

# A General Verification Framework for Dynamical and Control Models via Certificate Synthesis

Alec Edwards<sup>1</sup>, Andrea Peruffo<sup>2</sup>, Alessandro Abate<sup>1</sup>

**Abstract**—An emerging branch of control theory specialises in *certificate learning*, concerning the specification of a desired (possibly complex) system behaviour for an autonomous or control model, which is then analytically verified by means of a function-based proof. However, the synthesis of controllers abiding by these complex requirements is in general a non-trivial task and may elude the most expert control engineers. This results in a need for automatic techniques that are able to design controllers and to analyse a wide range of elaborate specifications. In this paper, we provide a general framework to encode system specifications and define corresponding certificates, and we present an automated approach to formally synthesise controllers and certificates. Our approach contributes to the broad field of safe learning for control, exploiting the flexibility of neural networks to provide candidate control and certificate functions, whilst using SMT-solvers to offer a formal guarantee of correctness. We test our framework by developing a prototype software tool, and assess its efficacy at verification via control and certificate synthesis over a large and varied suite of benchmarks.

**Index Terms**—control systems; formal verification; machine learning; Lyapunov stability; safety; reachability

## I. INTRODUCTION

THE analysis of the behaviour of continuous-time dynamical systems focuses on a wide range of properties, which are themselves suitable for an even wider range of applications. From arguably the most common property, (asymptotic) stability, namely the convergence of trajectories to an equilibrium [1]; to safety, namely the avoidance of an unsafe region of the state space at all time [2]; to its dual, reachability, that is the hitting of a target region of the state space in finite time [3]. As we shall see, with the combination and the slight modification of these basic properties (we shall alternatively denote them as specifications or requirements), an engineer can in practice design a broad range of desired dynamical behaviours for any model at hand.

Given a model of a dynamical system, such spectrum of properties can be investigated from different perspectives and with diverse approaches: either analytical (e.g., via local linearisation and eigenvalues computation) [1], or computational

ones (e.g., via dynamic flow propagation or via reach-set computation). In general, non-linearity in dynamical models is difficult to deal with: on account of this, approaches that are “indirect” or “sufficient” can be successful: a proof that the system actually fulfils a given requirement can be offered in the form of a *certificate*: the onus is thus to find, or to synthesise, a real-valued function defined over the state space with proper characteristics. A celebrated instance of indirect methods is the synthesis of Lyapunov functions [4], whereby one ought to hand-craft a bespoke energy function, oftentimes based on intuition and on physical properties of the underlying dynamical model.

In recent years numerical optimisation methods have automated the synthesis of certificates, employing *templates*, i.e. candidate functions where only the coefficients (parameters) ought to be determined. Commonly, the choice falls onto polynomial templates framed as sum-of-squares convex problems [5], [6], [7], [8], which are known to admit globally optimal solutions. However, these techniques operate solely on models with polynomial dynamics and various convexity assumptions. Alternative formulations include linear programs [9] [10], [11], and semi-algebraic systems [12] [13], all of which raise structural requirements on the dynamical models at hand.

Despite the usefulness of the mentioned synthesis approaches, they are numerically sensitive and generally unsound, and thus undesirable for robust solutions and safety-critical applications [14], [15], [16]. Consequently, in recent years interest has grown in approaches for synthesis that can yield *provably-correct* certificates, much in the same line of research as *correct-by-design* control synthesis [17], [18]. A powerful technique to reason formally about correctness involves SMT-solving [19]. SAT-modulo-theory (SMT) extends satisfiability (SAT) solving, namely finding feasible binary assignments to variables that make Boolean functions evaluate to true, to richer theories, such as linear or non-linear arithmetic over real numbers.

SMT can be in particular leveraged for synthesis tasks: [20], [21], [22], [23] have investigated the formal synthesis of controllers. *Inductive* approaches [24], leveraging SMT, have been used to synthesise certificates for dynamical models. Such techniques have been used first for stability certification of dynamical models using polynomial Lyapunov functions and later extended to more general reach-avoid requirements [25], [26], [27], [20]. Related to SMT-based solutions, approaches that formulate synthesis problems as a mixed-integer linear

This work was supported by the European Research Council through the SENTIENT project (ERC-2017-STG #755953) and the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems (EP/S024050/1).

<sup>1</sup>University of Oxford, UK -

{alec.edwards, alessandro.abate}@cs.ox.ac.uk

<sup>2</sup>Delft University of Technology, NL - a.peruffo@tudelft.nl

programs (MILP) have also been used to synthesise provably-correct Lyapunov functions [28] [29] for stability analysis, encompassing linear matrix inequalities for uncertain systems [30], and barrier certificates for safety [31], [32], [33]. Notably, mixed-integer problems also encompass candidates in the form of neural networks with ReLU activation functions, and may employ an optimisation engine like Gurobi [34] to certify the soundness of the proposed functions [35].

**Related work** The flexibility of neural networks has permeated throughout fields, including their use for synthesis of Lyapunov-like functions. For instance, [36], [37], [38] describe generally unsound procedures for gradient descent-based training of a Lyapunov neural network (LNN). Sound, counter-example based techniques are proposed in e.g. [25], [39], [40], [41], specifically for Lyapunov functions, and solely for barrier certificates in e.g. [42], [43], [44]. The choice of SMT solver depends on the models under consideration and desired certificate template: Z3 [45] handles polynomial functions, dReal [46] and iSat3 [47] enable analysis of non-polynomial functions as well as polynomials. The synthesis of certificates includes more complex properties, as proposed in [48], [49] where bespoke genetic algorithms are leveraged to generate *reach-while-stay* and *reach-and-stay-while-stay* functions, alongside controllers, for hybrid systems. Certificates for reach-avoid properties have been previously synthesised using counterexample-based approaches [20]. Meanwhile reach-avoid-stay properties and corresponding certificates have been well studied from a theoretical perspective [50], [51]. In this work, we collate certificates for these more complex properties, and categorise them in order to unify them within simpler certificates for stability, reachability and safety. We also generalise the concept of reach-avoid-stay property by separating allowing the “stay” set to be different from the “reach” set. The interested reader may find a survey on neural certificates with application in control synthesis and robotics in [52].

**Contributions** We summarise our contributions as follows:

- We collate and add to existing certificates across literature for nonlinear continuous-time dynamical models.
- We categorise the properties these certify into a simplified and general framework, which we newly describe through notions *arrive*, *avoid* and *remain*.
- We describe a unified algorithm to concurrently synthesise *both* controllers and certificates in parallel for dynamical models, to prove they satisfy these properties.
- We implement our framework<sup>1</sup> on top of the computational library Fossil [53], offering a new prototype software tool that can verify general *arrive*, *avoid* and *remain* properties for nonlinear control models using controllers and certificates based on neural networks.

**Organisation** This manuscript is organised as follows. Section II provides relevant background information and notation used across this work. In Section III, we describe a range of properties for dynamical models and certificates which prove that they hold. Next, we describe a unified and computationally correct algorithm for synthesising these certificates in Section

IV. We present experimental results for a prototype tool to synthesise these certificates in Section V, before discussing the limitations of our framework in Section VI and providing concluding remarks in Section VII. Finally, we outline the proofs of all theorems in Appendix I, and we reserve a discussion on the broader taxonomic connections of our work to Appendix II.

## II. PRELIMINARIES

### A. Dynamical Models

We denote the set of positive real numbers and its extended version as  $\mathbb{R}_+$  and  $\overline{\mathbb{R}} = \mathbb{R}_+ \cup \{+\infty\}$ , respectively. A function is said to be of class  $\mathcal{C}^1$  if its first derivative exists and is continuous. Let us consider models described by

$$\dot{\xi}(t) = f(\xi(t), u(t)), \quad \xi(t_0) = x_0 \in \mathcal{X}_I \subseteq \mathcal{X} \quad (1)$$

where  $x \in \mathcal{X} \subseteq \mathbb{R}^n$  is the state of the system,  $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}^n$  is a Lipschitz-continuous vector field describing the model dynamics. We refer to these models as control models. We denote a trajectory over a time horizon  $T \in \overline{\mathbb{R}}$  as  $\xi(t) : [0, T] \rightarrow \mathbb{R}^n$ , where  $\xi(t)$  admits a time derivative everywhere, and such that  $\dot{\xi}(t) = f(\xi(t), u(t))$  and  $\xi(t) \in \mathcal{X}$ , namely the trajectory is a solution of the model in (1). Finally,  $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$  is the input and  $\mathcal{X}_I$  denotes the set of initial conditions. Once a state-feedback controller  $u(t) = k(\xi(t))$  has been specified, we may interpret the dynamics described by (1) as those of a closed-loop model, as follows

$$\dot{\xi}(t) = f(\xi(t)), \quad \xi(t_0) = x_0 \in \mathcal{X}_I \subseteq \mathcal{X}. \quad (2)$$

We refer to models of this kind simply as autonomous dynamical models. We denote with  $x^*$  an equilibrium point of (2), namely where  $f(x^*) = 0$ .

### B. Systems and Properties

The goal of this work is to find a feedback controller, namely a signal  $u(t)$  in time, such that the dynamics above satisfy some desired temporal requirements (e.g., safety), or to show that given closed-loop (autonomous) dynamics are endowed with some given property (e.g., asymptotic stability).

We interpret properties of dynamical models in (2) in terms of their trajectories, and of binary relations that these trajectories have with given sets within the state space  $\mathcal{X}$ . To this end, we now introduce the notations and the semantics of the sets characterising properties of dynamical models. We denote as  $\mathcal{X}_U$  an unsafe set, indicating a region of the state space where the system’s trajectories should avoid;  $\mathcal{X}_G$  represents a goal set, indicating the region that the system’s trajectories should enter; and  $\mathcal{X}_F$  represents a final set, indicating a set where the system’s trajectories should remain for all times after arriving at the goal set. These sets will be employed in the next section for the definition of properties, as depicted in Fig. 1. Implicitly, we consider that the unsafe and final sets are disjoint, i.e.  $\mathcal{X}_U \cap \mathcal{X}_F = \emptyset$  and that the goal set is contained within the final set, i.e.  $\mathcal{X}_G \subset \mathcal{X}_F$ . Additional topological properties of sets will be clarified later, within formal statements. Given a set  $S$  in a domain  $\mathcal{X}$ , we denote by  $S^c$  its complement, i.e.

<sup>1</sup>Available at <https://github.com/oxford-oxcav/fossil>

$\mathcal{X} \setminus S$ , and by  $\text{int}(S)$  its interior, namely the set without its border, i.e.  $\text{int}(S) = S \setminus \partial S$ .

We consider a set  $S$  to be (forward) *invariant* if at some initial time  $t_0$ ,  $\xi(t_0) \in S$  implies that for all  $t > t_0$ ,  $\xi(t) \in S$  [2]. We consider a set  $S_A$  to be attracting [2] with *region of attraction*  $S_B$  if for some initial time  $t_0$  and any initial state  $\xi(t_0) \in S_B$ , the trajectory  $\xi(t)$  converges to  $S_A$  as  $t \rightarrow \infty$ , i.e. if  $\lim_{t \rightarrow \infty} \text{dist}(\xi(t), S_A) = 0$ .

### C. Neural Networks

Denote a neural network  $\mathcal{N}$  with input layer  $z_0 \in \mathbb{R}^n$ , corresponding to the dimension of the dynamical model in (2). This is followed by  $k$  hidden layers  $z_1, \dots, z_k$  with dimensions  $h_1, \dots, h_k$  respectively, and finally followed by an output layer  $z_{k+1} \in \mathbb{R}^d$ . In this work,  $d \in \{1, m\}$ , where  $d = 1$  corresponds to a scalar-valued certificate, whereas  $d = m$  is used for a state-feedback controller with  $m$  control variables.

To each hidden or output layer with index  $i$  are associated matrices of weights  $W_i \in \mathbb{R}^{h_i \times h_{i-1}}$  and a vectors of biases  $b_i \in \mathbb{R}^{h_i}$  [54]. Every  $i$ -th hidden layer is associated with an activation function  $\sigma_i: \mathbb{R} \rightarrow \mathbb{R}$ . The valuation of output and hidden layers is given by

$$z_i = \sigma_i(W_i \cdot z_{i-1} + b_i), \quad i = 1, \dots, k, \quad (3)$$

$$z_{k+1} = W_{k+1} \cdot z_k + b_{k+1}, \quad (4)$$

where each  $\sigma_i$  is applied element-wise to its  $h_i$ -dimensional argument.

## III. PROPERTIES AND CERTIFICATES

We present a number of properties (or requirements) for dynamical models defined over their trajectories, alongside definitions of corresponding certificates, whose existence serve as *sufficient conditions* for the satisfaction of the desired properties. We focus on continuous-time models; while the presented properties may be seamlessly applied to discrete time models, the corresponding certificates would not necessarily align with the presentation of the work, hence we omit their discussion as outside the scope of this work. The proofs of the theorems relating certificates to dynamical properties are outlined in Appendix I. Several properties (and corresponding certificates) are ubiquitous across the control theory literature, whilst others have been more recently introduced or inherited from analogues in formal verification. Further, we include a practical variant on Lyapunov functions to allow for “constrained” local stability and an original certificate called Reach-Avoid-Remain, and suggest that more can be obtained in a modular, composable fashion. We emphasise that while many of these properties are similar to each other, they are subtly, and importantly, different and distinguished. Later in the section we provide a summary and clarification on their distinguishing features, which may also be seen in Fig. 1.

### A. Stability

Stability is the most-studied property of dynamical models, and many definitions of this property exist. Stability is most commonly characterised in a Lyapunov (asymptotic) sense

[1], namely in terms of the distance of a trajectory from the equilibrium point. Conversely, in order to achieve a consistent characterisation of properties in terms of set containment, here we characterise stability as follows:

$$\exists \mathcal{X}_I : \forall \xi(t_0) \in \mathcal{X}_I, \exists T \in \overline{\mathbb{R}}, \forall \tau \geq T, \xi(\tau) \in \{x^*\}, \quad (5)$$

where  $\mathcal{X}_I$  has non empty interior. In words, there exists some initial set  $\mathcal{X}_I$  such that for all trajectories initialised in  $\mathcal{X}_I$ , there exists a time instant  $T$  (possibly at infinity) when the trajectory reaches the equilibrium state  $x^*$  and remains there for all times after  $T$ . A model can be proven to satisfy this property using a Lyapunov function, which we introduce next.

