BURNS: Backward Underapproximate Reachability for Neural-Feedback-Loop Systems

Chelsea Sidrane CHELSE@KTH.SE

Jana Tumova TUMOVA@KTH.SE

KTH Royal Institute of Technology, Digital Futures

Editors: G. Sukhatme, L. Lindemann, S. Tu, A. Wierman, N. Atanasov

Abstract

Learning-enabled planning and control algorithms are increasingly popular, but they often lack rigorous guarantees of performance or safety. Frameworks such as reachability analysis can be used to provide such guarantees. We introduce an algorithm for computing underapproximate backward reachable sets of nonlinear discrete time neural feedback loops. We then use the backward reachable sets to check goal-reaching properties. Our algorithm is based upon ideas from robustness analysis for vision networks, and on overapproximating the system dynamics function. Together these enable computation of underapproximate backward reachable sets through solutions of mixed-integer linear programs. We rigorously analyze the soundness of our algorithm and demonstrate it on a numerical example. Our work expands the class of properties that can be verified for learning-enabled systems.

Keywords: Neural Networks, Verification, Reachability, Control

1. Introduction

Neural network control and planning are becoming increasingly prevalent in complex robotic systems (Krinner et al., 2024; An et al., 2024; Lee et al., 2024) and are frequently able to demonstrate superior performance to traditional methods (Song et al., 2023). However, learning-based methods often lack guarantees of reliability and correctness. It is difficult to establish such guarantees because neural networks are difficult to analyze – they are typically non convex and may have anywhere from hundreds to billions of parameters.

One essential tool for analyzing the reliability and correctness of these systems is reachability analysis. Reachability analysis involves computing all possible states that the system may reach, and then using those sets to check a *property*. Two classic properties analyzed in reachability analysis are *reach* properties, which involve reaching a goal set, and *avoid* properties, which involve avoiding an unsafe set. Reachability analysis can be performed either forward or backward. *Forward* reachability analysis assumes that the system begins within a starting set $\mathcal{X}_s$ and computes subsequent reachable sets into the future. *Backward* reachability analysis begins from some set $\mathcal{X}_f$ and computes reachable sets into the past that will reach set $\mathcal{X}_f$ (Baier and Katoen, 2008).

It is typically intractable to perform exact reachability analysis for anything more complex than linear or affine systems, so approximation is commonly used. Approximate reachability algorithms generally compute either underapproximations, where the approximate set at time t is a subset of the true reachable set at time t, or overapproximations, where the approximate set is a superset. Different combinations of underapproximation, overapproximation, forward reachability and backward reachability are useful for checking different kinds of properties. Overapproximate forward

and backward reachability are both useful for *avoid* properties because these methods "inflate" the unsafe set, capturing all unsafe states and some safe states too. Underapproximate backward reachability is useful for *reach* properties because this method "shrinks" the goal set backwards in time, guaranteeing to capture a subset of the states that will reach the goal.

Related Work There is ongoing work on reachability analysis for dynamical systems without neural networks in both continuous and discrete time; including autonomous linear (Wetzlinger and Althoff, 2023), piecewise affine (Kloetzer and Belta, 2006), and general nonlinear (Chen et al., 2013; Abate et al., 2024) systems. There is also extensive work on Hamilton-Jacobi (HJ) reachability wherein one computes the reachable set and a control policy (Bansal et al., 2017).

In this work, we focus on reachability for discrete-time autonomous systems with neural network control policies, known as *neural feedback loops*. There is work on exact reachability analysis for NN dynamics (Vincent and Schwager, 2021; Zhang et al., 2023), on overapproximate forward (Everett et al., 2021; Sidrane et al., 2022b) and overapproximate backward (Rober et al., 2023; Kotha et al., 2023) reachability analysis. There is even underapproximate backward reachability for linear systems (Everett et al., 2021; Zhang et al., 2023), but to the best of the authors' knowledge, there is no work on underapproximate backward reachability analysis for general nonlinear neural feedback loops. We focus on addressing this gap in literature.

We take inspiration from methods that seek to verify neural networks in isolation. Many neural network verification approaches may be seen as forward reachability problems where the algorithm computes the image of the neural network function over an input set, and then reason about whether that image satisfies a desired property (Liu et al., 2021). Homologously, we propose that one may conceptualize of robustness analysis of neural networks as backwards reachability. Typically, robustness verification for a classification task such as mapping an image to a label like 'cat', computes a region around a specific training image where the label does not change (Dvijotham et al., 2018). A single robustness region does not capture all images labeled 'cat' by the network, and thus underapproximates the backward reachable set of the label 'cat'. Recent work has explored related ideas to compute preimages of a network in isolation (Zhang et al., 2024), but the goal of our work is distinct in that we consider dynamical systems in feedback with neural network controllers, compute multi-step backward reachable sets, and check temporal properties.

Contributions The contributions of this work are:

- An algorithm for underapproximate backward reachability of discrete time nonlinear neural feedback loops
- A rigorous theoretical analysis of the algorithm demonstrating its soundness
- A numerical demonstration of the algorithm
- A method for checking inclusion of a polytope in arbitrary non convex sets formed from unions of polytopes which enables the checking of goal reaching properties

2. Preliminaries

A **polytope** is $\{x \mid Ax \leq b\}$ with $A \in R^{m \times n}$, $x \in R^n$, $b \in R^m$. A **p-norm ball** $Ball_p(c, \epsilon, M)$ is a set $\{x \mid ||M(x-c)||_p \leq \epsilon\}$ with $x, c \in R^n$, $\epsilon \in R$, $M \in R^{n \times n}$, $p \in \{1, \infty\}$. Set $\mathcal{C}$ **overapproximates** set $\mathcal{S}$ if $\mathcal{S} \subseteq \mathcal{C}$; C **underapproximates** $\mathcal{S}$ if $\mathcal{C} \subseteq \mathcal{S}$. $f \circ^k (x) = f(f(\cdots f(x))$ denotes k repeated applications of function f. A **feed-forward neural network** maps $R^n \rightarrow R^m$

where the output $z_p = NN(z_0)$ is computed $z_i = \sigma_i(W_i z_{i-1} + b_i), i \in 1 \dots p$ with $W_i \in R^{d_i \times d_{i-1}}$, $b_i \in R^{d_i}$, neurons $z_i \in R^{d_i}$ and $\sigma_i \in R \to R$ acts elementwise. $\mathbb{I}\{\cdot\}$ is an indicator function; $\mathbf{int}S$ is the interior of set $\mathcal{S}$; $(\mathcal{S})^c$ is the complement; and $f(\mathcal{S})$ is the image $\{f(s) \mid s \in \mathcal{S}\}$.

3. Problem Setting

Consider a discrete-time **dynamical system** with system state $x_t \in R^n$ and dynamics function f that produces the state at the next step as a function of the current state and the current control input: $x_{t+1} = f(x_t, u_t)$; $u_t \in R^m$. If the control signal u_t at each time step is generated by a neural network policy $u_t = NN(x_t)$, we call this dynamical system a **neural feedback loop** (NFL). The **closed-loop** dynamics function, also known as the plant, is defined $f_{cl}(x_t) = f(x_t, NN(x_t))$. We are interested in computing backward reachable sets under the closed-loop plant f_{cl}.

