Polynomial Neural Barrier Certificate Synthesis of Hybrid Systems via Counterexample Guidance

Hanrui Zhao, Banglong Liu, Lydia Dehbi, Huijiao Xie, Zhengfeng Yang*, Haifeng Qian
Shanghai Key Lab of Trustworthy Computing, East China Normal University, Shanghai, China

***Abstract*—This paper presents a novel approach to the safety verification of hybrid systems by synthesizing neural barrier certificates (BCs) via counterexample-guided neural network (NN) learning combined with sum-of-square (SOS) based verification. We learn more easily verifiable BCs with NN polynomial expansions in a high-accuracy counterexamples guided framework. By leveraging the polynomial candidates yielded from the learning phase, we reformulate the identification of real BCs as convex Linear Matrix Inequality (LMI) feasibility testing problems, instead of directly solving the inherently NP-hard non-convex Bilinear Matrix Inequality (BMI) problems associated with SOS-based BC generation. Furthermore, we decompose the large SOS verification programming into several manageable sub-programmings. Benefiting from the efficiency and scalability advantages, our proposed approach can synthesize barrier certificates not amenable to existing methods and handle more general hybrid systems.**

***Index Terms*—Safety verification, Hybrid system, Polynomial neural barrier certificate, Counterexample guidance**

I. Introduction

Cyber-physical systems (CPS) seamlessly integrate physical components and software systems, allowing modeling as hybrid systems that describe the interaction of discrete transitions and continuous dynamics. With widespread applications in safety-critical domains such as air traffic control, automotive systems, and unmanned aerial vehicles, the safety verification of hybrid systems has garnered significant attention. In principle, the goal of safety verification is to assess whether a system, starting from an initial region, evolves into states that violate the desired safety properties.

Ensuring the safety of hybrid systems poses a formidable challenge. Many research efforts have been devoted to barrier certificate generation to handle the safety verification problem. A barrier certificate is a real function of state that separates the unsafe region from all system trajectories starting from a set of initial states [1]. As the existence of barrier certificate provides a safety guarantee, the problem of safety verification can be converted to barrier certificate synthesis. Compared with reachable set computation [2], barrier functions give more exact results over an infinite horizon.

The authors acknowledge the support of the National Key R&D Program of China (2023YFA1009402), the National Natural Science Foundation of China under Grant (No. 12171159), the Innovation Program of Shanghai Municipal Education Commission(2021-01-07-00-08-E00101), "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software under Grant (No. 22510750100), and ECNU Academic Innovation Promotion Program for Excellent Doctoral Students (No. YBNLTS2024-011). *Corresponding author: Zhengfeng Yang. Email: zfyang@sei.ecnu.edu.cn.

Barrier certificates (BCs) must consistently evaluate to nonnegative or negative values, regardless of their specific type. The conventional computational approach involves formulating a sum-of-squares (SOS) program for BC generation, which results in a bilinear matrix inequality (BMI) problem, a known NP-hard challenge [1], [3]. To make this problem more tractable, convex SOS relaxation techniques are employed, which convert the non-convex BMI problem into a convex Linear Matrix Inequality (LMI) one by fixing certain multipliers. Nonetheless, this approach may introduce conservatism in the verification conditions, potentially rendering solutions to the original BMI problem inaccessible within the derived LMI framework. Recently, data-driven methods that integrate BC learning with verification processes have emerged as a promising alternative. Zhao et al. initially developed neural BCs for continuous systems [4]. Peruffo et al. [5] made the pioneering effort to synthesize neural BCs for switch hybrid systems using a Counterexample-Guided Inductive Synthesis (CEGIS) [6] engine. However, their method was constrained by high-dimensional tasks due to limitations of the interval-based *dReal* [7] verifier. Zhao et al. [8] addressed scalability issues in solving neural BCs by substituting the less scalable SMT solver with an SOS-based verification approach, yet their method remains inadequate for handling hybrid systems.

This paper focus on synthesizing polynomial neural BC for each location in more general hybrid system encompassing guard conditions and reset functions. We propose a novel counterexample-guided framework that incorporates high-accuracy counterexamples refinement. In the BC learning phase, we employ interpretable polynomial NNs [9] to capture high-order nonlinear relationships among input variables through polynomial transformations and expansions, to facilitate more efficient acquisition of highly expressive and easily verifiable polynomial BCs of arbitrary degrees. During the verification phase, we convert the inherently non-convex BMI problem of BC generation into a convex LMI feasibility problem and further decompose large SOS programming into several manageable sub-programmings, which not only reduces computational time but also enhances the potential to identify more solutions beyond the reach of existing methods.

The main contributions of this paper are summarized as follows:

- Our proposed novel counterexample-guided framework combines polynomial neural BC learning and efficient SOS-based verification, which generates easily verifiable BC refined by high-accuracy counterexample set constructed by computing a minimum-volume ellipsoid.