**Certificate 1 (Lyapunov Function):** Given a model  $f$  with unique equilibrium point  $x^* \in \mathcal{X}$ , consider a function  $V : \mathcal{X} \subset \mathbb{R}^n \rightarrow \mathbb{R}, V \in \mathcal{C}^1$ .  $V$  is a Lyapunov function if:

$$V(x^*) = 0, \quad (6a)$$

$$V(x) > 0 \quad \forall x \in \mathcal{X} \setminus \{x^*\}, \quad (6b)$$

$$\dot{V}(x) = \langle \nabla V(x), f(x) \rangle < 0 \quad \forall x \in \mathcal{X} \setminus \{x^*\}. \quad (6c)$$

**Theorem 1 (Stability):** Given a model (2), if a Lyapunov function exists, then (5) holds for some set of initial conditions  $\mathcal{X}_I$ . ■

Let us clarify the issue of finding  $\mathcal{X}_I$  in the next section.

### B. Region of Attraction

Thm. 1 proves the existence of some region of the state space in which initialised trajectories will converge asymptotically towards the origin – this region is known as a *region of attraction* (ROA). In general, Lyapunov functions merely prove the existence of a region of attraction within the state space, without additional information about its size or shape. Lyapunov functions may prove global asymptotic stability under additional conditions, but these can be difficult to synthesise and verify automatically for some models. Here, we offer a certificate the acts as a “middle ground” between local and global asymptotic stability.

Proving that all trajectories initialised within a given  $\mathcal{X}_I$  are stable amounts to proving that  $\mathcal{X}_I$  is contained wholly within a sub-level set a Lyapunov function, and that the Lyapunov conditions in (6) hold over this entire sub-level set. This is treated in the following equation and corollary.

First, we modify (5) to now require a specified set of initial states  $\mathcal{X}_I$ , which should be a region of attraction for an equilibrium point, as follows:

$$\forall \xi(t_0) \in \mathcal{X}_I, \exists T \in \overline{\mathbb{R}}, \forall \tau \geq T, \xi(\tau) \in \{x^*\}. \quad (7)$$

We can certify that an autonomous model satisfies this property using the following certificate.

**Certificate 2 (ROA Certificate):** Let a dynamical model  $f$  be given with unique equilibrium point  $x^* \in \mathcal{X}$ . A Lyapunov function  $V$  is an ROA certificate if there exists  $\beta$  such that  $\mathcal{X}_I \subset \{x : V(x) \leq \beta\}$  and that the conditions in (6) hold over the set  $\{x : V(x) \leq \beta\}$ . ■

Alternatively, a Lyapunov function can be interpreted as a proof that a region of attraction (here  $\mathcal{X}_I$ ) exists within some larger set (here  $\mathcal{X}$ ), whereas a ROA certificate proves

the converse: that a given initial set  $\mathcal{X}_I$  lies within a larger region of attraction, which is defined by a sublevel set of  $V$ . This certificate offers a more practical guarantee over classical Lyapunov functions: all trajectories initialised within the user-defined set  $\mathcal{X}_I$  indeed converge to  $x^*$ .

**Corollary 2 (Region of Attraction):** Given a model (2), a bounded set of initial conditions  $\mathcal{X}_I$ , and a ROA certificate, then (7) holds. ■

### C. Safety

Safety is another fundamental property we can require from dynamical models. It involves the *avoidance* of some unsafe region: namely, that no trajectory starting from  $\mathcal{X}_I$  may enter the unsafe set  $\mathcal{X}_U$ <sup>2</sup>; formally

$$\forall \xi(t_0) \in \mathcal{X}_I, \forall t \in \mathbb{R}, t \geq t_0, \xi(t) \in \mathcal{X}_U^c. \quad (8)$$

For continuous-time models, safety over an unbounded time horizon can be proved via barrier certificates [55], [7].

**Certificate 3 (Barrier Certificate):** Consider a dynamical model  $f$ , a compact unsafe set  $\mathcal{X}_B$  and compact initial set  $\mathcal{X}_I$ . A function  $B : \mathcal{X} \subset \mathbb{R}^n \rightarrow \mathbb{R}, B \in \mathcal{C}^1$ , is a Barrier certificate if the following holds:

$$B(x) \leq 0 \quad \forall x \in \mathcal{X}_I, \quad (9a)$$

$$B(x) > 0 \quad \forall x \in \mathcal{X}_U, \quad (9b)$$

$$\dot{B}(x) < 0 \quad \forall x \in \{x : B(x) = 0\}. \quad (9c)$$

Many characterisations of barrier certificates exist, to cover different applications or to abide by additional constraints. These include reciprocal [56], high-order (zeroing) [57], or barrier conditions with modifications on the Lie derivative (9c) [55]. These certificates are alike in that they certify safety properties. However, the provided, more conventional barrier certificate formulation is most suitable for this work in terms of simplicity and usability.

**Theorem 3 (Safety):** Given a model (2), a compact domain  $\mathcal{X}$ , a compact initial set  $\mathcal{X}_I \subset \mathcal{X}$  and an unsafe set  $\mathcal{X}_U$ , alongside a barrier certificate, then (8) holds. ■

### D. Stable While Avoid

It is natural to extend the aforementioned notions of stability and safety towards a combination of both, whereby all relevant trajectories converge towards an equilibrium point, while also avoiding a given unsafe set. Such a property is formally described as follows:

$$\forall \xi(t_0) \in \mathcal{X}_I, \exists T \in \mathbb{R}, \forall t \in [t_0, T), \xi(t) \in \mathcal{X}_U^c \\ \wedge \forall \tau \geq T, \xi(\tau) \in \{x^*\}. \quad (10)$$

Since (10) is simply the conjunction of a stability property and a safety property, certifying this is equivalent to concurrently certifying both stability and safety hold: this task can be thus formally tackled by the following corollary.

**Corollary 4 (Stable while avoid (SWA)):** Given a model (2) with unique equilibrium point  $x^* \in \mathcal{X}$ , a compact domain

<sup>2</sup>We shall later draw connections between the concept of safety and the dual notion of (unconstrained) *reachability*. Please refer to the discussions in Section III-H and in Appendix II.

$\mathcal{X}$ , a compact initial set  $\mathcal{X}_I \subset \mathcal{X}$  and an unsafe set  $\mathcal{X}_U$  alongside a ROA certificate  $V$  and barrier certificate  $B$ , then (10) holds. ■

Notice that it is alternatively possible to combine the conditions of (6) and (9) into that of a *single* certificate for stability and safety. Such a function is sometimes referred to as a Lyapunov-Barrier certificate [58], [59]. However, in this work we choose to use two separate functions, as this makes synthesis easier and more modular.

### E. Reach While Avoid

Let us now set asymptotic stability aside, and instead study properties over a finite time horizon: namely, we require that trajectories enter a non-singleton set in finite time. These are reachability-like properties, which require trajectories to reach a region known as a goal set. A reach-while-avoid (RWA) property requires a non-singleton goal set to be reached within a finite time horizon  $T$ , while avoiding an unsafe region; in formal terms,

$$\forall \xi(t_0) \in \mathcal{X}_I, \exists T \in \mathbb{R}, \forall t \in [t_0, T] : \\ \xi(t) \in \mathcal{X}_U^c \wedge \xi(T) \in \mathcal{X}_G. \quad (11)$$

Next, we introduce an RWA certificate to guarantee that this property holds for a model under consideration.

**Certificate 4 (RWA):** Define an unsafe set  $\mathcal{X}_U = \mathcal{X} \setminus \mathcal{X}_S$ , where  $\mathcal{X}_S$  is a compact safe set, a compact initial set  $\mathcal{X}_I \subset \text{int}(\mathcal{X}_S)$ , and a compact goal set  $\mathcal{X}_G \subset \text{int}(\mathcal{X}_S)$  with non-empty interior. A reach-while-avoid (RWA) certificate [49] is a function  $V : \mathbb{R}^n \rightarrow \mathbb{R}, V \in \mathcal{C}^1$ , such that

$$V(x) \leq 0 \quad \forall x \in \mathcal{X}_I, \quad (12a)$$

$$V(x) > 0 \quad \forall x \in \partial \mathcal{X}_S, \quad (12b)$$

$$\dot{V}(x) < 0 \quad \forall x \in \{x \in \mathcal{X}_S | V(x) \leq 0\} \setminus \mathcal{X}_G. \quad (12c)$$

**Theorem 5 (Reach-While-Avoid):** Given a model (2) and a RWA Certificate corresponding to the given sets of interest, then (11) holds. ■

**Remark 6 (Unconstrained Reachability):** Unconstrained reachability can be defined as a special case of RWA, where we set  $\mathcal{X}_U = \emptyset$  (i.e.  $\mathcal{X}_S = \mathcal{X}$ ). Hence, a certificate can be provided accordingly, as special instance of Certificate 4. ■

We emphasise that a RWA certificate does not prove that trajectories will *remain* within the goal set, or that trajectories shall avoid the unsafe set for all (unbounded) time, nor does the specification in (11) indeed encode these requirements. In fact, since the Lie derivative condition (12c) does not hold across the goal set  $\mathcal{X}_G$ , it is possible for trajectories to leave the goal set after entering it, and thereafter possibly enter the unsafe set  $\mathcal{X}_U$ . This alternative, more restrictive scenario is addressed by the next certificate.

### F. Reach-and-Stay While Avoid

A reach-and-stay while avoid (RSWA) property is similar to the RWA property, consisting of RWA with an additional requirement that trajectories remain in the goal set indefinitely



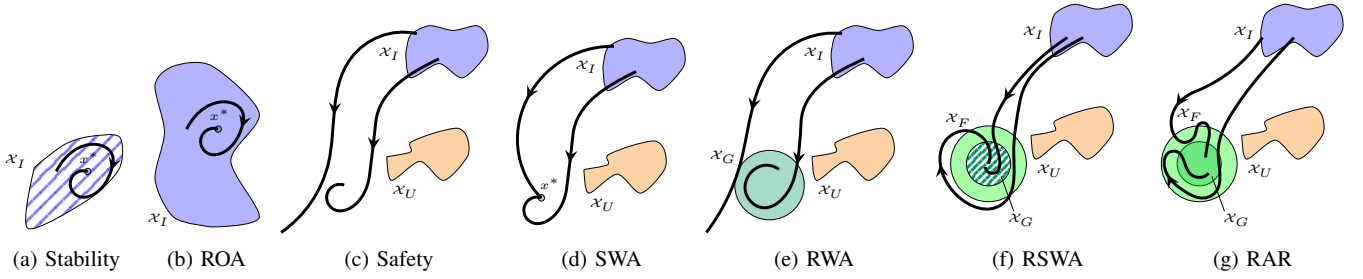


Fig. 1: Pictorial depiction of relevant properties in this work. Here,  $\mathcal{X}_I$  is the initial set,  $\mathcal{X}_U$  the unsafe set ( $\mathcal{X}_S$  is its safe complement),  $\mathcal{X}_G$  the goal/target set,  $\mathcal{X}_F$  the final set. (The entire state space is  $\mathcal{X}$ .) Here, a dashed background denotes that the corresponding set's existence is implied by the corresponding certificate, but that it is not explicitly defined in the property.

after reaching it. Formally, trajectories should satisfy the property

$$\exists \mathcal{X}_G : \forall \xi(t_0) \in \mathcal{X}_I, \exists T \in \mathbb{R}, \forall t \in [t_0, T], \xi(t) \in \mathcal{X}_U^c \wedge \xi(T) \in \mathcal{X}_G \wedge \forall \tau \geq T : \xi(\tau) \in \mathcal{X}_F. \quad (13)$$

Notably, this property does not require that trajectories reach the final set and remain within it, and may enter and leave the final set as long as they eventually remain within the final set. However, in finite time trajectories must reach some subset of the final set a goal set, after which point they must remain within the final set for all time.

**Certificate 5 (RSWA):** Define an unsafe set  $\mathcal{X}_U = \mathcal{X} \setminus \mathcal{X}_S$ , where  $\mathcal{X}_S$  is a compact safe set, then define a compact initial set  $\mathcal{X}_I \subset \text{int}(\mathcal{X}_S)$ , and a compact final set  $\mathcal{X}_F \subset \text{int}(\mathcal{X}_S)$ . A RSWA [49] certificate is a function  $V : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $V \in \mathcal{C}^1$ , that satisfies the following:

$$V(x) \leq 0 \quad \forall x \in \mathcal{X}_I, \quad (14a)$$

$$V(x) > 0 \quad \forall x \in \partial \mathcal{X}_S, \quad (14b)$$

$$\dot{V}(x) < 0 \quad \forall x \in \{x \in \mathcal{X}_S | V(x) \leq 0\} \setminus \mathcal{X}_F, \quad (14c)$$

$$V(x) > \beta \quad \forall x \in \partial \mathcal{X}_F, \quad (14d)$$

$$\dot{V}(x) < 0 \quad \forall x \in \mathcal{X}_F \setminus \text{int}(\{x \in \mathcal{X}_S | V(x) \leq \beta\}), \quad (14e)$$

for some constant  $\beta \in \mathbb{R}$ .

**Theorem 7 (Reach-and-stay while avoid):** Given a model (2) and a certificate corresponding to the given sets of interest, then (13) holds. ■

The sub-level set of  $V$  given by  $\beta$  defines an invariant set contained within the final set, and ensures that trajectories reach this set in finite time without entering an unsafe region. Note that the specification described in (13) - and the corresponding certificate - permit trajectories to enter and leave the final set, as long as trajectories eventually enter a goal set and do not leave the final set again.

