 2021) the authors reported that their tool runs out of memory before the analysis finishes. The plot in Figure 5 contains intermediate results, which show that the precision declines quickly.

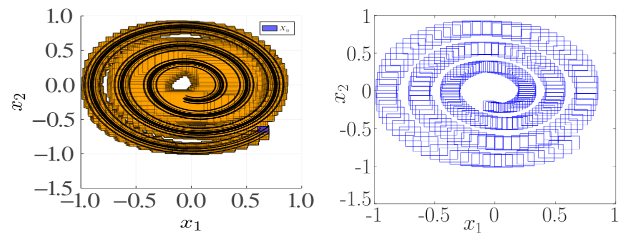
Other Problems From ARCH-COMP

Here we shortly summarize the results on the remaining benchmark problems from ARCH-COMP 2021: a translational oscillator with a rotational actuator (TORA), an adaptive cruise control (ACC), a single and a double pendulum, and an airplane model. We summarize the core model properties in Table 1. Since NNV cannot solve many of the problems, we only discuss the results of our implementation and Sherlock. The reachability results are plotted in Figure 6.

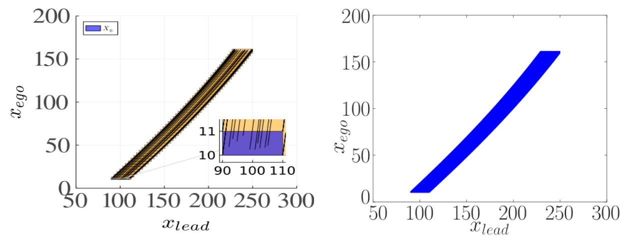
The TORA problem was also proposed by the Sherlock authors. Here $c2p(u) = u - 10$. Again the zonotope algorithm produces relatively coarse results and our tool JuliaReach has to split heavily to yield the required precision, which makes it slow. For all other problems, JuliaReach is precise enough without splitting and is faster than Sherlock.

The implementation of Sherlock requires ReLU activation functions at every layer, including the output layer. This is not common, but we modified the controllers of the last four problems accordingly for a fair comparison. The original specifications are not satisfied by the modified controllers and hence we only compare reachability results and run time here for these problems. Our implementation can solve all benchmark problems with the original controllers, as shown in the ARCH-COMP 2021 report (Johnson et al. 2021).

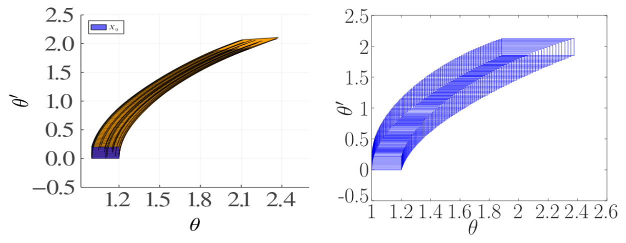
For the ACC problem, $p2c(x) = (30, 1.4, x_5, x_1 - x_4, x_2 - x_5)^T$ (an affine map); the tools have similar precision but ours is faster. For the single-pendulum problem, our implementation is more precise. On the remaining two prob-



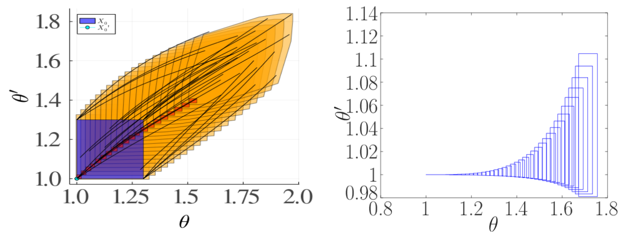
(a) TORA.



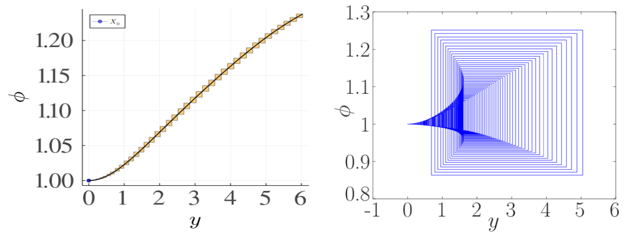
(b) ACC.



(c) Single pendulum.



(d) Double pendulum. Left: Additional result from a smaller initial set (in red). Right: Incomplete result from the smaller initial set.



(e) Airplane. Right: Incomplete result.

Figure 6: Results from JuliaReach (left) and Sherlock (right). We additionally plot simulations from the extreme points of $\mathcal{X}_0$ and ten additional random points.

lems with multiple control variables (double pendulum and airplane), Sherlock diverges. As discussed before, Sherlock uses an approximate analysis in those cases. For the double-pendulum problem, it stops after the first control cycle and returns with an error about divergence. Splitting $\mathcal{X}_0$ helps for a while, but Sherlock diverges after seven control cycles even when splitting each dimension into 30,000 pieces. For the airplane problem, the trajectories obtained with the modified controller expand fast and hence we define a smaller initial set. Sherlock is slow due to the large number of control variables and diverges after ten control cycles.