- The SOS-based BC verification approach guarantees formal soundness, offering lower computational complexity than non-convex BMI problem and uncovering more feasible solutions compared to conventional LMI relaxation method for BC generation.
- We implement a tool named *SynHbc* and evaluate its performance on a series of benchmarks. The experiments demonstrate our *SynHbc* proves superior effectiveness and practicality compared with conventional BMI and LMI-based methods, as well as state-of-the-art counterexample-guided neural BC synthesis approaches.

The rest of the paper is organized as follows. We start by defining hybrid systems and safety in Section II. Section III illustrates how to transform the problem of barrier certificate generation into a BMI-solving problem. Section IV introduces the framework for synthesizing polynomial neural barrier certificates of hybrid systems. Section V presents experimental evaluations of our algorithm over a set of benchmark examples. Section VI reviews related work, and we discuss the limitations and conclude the paper in Section VII.

II. Preliminaries

Notations. Let $\mathbb{R}$ be the real numbers field and $\mathbb{N}$ be the natural numbers. Let the symbol $\mathbb{R}[\mathbf{x}] := \mathbb{R}[x_1, \ldots, x_n]$ represent the polynomial ring with coefficients in $\mathbb{R}$ over the variable $\mathbf{x} = [x_1, \ldots, x_n]^T$, and $\mathbb{R}[\mathbf{x}]^m$ denote the polynomial ring vector in m-dimensional. $\mathbb{R}[\mathbf{x}]_d$ is the set consisting of all real polynomials in $\mathbf{x}$ with degree at most d. S^n denotes the set of $n \times n$ symmetric matrices, and the notation $B \succeq 0$ means that the matrix $B \in S^n$ is positive semidefinite. Let $\Sigma[\mathbf{x}] \subset \mathbb{R}[\mathbf{x}]$ define the space of SOS polynomials. The notation $\Sigma[\mathbf{x}]_d$ refers to the d-th truncation of $\Sigma[\mathbf{x}]$, defined as $\Sigma[\mathbf{x}]_d := \Sigma[\mathbf{x}] \cap \mathbb{R}[\mathbf{x}]_d$.

A continuous dynamical system is modeled by a finite number of first-order ordinary differential equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{1}$$

where $\dot{\mathbf{x}}$ represents the derivative of $\mathbf{x}$ with respect to the time variable t and $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_n(\mathbf{x})]^T$ denotes the vector field defined on the state space $\Psi \subseteq \mathbb{R}^n$. The vector field $\mathbf{f}$ satisfies the local Lipschitz condition, ensuring that for a given initial state $\mathbf{x}(0) = \mathbf{x}_0$, there exists a time $T > 0$ and a unique function $\tau : [0, T) \mapsto \mathbb{R}^n$ such that $\tau(t) = \mathbf{x}(t)$. The trajectory of (1) from $\mathbf{x}_0$ is also denoted as $\mathbf{x}(t)$.

Hybrid systems demonstrate a combination of continuous dynamics and discrete transitions. We adopt the notion of hybrid automaton [10] to model hybrid systems as follows:

Definition 1 (Hybrid System): A hybrid system is defined as a tuple $\mathbf{H} : \langle L, X, F, \Psi, E, \mathcal{G}, \mathcal{R}, \Theta, \ell_0 \rangle$,where

- L, a finite set of locations;
- $X \subseteq \mathbb{R}^n$ is the continuous state space. The hybrid state space of the system is defined by $\mathcal{X} = L \times X$ and a state is defined by $(\ell, \mathbf{x}) \in \mathcal{X}$;
- $F : L \to (\mathbb{R}^n \to \mathbb{R}^n)$, assigns to each location $\ell \in L$ a locally Lipschitz continuous vector field $\mathbf{f}_\ell$;
- Ψ assigns to each location $\ell \in L$ a location invariant $\Psi(\ell) \subseteq \mathbb{R}^n$;
- $E \subseteq L \times L$ is a finite set of discrete transitions;
- $\mathcal{G}$ assigns to each transition $e \in E$ a switching guard $\mathcal{G}_e \subseteq \mathbb{R}^n$;
- $\mathcal{R}$ assigns to each transition $e \in E$ a reset function $\mathcal{R}_e : \mathbb{R}^n \to \mathbb{R}^n$;
- $\Theta \subseteq \mathbb{R}^n$, an initial continuous state set;
- $\ell_0 \in L$, an initial location. The initial state space is defined by $\ell_0 \times \Theta$.

The hybrid system undergoes evolution through either a discrete transition or a continuous flow. During a continuous flow, the system remains within a location ℓ, and its continuous state variables $\mathbf{x}$ evolve according to the differential equation $\dot{\mathbf{x}} = \mathbf{f}_\ell(\mathbf{x})$ while adhering to the location invariant $\mathbf{x} \in \Psi(\ell)$. An allowed transition $e = (\ell, \ell')$ in E enables the system to transit from one location ℓ to another ℓ', subject to the satisfaction of the switching guard $\mathcal{G}_e$ by the state variables $\mathbf{x}$. Furthermore, the reset function $\mathcal{R}_e$ assigns new values to the state variables $\mathbf{x}$ during this transition.