### G. Reach, Avoid and Remain

The final property we consider is again similar to the previous *Reach and Stay While Avoid* property, but, as with the ROA certificate, we seek to remove the existential quantifier over the goal set from (13). This means that the Reach Avoid Remain (RAR) property requires that trajectories remain

within a final set after reaching a goal set, but for two given goal and final sets. We express this formally, as follows:

$$\forall \xi(t_0) \in \mathcal{X}_I, \exists T \in \mathbb{R}, \forall t \in [t_0, T] : \xi(t) \in \mathcal{X}_U^c \wedge \xi(T) \in \mathcal{X}_G \wedge \forall \tau \geq T : \xi(\tau) \in \mathcal{X}_F. \quad (15)$$

**Certificate 6 (RAR):** Define an unsafe set  $\mathcal{X}_U = \mathcal{X} \setminus \mathcal{X}_S$ , where  $\mathcal{X}_S$  is a compact safe set, a compact initial set  $\mathcal{X}_I \subset \text{int}(\mathcal{X}_S)$ , a compact final  $\mathcal{X}_F \subset \text{int}(\mathcal{X}_S)$ , and a compact goal set  $\mathcal{X}_G \subset \text{int}(\mathcal{X}_F)$  with non-empty interior. Let  $V : \mathbb{R}^n \rightarrow \mathbb{R}$  be a RWA certificate, and a function  $B : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $B \in \mathcal{C}^1$ , such that:

$$B(x) \leq 0 \quad \forall x \in \mathcal{X}_G, \quad (16a)$$

$$B(x) > 0 \quad \forall x \in \partial \mathcal{X}_F, \quad (16b)$$

$$\dot{B}(x) < 0 \quad \forall x \in \{x : B(x) = 0\}. \quad (16c)$$

The pair  $(V, B)$  define a Reach-Avoid-Remain certificate. ■

As with the Stable-While-Avoid certificate, we choose to formulate this certificate as a pair of separate functions, rather than collapsing the conditions to a single function. This choice, which of course does not affect the soundness of the approach, renders synthesis practically easier and more modular.

**Theorem 8 (Reach-Avoid-Remain):** Given a model (2) and a certificate pair  $V, B$  satisfying the conditions in Certificate 6, then (15) holds. ■

We note that here, the certificate  $B$  is similar to a Barrier certificate as defined in (9), though with  $\mathcal{X}_G$  as the initial set and  $\partial \mathcal{X}_F$  as the unsafe set. We have restated the function in this context for clarity.

### H. Summary and Classification of Properties

So far, we have presented a number of different properties that a dynamical model may conform to. These properties, and the certificates that sufficiently prove them to hold, can be complex and subtly different. However, we observe the following similarities between them:

- All certificates rely on a set on initial conditions  $\mathcal{X}_I$ . In the case of a Lyapunov function, this set is implicitly defined a-posteriori to the synthesis of the certificate.
- $\mathcal{X}_U$  denotes a region trajectories should *avoid* (and thus relate to a safety requirement).
- Either  $\mathcal{X}_G$  or  $\{x^*\}$  denote a region which trajectories should enter or *arrive* at. We leverage this notion to

encompass both finite-time reachability and asymptotic stability, and note that it can be thought as the dual of the *avoid* category.

- Either  $\mathcal{X}_F$  or  $\{x^*\}$  denote a region which trajectories should eventually *remain* in, for all time, as soon as they have *arrived* in a goal set. This notion is thus related to both (forward) set invariance and asymptotically stable equilibria.

Based on these analogies, we introduce three labels *avoid*, *arrive* and *remain*, and assign them to each certificate, in order to clarify their purpose and differences. We portray this relationships in Fig. 2, where the label *arrive* encompasses both stability and finite-time reachability.

We note that this classification does not distinguish between some certificates, which we clarify here. Firstly, as previously mentioned, a Lyapunov function and a ROA certificate differ in that for the latter an initial set is explicitly specified a priori to synthesis. Meanwhile, the SWA, RSWA and RAR certificates satisfy all three labels. Stable while Avoid is easy to distinguish, as it handles asymptotic stability, whereas the others treat finite time reachability. Finally, we differentiate the RSWA and RAR certificates as follows. A RSWA certificate proves that there exists a goal set, contained within a given final set, that trajectories will reach and afterwards never leave the final set. Meanwhile a RAR certificate explicitly defines both the goal set and final set associated with this, allowing for a more elaborate specification.

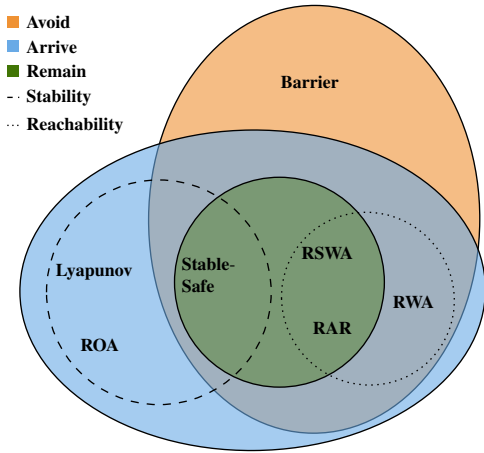


Fig. 2: Euler Diagram depicting the semantic labels, *arrive*, *avoid*, and *remain*, associated with each certificate in this work. By the dashed line, we group properties that exhibit asymptotic stability. By the dotted line, we denote properties that exhibit (finite-time) reachability.

### I. Certificates for Control Models

Much work in the literature of certificate synthesis refers to, e.g., *control* Lyapunov functions and *control* Barrier certificates. In this work, we consider such terms to concern certificates that refer to a model expressed in terms of both state  $x$  and control input  $u$ , as in (1). Existing approaches often

specify a control set, over which the specifications quantify existentially. In other words, they seek certificates for which there always exists a valid control action that allows for the property to hold. A control Lyapunov function therefore proves that there always exists a suitable control input such that the Lyapunov conditions hold, and hence the system is (asymptotically) stabilisable. This approach entails that, after a valid control-certificate is synthesised, control actions can be determined, e.g. by solving an optimisation program over the input space and the found certificate.

We emphasise that this is *not* the approach taken in this work. Instead, we synthesise a feedback control law for the model described by (1), and “apply” this state feedback to obtain a closed-loop model of the form of (2), for which we synthesise a certificate. Whilst we synthesise the control law *concurrently* with the certificate, we do not refer to these as “control certificates”, as in literature. Hence, we verify properties for control models using both a controller and a certificate, and in particular do not delegate the controller synthesis a-posteriori.

## IV. SYNTHESIS OF CERTIFICATES AND CONTROLLERS

Following the introduction of specifications that are salient for verification purposes, and certificates whose existence prove that a given model satisfies these conditions, we describe next a unified, efficient, and sound algorithm for the synthesis of these certificates, for both dynamical and controlled models. The objective of the synthesis task is indeed twofold: we seek a feedback controller  $k(x)$  that ensures a control model satisfies a desired specification, and concurrently we synthesise a certificate  $C(x)$  that serves as a proof that the specification holds. As such, allow us to use  $C(x)$  to refer to each function in a possible pair separately. Call with  $\theta_u$  the parameters of the state feedback law and  $\theta_c$  the parameters of the certificate function. The synthesis task therefore amounts to finding values for the parameters  $\theta_u$  and  $\theta_c$  such that the certificate conditions hold, i.e.,

$$\exists \theta_u, \theta_c \forall x \in \mathcal{X} : \phi(C(x)), \quad (17)$$

where  $\phi(C(x))$  denotes the conditions, related to a given specification (as formalised in the previous section), that the certificate ought to satisfy. Our procedure is based on counter-example guided inductive synthesis (CEGIS) [24], an established approach to formally solving such exists-forall queries. CEGIS consists of two opposing components: a *learner* and a *verifier* (cf. Figure 4), which we detail in turn.

### A. Learner

The learner fulfils the twofold task of designing both a stabilising control law (when required) and the desired certificate, as exemplified in Fig. 3.

In this work, we seek *feedback* control laws and employ neural networks as a general template for these functions: neural architectures allow for a wide variety of control laws, as determined by the choice of their activation functions and of the number of neurons.

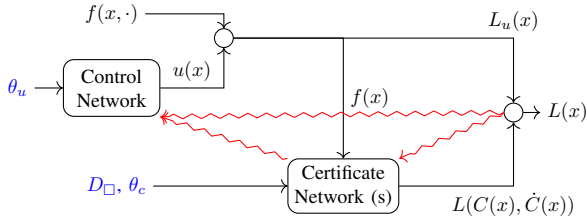


Fig. 3: Loss calculation block diagram. Blue indicates the inputs required for the loss calculation: namely the sampled data sets  $D_{\square}$ , the control and certificate network parameters  $\theta_u$  and  $\theta_c$ , respectively. Red arrows represent the back-propagation steps, updating the networks' parameters during training.  $f(x, \cdot)$  corresponds to the model in (1), while  $\dot{f}(x)$  corresponds to the model in (2).

Our candidate controller is thus simply the output of a neural network, contributing to the closed-loop dynamics, and which is then employed within the certificate synthesis procedure.

For the certificate synthesis, we also employ a neural network as a template for  $C(x)$ : this allows not only for polynomial templates akin to classical (e.g., SOS-based) certificates, but also for highly non-linear functions, leveraging the expressive power of neural networks. One of the crucial components of a successful training within the learner is the definition of a tailored loss function.

1) *Certificate Loss*: We note that all of the certificate conditions described in the previous section can be expressed in terms of inequalities, either as

$$C(x) \bowtie c \quad \forall x \in \mathcal{X}_{\square}, \text{ or as } \dot{C}(x) \bowtie c \quad \forall x \in \mathcal{X}_{\square}, \quad (18)$$

where  $\bowtie := \{<, >, \geq, \leq\}$ ,  $C(x)$  represents the certificate,  $\mathcal{X}_{\square}$  represents the relevant set (e.g.,  $\mathcal{X}_I$  or  $\mathcal{X}$ ) and  $c \in \mathbb{R}$ . We can then construct a loss function based on this observation, as follows. Consider a monotonically increasing function  $m(\cdot)$  and suppose we have a finite set of sampled data points over the set  $\mathcal{X}_{\square}$ , which we denote as  $D_{\square}$ . A general loss function for any of the discussed certificate conditions is thus

$$\sum_{d \in D_{\square}} m(p \cdot C(d)), \quad (19)$$

where  $p = 1$  if  $\bowtie = \{<, >\}$  and  $p = -1$  if  $\bowtie = \{\geq, \leq\}$ . Note that this function penalises points in  $D_{\square}$  where the required condition is not satisfied. Suitable choices for function  $m$  are leaky-ReLU, which is piecewise linear, and softplus, which is smooth. Since several of the sets  $\mathcal{X}_{\square}$  are boundaries of sets, or represent level sets, in practice we consider a small “band” around them, in order to encompass a sufficient number of data points in  $D_{\square}$ .

*Example.* Let us consider a barrier certificate  $B(x)$  for safety verification (see (9)). We create separate sets of finite samples for each set defined by the property (initial, unsafe, state-

space), and the resulting loss function is given by

$$L = \frac{1}{N_I} \sum_{d \in D_I} m(B(d)) + \frac{1}{N_U} \sum_{d \in D_U} m(-B(d)) + \frac{1}{N} \sum_{d \in Z_B} m(\dot{B}(d)). \quad (20)$$

Here  $Z_B(d) = \{d \in D : \dot{B}(d) = 0\}$ , where  $D$  is the set of  $N$  samples over the whole state space,  $D_I$  is the set of  $N_I$  samples over the initial set and  $D_U$  is the set  $N_U$  of samples of the unsafe set. ■

2) *Controller Loss*: Let us now discuss controlled models. Note that, as described in Figure 3, the parameters of the neural network controller appear in the loss function for the certificate, as described previously. Specifically, they manifest themselves in the terms corresponding to conditions on the Lie derivative of the certificate, since these in turn depend on the control-dependent dynamics  $f$ . This should encourage the learner to seek a controller that enables a valid certificate. However, in practice we find that this is insufficient for robust synthesis in the case of *arrive* requirements. While our certificates and properties make no assumption on the location of the goal set or of the equilibrium, oftentimes an equilibrium point (without loss of generality, the origin) lies within the goal set. If this is the case, trajectories converging to the origin exhibiting a negative Lie derivative are in turn attracted to the goal set. Since the Lie derivative consists of two components,  $f(x)$  and  $\nabla C(x)$ , it can be helpful to separate these elements within the loss function to encourage better learning. We thus propose the additional term for the loss function:

$$L_u = \frac{1}{N} \sum_{d \in D} \frac{\langle d, f(d) \rangle}{\|d\| \cdot \|f(d)\|}, \quad (21)$$

where  $\langle \cdot, \cdot \rangle$  is the inner product of its inputs and  $\|\cdot\|$  is the 2-norm of its input, and  $D$  is the set of  $N$  samples over the state space. This loss is known as the *cosine similarity* of the two vectors  $d$  and  $f(d)$ , and rewards vectors for pointing in opposite directions. We interpret this loss function as separating  $f(x)$  from  $\nabla C(x)$  when calculating the Lie derivative, and instead replacing the corresponding certificate with a “default” positive definite function (that of Euclidean distance). This encourages the loss function to specifically focus on the parameters in the feedback law to learn a desirable  $f(x)$ . As an alternative interpretation, since any vector  $d$  is fixed and points away from the origin, this encourages a controller which guides the dynamics to point towards to origin, and hence eventually converge towards it. We demonstrate the efficacy of this loss component in Section V-B.

## B. Verifier

We now discuss the dual component of the CEGIS architecture, as portrayed in Figure 4. The verifier's role is assumed by an SMT solver and its operation is described as follows: let  $\phi$  denote the conditions required for a given certificate. We seek a point  $x \in \mathcal{X}$  that violates any of the constraints  $\phi$  associated to the certificate. To this end, we express the *negation* of such requirements  $\phi$ , and formulate a nonlinear

constrained problem over real numbers. Formally, we ask an SMT solver to find a witness to

$$\exists x \in \mathcal{X} : \neg\phi(C(x)), \quad (22)$$

where any such witness  $x$  would be considered a counterexample to the validity of the certificate  $\phi$ .