Definition 1 *The exact* **k-*step backward reachable set*** *of $\mathcal{G}$ for closed-loop system f_{cl} is defined*

$$\mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G}) \triangleq \{x_{-k} \mid x_{-k+1} = f_{cl}(x_{-k}) \wedge \dots \wedge x_0 = f_{cl}(x_{-1}) \wedge x_0 \in \mathcal{G}\} \tag{1}$$

After computing backward reachable sets, we want to check goal-reaching properties.

Definition 2 *An NFL f_{cl} satisfies a **goal-reaching property** with goal $\mathcal{G}$ for k timesteps starting from set $\mathcal{X}_s$ if $\forall x \in \mathcal{X}_s . \exists t \in [0 \dots k] . f \circ^t (x) \in \mathcal{G}$, or equivalently, if $\mathcal{X}_s \subseteq \bigcup_{t=1}^{k} \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$.*

However, if we assume f is an arbitrary nonlinear function, it is not possible to compute exact backward reachable sets. We must therefore approximate, and our choice of approximation is guided by our intention to check goal-reaching properties. Computing underapproximate backward reachable sets $\mathcal{R}^{f_{cl}}_{(-t)_{\text{under}}}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$ allows for the sound verification of goal-reaching properties: $\left(\mathcal{X}_s \subseteq \bigcup_{t=1}^{k} \mathcal{R}^{f_{cl}}_{(-t)_{\text{under}}}(\mathcal{G})\right) \wedge \left(\mathcal{R}^{f_{cl}}_{(-t)_{\text{under}}}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})\right) \Rightarrow \mathcal{X}_s \subseteq \bigcup_{t=1}^{k} \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$.

Problem Statement Given a nonlinear NFL $x_{t+1} = f(x_t, NN(x_t))$ where $x_t \in R^n$, $u_t \in R^m$ and a goal set $\mathcal{G} \subseteq R^n$, compute a sequence of k underapproximate backward reachable sets $\mathcal{R}^{f_{cl}}_{(-t)_{\text{under}}}(\mathcal{G})$ such that $\forall t \in 1 \dots k . \mathcal{R}^{f_{cl}}_{(-t)_{\text{under}}}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$.

4. Method

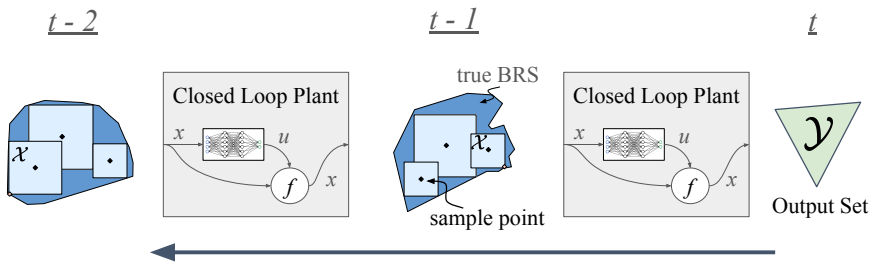


Figure 1: Illustration of underapproximate backward reachability algorithm.

Our approach to the problem defined above is to compute multiple maximal norm balls underapproximating the backward reachable set at each time step t. We formulate the computation of these sets as a series of mixed-integer optimization problems. The algorithm is presented in Algorithm 1 and illustrated in Figure 1; details and analysis follow.

4.1. Backward Reachability Algorithm

Algorithm 1 begins in line 2 by computing tight bounds for each neuron by solving an MIP. Line 3 enters a loop to iteratively compute the backward reachable set at each time t. In lines 4-5, the neural network and dynamics are encoded. Line 6 samples a set of points $\{x_d\}$ from the backward reachable set to serve as centers for norm balls. Line 7 uses the samples to estimate the shape of the backward reachable set and calculate M. Line 8 calculates the center of the samples to use in the first norm ball. Line 10 encodes that the input at time $-t$ must lie in the norm ball. Line 11 encodes the constraint that the state at the final time does not lie within the output set, and line 12 solves for the optimal radius ϵ^*, which is used to construct a set in line 13. In lines 16-189, if the horizon h has been reached, the optimization problem is reset and computation continues backward from time t_o. Resetting the optimization increases conservatism but improves computational tractability and is known as *hybrid-symbolic* reachability (Sidrane et al., 2022b).

Algorithm 1: Underapproximate Backward Reachability

Data: f, NN, k, h, $\mathcal{G}$, $\mathcal{D}$
Result: $\{\mathcal{R}\}_k$

1 $m \leftarrow$ init_opt_problem(); $\mathcal{O} \leftarrow \mathcal{G}$; $t \leftarrow 1$; $t_o \leftarrow 0$;
2 $b \leftarrow$ get_network_bounds_MIP($\mathcal{D}$, NN);
3 **while** $t < k$ **do**
4 $\mathcal{D}_u \leftarrow$ encode_control(m, NN, $\mathcal{D}$, b);
5 $x_t =$ encode_dynamics(m, f, $\mathcal{D}$, $\mathcal{D}_u$);
6 $\{x_\mathrm{d}\}_{j=1}^{1000} \leftarrow$ rejection_sampling($\mathcal{O}, \mathcal{D}, f, t - t_o$);
7 $M \leftarrow$ estimate_shape($\{x_\mathrm{d}\}$);
8 pushfirst!($\{x_\mathrm{d}\}$, mean($\{x_\mathrm{d}\}$));
9 **for** $j \in 1 : n_s$ **do**
10 $c \leftarrow$ encode $x_t \in Ball(\epsilon, x_\mathrm{d}, M)$ `// input constraint`
11 encode $x_{t_o} \notin \mathcal{O}$;
12 $\epsilon^* \leftarrow$ solve_opt_prob_2(); delete(c);
13 Ball$_j \leftarrow Ball(\epsilon^*, x_\mathrm{d}, M)$;
14 **end**
15 $\{\mathcal{R}\}_k[t] \leftarrow \bigcup_j$ Ball$_j$;
16 **if** *modulo*$(t, h) == 0$ **then**
17 $\mathcal{O} \leftarrow \mathcal{R}_t$; $t_o \leftarrow t$;
18 $m \leftarrow$init_opt_problem();
19 **end**
20 $t++$;
21 **end**

The complexity of Alg. 1 is dominated by the solution of the MILP (line 12). Further, the size of the MILP is dominated by the encoding of the dynamics and control policy. If the encoding of the dynamics and control policy (lines 3-4) requires $q = q_d + q_c$ variables, the algorithm solves n_s MILP problems each containing approximately $t * q$ variables at each timestep t, with the final problem containing $\sim k * q$ variables. I.e., the size of the MILP scales linearly with the number of neurons in the control policy network, q_c, and with the timesteps k. The only dependence on

state dimension comes from the sampling in line 6, which scales exponentially, but more efficient sampling techniques can be used for higher dimensional problems. Next we elaborate on specific subprocedures.