Discussion

To summarize, our approach generally produces precise results, as can be seen from the simulations in the plots covering most of the reachable states, and can additionally benefit from splitting the initial set; for Sherlock, the effect of splitting is smaller and does not help solving the double pendulum and airplane problems. Sherlock is typically precise and sufficiently fast for controllers with a single output, although not always more precise than our implementation. For multiple outputs, Sherlock is slower and often diverges.

Our algorithm relies on two reachability algorithms and inherits their scalability. We shortly discuss the most relevant parameters for NNCS reachability. 1) Plant dimension: Reachability methods for high-dimensional nonlinear systems are generally not available. 2) Neural-network dimension: The algorithm from (Singh et al. 2018) scales to realistic neural networks in control applications. 3) Number of iterations: Each iteration incurs a conversion between set representations, which makes the task more challenging.

Conclusion

In this paper we have addressed the reachability problem for neural-network control systems. When combining successful reachability tools for the ODE and neural-network components, the main obstacle is the conversion of sets at the tool interface. We have proposed a conversion scheme when the ODE analyzer uses Taylor models and the neural-network analyzer uses zonotopes. Our approach is able to preserve most dependencies between the control cycles. Our implementation is the first to successfully analyze all benchmark problems of the verification competition ARCH-COMP 2021. Compared to Sherlock, our approach works reliably for neural networks with multiple output dimensions.

For future work, we plan to investigate the interface for other set representations. For example, CORA (Althoff 2015) can use polynomial zonotopes (Althoff 2013; Kochdumper and Althoff 2021), which are as expressive as Taylor models and conversion to and from zonotopes works well. Another direction is to combine different reachability algorithms, e.g., the one in (Gehr et al. 2018) or the “polynomialization” approach used in Sherlock (Dutta, Chen, and Sankaranarayanan 2019); thus we could compute several output sets and then choose one or even combine them.

Acknowledgments

We thank Luis Benet for fruitful discussions about the Taylor-model implementation. We also thank the anonymous reviewers for their feedback. S.G. acknowledges funding from Julia Seasons of Contributions.

References

Akintunde, M. E.; Botoeva, E.; Kouvaros, P.; and Lomuscio, A. 2020. Formal Verification of Neural Agents in Nondeterministic Environments. In *AAMAS*, 25–33. International Foundation for Autonomous Agents and Multiagent Systems.

Althoff, M. 2013. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *HSCC*, 173–182. ACM.

Althoff, M. 2015. An Introduction to CORA 2015. In *ARCH*, volume 34 of *EPiC Series in Computing*, 120–151. EasyChair.

Althoff, M.; Frehse, G.; and Girard, A. 2021. Set Propagation Techniques for Reachability Analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1): 369–395.

Alur, R.; Courcoubetis, C.; Henzinger, T. A.; and Ho, P. 1992. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In *Hybrid Systems*, volume 736 of *LNCS*, 209–229. Springer.

Bacci, E.; Giacobbe, M.; and Parker, D. 2021. Verifying Reinforcement Learning up to Infinity. In *IJCAI*, 2154–2160. ijcai.org.

Benet, L.; Forets, M.; Sanders, D. P.; and Schilling, C. 2019. TaylorModels.jl: Taylor models in Julia and its application to validated solutions of ODEs. In *SWIM*.

Benet, L.; and Sanders, D. P. 2019. TaylorSeries.jl: Taylor expansions in one and several variables in Julia. *Journal of Open Source Software*, 4(36): 1043.

Bogomolov, S.; Forets, M.; Frehse, G.; Potomkin, K.; and Schilling, C. 2019. JuliaReach: a toolbox for set-based reachability. In *HSCC*, 39–44. ACM.

Chen, X.; Ábrahám, E.; and Sankaranarayanan, S. 2012. Taylor Model Flowpipe Construction for Non-linear Hybrid Systems. In *RTSS*, 183–192. IEEE Computer Society.

Chen, X.; Ábrahám, E.; and Sankaranarayanan, S. 2013. Flow*: An Analyzer for Non-linear Hybrid Systems. In *CAV*, volume 8044 of *LNCS*, 258–263. Springer.

Clavière, A.; Asselin, E.; Garion, C.; and Pagetti, C. 2021. Safety Verification of Neural Network Controlled Systems. In *DSN*, 47–54. IEEE.

Cousot, P.; and Cousot, R. 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *POPL*, 238–252. ACM.

Dutta, S.; Chen, X.; Jha, S.; Sankaranarayanan, S.; and Tiwari, A. 2019. Sherlock - A tool for verification of neural network feedback systems: demo abstract. In *HSCC*, 262–263. ACM.