*Example (Cont'd).* Consider the negation of barrier certificate conditions  $\phi$  as in (9), namely

$$(x \in \mathcal{X}_I \wedge B(x) > 0) \vee (x \in \mathcal{X}_U \wedge B(x) \leq 0) \vee (B(x) = 0 \wedge \dot{B}(x) \geq 0). \quad (23)$$

The verifier searches for solutions  $x$  of the constraints in (23). This in general requires manipulating non-convex functions and is therefore handled by an SMT solver. Whenever such  $x$  is found, it represents a witness that the candidate  $B(x)$  is not a valid certificate function. ■

The correctness of our algorithm hinges upon the soundness of the verification engine: we use two solvers, Z3 [45] and dReal [46], both of which are sound over nonlinear real arithmetic. Z3 is restricted to polynomial reasoning, whereas dReal can handle non-polynomial expressions, for instance containing trigonometric or exponential terms, thus allowing for more complex models and certificates (via their activation functions). We note that dReal is a  $\delta$ -complete SMT-solver: while this guarantees dReal will always find a counterexample if one exists, it may return also spurious counterexamples within a  $\delta$ -perturbation of the original formula [46]. This implies that our procedure may not terminate, even when the candidate is valid. However quite importantly it does not compromise the correctness of certificates we verify successfully.

### C. Enhanced Communication amongst Components

Our CEGIS approach builds on that of the software tool Fossil [53]. As part of its CEGIS loop, Fossil adds two elements to enhance the communication between components.

*Translator.* The translator is tasked with the conversion of the neural networks into a symbolic candidate  $C(x)$  and the corresponding  $\dot{C}(x)$ , ready to be processed by the verifier (see Fig. 4). The efficiency of SMT solvers depends also upon the numerical expressions of the formulae to be verified. Oftentimes the training returns numerically ill-conditioned expressions, e.g. unreasonably small coefficients (e.g., in the order of  $10^{-8}$ ); these candidates might slow the verification step, and thus the whole procedure. The translator thus rounds the coefficients of the candidate function to a specified precision, in order to help human interpretation and the verification process.

*Consolidator.* Generating counterexamples is, in general, an expensive procedure, and the verification engine returns a single counterexample ( $\text{cex}$  in Fig. 4); e.g. an instance satisfying (23). Naturally, an isolated sample does not provide enough information for the learner to improve the candidate certificate. To overcome this issue, we randomly generate a *cloud* of points around the  $\text{cex}$  point, since samples around a counterexample are also likely to invalidate the certificate conditions. Secondly, starting from  $\text{cex}$ , we compute the

gradient of  $C$  (or of  $\dot{C}$ ) thanks to an automatic differentiation feature, and follow the direction that maximises the violation of the certificate constraints. The consolidator then aggregates the original counterexample and the newly generated points (denoted  $\text{cex}^+$  in Fig. 4) with the sample set  $S$  for a new synthesis round by the learner.

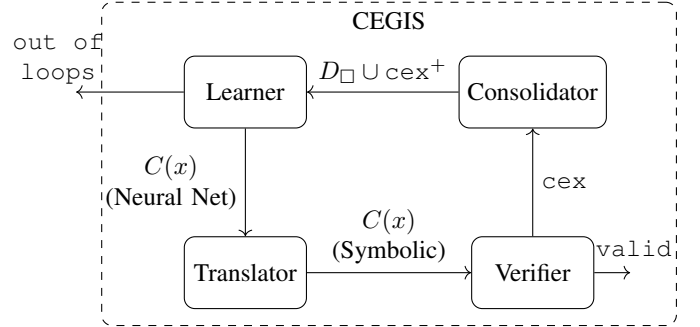


Fig. 4: Enhanced CEGIS architecture within Fossil.

### D. Comments on Specific Certificates

We add a few remarks next, elaborating on practical synthesis details that are unique to specific certificates.

1) *Lyapunov functions:* We assume without loss of generality that the given model has an equilibrium at the origin. We aid synthesis by enforcing the positivity condition by construction: in the case of positive-definite polynomial functions this is done by fixing the output layer's weights  $W_{k+1}$  to be all positive.

2) *ROA:* For verification, we estimate the smallest level set that contains  $\mathcal{X}_I$  using sample points. Then, we verify that  $\mathcal{X}_I$  is contained wholly within this level set and that the Lyapunov conditions hold over the entire level set. Due to dReal's inner workings, when verifying either Lyapunov or ROA certificates, we remove a very small region around the origin from the verification domain. This is common practice [39] across similar works, and does not affect sub-level sets outside this region. In fact, this notion is studied as  $\epsilon$ -stability [60]. This caveat does not apply when using Z3.

3) *RSA:* The conditions described in (14d, 14e) depend on the existence of the parameter  $\beta$ . We must prove a value for this parameter exists such that the relevant conditions hold in order to prove the certificate is valid. After successfully verifying the conditions which do not depend on  $\beta$ , we perform a line search for  $\beta$  to find a suitable value [49]. If we do not find a valid  $\beta$ , we return to synthesis and keep training.

## V. COMPUTATIONAL EXPERIMENTS AND BENCHMARKS

We have implemented the proposed framework based on the computational library of Fossil. We have tested our approach across a large number of case studies, and benchmarked it against the first release of Fossil, showing improved results.

### A. Main results

Our new prototype tool, which we refer to as Fossil 2.0, is able to verify all properties described in Section III for



	$N_s$	$N_u$	Property	Neurons	Activations	$T$ (s)			Success (%)
						min	$\mu$	max	S
1	2	0	Stability	[6]	$[\varphi_2]$	0.01 ( $\approx 0.00$ )	0.16 (0.15)	1.50 (1.48)	100
2	3	0	Stability	[8]	$[\varphi_2]$	0.28 ( $\approx 0.00$ )	2.22 (0.45)	12.57 (3.31)	100
3	2	2	Stability	[4]	$[\varphi_2]$	0.07 (0.01)	0.19 (0.02)	0.47 (0.04)	100
4	2	2	Stability	[5]	$[\varphi_2]$	0.09 (0.01)	0.26 (0.02)	0.54 (0.03)	100
5	2	0	ROA	[5]	$[\sigma_{\text{soft}}]$	0.21 (0.12)	14.09 (12.59)	25.32 (22.13)	40
6	3	3	ROA	[8]	$[\varphi_2]$	1.24 (0.02)	39.08 (0.03)	287.89 (0.04)	100
7	2	0	Safety	[15]	$[\sigma_t]$	0.44 (0.35)	3.36 (2.90)	7.61 (7.11)	100
9	8	0	Safety	[10]	$[\varphi_1]$	12.63 (7.71)	51.97 (32.75)	70.59 (44.66)	70
10	3	1	Safety	[15]	$[\sigma_t]$	1.57 (0.19)	11.87 (2.50)	51.08 (7.52)	90
11	3	0	SWA	[6], [5]	$[\varphi_2], [\sigma_t]$	0.19 (0.05)	2.46 (0.100)	12.10 (0.20)	90
12	2	0	SWA	[5], [5, 5]	$[\varphi_2], [\sigma_{\text{sig}}, \varphi_2]$	0.13 (0.06)	0.27 (0.14)	0.39 (0.20)	100
13	2	1	SWA	[8], [5]	$[\varphi_2], [\varphi_2]$	0.06 (0.03)	0.20 (0.10)	0.58 (0.24)	90
14	3	1	SWA	[10], [8]	$[\varphi_2], [\sigma_t]$	4.06 (0.87)	19.81 (2.73)	103.49 (7.23)	90
15	2	0	RWA	[4]	$[\varphi_2]$	0.14 (0.09)	1.81 (1.75)	4.70 (4.63)	100
16	3	0	RWA	[16]	$[\varphi_2]$	1.36 (0.09)	14.10 (0.14)	72.97 (0.20)	90
17	2	1	RWA	[4, 4]	$[\sigma_{\text{sig}}, \varphi_2]$	0.59 (0.27)	6.82 (3.32)	20.07 (11.46)	100
18	3	1	RWA	[5]	$[\varphi_2]$	0.46 (0.11)	16.06 (5.81)	72.47 (44.64)	80
19	2	2	RWA	[5]	$[\sigma_{\text{sig}}]$	0.69 (0.40)	1.38 (0.94)	2.14 (1.90)	100
20	2	0	RSWA	[4]	$[\varphi_2]$	0.19 (0.03)	1.29 (1.04)	3.79 (3.37)	100
21	3	0	RSWA	[16]	$[\varphi_2]$	4.81 (0.13)	27.14 (0.19)	80.95 (0.25)	100
22	2	0	RSWA	[5, 5]	$[\sigma_{\text{sig}}, \varphi_2]$	1.52 (0.06)	4.45 (0.19)	10.97 (0.35)	100
23	2	1	RSWA	[8]	$[\varphi_2]$	0.21 (0.05)	0.67 (0.25)	1.19 (0.91)	100
24	2	2	RSWA	[5, 5]	$[\sigma_{\text{sig}}, \varphi_2]$	0.98 (0.16)	1.23 (0.28)	1.61 (0.46)	100
25	2	0	RAR	[6], [6]	$[\sigma_{\text{soft}}], [\varphi_2]$	6.65 (1.08)	24.74 (6.46)	77.80 (15.06)	100
26	2	2	RAR	[6, 6], [6, 6]	$[\sigma_{\text{sig}}, \varphi_2], [\sigma_{\text{sig}}, \varphi_2]$	5.13 (1.34)	26.99 (9.90)	101.23 (60.14)	100

TABLE I: Results of synthesising certificates for all properties presented in this work. The first column indexes the benchmarks.  $N_s$ : Number of states,  $N_u$ : number of control inputs. We show the *Property* being verified and network structure (*Neurons* and *Activations*). For certificates of two functions, comma-separated lists shows the different structures. Finally, we report success rate ( $S$ ) and the minimum, mean ( $\mu$ ) and maximum computation time  $T$  over successful runs, in seconds. In brackets we show the time spent during the *learning* phase.

continuous-time models, both autonomous and controlled. Further, Fossil 2.0 can verify stability and safety properties for discrete-time models (the discussion of which is omitted for brevity). The extension to all the presented properties in discrete time is matter of future work. We showcase the efficacy of our framework and corresponding tool across 26 benchmarks, borrowed from existing literature on certificate synthesis [10], [48], [53], [61]. Note that, in some cases, we have modified these benchmarks to further challenge our approach, for instance by using disjoint, non-convex sets in the specifications. We consider a key strength of our approach to be its flexibility - we are able to perform well on straightforward and challenging benchmarks using certificates that represent both polynomials and more complex non-polynomial functions (as determined by the activation function of the neural network). We reflect this in our selection of benchmarks, including dynamics that are relatively simple and dynamics that involve transcendental and trigonometric functions. Due to the large number of benchmarks, details on the dynamics and sets can be found in Appendix III, and in the corresponding code-base, <https://github.com/oxford-oxcav/fossil>, where additional benchmarks can be also found.

The results are reported in Table I, where for each benchmark we outline the number of variables  $N_s$  and of control input  $N_u$ , the property to be verified (cf. acronyms introduced earlier), the number of neurons in each hidden layer and

the corresponding activation functions for these layers. The number of neurons is denoted as a list, e.g.  $[n_1, n_2]$  indicates that the first and second hidden layers are composed of  $n_1$  and  $n_2$  neurons, respectively. The activation functions for these hidden layers are denoted similarly. As mentioned, a strength of our methodology is its flexibility in terms of the form that certificates may take: we are able to synthesise polynomial certificates as well as non-polynomial certificates that represent more “neural-typical” functions – this is illustrated in the “Activations” column of Table I. By  $\varphi_j$  we denote that the layer represents a polynomial function of order  $j$ ;  $\sigma_{\text{sig}}$  represents the sigmoid function,  $\sigma_t$  represents the hyperbolic tangent function and  $\sigma_{\text{soft}}$  is the softplus function.

For almost all benchmarks, we use a linear control function. Our approach can handle more general nonlinear templates, but we emphasise that, as we solve a verification problem, rather than a control problem, we only seek a feedback law such that the property is satisfied by the closed-loop dynamics, and thus offer no guarantee on the optimality of this controller. Still, we use a nonlinear controller employing  $\sigma_t$  functions for the benchmark number 10 of Table I.

We measure the robustness performance by running each experiment 10 times, where we initialise the network with different weights and a new dataset across separate random seeds. Our procedure is not guaranteed to terminate, so after a maximum number of CEGIS loops we stop it and consider the

overall run a failure. In general, we allow 25 CEGIS loops; for the SWA and RAR certificates we allow 100 CEGIS loops as these certificates are composed of two functions.

We consider two metrics to assess the quality of our framework when attempting to synthesise a certificate and controller: how often it returns a successful result, the success rate  $S$ , and how long the algorithm takes for these successful runs, the time  $T$ . Table I thus reports  $S$ , along with the average, minimum and maximum time for the procedure to terminate, under the  $T$  column, denoted  $\mu$ , min and max, respectively. In brackets, we also denote the amount of time spent during the learning phase of our procedure, with approximately all remaining time spent during the verification phase.

The success rate is consistently close to 100%, with the minimum standing at 40% for a single benchmark, highlighting the robustness of our approach over the presented broad range of complex properties.

### B. Control Loss Evaluation

We have presented a novel loss function with the purpose of encouraging trajectories to converge to the origin, which is often desirable in the case of *arrive* conditions, as discussed in Section IV-A.2 We evaluate the inclusion of this loss function in two ways: first, we consider the effect on success rate and computation time relative to not having the term across all control benchmarks for RWA, RSWA and RAR properties, of which there are 6. These results are presented in Table II, where as before we present the success rate (this time over all 60 runs), and the average computation time for successful runs. It is clear that the inclusion of this loss term improves both the robustness and efficiency of the automated synthesis.

	Success (%)	Time (s)
With $L_u$	<b>96.67</b>	<b>7.14</b>
Without $L_u$	80.00	10.64

TABLE II: Comparison of success rate and computation time for the combined RWA, RSWA and RAR benchmarks with control, with and without the loss term described in (21).