4.2. Computation of a Single Norm Ball

Line 12 in algorithm 1 computes a single norm ball within the backward reachable set, which corresponds to solving the optimization problem in eq. (2). Wlog, we consider the computation of the underapproximate backward reachable set of $\mathcal{G}$, k steps backward: $\mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$.

$$\underset{x,\, x_0,\, \epsilon}{\text{minimize}} \quad \epsilon \tag{2a}$$

$$\text{subject to} \quad ||M(x - x_{\text{d}})||_p \leq \epsilon, \tag{2b}$$

$$x_0 = f_{cl} \circ^k (x), \tag{2c}$$

$$x_0 \notin \mathbf{int}\mathcal{G} \tag{2d}$$

Constraint 2b parameterizes a norm ball centered at $x_{\text{d}} \in \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$. To connect x at time t=$-k$ to x_0 at time t=0, constraint 2c enforces that x_0 is generated by repeated application of the closed-loop plant $x_0 = f_{cl} \circ^k (x)$. The final constraint, 2d, enforces that x_0 does not lie within the goal, $x_0 \notin \mathbf{int}\mathcal{G}$. The objective $\min \epsilon$ finds the smallest norm ball satisfying the constraints. The intuition is that any smaller and x_0 would lie within $\mathcal{G}$, implying that every point in the ϵ-$Ball$ is within the backwards reachable set. Although optimization problem 2 contains nonlinear constraints, we assume momentarily that we can obtain the global optimum and analyze the set that would result.

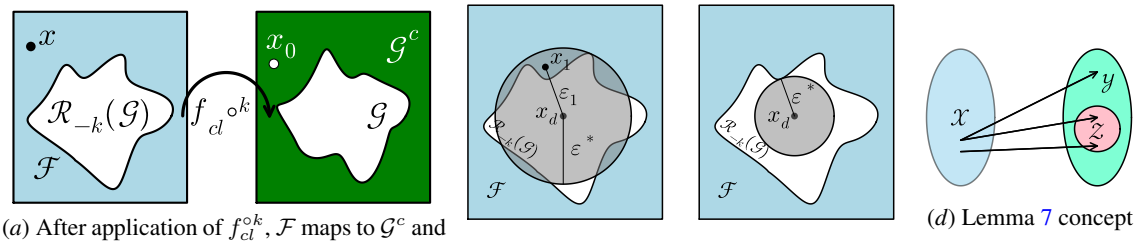
4.2.1. Theoretical Analysis of Computing a Single Norm Ball

We prove that the optimum solution $Ball_p(x_d, \epsilon^*, M)$ to eq. (2) is a sound underapproximation of the backwards reachable set, and is the largest p-norm ball centered at x_d under transformation M with this property. Computing the largest underapproximation at a given point reduces conservatism of the resulting reachable set. We show this property in the proof of lemma 4 by arguing that $Ball_p(x_d, \epsilon^*, M)$ is what we call a boundary coincident subset centered at x_d. We assume the optimal solution to opt. prob. 2 is obtained, and that $\mathcal{G}$ and $\mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$ are closed sets containing both interior and boundary. $\mathcal{G}$ may be non-convex and/or a union of disjoint sets.

Definition 3 ***Boundary Coincident Subset*** *$A \subseteq_{\partial c} B$ if $A \subseteq B$ and $\partial A \cap \partial B \neq \emptyset$.*

Lemma 4 $Ball_p(x_d, \epsilon^*, M) \subseteq_{\partial c} \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$.

Proof Sketch (i) $Ball_p(x_d, \epsilon^*, M) \subseteq \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$: Assume there exists a point within the interior of the norm ball but outside of the backward reachable set of $\mathcal{G}$, i.e. $\exists x.x \in \mathbf{int}Ball_p(x_d, \epsilon^*, M) \bigwedge x \notin \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$. Such a point is illustrated in fig. 2(*a*) as x and fig. 2(*b*) as x_1 and would have a smaller distance to x_{d} than ϵ^*, presenting a contradiction (see fig. 2(*b*)). Therefore we can assert that $Ball_p(x_d, \epsilon^*, M) \subseteq \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$. (ii) $\partial Ball_p(x_d, \epsilon^*, M) \bigcap \partial\mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G}) \neq \emptyset$: Now either (1) the norm ball is in the interior of the backward reachable set, or (2) the norm ball is boundary coincident to the backward reachable set. If we assume (1), the constraint eq. (2d) that $x_0 \notin \mathcal{G}$ presents a contradiction. Therefore, we can conclude (2). See fig. 2(*c*) for an illustration and appendix A for complete proof. ■



(*a*) After application of $f_{cl}^{\circ k}$, $\mathcal{F}$ maps to $\mathcal{G}^c$ and $\mathcal{R}_{-k}(\mathcal{G})$ maps to $\mathcal{G}$.

(*b*) Contradiction

(*c*) Optimum

(*d*) Lemma 7 concept

Figure 2: a) Sets relevant to opt. prob. 2; b-c) Illustrate proof of lemma 4; d) Illustrates lemma 7

4.3. Constructing a Tractable Optimization Problem

As stated, optimization problem 2 is not tractable to solve due to constraints 2c and 2d. Constraint 2c contains nonlinear, nonconvex functions, and constraint 2d is represented abstractly. We provide solutions in the following subsections.

4.3.1. Encoding a Nonlinear NFL into an MILP

We encode the closed-loop nonlinear dynamics function $x_{t+1} = f(x_t, NN(x_t))$ in constraint 2c using the abstraction method from Sidrane et al. (2022b). This results in a piecewise-linear **multi-valued function** $\hat{f}$ that **overapproximates** the dynamics function f. Further detail in appendix B.

Definition 5 *As opposed to a single-valued function, a **multi-valued function** is a function g mapping $\mathcal{X} \subseteq R^d \to \mathcal{Y} \subseteq R^p$ where for each element $x \in \mathcal{X}$, multiple values of $y \in \mathcal{Y}$ may belong to the mapping, e.g., $(x, y_1) \in g$, $(x, y_2) \in g$, and $(x, y_3) \in g$.*

Definition 6 *An **overapproximation** $\hat{f}$ of a function f where both map $x \in R^d \to y \in R^p$ is such that $\forall x, y.(x, y) \in f \implies (x, y) \in \hat{f}$. As a corollary: $f(\mathcal{X}) \subseteq \hat{f}(\mathcal{X})$.*

We only consider activation functions σ_i for the neural network control policy that are piecewise linear, such as ReLU, $z_i^j = \max(\hat{z}_i^j, 0)$ where z_i^j is the j^{th} neuron in layer i. [1] As a result, the closed loop system abstraction $\hat{f}_{cl}(x) = \hat{f}(x, NN(x))$ is a multi-valued piecewise-linear function, and can be represented in the optimization problem with mixed-integer linear constraints. To encode ReLU functions from the control policy and $\max$ (and $\min$) functions from the abstracted dynamics $\hat{f}$, we use the encodings presented by Tjeng et al. (2017). Bounds for each neuron z_i^j in the network are computed by solving MIPs over domain $\mathcal{D}$ with objective $\{\min z_i^j, \max z_i^j\}$ where all layers $1 \ldots i-1$ are encoded as constraints. Further detail can be found in Appendix C.