Dutta, S.; Chen, X.; and Sankaranarayanan, S. 2019. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *HSCC*, 157–168. ACM.

Dutta, S.; Jha, S.; Sankaranarayanan, S.; and Tiwari, A. 2018a. Learning and Verification of Feedback Control Systems using Feedforward Neural Networks. In *ADHS*, volume 51, 151–156. Elsevier.

Dutta, S.; Jha, S.; Sankaranarayanan, S.; and Tiwari, A. 2018b. Output Range Analysis for Deep Feedforward Neural Networks. In *NFM*, volume 10811 of *LNCS*, 121–138. Springer.

Fan, J.; Huang, C.; Chen, X.; Li, W.; and Zhu, Q. 2020. ReachNN*: A Tool for Reachability Analysis of Neural-Network Controlled Systems. In *ATVA*, volume 12302 of *LNCS*, 537–542. Springer.

Forets, M.; and Schilling, C. 2021. LazySets.jl: Scalable Symbolic-Numeric Set Computations. *Proceedings of the JuliaCon Conferences*, 1(1): 11.

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. T. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *SP*, 3–18. IEEE Computer Society.

Ghorbal, K.; Goubault, E.; and Putot, S. 2009. The Zonotope Abstract Domain Taylor1+. In *CAV*, volume 5643 of *LNCS*, 627–633. Springer.

Girard, A. 2005. Reachability of Uncertain Linear Systems Using Zonotopes. In *HSCC*, volume 3414 of *LNCS*, 291–305. Springer.

Hainry, E. 2008. Reachability in Linear Dynamical Systems. In *CiE*, volume 5028 of *LNCS*, 241–250. Springer.

Huang, C.; Fan, J.; Li, W.; Chen, X.; and Zhu, Q. 2019. ReachNN: Reachability Analysis of Neural-Network Controlled Systems. *ACM Trans. Embed. Comput. Syst.*, 18(5s): 106:1–106:22.

Ivanov, R.; Weimer, J.; Alur, R.; Pappas, G. J.; and Lee, I. 2019. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *HSCC*, 169–178. ACM.

Johnson, T. T.; Lopez, D. M.; Benet, L.; Forets, M.; Guadalupe, S.; Schilling, C.; Ivanov, R.; Carpenter, T. J.; Weimer, J.; and Lee, I. 2021. ARCH-COMP21 Category Report: Artificial Intelligence and Neural Network Control Systems (AINNCS) for Continuous and Hybrid Systems Plants. In *ARCH*, volume 80 of *EPiC Series in Computing*, 90–119. EasyChair.

Kochdumper, N.; and Althoff, M. 2021. Sparse Polynomial Zonotopes: A Novel Set Representation for Reachability Analysis. *IEEE Trans. Autom. Control.*, 66(9): 4043–4058.

Liu, C.; Arnon, T.; Lazarus, C.; Strong, C. A.; Barrett, C. W.; and Kochenderfer, M. J. 2021. Algorithms for Verifying Deep Neural Networks. *Found. Trends Optim.*, 4(3-4): 244–404.

Makino, K.; and Berz, M. 2003. Taylor models and other validated functional inclusion methods. *Int. J. Pure Appl. Math*, 4(4): 379–456.

Neumaier, A. 1993. *The wrapping effect, ellipsoid arithmetic, stability and confidence regions*, 175–190. Springer Vienna.

Rackauckas, C.; and Nie, Q. 2017. DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. *The Journal of Open Research Software*, 5(1).

Sidrane, C.; Maleki, A.; Irfan, A.; and Kochenderfer, M. J. 2021. OVERT: An Algorithm for Safety Verification of Neural Network Control Policies for Nonlinear Systems. *CoRR*, abs/2108-0122.

Singer, A. B.; and Barton, P. I. 2006. Bounding the Solutions of Parameter Dependent Nonlinear Ordinary Differential Equations. *SIAM J. Sci. Comput.*, 27(6): 2167–2182.

Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. T. 2018. Fast and Effective Robustness Certification. In *NeurIPS*, 10825–10836.

Tran, H.; Bak, S.; Xiang, W.; and Johnson, T. T. 2020a. Verification of Deep Convolutional Neural Networks Using ImageStars. In *CAV*, volume 12224 of *LNCS*, 18–42. Springer.

Tran, H.; Yang, X.; Lopez, D. M.; Musau, P.; Nguyen, L. V.; Xiang, W.; Bak, S.; and Johnson, T. T. 2020b. NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems. In *CAV*, volume 12224 of *LNCS*, 3–17. Springer.

Xiang, W.; Tran, H.; Rosenfeld, J. A.; and Johnson, T. T. 2018. Reachable Set Estimation and Safety Verification for Piecewise Linear Systems with Neural Network Controllers. In *ACC*, 1574–1579. IEEE.

Ziegler, G. M. 1995. *Lectures on polytopes*, volume 152 of *Graduate Texts in Mathematics*. New York: Springer-Verlag.