Secondly, we compare again these two metrics when instead using an LQR feedback controller. This is the purpose of benchmarks 22 and 24, which are identical except for one key difference: benchmark 22 is equipped with a pre-computed LQR controller, *de facto* representing an autonomous model, whilst we shall compute a control law in benchmark 24. This allows us to compare our framework's ability to learn feedback laws which satisfy properties relative to a common baseline controller, the known LQR. The results for these two benchmarks are very similar, with our control approach performing slightly more efficiently. Note, we do not claim to outperform LQR controllers, as other cost matrices may perform better or worse; further, we do not provide the cost minimisation guarantees - we simply use these benchmarks as a baseline comparison.

### C. Comparison to Fossil 1.0 Baseline

We have built a prototype tool based on our framework, improving on the work initially presented in [53]. This tool compares against competitive state-of-the-art techniques such as SOS-tools, proving to deliver a faster synthesis for stability and safety properties for autonomous models. Table III collects works providing an automated and sound synthesis of relevant certificates. However, these related works are not proper software tools, and not equipped with a user-friendly way to e.g., tune network architecture and activation functions or able to handle several certificates. We therefore benchmark against Fossil 1.0 as the state-of-the-art for certificate synthesis. These results are presented in Table IV.

	Stability	ROA	Safety	SWA	RWA	RSWA	RAR	Control
Fossil 2.0	✓	✓	✓	✓	✓	✓	✓	✓
Fossil 1.0 [53]	✓	✗	✓	✗	✗	✗	✗	✗
F4CS [48],[49]	✓	✗	✓	✗	✓	✓	✗	✓
NLC [39]	✓	✗	✗	✗	✗	✗	✗	✓
[21], [27], [20]	✗	✗	✗	✗	✓	✓	✗	✓
[35]	✗	✗	✓	✗	✗	✗	✗	✗
[43], [31]	✗	✗	✓	✗	✗	✗	✗	✓
[44]	✗	✗	✓	✗	✗	✗	✗	✗
[26]	✓	✗	✓	✗	✗	✗	✗	✗
[28]	✓	✗	✗	✗	✗	✗	✗	✗

TABLE III: Comparison of works for automated (and sound) synthesis of certificates for continuous-time dynamical models. We show the properties verified in the respective works, and whether they are also able to verify these properties for control models (either using control certificates or controller and certificate). Similar works are grouped together.

We employ the benchmarks originally outlined in [53], and for a fair comparison we use the same network structure (width and activations) for both tools. It is clear that the approaches are very similar when synthesising the more straightforward Lyapunov functions. However, we achieve significant improvements in terms of both success rate and synthesis time relative to the baseline, on account of a fine-tuned loss function, an enhanced communication between the CEGIS components, and an overall improved software implementation.

## VI. DISCUSSIONS ON GENERALITY

### A. Asymptotic Reachability

Oscillations are important and common phenomena occurring in dynamical systems, typically linked to (stable) limit cycles. Certificates for stability analysis in the presence of limit cycles, or equivalently for the asymptotic convergence to such set, have been so far notably absent from the synthesis framework in this manuscript. We discuss them next, recalling Barbashin-Krasowskii-Lasalle's Principle [62].

**Theorem 9 (Invariance Principle [62]):** Let  $\Omega \subset \mathcal{X}$  be a compact set that is positively invariant with respect to (2). Let  $V : \mathcal{X} \rightarrow \mathbb{R}$  be a continuously differentiable function such that  $\dot{V}(x) \leq 0$ . Let  $E$  be the set of all points in  $\Omega$  where  $V(x) = 0$ . Let  $M$  be the largest invariant set in  $E$ . Then every solution starting in  $\Omega$  approaches  $M$  as  $t \rightarrow \infty$ . ■

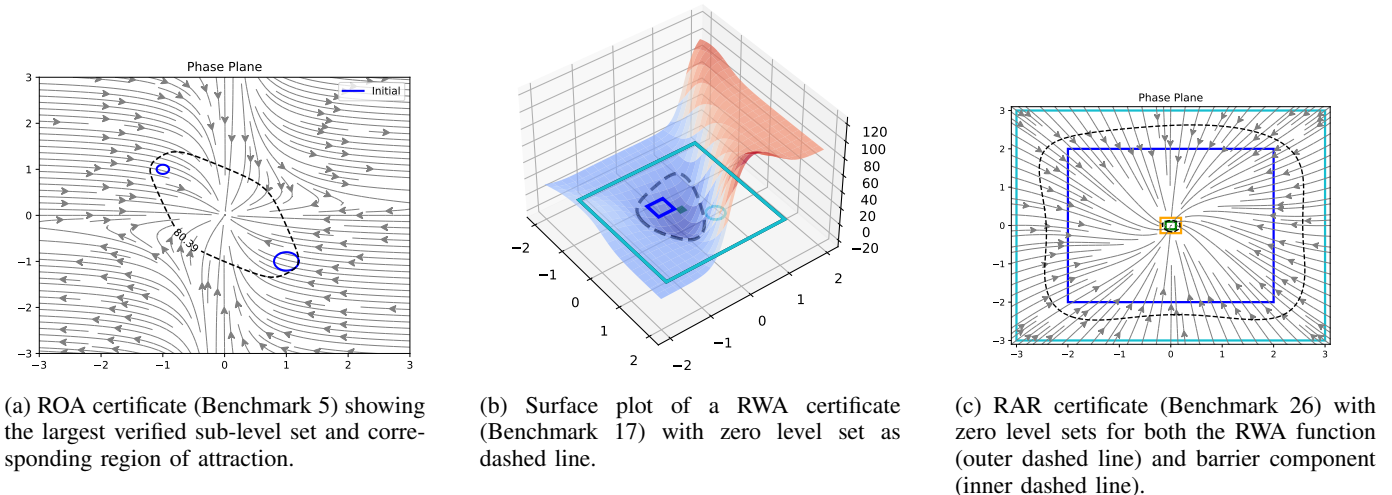


Fig. 5: Visualisations of a ROA, RWA and RAR certificate as a phase portrait, surface plot and phase portrait. We show salient level sets as dashed lines, and denote others sets as follows. Dark blue:  $\mathcal{X}_I$ ; light blue:  $\mathcal{X}_S$ ; green:  $\mathcal{X}_G$ ; orange:  $\mathcal{X}_F$ . TODO: add example Lyap level set to ROA

Benchmark	$N_s$	Property	Neurons	Activations	Fossil 1.0				Fossil 2.0			
					min	$\mu$	max	$S$	min	$\mu$	max	$S$
NonPoly0	2	Stability	[5]	$[\varphi_2]$	0.04	0.21	1.58	100	0.01	0.16	1.54	100
Poly2	2	Stability	[5]	$[\varphi_2]$	0.35	11.71	70.39	90	0.08	4.77	6.50	100
Barr1	2	Safety	[10]	$[\varphi_1]$	0.34	1.00	2.72	40	0.02	0.27	0.63	100
Barr3	2	Safety	[10, 10]	$[\sigma_{\text{sig}}, \sigma_{\text{sig}}]$	16.80	101.72	334.79	50	3.81	14.14	30.63	100

TABLE IV: Comparison of Fossil 1.0 vs Fossil 2.0 (the present work). Here, we use the same naming scheme for benchmarks as used in Fossil 1.0, rather than indexing by number. See Table I for details on the columns.

Crucially, the principle states that we can prove asymptotic convergence towards a set thanks to a Lyapunov-like function which is equal to zero *exactly* at the limit cycle. Such certificates have recently been studied from the perspective of disturbed models [50], [51]. Nonetheless, without prior knowledge of the existence and location of a limit cycle it is, in our experience, impractical to automatically synthesise such certificates, as they would need to be strongly templated based on the limit cycle (namely, precisely tailored to that set). Whilst in principle our approach could offer certificates for such properties, in this work we omit certificates requiring such an extensive analysis of the model dynamics.

### B. Sufficiency of the Certificates and Completeness of their Synthesis

The certificates provided in this work are *sufficient* proofs for the corresponding properties. We do not in general provide guarantees that a certificate exists if the property is satisfied, i.e. *necessary* proofs. Such converse results exist for Lyapunov and barrier functions. In particular, a construction method of Lyapunov functions is known for globally exponentially stable models [62], whilst the necessity of barrier certificates is studied in, e.g., [63], [64], [65].

Contextually, whilst *sound*, our counterexample-based inductive synthesis method is not *complete*: whenever it finds a certificate, the desired property formally holds for the model under consideration. On the other hand, if our algorithm fails

to find a certificate, we cannot draw any conclusion about the validity of the property for the given model. We show that in practice our procedure consistently terminates with a successful outcome.

### C. Modularity of the Synthesis and Nested Properties

We have not considered properties describing that of sequential reachability - namely trajectories arriving at a series of target sets before the goal set. We have considered properties obtained by conjunction of requirements, and we shall comment in Appendix II instances obtained via disjunction. We could similarly obtain certificates by manipulating via propositional logic, (e.g., conjunction and disjunction, rather than temporally, as suggested previously) requirements and corresponding certificates. Such properties tie into our approach of using certificates in a “modular” fashion, which is a relatively unexplored concept and represents an area of future work.

### D. Issues of Scale

Our CEGIS-based approach consists of a gradient descent based learning phase followed by an SMT dependent verification phase. While gradient descent is known to scale well to higher dimensions, nonlinear real arithmetic SMT does not in general scale well to higher dimensions. This problem is shared to all methodologies which rely on SMT. Possible mitigations, beyond an improvement of SMT performance,

include the use of an alternative verification method, e.g., software as Marabou [66] for ReLU networks, or interval bound propagation based techniques.

### E. Broader Connections and Taxonomy of Properties

In Section III we have presented a diverse set of certifiable properties in terms of the behaviour of trajectories (that is, of solutions) of a given dynamical model. Furthermore, in Section III-H, we have laid connections across such properties through the notions of *avoid*, *arrive*, and *remain*. In Appendix II, we further frame such requirements in broader contexts, providing a categorisation of these properties within known classes of specifications. To this end, we draw connections with formal languages (in particular, regular expressions) and with automata theory (specifically, deterministic finite automata) [67], and in passing we also informally relate to temporal logic for specifications of reactive models [68], [69], [70].

Appendix II is written for the benefit of readers with a background in the mentioned areas, or with an interest in a perspective on dynamical models grounded upon formal methods [17], [18] - this part can be otherwise dispensed with, at no loss of understanding of the overall material.

## VII. CONCLUDING REMARKS

We have presented a general framework to formally verify dynamical and control models via certificate synthesis, and introduced Fossil 2.0, a prototype software tool for the automated formal synthesis of a broad range of certificates. Certificate synthesis is based on a CEGIS loop, exploiting neural networks to provide candidate functions, which are then formally verified with the help of SMT solvers. Our approach is able to efficiently synthesise certificates for a wide range of properties for both autonomous and control models, with varying complexity of dynamics and sets structure. We test our framework and corresponding tool on a number of benchmarks, outperforming a state-of-the-art tool for certificate synthesis and showing robustness to initialisation.

In future work, we hope to explore further the modular synthesis of certificates, as well as to extend our framework to stochastic models.

## REFERENCES

- [1] S. Sastry, *Nonlinear Systems*, ser. Interdisciplinary Applied Mathematics, J. E. Marsden, L. Sirovich, and S. Wiggins, Eds. New York, NY: Springer New York, 1999, vol. 10.
- [2] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*. Boston, MA: Birkhäuser Boston, 2008.
- [3] T. Henzinger, "The theory of hybrid automata," in *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, Jul. 1996, pp. 278–292.
- [4] A. M. LYAPUNOV, "The general problem of the stability of motion," *International Journal of Control*, vol. 55, no. 3, pp. 531–534, Mar. 1992.
- [5] A. Papachristodoulou and S. Prajna, "On the construction of Lyapunov functions using the sum of squares decomposition," in *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002., vol. 3. Las Vegas, NV, USA: IEEE, 2002, pp. 3482–3487.
- [6] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. Parrilo, "SOSTOOLS Version 3.00 Sum of Squares Optimization Toolbox for MATLAB," *arXiv:1310.4716 [cs, math]*, Oct. 2013.
- [7] S. Prajna, "Barrier certificates for nonlinear model validation," *Automatica (Journal of IFAC)*, vol. 42, no. 1, pp. 117–126, Jan. 2006.
- [8] E. Goubault, J.-H. Jourdan, S. Putot, and S. Sankaranarayanan, "Finding Non-Polynomial Positive Invariants and Lyapunov Functions for Polynomial Systems through Darboux Polynomials," in *American Control Conference (ACC)*, Portland, United States, Jun. 2014.
- [9] M. A. Ben Sassi, S. Sankaranarayanan, X. Chen, and E. Ábrahám, "Linear relaxations of polynomial positivity for polynomial Lyapunov function synthesis," *IMA Journal of Mathematical Control and Information*, vol. 33, no. 3, pp. 723–756, Sep. 2016.
- [10] S. Sankaranarayanan, X. Chen, and E. Ábrahám, "Lyapunov Function Synthesis using Handelman Representations," *IFAC Proceedings Volumes*, vol. 46, no. 23, pp. 576–581, 2013.
- [11] S. Ratschan and Z. She, "Providing a Basin of Attraction to a Target Region of Polynomial Systems by Computation of Lyapunov-Like Functions," *SIAM Journal on Control and Optimization*, vol. 48, no. 7, pp. 4377–4394, Jan. 2010.
- [12] Z. She, B. Xia, R. Xiao, and Z. Zheng, "A semi-algebraic approach for asymptotic stability analysis," *Nonlinear Analysis: Hybrid Systems*, vol. 3, no. 4, pp. 588–596, Nov. 2009.
- [13] Z. She, H. Li, B. Xue, Z. Zheng, and B. Xia, "Discovering polynomial Lyapunov functions for continuous dynamical systems," *Journal of Symbolic Computation*, vol. 58, pp. 41–63, Nov. 2013.
- [14] A. Abate, "Formal verification of complex systems: Model-based and data-driven methods," *MEMOCODE 2017 - 15th ACM-IEEE International Conference on Formal Methods and Models for System Design*, pp. 91–93, 2017.
- [15] R. Bohrer, Y. K. Tan, S. Mitsch, M. O. Myreen, and A. Platzer, "Veriphy: verified controller executables from verified cyber-physical system models," in *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018*, 2018, pp. 617–630.
- [16] J. C. Knight, "Safety critical systems: challenges and directions," in *ICSE. ACM*, 2002, pp. 547–550.
- [17] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.
- [18] C. Belta, B. Yordanov, and E. A. Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017, vol. 15.
- [19] C. Barrett, A. Stump, C. Tinelli *et al.*, "The smt-lib standard: Version 2.0," in *Proceedings of the 8th international workshop on satisfiability modulo theories (Edinburgh, UK)*, vol. 13, 2010, p. 14.
- [20] H. Ravanbakhsh and S. Sankaranarayanan, "Counterexample Guided Synthesis of Switched Controllers for Reach-While-Stay Properties," *arXiv:1505.01180 [cs]*, Sep. 2015.
- [21] —, "Learning control lyapunov functions from counterexamples and demonstrations," *Autonomous Robots*, vol. 43, no. 2, pp. 275–307, Feb. 2019.
- [22] Z. Huang, Y. Wang, S. Mitra, G. E. Dullerud, and S. Chaudhuri, "Controller synthesis with inductive proofs for piecewise linear systems: An smt-based algorithm," in *2015 54th IEEE conference on decision and control (CDC)*. IEEE, 2015, pp. 7434–7439.
- [23] A. Abate, I. Bessa, D. Cattaruzza, L. Cordeiro, C. David, P. Kesseli, D. Kroening, and E. Polgreen, "Automated formal synthesis of provably safe digital controllers for continuous plants," *Acta Informatica*, vol. 57, no. 3, p. 223–244, 2020.
- [24] A. Solar-Lezama, L. Tancau, R. Bodik, S. Seshia, and V. Saraswat, "Combinatorial sketching for finite programs," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 5, pp. 404–415, Oct. 2006.
- [25] D. Ahmed, A. Peruffo, and A. Abate, "Automated and Sound Synthesis of Lyapunov Functions with SMT Solvers," 2018.
- [26] J. Kapinski, J. V. Deshmukh, S. Sankaranarayanan, and N. Arechiga, "Simulation-guided lyapunov analysis for hybrid dynamical systems," in *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '14. New York, NY, USA: Association for Computing Machinery, Apr. 2014, pp. 133–142.
- [27] H. Ravanbakhsh and S. Sankaranarayanan, "Counter-Example Guided Synthesis of control Lyapunov functions for switched systems," in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec. 2015, pp. 4232–4239.
- [28] H. Dai, B. Landry, M. Pavone, and R. Tedrake, "Counter-example guided synthesis of neural network Lyapunov functions for piecewise linear systems," in *2020 59th IEEE Conference on Decision and Control (CDC)*, Dec. 2020, pp. 1274–1281.
- [29] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control," in *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, Jul. 2021.