4.3.2. Output Constraints

We express output constraint eq. (2d) as follows. If $\mathcal{G}$ is a convex polytope $\mathcal{G} = \{x \mid Ax \leq b\}$, $x \in R^n, A \in R^{m \times n}, b \in R^m$, the constraint $x \notin \mathbf{int}\mathcal{G}$ where $\mathbf{int}\mathcal{G} = \{x \mid Ax < b\}$, may be expressed $x \in (\mathbf{int}\mathcal{G})^c$ where $(\mathbf{int}\mathcal{G})^c = \{x \mid \bigvee_i a_i^T x \geq b_i\}$ and where a_i is the i^{th} row of A and b_i is the i^{th} element of b. We can re-formulate this expression as

$$\{x \mid \max(a_1 x - b_1, a_2 x - b_2, \ldots, a_m x - b_m) \geq 0\} \tag{3}$$

1. It is possible to use smooth nonlinear activation functions such as $\tanh$, but every activation function would have to be overapproximated, making the problem larger (Sidrane, 2022).

and encode it into the optimization problem using the $\max(\cdot)$ encoding from (Tjeng et al., 2017). If $\mathcal{G}$ is a union of convex polytopes $\mathcal{G} = \{x \mid \bigcup_j P^j\}$ where $P^j = \{x \mid A^j x \leq b^j\}$, to express $x \notin \mathbf{int}\mathcal{G}$, we equivalently write $\bigwedge_j x \notin \mathbf{int}P^j$ or $\bigwedge_j x \in (\mathbf{int}P^j)^c$. Each constraint $x \in (\mathbf{int}P^j)^c$ may be expressed using eq. (3). The number of binary variables needed to represent the constraint $x \in (\mathbf{int}\mathcal{G})^c$ grows linearly in the number of halfspaces defining $\mathcal{G}$ because each $\max(\cdot)$ encoding uses n_{args} binary variables. Additionally, using an approximate $\tilde{\mathcal{G}}$ as a result of hybrid-symbolic computation preserves the underapproximation because $\tilde{\mathcal{G}} \subseteq \mathcal{G}$ implies $\mathcal{R}^f_{(-t)}(\tilde{\mathcal{G}}) \subseteq \mathcal{R}^f_{(-t)}(\mathcal{G})$.

4.3.3. Theoretical Analysis of Using Abstraction

We reason that it is sound to replace f with abstraction $\hat{f}$ in the computation of the backward reachable set underapproximation.

Lemma 7 *If $\hat{f} : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X} \subseteq R^d$, $\mathcal{Y} \subseteq R^k$ and $\mathcal{Y} = \hat{f}(\mathcal{X})$ is a multivalued function that overapproximates the (single-valued) function $f : \mathcal{X} \to \mathcal{Z}$, where $\mathcal{Z} \subseteq R^k$, and $\mathcal{Z} = f(\mathcal{X})$, then $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{Z}) \subseteq \mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{Z})$.*

Proof Sketch If $\hat{f}$ is a multivalued function overapproximation of f, this means every tuple (x, y) in f is also in $\hat{f}$, plus potentially more. In other words, $\mathcal{Z} \subseteq \mathcal{Y}$. Given that $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{Y}) = \mathcal{X}$, we can say $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{Z}) \subseteq \mathcal{X}$. Given $\mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{Z}) = \mathcal{X}$, we can say therefore that $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{Z}) \subseteq \mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{Z})$. ■

See fig. 2(*d*) for illustration of sets relevant to lemma 7 and full proof in appendix D. Next, we reason about the soundness of using multiple copies of $\hat{f}$ to compute multiple step backward reachable sets.

Lemma 8 $\forall t \in 1 \ldots k. \mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$.

Proof Sketch We show this by induction. For the base case, we can say from lemma 7, that $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{G})$. For the inductive case, we assume: $\mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$ for some t. We then seek to show that $\mathcal{R}^{\hat{f}_{cl}}_{(-t-1)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t-1)}(\mathcal{G})$. The backward reachable set $\mathcal{R}^{f_{cl}}_{(-t-1)}(\mathcal{G})$ may be written $\mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G}))$. Invoking lemma 7, we can state that the one-step backward reachable set of $\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$ under $\hat{f}$ is a subset of that under f: $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})) \subseteq \mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G}))$. And then invoking our assumption $\mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$, we can state $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G})) \subseteq \mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G}))$. Putting these together, we can state $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G})) \subseteq \mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G}))$, or equivalently $\mathcal{R}^{\hat{f}_{cl}}_{(-t-1)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t-1)}(\mathcal{G})$. Thus the induction has been shown and we can state $\forall t \in 1 \ldots k. \mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$. ■

See appendix D for full proof. And in conclusion we can state:

Theorem 9 *Algorithm 1 returns sets $\{\mathcal{R}_t \mid t \in 1 \ldots k\}$ such that $\forall t \in 1 \ldots k. \mathcal{R}_t \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$*

Proof Algorithm 1 produces sets at each time t, $\mathcal{R}_t := \bigcup_i^{n_s} Ball_p(x^{(i)}_{d_t}, \epsilon^*, M)$. Per lemma 4 and lemma 8, for any i, $\forall t. Ball_p(x^{(i)}_{d_t}, \epsilon^*, M) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$. Therefore we can state $\forall t. \bigcup_i^{n_s} Ball_p(x^{(i)}_{d_t}, \epsilon^*, M) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$ and therefore $\forall t \in 1 \ldots k. \mathcal{R}_t \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$. ■

Algorithm 2: Rejection Sampling

Data: $\mathcal{O}, \mathcal{D}, f, t$
Result: x_d
while $true$ **do**
  $x_d \leftarrow sample(\text{global\_sobol\_sampler}, \mathcal{D})$;
  $y \leftarrow f \circ^t (x_d)$;
  if $y \in \mathcal{O}$ **then**
    return x_d;
  end
end

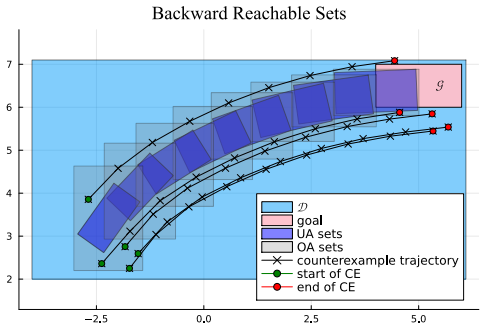


Figure 3: Comparison with Rober et al. (2023). Trajectories from OA sets fail to reach the goal.