- [30] D. Masti, F. Fabiani, G. Gnecco, and A. Bemporad, “Counter-example guided inductive synthesis of control lyapunov functions for uncertain systems,” *IEEE Control Systems Letters*, 2023.
- [31] H. Zhao, X. Zeng, T. Chen, Z. Liu, and J. Woodcock, “Learning safe neural network controllers with barrier certificates,” *Formal Aspects of Computing*, vol. 33, no. 3, pp. 437–455, Jun. 2021.
- [32] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, “Learning Lyapunov Functions for Piecewise Affine Systems with Neural Network Controllers,” *arXiv:2008.06546 [math]*, Aug. 2020.
- [33] —, “Learning lyapunov functions for hybrid systems,” in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*. Nashville Tennessee: ACM, May 2021, pp. 1–11.
- [34] Gurobi Optimization, LLC, “Gurobi Optimizer Reference Manual,” 2021.
- [35] Q. Zhao, X. Chen, Y. Zhang, M. Sha, Z. Yang, W. Lin, E. Tang, Q. Chen, and X. Li, “Synthesizing ReLU neural networks with two hidden layers as barrier certificates for hybrid systems,” in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*. New York, NY, USA: Association for Computing Machinery, May 2021, no. 17, pp. 1–11.
- [36] S. M. Richards, F. Berkenkamp, and A. Krause, “The Lyapunov Neural Network: Adaptive Stability Certification for Safe Learning of Dynamical Systems,” *arXiv:1808.00924 [cs]*, Oct. 2018.
- [37] N. Noroozi, P. Karimaghaee, F. Safaei, and H. Javadi, “Generation of Lyapunov Functions by Neural Networks,” p. 5, 2008.
- [38] W. Jin, Z. Wang, Z. Yang, and S. Mou, “Neural Certificates for Safe Control Policies,” *arXiv:2006.08465 [cs, eess]*, Jun. 2020.
- [39] Y.-C. Chang, N. Roohi, and S. Gao, “Neural Lyapunov Control,” *arXiv:2005.00611 [cs, eess, stat]*, Dec. 2020.
- [40] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, “Automated Formal Synthesis of Lyapunov Neural Networks,” 2020.
- [41] P. Samanipour and H. A. Poonawala, “Stability analysis and controller synthesis using single-hidden-layer relu neural networks,” *IEEE Transactions on Automatic Control*, pp. 1–12, 2023.
- [42] A. Peruffo, D. Ahmed, and A. Abate, “Automated and Formal Synthesis of Neural Barrier Certificates for Dynamical Models,” Mar. 2021, pp. 370–388.
- [43] H. Zhao, X. Zeng, T. Chen, and Z. Liu, “Synthesizing barrier certificates using neural networks,” in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, ser. HSCC ’20. New York, NY, USA: Association for Computing Machinery, Apr. 2020, pp. 1–11.
- [44] S. Ratschan, “Simulation Based Computation of Certificates for Safety of Hybrid Dynamical Systems,” *arXiv:1707.00879 [cs]*, Oct. 2018.
- [45] L. de Moura and N. Björner, “Z3: An Efficient SMT Solver,” in *Tools and Algorithms for the Construction and Analysis of Systems*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, C. R. Ramakrishnan, and J. Rehof, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 4963, pp. 337–340.
- [46] S. Gao, S. Kong, and E. M. Clarke, “dReal: An SMT Solver for Nonlinear Theories over the Reals,” in *Automated Deduction – CADE-24*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and M. P. Bonacina, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 7898, pp. 208–214.
- [47] “isat3,” <https://projects.informatik.uni-freiburg.de/projects/isat3/>.
- [48] C. F. Verdier and M. Mazo Jr, “Formal controller synthesis for hybrid systems using genetic programming,” *arXiv:2003.14322 [cs, eess]*, Sep. 2020.
- [49] C. F. Verdier, “Formal synthesis of analytic controllers: An evolutionary approach,” 2020.
- [50] Y. Meng, Y. Li, and J. Liu, “Control of Nonlinear Systems with Reach-Avoid-Stay Specifications: A Lyapunov-Barrier Approach with an Application to the Moore-Greizer Model,” in *2021 American Control Conference (ACC)*, May 2021, pp. 2284–2291.
- [51] Y. Meng, Y. Li, M. Fitzsimmons, and J. Liu, “Smooth Converse Lyapunov-Barrier Theorems for Asymptotic Stability with Safety Constraints and Reach-Avoid-Stay Specifications,” Dec. 2021.
- [52] C. Dawson, S. Gao, and C. Fan, “Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods,” *arXiv preprint arXiv:2202.11762*, 2022.
- [53] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo, “FOSSIL: A software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks,” in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC ’21. New York, NY, USA: Association for Computing Machinery, May 2021, pp. 1–11.
- [54] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge, UK ; New York: Cambridge University Press, 2003.
- [55] S. Prajna, A. Jadbabaie, and G. Pappas, “Stochastic safety verification using barrier certificates,” in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*. Nassau, Bahamas: IEEE, 2004, pp. 929–934 Vol.1.
- [56] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control Barrier Function Based Quadratic Programs for Safety Critical Systems,” *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.
- [57] X. Tan, W. S. Cortez, and D. V. Dimarogonas, “High-order barrier functions: Robustness, safety, and performance-critical control,” *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 3021–3028, 2022.
- [58] Z. Wu, F. Albalawi, Z. Zhang, J. Zhang, H. Durand, and P. D. Christofides, “Control Lyapunov-Barrier function-based model predictive control of nonlinear systems,” *Automatica*, vol. 109, p. 108508, Nov. 2019.
- [59] M. Z. Romdlony and B. Jayawardhana, “Stabilization with guaranteed safety using control lyapunov–barrier function,” *Automatica*, vol. 66, pp. 39–47, 2016.
- [60] S. Gao, J. Kapinski, J. Deshmukh, N. Roohi, A. Solar-Lezama, N. Arechiga, and S. Kong, “Numerically-robust inductive proof rules for continuous dynamical systems,” in *Computer Aided Verification*, I. Dillig and S. Tasiran, Eds. Cham: Springer International Publishing, 2019, pp. 137–154.
- [61] A. Vannelli and M. Vidyasagar, “Maximal lyapunov functions and domains of attraction for autonomous nonlinear systems,” *Automatica*, vol. 21, no. 1, pp. 69–80, Jan. 1985.
- [62] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, N.J: Prentice Hall, 2002.
- [63] S. Prajna and A. Rantzer, “On the necessity of barrier certificates,” *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 526–531, 2005.
- [64] R. Wisniewski and C. Sloth, “Converse barrier certificate theorems,” *IEEE Transactions on Automatic Control*, vol. 61, no. 5, pp. 1356–1361, 2015.
- [65] S. Ratschan, “Converse theorems for safety and barrier certificates,” *IEEE Transactions on Automatic Control*, vol. 63, no. 8, pp. 2628–2632, 2018.
- [66] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, D. L. Dill, M. J. Kochenderfer, and C. Barrett, “The Marabou Framework for Verification and Analysis of Deep Neural Networks,” in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, I. Dillig and S. Tasiran, Eds. Cham: Springer International Publishing, 2019, pp. 443–452.
- [67] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing, 1979.
- [68] A. Pnueli, “The temporal logic of programs,” in *18th Annual Symposium on Foundations of Computer Science*, 1977, pp. 46–57.
- [69] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.
- [70] E. Clarke, O. Grumberg, D. Kroening, D. Peled, and H. Veith, *Principles of model checking*, 2nd ed. MIT Press, 2018.
- [71] Z. Manna and A. Pnueli, “A hierarchy of temporal properties,” in *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC ’90, 1990, p. 377–410.
- [72] O. Kupferman and M. Vardi, “Model checking of safety properties,” *Formal Methods in System Design*, vol. 19, 1999.
- [73] A. Abate, M. Giacobbe, and D. Roy, “Learning probabilistic termination proofs,” in *CAV (2)*, ser. Lecture Notes in Computer Science, vol. 12760. Springer, 2021, pp. 3–26.

## APPENDIX I PROOF OF THEOREMS

*Proof:* (Proof of Thm. 1) We state without proof from Lyapunov theory [1] that the conditions in (6) imply that  $f(x)$  is asymptotically stable, and that all sub-level sets of  $V(x)$  fully contained within  $\mathcal{X}$  are forward invariant. Since  $V$  is continuous and  $\mathcal{X}$  is non-empty, then there exists some  $\beta$  such that the sub-level set  $\Omega_\beta = \{x \in \mathcal{X} \mid V(x) \leq \beta\}$  is fully

contained within  $\mathcal{X}$  and  $x^* \in \Omega_\beta$ . Since  $x^*$  is asymptotically stable, all trajectories in  $\Omega_\beta$  converge towards it, and (5) holds for the initial set  $\Omega_\beta$ .  $\square$

*Proof:* (Proof of Cor. 2) Define the  $\beta$  sub-level of the ROA certificate  $V$  as  $\Omega_\beta = \{x \in \mathcal{X} \mid V(x) \leq \beta\}$ . By construction,  $\mathcal{X}_I \subset \Omega_\beta$ , and the condition of (6) hold. The proof then follows directly from Thm. 1.  $\square$

**Theorem 10 (Nagumo’s Theorem):** Let us state without proof Nagumo’s Theorem. For a proof, see [2]. Consider the system  $\dot{\xi}(t) = f(\xi(t))$ , where  $f$  is Lipschitz continuous such that for each initial condition  $\xi(t_0) \in \mathcal{X}$  it admits a unique solution. Let  $\Omega \in \mathcal{X}$  be a closed set.  $\Omega$  is positively invariant if and only if for every exterior normal vector  $v$  at point  $x$  on the border of  $\partial\Omega$ , the inner product satisfies  $\langle f(x), v \rangle \leq 0$ .

*Proof:* (Proof of Thm. 3) By definition we have that  $\mathcal{X}_I \cap \mathcal{X}_U = \emptyset$ . The set  $\Omega_0 = \{x \in \mathcal{X} \mid \dot{B}(x) \leq 0\}$  defines a closed set for which (9c) ensures that  $f(x)$  points inwards along its border. It follows from 10 that  $\Omega_0$  is an forward invariant set which contains  $\mathcal{X}_I$ , and that at all trajectories initialised in  $\mathcal{X}_I$  remain within  $\Omega_0$  for all time  $t > t_0$ . (9b) ensures that  $\Omega_0 \cap \mathcal{X}_U = \emptyset$ , and hence (8) holds.  $\square$

*Proof:* (Proof of Cor. 4) We note that the specification described by (10) is the conjunction of those in (5) and (8). Therefore, the proof follows directly from Cor. 2 and Thm. 3.  $\square$

*Proof:* (Proof of Thm. 5 (from [48])) Let  $A = \{x \in \mathcal{X}_S \mid V(x) \leq 0\}$ . For  $\xi(t_0) \in \mathcal{X}_I$ , it follows from (12a) and the definition of  $A$  that  $\xi(t_0) \in A$ . From (12c), for all  $\xi(t_k) \in A \setminus \mathcal{X}_G$ ,  $\dot{V}(\xi(t_k)) < 0$ . Using  $\forall x \in A, V(x) \leq 0$  and the comparison principle [62], it follows that  $\forall k \in \mathbb{Z}_{\geq 0}, \forall t \in [t_k, t_k + h], \forall \xi(t_k) \in A \setminus \mathcal{X}_G : V(\xi(t)) \leq V(\xi(t_k)) < 0$ . Therefore,  $\xi(t_k) \in A \setminus \mathcal{X}_G$  implies  $\forall t \in [t_k, t_k + h], V(\xi(t))$  will decrease and thus cannot reach  $\partial\mathcal{X}_S$ , since by (12b)  $\forall z \in \partial\mathcal{X}_S : V(x) > 0$ . Since  $\mathcal{X}_S$  is compact and  $V(x)$  is continuous,  $\exists e \in \mathbb{R}$  s.t.  $e = \inf_{x \in \mathcal{X}_S \setminus \mathcal{X}_G} V(x)$  and the sublevel sets of  $V(x)$  is compact. It follows that  $V(x)$  is lower bounded on  $A \setminus \mathcal{X}_G \subset \mathcal{X}_S \setminus \mathcal{X}_G$ , so  $V(\xi(t))$  will decrease until in finite time  $\xi(t)$  leaves  $A \setminus \mathcal{X}_G$  and may only enter  $\mathcal{X}_G$ .  $\square$

*Proof:* (Proof of Thm. 7 (from [48])) Let  $B = \{x \in \mathcal{X}_S \mid V(x) \leq \beta\}$ . From Thm. 5, there exists a time  $t_K \geq t_0$  such that  $\xi(t_K) \in \mathcal{X}_F$ . Using a similar argument as before, we conclude that  $\forall \xi(t_K) \in \mathcal{X}_F, \xi(t)$  with  $t \geq t_K$  enters in finite time  $\mathcal{X}_F \cap B$ . Since  $B$  is a sub-level set of the compact set  $S$  and continuous  $V$ , then  $B$  is also compact and so is  $\mathcal{X}_F \cap B$ . From (14e) we have  $\forall x \in \partial(\mathcal{X}_F \cap B) : \dot{V}(x) < 0$ . Combining with (14d), we have that all states  $\xi(t) \in \partial(\mathcal{X}_F \cap B)$  cannot reach  $\partial\mathcal{X}_F$  and  $V(\xi(t))$  decreases, meaning these trajectories remain in  $\mathcal{X}_F \cap B$ . Therefore,  $\mathcal{X}_F \cap B$  is forward invariant. Since  $\mathcal{X}_F \subset \text{int}(\mathcal{X}_S)$ , we have that (13) holds.  $\square$

*Proof:* (Proof of 8) This proof follows directly from Thms. 3 and 5.  $\square$

## APPENDIX II

### BROADER CONNECTIONS AND TAXONOMY OF PROPERTIES

In this part, we further frame the presented requirements in broader contexts, thus providing a categorisation of the

properties under study within specific classes of *formal specifications* [69], [70]. In order to do so, we draw connections with formal languages (in particular, regular expressions) and with automata theory (specifically, deterministic finite automata) [67], and in passing we also informally relate to the use of temporal logic for specifications of reactive models [68], [69], [70]. However, these connections can be drawn under an important proviso: that is, in formal methods, properties by and large are expressed over finite alphabets, namely over a finite set of labels (which can be related to our sets – or complement thereof – of interest), but most importantly concern traces in discrete time. Similarly, formal languages and automata deal with finite or countably-infinite strings of (finite) characters.

*Safety and reachability - language and semantic duality:* We remark that all the presented properties consist of combinations of two fundamental specifications in formal verification [71], [72]. The first is *safety*, which qualitatively concerns trajectories always staying clear from a nominative region in the state space that is deemed to be “unsafe”. Thus, safety specifications are infinite-horizon requirements, which however can be equivalently defined as requirements admitting finite-horizon counter-examples: namely, they are specifications that are necessarily invalidated by trajectories that enter the given unsafe in finite time.

The second specification we refer to as *reachability*, and comprise trajectories reaching a given desirable goal set (which is also known as “reach” or “target” set). Reachability is notably also employed to prove finite-time termination of programs, by means of certificates known as “ranking functions” [73] that are reminiscent to Lyapunov or RWA ones. In the area of formal languages, reachability is known to be a *co-safety* property, namely a dual to a safety requirement and, as such, it is a finite-horizon property. Indeed, in formal verification co-safe specifications admit finite-horizon witnesses, that is satisfying trajectories that enter the goal set in finite time. In Section III we have defined this as “unconstrained” reachability, and as a special instance of the reach-while avoid specification. We could have alternatively introduced it as the logical dual of safety, except that the certificate synthesis would not have followed as seamlessly. We should mention that, more generally, co-safety properties (and, in particular, reachability) are special instances of another broad class of specifications, known as *liveness* properties [69], [70]: in this work we do not deal with general liveness properties, and in particular the synthesis of sufficient certificates for this class is left as future work.

*Finite- and infinite-horizon requirements:* We have discussed that safety and reachability properties are dual in the sense that the earlier raises a requirement over infinite-horizon trajectories, asking that “nothing bad ever happens”, whereas the latter requires reasoning about finite-horizon solutions, asking that over a finite time horizon “something good eventually happens” [69]. However note that, whenever safety requirements are added to finite-time “reachability” properties, as in the case of the “avoid” constraint in RWA, RSWA, and RAR, these safety requirements ought to hold only over the finite time spans inherited from the reachability requirements.

Dually, as much as reachability requirements natively en-

compass co-safety properties, in this work we have discussed extension of such reachability requirements over *infinite-time* horizons: this is quite natural in control theory, namely in the context of asymptotic stability. Similarly, we have not only raised requirements that trajectories reach a goal set within a finite time horizon, which we have simply denoted as *reachability*, but additionally discussed (cf. VI-A) that they may approach the set (e.g., a set of equilibria, or a limit cycle) asymptotically, which we have denoted as *asymptotic reachability*.

*Automata theory:* Next, we qualitatively relate these classes of properties to automata theory. Through their duality, both safety and co-safety properties can be expressed by the same class of finite-state models, namely deterministic finite automata (DFA) [67]. In other words, traces within these two classes of specifications can be “compiled” by a DFA model, which “reads” them when they are started in one of its initial states, and when they terminate upon hitting one its accepting states. (Conversely, liveness properties, which we saw generalise co-safety specifications, require automata of different nature, such as Büchi or Rabin automata, which specifically accept infinite traces.) Summarising this discussion, we can conclude that we can provide sufficient certificates for properties that can be expressed as DFAs: an enticing extension of our work is looking at “richer” specifications obtained by modularly “composing” such finite-state models.

*Linear-time properties and temporal logic:* Finally, let us draw connections to temporal logic, a class of modal logic that was natively developed to specify requirements of (models of) reactive programs [68]. Whilst originally employed for finite-state programs evolving over discrete-time steps, temporal logics, such as LTL (linear TL), have been also widely employed in the context of dynamical and control models [17], [18]. Bearing in mind that the above caveat on discrete-time semantics holds also in this context (which for instance renders its “next” operator useless for our purposes), we can express safety requirements over safe set  $S$  as “always  $S$ ” - in LTL syntax this is expressed as the formula  $\Box S$  (or  $GS$ ) - and, dually, reachability requirements over target set  $R$  as “eventually  $T$ ” - in LTL this is  $\Diamond R$  (or  $FR$ ); incidentally, the discussed duality between these requirements is crisply expressed logically as:  $\Box S \equiv \neg \Diamond \neg S \equiv \neg \Diamond S^c$ . Similar discussions follow through for finite-time properties, which however in LTL require selecting a (finite) horizon  $T \in \mathbb{R}$ , thus yielding for example  $\Box^{\leq T} S$ . Reach-avoid can be expressed via the “until” operator  $U$  in LTL, say  $S UR$ , which by the way can be tailored to “unconstrained reachability” as  $\Diamond R \equiv \text{true } UR$ .

Dovetailing to the discussion above, we can thus flexibly and modularly synthesise certificates also for the known “weak until”  $W$ , and even for the “release”  $R$  specifications in LTL. Indeed, recall that [69]

$$SWR = (SUR) \vee \Box S,$$

We can thus obtain a certificate for weak until from either a certificate for reach-avoid or one for safety, respectively. Additionally notice that, in particular, that  $\Box S \equiv SW \text{false}$ . Additionally,

$$SRT = \neg(\neg S U \neg T),$$

and in fact (the next equivalent expression is easier for finding certificates

$$SRT = (\neg S \wedge T)W(S \wedge T),$$

and so, in particular,  $\Box S \equiv \text{false } RS$ .



**Alec Edwards** is a PhD student in Computer Science at the University of Oxford, as part of the EPSRC CDT in Autonomous Intelligent Machines and Systems. He received an MEng in Engineering Science from the University of Oxford in 2018, and worked as a research assistant at the Energy Futures Lab, Imperial College London. His research interests include the intersection of formal verification with artificial intelligence.



**Andrea Peruffo** is a postdoc at the Delft Center for Systems and Control. He obtained the DPhil in Computer Science at the University of Oxford. He received a Laurea in Information Engineering in 2012 and an MS in 2015 in Automation Engineering, both from the University of Padova (IT). He worked as a research engineer at Inria Rennes in 2016 and visited the Erato MMSD Group at NII, Tokyo, in 2020. His research interests include hybrid systems, formal verification and control theory.



**Alessandro Abate** (S'02-M'08-SM'19) is Professor of Verification and Control in the Department of Computer Science at the University of Oxford (UK). He received a Laurea in Electrical Engineering in 2002 from the University of Padua (IT), an MS in 2004 and a PhD in 2007, both in Electrical Engineering and Computer Sciences, at UC Berkeley (USA). He has been an International Fellow in the CS Lab at SRI International (USA) and a PostDoctoral Researcher at Stanford University (USA), in the Department of Aeronautics and Astronautics. He has also been an Assistant Professor at the Delft Centre for Systems and Control, TU Delft (NL).

### APPENDIX III COLLECTION OF BENCHMARKS

Here, we present the benchmarks presented in this work. These benchmarks are indexed with a number, corresponding to the row index from Table I, along with the property that is being verified for the benchmark. We show the salient sets for the property; note that in some cases it is simpler to show the safe set  $\mathcal{X}_S$ , and in other cases the unsafe set  $\mathcal{X}_U$ . These are always the complement of each other:  $\mathcal{X}_S = \mathcal{X} \setminus \mathcal{X}_U$ .

We also use shorthand notation to describe sets of familiar geometric structure:

- $Rectangle(lb, ub)$  denotes a hyper rectangle characterised by its lower bounds ( $lb$ ) for each dimension, and its upper bounds ( $ub$ ) in each dimension.
- $Sphere(c, r)$  denotes a hyper sphere with centre  $c$  and radius  $r$ . The dimension of the hyper-sphere corresponds to the size of  $c$ .
- $Torus(c, r_i, r_o)$  denotes the set defines by  $Sphere(c, r_o) \setminus Sphere(c_i)$ .

#### [1] NonPoly0 Stability

$$f(x) = [x_0x_1 - x_0, -x_1]$$

$$\mathcal{X} : Torus([0, 0], 1, 0.01)$$

Activation:  $[\varphi_2]$ , Neurons: [6]

#### [2] Poly1 Stability

$$f(x) = [-x_0^3 - x_0x_2^2, -x_0^2x_1 - x_1, 3x_0^2x_2 - 4x_2]$$

$$\mathcal{X} : Torus([0.0, 0.0, 0.0], 10.0, 0.1)$$

Activation:  $[\varphi_2]$ , Neurons: [8]

#### [3] Benchmark1 Stability

$$f(x) = [u_0 + x_0 + x_1, u_1 - x_0 - x_1]$$

$$\mathcal{X} : Torus([0.0, 0.0], 10.0, 0.1)$$

Activation:  $[\varphi_2]$ , Neurons: [4] CTRLActivation:  $[\varphi_1]$ , Neurons: [15, 2]

#### [4] InvertedPendulum Stability

$$f(x) = \left[ u_0 + x_1, u_1 - \frac{8}{3}x_1 + 19.62 \sin(x_0) \right]$$

$$\mathcal{X} : Torus([0.0, 0.0], 1, 0.1)$$

Activation:  $[\varphi_2]$ , Neurons: [5] CTRLActivation:  $[\varphi_1]$ , Neurons: [25, 2]

#### [5] NonPoly1 ROA

$$f(x) = [2x_0^2x_1 - x_0, -x_1]$$

$$\mathcal{X}_I : (Sphere([-1, 1], 0.1) | Sphere([1, -1], 0.2))$$

Activation:  $[\sigma_{\text{soft}}]$ , Neurons: [5]

#### [6] LorenzSystem ROA

$$f(x) = \left[ u_0 - 10.0x_0 + 10.0x_1, u_1 - x_0x_2 + 28.0x_0 - x_1, u_2 + x_0x_1 - \frac{8}{3}x_2 \right]$$

$$\mathcal{X}_I : Sphere([0, 0, 0], 0.3)$$

Activation:  $[\varphi_2]$ , Neurons: [8] CTRLActivation:  $[\varphi_1]$ , Neurons: [8, 3]



**[7] Barr2 Safety**

$$f(x) = [x_1 - 1 + e^{-x_0}, -\sin^2(x_0)]$$

$$\mathcal{X} : x_0 \geq -2 \wedge x_1 \leq 2$$

$$\mathcal{X}_I : Sphere([-0.5, 0.5], 0.4)$$

$$\mathcal{X}_U : Sphere([0.7, -0.7], 0.3)$$

Activation:  $[\sigma_t]$ , Neurons: [15]

**[8] ObstacleAvoidance Safety**

$$f(x) = \left[ \sin(x_2), \cos(x_2), \frac{3x_0 \sin(x_2) + 3x_1 \cos(x_2)}{x_0^2 + x_1^2 + 0.5} - \sin(x_2) \right]$$

$$\mathcal{X} : x_0 \geq -2 \wedge x_1 \geq -2 \wedge x_2 \geq -1.57 \wedge x_0 \leq 2 \wedge x_1 \leq 2 \wedge x_2 \leq 1.57$$

$$\mathcal{X}_I : x_0 \geq -0.1 \wedge x_1 \geq -2 \wedge x_2 \geq -0.52 \wedge x_0 \leq 0.1 \wedge x_1 \leq -1.8 \wedge x_2 \leq 0.52$$

$$\mathcal{X}_U : x_0^2 + x_1^2 \leq 0.04$$

Activation:  $[\varphi_4]$ , Neurons: [25]