4.4. Checking Goal Reaching Properties

Checking goal-reaching properties, $\mathcal{X}_s \subseteq \bigcup_{t=1}^{k} \mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G})$ (def. 2), reduces to checking if a (convex) polytope is a subset of a (non-convex) union of overlapping polytopes. We introduce an optimization-based approach to test this. The property $P \subseteq Q$ is expressed as $\nexists x.x \in P \wedge x \notin Q$, and a feasible point x is sought. If no such point exists, we conclude $P \subseteq Q$. For goal-reaching, we check $\nexists x.x \in \mathcal{X}_s \wedge x \notin \bigcup_{t=1}^{k} \mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G})$. For polytope $\mathcal{X}_s = \{Cx \leq d\}$ with $C \in R^{m \times n}$, $d \in R^n$, the constraint $x \in \mathcal{X}_s$ is easily expressed. The formula $x \notin \bigcup_{t=1}^{k} \mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G})$ may be encoded into the optimization problem using the method in section 4.3.2 as Q is a union of a union of polytopes: $\bigcup_{t=1}^{k} \mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G}) = \bigcup_{t=1}^{k} \bigcup_{j=1}^{n_s} Ball_j^{(t)}$. The optimization problem is then[2]:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & 0 \\ \text{subject to} \quad & Cx \leq d, \\ & x \in \left(Ball_j^{(t)}\right)^c \quad j = 1 \ldots n_s, t = 1 \ldots k \end{aligned} \tag{4}$$

where each set $\left(Ball_j^{(t)}\right)^c$ is represented by $2n$ halfspaces. The optimization problem is an MILP with $2n \times k \times n_s$ binary variables, where n is the state dimension and k the number of timesteps.

4.5. Sampling and Shape-Matching Heuristics

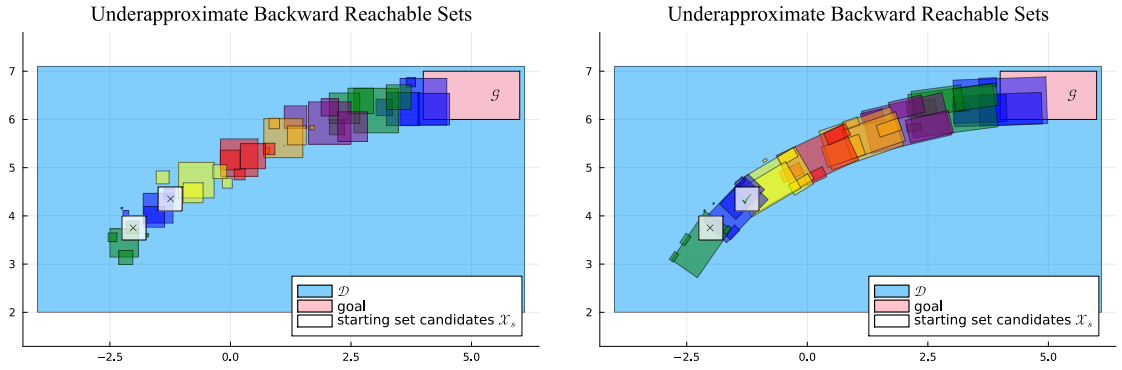
To encode the input constraint 2b, we sample points $x_\text{d} \in \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$. We achieve this through rejection sampling from a dense, space-filling Sobol' sequence (LaValle, 2006) over domain $\mathcal{D}$, where $\mathcal{D}$ is a hyperrectangle assumed to contain the backward reachable sets. The algorithm for rejection sampling is shown in algorithm 2. The Sobol' sequence used has the property that the sampled points are uniformly distributed (Niederreiter, 1992)(Theorem 4.17), a property which is also referred to as being *dense* (LaValle, 2006). A sequence that is dense in set $\mathcal{D}$ contains terms arbitrarily close to every element in $\mathcal{D}$. Thus we eventually sample centers x_d arbitrarily close to every point in $\mathcal{D}$, and therefore in $\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$ as $\forall t.\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G}) \subseteq \mathcal{D}$. Although a dense sampling

2. Note that precisely speaking, using $\geq$ constraints actually enforces $P \subset Q$ rather than the desired $P \subseteq Q$ but a stricter guarantee is acceptable.

strategy has desirable theoretical properties in the limit, we employ two heuristics to maximize the area of the backward reachable set underapproximation for finite sampling. First, we estimate the center of each $Ball_1^{(t)}$ for all t using the mean of a large number samples. Second, we adjust the shape and orientation of the ϵ-ball to match that of the backward reachable set. We do this by calculating the covariance of the centered, sampled data, $\Sigma = cov(X_c)$, extracting the eigenvectors and eigenvalues, $(E, \lambda) = eigdecomp(\Sigma)$, $E \in R^{n \times n}$, $\lambda \in R^n$, and calculating $M = (E * diagm(\sqrt{\lambda_i}))^{-1}$ where $diagm$ indicates construction of a diagonal matrix.

5. Numerical Example

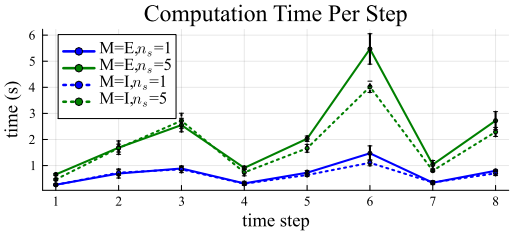
We demonstrate the algorithm on a nonlinear 2-D robot navigation problem from literature (Rober et al., 2023). The state is position $\boldsymbol{x}_t = [x_t, y_t]^T$ and the dynamics f are $[x_{t+1} = v\cos(\theta_t),\ y_{t+1} = v\sin(\theta_t)]^T$ where heading angle $\theta_t = NN(\boldsymbol{x}_t)$ is given by a neural network with 3 layers of 10 neurons and ReLU activations, and v is a constant. A hyperrectangular goal set of $[x_0, y_0]^T \in [4, 6] \times [6, 7]$ and $p = \infty$ was used to compute underapproximate backward reachable sets for 8 timesteps. Figure 3 demonstrates that trajectories beginning in an overapproximate set produced using the methods in (Rober et al., 2023) may fail to reach the goal, in comparison to the sets produced by our method, which are guaranteed to only contain goal-reaching points.



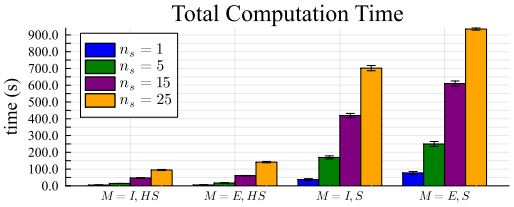
(*a*) $M = I$, $n_s = 5$, Hybrid-symbolic (*b*) $M = E$, $n_s = 5$, Hybrid-symbolic

Figure 4: Underapproximate backward reachable sets for 8 steps, showing two possible starting sets.

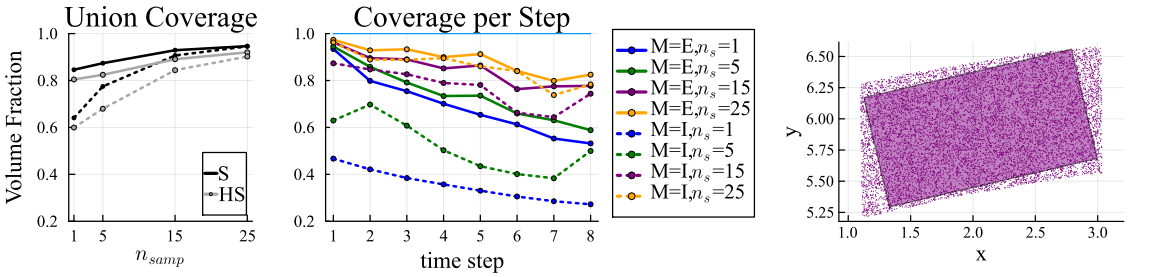
Next, we run an ablation on number of samples with $n_s = \{1, 5, 15, 25\}$, a comparison of our shape matching heuristic, M=E, vs. M=I, and a comparison of pure symbolic reachability ($h = k$) vs. hybrid-symbolic reachability with $h = 3$, wherein we run 5 trials of each configuration. We check the potential starting sets $\mathcal{X}_{s_1} = [-1.5, -1.] \times [4.1, 4.6]$ and $\mathcal{X}_{s_2} = [-2.25, -1.75] \times [3.5, 4]$. Figure 4(*a*) and Figure 4(*b*) show the reachable sets; color denotes timestep. M=E can guarantee goal-reaching from $\mathcal{X}_{s_1}$ for all numbers of samples and from $\mathcal{X}_{s_2}$ for $\{15, 25\}$; M=I can guarantee goal-reaching from $\mathcal{X}_{s_1}$ for $\{15, 25\}$ and from $\mathcal{X}_{s_2}$ for $\{25\}$. In Figure 5(*a*), we show the computation time for the 4 fastest configurations – hybrid symbolic (HS), $n_s \in \{1, 5\}$, $M \in \{E, I\}$. Computation time grows quickly with timestep between t=1-3 and between t=4-6 but the HS approach keeps average total computation time for n_s=1 to $4.95s \pm .14s$ for M=I and $5.51s \pm 0.36s$ for M=E. Checking starting sets and time to bound the NN are not included in runtime. Checking starting sets takes on average at most $0.3s \pm 0.05s$ over all trials in the ablation. NN bounding takes on average $0.06s \pm 0.03s$ with maximum observed $0.3s$ over 1000



(*a*) Hybrid-Symbolic is efficient.



(*b*) Scalability Evaluation



(*c*) (left) Symbolic is less conservative than hybrid symbolic; (right) Coverage decreases with each step (symbolic shown).

(*d*) Visualizing 75% coverage of BRS for n_s=1, $t = 3$, M=E

Figure 5: Underapproximation conservatism and computation time.

trials. Figure 5(*b*) evaluates the method's scalability. Notably, solving with M=E takes more time; we hypothesize that this is due to non-axis-aligned constraints generating a less sparse optimization problem.

Figure 5(*c*) (left) displays the fraction of $\bigcup_{t=1}^{8} \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$ that is covered by underapproximate sets.[3] We measure the coverage using volume fraction (vf) of 10,000 uniform random samples per timestep, $vf = \sum_i^{ke4} \mathbb{I}\{\bigvee_t x_i \in \hat{\mathcal{R}}^{f_{cl}}_{(-t)}(\mathcal{G})\}/(k * 1e4)$. Using M=E, (S or HS), results in high coverage (85%, 80%, respectively) with just 1 sample. In fig. 5(*c*) (right) we display coverage at each timestep $t \in 1 \ldots k$, demonstrating that conservatism increases with k as aproximation error accumulates. Coverage is visualized for a single timestep in Figure 5(*d*). More details in appendix E.

6. Discussion and Conclusion

To the best of the author's knowledge, this paper presents the first algorithm for computing underapproximate backward reachable sets of nonlinear discrete-time neural feedback loops. The algorithm allows one to check goal reaching properties for a class of learning enabled systems that was not previously possible. The soundness of the algorithm is rigorously analyzed in Section 4, and a numerical example is presented in Section 5.

The main limitation of the methodology is scalability; similar to other verification literature. The algorithm can only analyze a finite number of timesteps before the MILP becomes too large to solve. Future work will address this issue. However, Figure 5(*c*) demonstrates that a limited number of samples combined with a shape-matching heuristic, which is fast to run, can cover most of the backward reachable set. Lastly, future work will test algorithm performance over a larger number of example problems. Overall, our work advances the state of the art in verification of learning-enabled systems and lays theoretical groundwork for future algorithms.

3. We omit error bars as they are on the order of 1e-5 or smaller.

Acknowledgements

The authors would like to thank Professor Christiane Lemieux for helpful discussion on dense sequences. This work was partially supported by Digital Futures, FMV and Sweden's Innovation Agency Vinnova through the SHARCEX grant and by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

Alessandro Abate, Matthias Althoff, Lei Bu, Gidon Ernst, Goran Frehse, Luca Geretti, Taylor T Johnson, Claudio Menghi, Stefan Mitsch, Stefan Schupp, et al. The arch-comp friendly verification competition for continuous and hybrid systems. In *International TOOLympics Challenge*, pages 1–37. Springer, 2024.

Tianxu An, Joonho Lee, Marko Bjelonic, Flavio De Vincenti, and Marco Hutter. Scalable multi-robot cooperation for multi-goal tasks using reinforcement learning. *IEEE Robotics and Automation Letters*, 2024.

Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25*, pages 258–263. Springer, 2013.

Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. In *UAI*, volume 1, page 3, 2018.

Michael Everett, Golnaz Habibi, Chuangchuang Sun, and Jonathan P How. Reachability analysis of neural feedback loops. *IEEE Access*, 9:163938–163953, 2021.

Marius Kloetzer and Calin Belta. Reachability analysis of multi-affine systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 348–362. Springer, 2006.

Suhas Kotha, Christopher Brix, J Zico Kolter, Krishnamurthy Dvijotham, and Huan Zhang. Provably bounding neural network preimages. *Advances in Neural Information Processing Systems*, 36:80270–80290, 2023.

Maria Krinner, Angel Romero, Leonard Bauersfeld, Melanie Zeilinger, Andrea Carron, and Davide Scaramuzza. Mpcc++: Model predictive contouring control for time-optimal flight with safety constraints. *arXiv preprint arXiv:2403.17551*, 2024.

Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

Joonho Lee, Marko Bjelonic, Alexander Reske, Lorenz Wellhausen, Takahiro Miki, and Marco Hutter. Learning robust autonomous navigation and locomotion for wheeled-legged robots. *Science Robotics*, 9(89):eadi9641, 2024.

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

Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. SIAM, 1992.