**[9] HighOrd8 Safety**

$$f(x) = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \\ -576x_0 - 2400x_1 - 4180x_2 - 3980x_3 - 2273x_4 - 800x_5 - 170x_6 - 20x_7]$$

$$\mathcal{X} : Rectangle([-2.2, -2.2, -2.2, -2.2, -2.2, -2.2, -2.2, -2.2], [2.2, 2.2, 2.2, 2.2, 2.2, 2.2, 2.2, 2.2])$$

$$\mathcal{X}_I : Rectangle([0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9], [1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1])$$

$$\mathcal{X}_U : Rectangle([-2.2, -2.2, -2.2, -2.2, -2.2, -2.2, -2.2, -2.2], \\ [-1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8, -1.8])$$

Activation:  $[\varphi_1]$ , Neurons: [10]

**[10] CtrlObstacleAvoidance Safety**

$$f(x) = [\sin(x_2), \cos(x_2), u_0 - \sin(x_2)]$$

$$\mathcal{X} : Rectangle([-10.0, -10.0, -\pi], [10.0, 10.0, \pi])$$

$$\mathcal{X}_I : Rectangle([4.0, 4.0, \frac{-\pi}{2}], [6.0, 6.0, \frac{\pi}{2}])$$

$$\mathcal{X}_U : Rectangle([-9, -9, \frac{-\pi}{2}], [-7.0, -6.0, \frac{\pi}{2}])$$

Activation:  $[\sigma_t]$ , Neurons: [15] CTRLActivation:  $[\sigma_t]$ , Neurons: [5, 1]

**[11] NonPoly3 SWA**

$$f(x) = [-0.1x_0x_1^3 - 3x_0, -x_1 + x_2, -x_2]$$

$$\mathcal{X} : \text{Tor}us([0, 0, 0], 3, 0.01)$$

$$\mathcal{X}_I : \text{Sphere}([-0.9, -0.9, -0.9], 1.0)$$

$$\mathcal{X}_U : (\text{Sphere}([0.4, 0.4, 0.4], 0.2) | \text{Sphere}([-0.4, 0.4, 0.4], 0.2))$$

Activation:  $[\varphi_2]$ , Neurons: [6] AltActivation:  $[\sigma_t]$ , AltNeurons: [5]

**[12] Barr3 SWA**

$$f(x) = \left[ x_1, \frac{1}{3}x_0^3 - x_0 - x_1 \right]$$

$$\mathcal{X} : \text{Rectangle}([-3, -2], [2.5, 1])$$

$$\mathcal{X}_I : \text{Rectangle}([0.4, 0.1], [0.8, 0.5])$$

$$\mathcal{X}_U : \text{Sphere}([-1, -1], 0.4)$$

Activation:  $[\varphi_2]$ , Neurons: [5] AltActivation:  $[\sigma_{\text{sig}}, \varphi_2]$ , AltNeurons: [5, 5]

**[13] SecondOrder SWA**

$$f(x) = [-x_0^3 + x_1, u_0]$$

$$\mathcal{X} : \text{Rectangle}([-1.5, -1.5], [1.5, 1.5])$$

$$\mathcal{X}_I : \text{Rectangle}([-0.5, -0.5], [0.5, 0.5])$$

$$\mathcal{X}_U : \text{Complement}(\text{Rectangle}([-1, -1], [1, 1]))$$

Activation:  $[\varphi_2]$ , Neurons: [8] AltActivation:  $[\varphi_2]$ , AltNeurons: [5] CTRLActivation:  $[\varphi_1]$ , Neurons: [8, 1]

**[14] ThirdOrder SWA**

$$f(x) = \left[ u_0 - 10x_0 + 10x_1, -x_0x_2 + 28x_0 - x_1, x_0x_1 - \frac{8}{3}x_2 \right]$$

$$\mathcal{X} : \text{Rectangle}([-6, -6, -6], [6, 6, 6])$$

$$\mathcal{X}_I : \text{Rectangle}([-1.2, -1.2, -1.2], [1.2, 1.2, 1.2])$$

$$\mathcal{X}_U : \text{Complement}(\text{Rectangle}([-5, -5, -5], [5, 5, 5]))$$

Activation:  $[\varphi_2]$ , Neurons: [10] AltActivation:  $[\sigma_t]$ , AltNeurons: [8] CTRLActivation:  $[\varphi_1]$ , Neurons: [8, 1]

---

**[15] SecondOrderLQR RWA**

$$f(x) = [-x_0^3 + x_1, -1.0x_0 - 1.73x_1]$$

$$\mathcal{X} : \text{Rectangle}([-1.5, -1.5], [1.5, 1.5])$$

$$\mathcal{X}_I : \text{Rectangle}([-0.5, -0.5], [0.5, 0.5])$$

$$\mathcal{X}_S : \text{Rectangle}([-1, -1], [1, 1])$$

$$\mathcal{X}_G : \text{Rectangle}([-0.05, -0.05], [0.05, 0.05])$$

Activation:  $[\varphi_2]$ , Neurons: [4]

**[16] ThirdOrderLQR RWA**

$$f(x) = \left[ -33.71x_0 - 8.49x_1, -x_0x_2 + 28x_0 - x_1, x_0x_1 - \frac{8}{3}x_2 \right]$$

$$\mathcal{X} : \text{Rectangle}([-6, -6, -6], [6, 6, 6])$$

$$\mathcal{X}_I : \text{Rectangle}([-1.2, -1.2, -1.2], [1.2, 1.2, 1.2])$$

$$\mathcal{X}_S : \text{Rectangle}([-5, -5, -5], [5, 5, 5])$$

$$\mathcal{X}_G : \text{Rectangle}([-0.3, -0.3, -0.3], [0.3, 0.3, 0.3])$$

Activation:  $[\varphi_2]$ , Neurons: [16]

**[17] SecondOrder RWA**

$$f(x) = [-x_0^3 + x_1, u_0]$$

$$\mathcal{X} : \text{Rectangle}([-1.5, -1.5], [1.5, 1.5])$$

$$\mathcal{X}_I : \text{Rectangle}([-0.5, -0.5], [-0.1, -0.1])$$

$$\mathcal{X}_S : (\text{Rectangle}([-1.5, -1.5], [1.5, 1.5]) \setminus \text{Sphere}([0.5, 0.5], 0.2))$$

$$\mathcal{X}_G : \text{Rectangle}([-0.05, -0.05], [0.05, 0.05])$$

Activation:  $[\sigma_{\text{sig}}, \varphi_2]$ , Neurons: [4, 4] CTRLActivation:  $[\varphi_1]$ , Neurons: [8, 1]

**[18] ThirdOrder RWA**

$$f(x) = \left[ u_0 - 10x_0 + 10x_1, -x_0x_2 + 28x_0 - x_1, x_0x_1 - \frac{8}{3}x_2 \right]$$

$$\mathcal{X} : \text{Rectangle}([-6, -6, -6], [6, 6, 6])$$

$$\mathcal{X}_I : \text{Rectangle}([-1.2, -1.2, -1.2], [1.2, 1.2, 1.2])$$

$$\mathcal{X}_S : \text{Rectangle}([-5, -5, -5], [5, 5, 5])$$

$$\mathcal{X}_G : \text{Rectangle}([-0.3, -0.3, -0.3], [0.3, 0.3, 0.3])$$

Activation:  $[\varphi_2]$ , Neurons: [5] CTRLActivation:  $[\varphi_1]$ , Neurons: [8, 1]

**[19] InvertedPendulum RWA**

$$f(x) = \left[ u_0 + x_1, u_1 - \frac{8}{3}x_1 + 19.62 \sin(x_0) \right]$$

$$\mathcal{X} : \text{Rectangle}([-3, -3], [3, 3])$$

$$\mathcal{X}_I : \text{Rectangle}([-0.6, -0.6], [0.6, 0.6])$$

$$\mathcal{X}_S : \text{Rectangle}([-2.5, -2.5], [2.5, 2.5])$$

$$\mathcal{X}_G : \text{Rectangle}([-0.01, -0.01], [0.01, 0.01])$$

Activation:  $[\sigma_{\text{sig}}]$ , Neurons: [5] CTRLActivation:  $[\varphi_1]$ , Neurons: [8, 2]

**[20] SecondOrderLQR RSWA**

$$f(x) = [-x_0^3 + x_1, -1.0x_0 - 1.73x_1]$$

$$\mathcal{X} : \text{Rectangle}([-1.5, -1.5], [1.5, 1.5])$$

$$\mathcal{X}_I : \text{Rectangle}([-0.5, -0.5], [0.5, 0.5])$$

$$\mathcal{X}_S : \text{Rectangle}([-1, -1], [1, 1])$$

$$\mathcal{X}_F : \text{Rectangle}([-0.05, -0.05], [0.05, 0.05])$$

Activation:  $[\varphi_2]$ , Neurons: [4]



**[21] ThirdOrderLQR RSWA**

$$f(x) = \begin{bmatrix} -33.71x_0 - 8.49x_1, & -x_0x_2 + 28x_0 - x_1, & x_0x_1 - \frac{8}{3}x_2 \end{bmatrix}$$

$$\mathcal{X} : \text{Rectangle}([-6, -6, -6], [6, 6, 6])$$

$$\mathcal{X}_I : \text{Rectangle}([-1.2, -1.2, -1.2], [1.2, 1.2, 1.2])$$

$$\mathcal{X}_S : \text{Rectangle}([-5, -5, -5], [5, 5, 5])$$

$$\mathcal{X}_F : \text{Rectangle}([-0.3, -0.3, -0.3], [0.3, 0.3, 0.3])$$

Activation:  $[\varphi_2]$ , Neurons: [16]

**[22] InvertedPendulumLQR RSWA**

$$f(x) = [-7.21x_0 - 0.34x_1, \quad -1.34x_0 - 2.997x_1 + 19.62 \sin(x_0)]$$

$$\mathcal{X} : \text{Rectangle}([-3, -3], [3, 3])$$

$$\mathcal{X}_I : \text{Rectangle}([-0.6, -0.6], [0.6, 0.6])$$

$$\mathcal{X}_S : \text{Rectangle}([-2.5, -2.5], [2.5, 2.5])$$

$$\mathcal{X}_F : \text{Rectangle}([-0.3, -0.3], [0.3, 0.3])$$

Activation:  $[\sigma_{\text{sig}}, \varphi_2]$ , Neurons: [5, 5]

**[23] SecondOrder RSWA**

$$f(x) = [-x_0^3 + x_1, \quad u_0]$$

$$\mathcal{X} : \text{Rectangle}([-1.5, -1.5], [1.5, 1.5])$$

$$\mathcal{X}_I : \text{Rectangle}([-0.5, -0.5], [0.5, 0.5])$$

$$\mathcal{X}_S : \text{Rectangle}([-1, -1], [1, 1])$$

$$\mathcal{X}_F : \text{Rectangle}([-0.05, -0.05], [0.05, 0.05])$$

Activation:  $[\varphi_2]$ , Neurons: [8] CTRLActivation:  $[\varphi_1]$ , Neurons: [8, 1]

**[24] InvertedPendulum RSWA**

$$f(x) = \left[ u_0 + x_1, u_1 - \frac{8}{3}x_1 + 19.62 \sin(x_0) \right]$$

$$\mathcal{X} : \text{Rectangle}([-3, -3], [3, 3])$$

$$\mathcal{X}_I : \text{Rectangle}([-0.6, -0.6], [0.6, 0.6])$$

$$\mathcal{X}_S : \text{Rectangle}([-2.5, -2.5], [2.5, 2.5])$$

$$\mathcal{X}_F : \text{Rectangle}([-0.3, -0.3], [0.3, 0.3])$$

Activation:  $[\sigma_{\text{sig}}, \varphi_2]$ , Neurons: [5, 5] CTRLActivation:  $[\varphi_1]$ , Neurons: [8, 2]

**[25] SecondOrderLQR RAR**

$$f(x) = [-x_0^3 + x_1, -1.0x_0 - 1.73x_1]$$

$$\mathcal{X} : \text{Rectangle}([-3.5, -3.5], [3.5, 3.5])$$

$$\mathcal{X}_I : \text{Rectangle}([-2, -2], [2, 2])$$

$$\mathcal{X}_S : \text{Rectangle}([-3, -3], [3, 3])$$

$$\mathcal{X}_G : \text{Rectangle}([-0.1, -0.1], [0.1, 0.1])$$

$$\mathcal{X}_F : \text{Rectangle}([-0.15, -0.15], [0.15, 0.15])$$

Activation:  $[\sigma_{\text{soft}}]$ , Neurons: [6] AltActivation:  $[\varphi_2]$ , AltNeurons: [6]

**[26] InvertedPendulum RAR**

$$f(x) = \left[ u_0 + x_1, u_1 - \frac{8}{3}x_1 + 19.62 \sin(x_0) \right]$$

$$\mathcal{X} : \text{Rectangle}([-3.5, -3.5], [3.5, 3.5])$$

$$\mathcal{X}_I : \text{Rectangle}([-2, -2], [2, 2])$$

$$\mathcal{X}_S : \text{Rectangle}([-3, -3], [3, 3])$$

$$\mathcal{X}_G : \text{Rectangle}([-0.1, -0.1], [0.1, 0.1])$$

$$\mathcal{X}_F : \text{Rectangle}([-0.2, -0.2], [0.2, 0.2])$$

Activation:  $[\sigma_{\text{sig}}, \varphi_2]$ , Neurons: [6, 6] AltActivation:  $[\sigma_{\text{sig}}, \varphi_2]$ , AltNeurons: [6, 6] CTRLActivation:  $[\varphi_1]$ , Neurons: [8, 2]