Nicholas Rober, Sydney M Katz, Chelsea Sidrane, Esen Yel, Michael Everett, Mykel J Kochenderfer, and Jonathan P How. Backward reachability analysis of neural feedback loops: Techniques for linear and nonlinear systems. *IEEE Open Journal of Control Systems*, 2:108–124, 2023.

Chelsea Sidrane, Sydney Katz, Anthony Corso, and Mykel J Kochenderfer. Verifying inverse model neural networks. *arXiv preprint arXiv:2202.02429*, 2022a.

Chelsea Sidrane, Amir Maleki, Ahmed Irfan, and Mykel J Kochenderfer. Overt: An algorithm for safety verification of neural network control policies for nonlinear systems. *Journal of Machine Learning Research*, 23(117):1–45, 2022b.

Chelsea Rose Sidrane. *Neural Network Verification for Nonlinear Systems*. Stanford University, 2022.

Yunlong Song, Angel Romero, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Science Robotics*, 8(82):eadg1462, 2023.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Joseph A Vincent and Mac Schwager. Reachable polyhedral marching (rpm): A safety verification algorithm for robotic systems with deep neural network components. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9029–9035. IEEE, 2021.

Mark Wetzlinger and Matthias Althoff. Backward reachability analysis of perturbed continuous-time linear systems using set propagation. *arXiv preprint arXiv:2310.19083*, 2023.

Xiyue Zhang, Benjie Wang, Marta Kwiatkowska, and Huan Zhang. Premap: A unifying preimage approximation framework for neural networks. *arXiv preprint arXiv:2408.09262*, 2024.

Yuhao Zhang, Hang Zhang, and Xiangru Xu. Backward reachability analysis of neural feedback systems using hybrid zonotopes. *IEEE Control Systems Letters*, 7:2779–2784, 2023.

Appendix A. Ext. on Theoretical Analysis of Computing a Single Norm Ball

Here we present the complete proof of lemma 4. We define the following supporting definition and lemma.

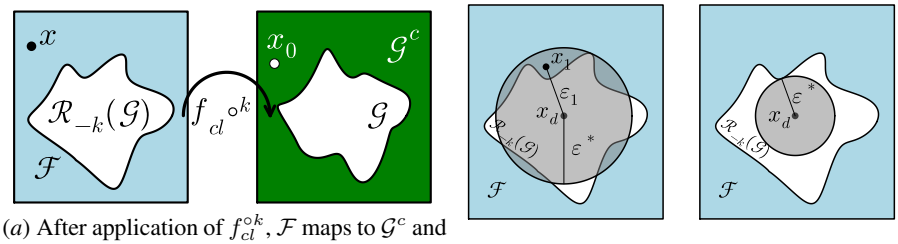
Definition 10 *The **closure** of a set S is the union of S and its boundary ∂S: $\mathbf{cl}S = S \bigcup \partial S$ or equivalently, the intersection of all closed sets containing S.*

Lemma 11 *If A is an open set and $\mathbf{cl}B$ is a closed set, $A \subseteq \mathbf{cl}B \implies \mathbf{cl}A \subseteq \mathbf{cl}B$.*

Proof By definition 10, $\mathbf{cl}A$ is a subset of any closed set containing A. $\mathbf{cl}B$ is a closed set containing A, therefore $\mathbf{cl}A \subseteq \mathbf{cl}B$. ■

And we reproduce lemma 4 and fig. 2 for context:

$$Ball_p(x_d, \epsilon^*, M) \subseteq_{\partial c} \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$$



(a) After application of $f_{cl}^{\circ k}$, $\mathcal{F}$ maps to $\mathcal{G}^c$ and $\mathcal{R}_{-k}(\mathcal{G})$ maps to $\mathcal{G}$.

(b) Contradiction

(c) Optimum

Figure 6: Reproduction of figures 2(a), 2(b) and 2(c).

Proof To demonstrate that lemma 4 is true, we first show that $Ball_p(x_d, \epsilon^*, M) \subseteq \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$ and then we show that $\partial Ball_p(x_d, \epsilon^*, M) \bigcap \partial \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G}) \neq \emptyset$.

Showing the subset property. First assume

$$\exists x.x \in \mathbf{int}Ball_p(x_d, \epsilon^*, M) \bigwedge x \notin \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$$

Such a point is illustrated in fig. 6(a) and would not map into $\mathcal{G}$ as illustrated. The proposition $x \notin \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$ may equivalently expressed as $x \in (\mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G}))^c$ and note that we denote $(\mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G}))^c$ as $\mathcal{F}$ in fig. 6 as it is the interior of the feasible set of the optimization problem.

If $\exists x_1.x_1 \in (\mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G}))^c \bigwedge x_1 \in \mathbf{int}Ball_p(x_d, \epsilon^*, M)$, we can then define its distance to x_d to be $\epsilon_1 = ||x_1 - x_d||_p$, as illustrated in fig. 6(b). It follows that $\epsilon_1 < \epsilon^*$ because all points $\in \mathbf{int}Ball_p(x_d, \epsilon^*, M)$ will have distance to the center less than the radius ϵ^* (see fig. 6(b)). Because there exists a point with smaller ϵ, the distance ϵ^* is therefore not the minimizer of eq. (2), presenting a contradiction. Therefore we can assert $\nexists x.x \in (\mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G}))^c \bigwedge x \in \mathbf{int}Ball_p(x_d, \epsilon^*, M)$. Note that both sets are open and we have not precluded their boundaries from overlapping. It follows that all points in the ball lie inside the backward reachable set, i.e.,

$$\forall x.x \in \mathbf{int}Ball_p(x_d, \epsilon^*, M) \implies x \in \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$$

By lemma 11 and the fact that $\mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$ is a closed set, we can say that $\mathbf{cl}(\mathbf{int}Ball_p(x_d, \epsilon^*, M)) \subseteq \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$ or rather that $Ball_p(x_d, \epsilon^*, M) \subseteq \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$.

Showing the Boundary Intersection Property. However, now two scenarios remain, (1) $Ball_p(x_d, \epsilon^*, M) \subseteq \mathbf{int}\mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$, or (2) $Ball_p(x_d, \epsilon^*, M) \subseteq_{\partial c} \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$. We show through contradiction that (2) must be the case.

Assume (1). In this case, constraint eq. (2d) that $x_0 \notin \mathcal{G}$ could not hold, because $\forall x.x \subseteq \mathbf{int}\mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G}) \implies x_0 \in \mathcal{G}$ by eq. (1). This is a contradiction. See fig. 6(*a*) for illustration. Therefore, we can conclude (2) $Ball_p(x_d, \epsilon^*, M) \subseteq_{\partial c} \mathcal{R}^{f_{cl}}_{(-k)}(\mathcal{G})$. See fig. 6(*c*) for an illustration. ■

Therefore, we can state that $Ball_p(x_d, \epsilon^*, M)$ is the largest possible p-norm ball under transformation M underapproximation of the backwards reachable set centered at x_d.

Appendix B. Function Abstraction

Previous literature has developed overapproximate forward reachability algorithms for nonlinear neural feedback loops (Sidrane et al., 2022a) that involve overapproximation of the dynamics function $f(x, u)$. To summarize, $f(x, u)$ is abstracted first through re-writing multi-dimensional nonlinear functions into one-dimensional nonlinear functions and multidimensional affine functions. Then, piecewise linear upper and lower bounds $\forall w \in \mathcal{S}.f_{i_{LB}}(w) \leq f_i(w) \leq f_{i_{UB}}(w)$ are constructed for each nonlinear function f_i resulting from rewriting. The process abstracts nonlinear function f to a piecewise-linear **multi-valued function** $\hat{f}$ that **overapproximates** the dynamics function f.

Appendix C. Mixed-Integer Encoding

For ReLU activations in the neural network controller, we use the following constraints (Tjeng et al., 2017):

$$t \geq 0 \bigwedge t \geq x \bigwedge t \leq u\delta \bigwedge t \leq x - l(1-\delta) \bigwedge \delta \in \{0, 1\} \tag{5}$$

where $x \in [l, u]$ and $\delta \in \{0, 1\}$.

The abstraction $\hat{f}$ contains functions $t = \max^m_{i=1}(x_i)$ and $t = \min^m_{i=1}(x_i)$. The $\max(x_i)$ functions with > 2 arguments are encoded using the following constraints (Tjeng et al., 2017):

$$x_i \leq t \leq x_i + (u_{\max_i} - l_i)(1 - \delta_i) \bigwedge \delta_1 + \cdots + \delta_m = 1 \bigwedge \delta \in \{0, 1\}^m \tag{6}$$

where $x_i \in [l_i, u_i]$, and $u_{\max_i} = \max_{j \neq i} u_j$. The $\min_i(x_i)$ functions can equivalently be written $-\max(-x_i)$) and encoded using the constraints in 6. When there are only two operands to a max or min operator, only a single binary variable is used.

Note that the number of variables in the encoding of the dynamics may be adjusted by tuning the tightness of the approximation of f. The solve time of the reachable set computation problem is also affected by the tightness of the bounds $[l, u]$ that appear above for each neuron in the control network. In our algorithm, we solve a mixed-integer optimization problem to compute each set of neuron bounds. To bound neuron z^j_i, we solve two MIPs over domain $\mathcal{D}$ with objectives $\{\min z^j_i, \max z^j_i\}$ where all layers $1 \ldots i-1$ are encoded. ReLU activations are encoded as constraints in the MIP using 5.

Appendix D. Ext. on Theoretical Analysis of Using Abstraction

In this appendix we present proofs of lemma 7 and lemma 8. We begin with a proof of lemma 7, which we reproduce for context:
If $\hat{f} : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X} \subseteq R^d$, $\mathcal{Y} \subseteq R^k$ and $\mathcal{Y} = \hat{f}(\mathcal{X})$ is a multivalued function that over-approximates the (single-valued) function $f : \mathcal{X} \to \mathcal{Z}$, where $\mathcal{Z} \subseteq R^k$, and $\mathcal{Z} = f(\mathcal{X})$, then $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{Z}) \subseteq \mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{Z})$.
Proof If $\hat{f}$ is a multivalued function overapproximation of f, both defined over $\mathcal{X}$, this means that for every $x \in \mathcal{X}$, there exists y such that $(x, y) \in f$ and $(x, y) \in \hat{f}$, and there may exist other y_i such that multiple tuples $(x, y_1), (x, y_2), \ldots$ may belong to $\hat{f}$. In other words, $\mathcal{Z} \subseteq \mathcal{Y}$. Note that due to the definition of multivalued function overapproximation,
$\nexists x \in \mathcal{X}.\exists y_1(x, y_1) \in \hat{f} \wedge \nexists y_2(x, y_2) \in f$.

From definition 1, we can say for the overapproximation $\hat{f}$ that $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{Y}) = \mathcal{X}$. Considering that $\mathcal{Z} \subseteq \mathcal{Y}$, we can assert $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{Z}) \subseteq \mathcal{X}$. Again from theorem 1, we can state for f that $\mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{Z}) = \mathcal{X}$, and therefore that $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{Z}) \subseteq \mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{Z})$. ■

We reproduce fig. 2(*d*) in fig. 7 to illustrate the relevant sets.

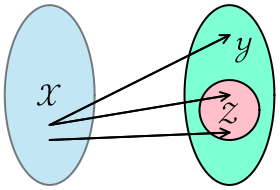


Figure 7: Visual illustration of lemma 7, which demonstrates how we are able to compute underapproximate backward reachable sets from function overapproximations.

Next, we present a proof of lemma 8, which is reproduced for context: $\forall t \in 1 \ldots k.\mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$.
Proof We show this by induction. For the base case, corresponding to the first reachable set backward from goal set $\mathcal{G}$, consider the setting of lemma 7, but now take $\mathcal{Z} = \mathcal{G}$ and $\mathcal{X} = \mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{G})$. We can then assert that $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{G})$.
For the inductive case, we assume that we have two sets such that for some arbitrary t, the following holds:

$$\mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G}) \tag{7}$$

We then show that $\mathcal{R}^{\hat{f}_{cl}}_{(-t-1)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t-1)}(\mathcal{G})$. The backward reachable set $\mathcal{R}^{f_{cl}}_{(-t-1)}(\mathcal{G})$ may equivalently be written $\mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G}))$. Invoking lemma 7, we can state that the one-step backward reachable set of $\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$ under $\hat{f}$ is a subset of that under f:

$$\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})) \subseteq \mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})) \tag{8}$$

And then invoking our assumption in eq. (7), we can state

$$\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G})) \subseteq \mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})) \tag{9}$$

Putting eq. (8) and eq. (9) together, we can then state that $\mathcal{R}^{\hat{f}_{cl}}_{(-1)}(\mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G})) \subseteq \mathcal{R}^{f_{cl}}_{(-1)}(\mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G}))$ and then rewriting using typical convention, that

$$\mathcal{R}^{\hat{f}_{cl}}_{(-t-1)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t-1)}(\mathcal{G})$$

Thus the induction has been shown and we can state that

$$\forall t \in 1 \dots k. \mathcal{R}^{\hat{f}_{cl}}_{(-t)}(\mathcal{G}) \subseteq \mathcal{R}^{f_{cl}}_{(-t)}(\mathcal{G})$$

■

Appendix E. Further Experiment Details

For the experiments presented here, we estimate the center of the backwards reachable set using the mean of 1000 samples. If the mean is not in the backward reachable set, the point is perturbed to find a nearby point that is in the backward reachable set. We use a relative error tolerance of $1e^{-6}$ to encode f with number of PWL segments automatically chosen to meet this error level. All experiment times reported are runtime, and do not include precompilation, as is standard practice when using Julia. Experiments were run on an Apple MacBook M2 Pro with 10 CPU cores and 32GB RAM.