In this paper, the vector fields corresponding to the hybrid system are expressed as polynomials. The semi-algebraic sets $\Psi(\ell), \mathcal{G}_e, \Theta$ in $\mathbf{H}$ are represented by polynomial equations and inequalities. For clarity sake, we denote $\Psi(\ell)$, $\mathcal{G}_e$, and Θ as

$$\begin{aligned}\Psi(\ell) : &= \{\mathbf{x} \in \mathbb{R}^n \mid \psi_{\ell,1}(\mathbf{x}) \geq 0, \ldots, \psi_{\ell,r}(\mathbf{x}) \geq 0\},\\ \mathcal{G}_e : &= \{\mathbf{x} \in \mathbb{R}^n \mid g_{e,1}(\mathbf{x}) \geq 0, \ldots, g_{e,s}(\mathbf{x}) \geq 0\},\\ \Theta : &= \{\mathbf{x} \in \mathbb{R}^n \mid \theta_1(\mathbf{x}) \geq 0, \ldots, \theta_q(\mathbf{x}) \geq 0\},\end{aligned}$$

where $\ell \in L, e \in E$ and $\psi_{\ell,j}(\mathbf{x}), g_{e,k}(\mathbf{x}), \theta_i(\mathbf{x})$ are vectors of polynomials, with the inequalities being satisfied entry-wise. Furthermore, these semi-algebraic sets are assumed to be compact. Suppose that the location ℓ has an unsafe region $\Xi(\ell)$, which is a compact semi-algebraic set:

$$\Xi(\ell) := \{\mathbf{x} \in \mathbb{R}^n \mid \xi_{\ell,1}(\mathbf{x}) \geq 0, \ldots, \xi_{\ell,p}(\mathbf{x}) \geq 0\},$$

where $\xi_{\ell,v}(\mathbf{x}) \in \mathbb{R}^n, 1 \leq v \leq p$. The safety of hybrid system $\mathbf{H}$ can be defined with respect to the unsafe region $\Xi(\ell)$.

Definition 2 (Safety): For a hybrid system $\mathbf{H} : \langle L, X, F, \Psi, E, \mathcal{G}, \mathcal{R}, \Theta, \ell_0 \rangle$ with a predefined unsafe state set $\Xi(\ell)$, the system $\mathbf{H}$ is considered *safe* if every trajectory of $\mathbf{H}$ originating from the initial set $\ell_0 \times \Theta$ avoids reaching any state specified by $\Xi(\ell)$.

For safety verification of hybrid systems, barrier certificate plays a crucial role. A barrier certificate maps all reachable states to nonnegative reals and all unsafe states to negative reals. In the following theorem, we define an exponential-type barrier certificate of a hybrid system modified from [11].

Theorem 1: Let $\mathbf{H} : \langle L, X, F, \Psi, E, \mathcal{G}, \mathcal{R}, \Theta, \ell_0 \rangle$ be a hybrid system and $\Xi(\ell)$ be an unsafe region. Let $\lambda_\ell(\mathbf{x})$ be given polynomials for all $\ell \in L$, and $\gamma_e(\mathbf{x})$ be given nonnegative polynomials for all $e \in E$. If there exists a polynomial $B_\ell(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ for each location $\ell \in L$, which satisfies the following conditions:

(i) $B_{\ell_0}(\mathbf{x}) \geq 0 \quad \forall \mathbf{x} \in \Theta$,

(ii) $\mathcal{L}_f B_\ell(\mathbf{x}) - \lambda_\ell(\mathbf{x}) B_\ell(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in \Psi(\ell)$, here $\mathcal{L}_f B_\ell(\mathbf{x})$ denotes the Lie-derivative of $B_\ell(\mathbf{x})$ with respect to the vector field $\mathbf{f}_\ell(\mathbf{x})$, i.e., $\mathcal{L}_f B_\ell(\mathbf{x}) = \sum_{i=1}^{n} \frac{\partial B_\ell}{\partial x_i} \cdot f_{\ell,i}(\mathbf{x})$,

(iii) $B_{\ell'}(\mathbf{x}') - \gamma_e(\mathbf{x})B_\ell(\mathbf{x}) \geq 0 \quad \forall \mathbf{x}' = \mathcal{R}_e(\mathbf{x}) \ \forall \mathbf{x} \in \mathcal{G}_e,\ \forall e = (\ell, \ell') \in E$,

(iv) $B_\ell(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in \Xi(\ell)$,

then $B_\ell(\mathbf{x})$ is as a barrier certificate at the location ℓ, thereby ensuring the safety of the hybrid system $\mathbf{H}$.

Proof 1: Condition (i) ensures the nonnegativity of $B_{\ell_0}(\mathbf{x})$ on Θ. Condition (ii) implies that $\mathcal{L}_f B_\ell(\mathbf{x}) > 0$ whenever $B_\ell(\mathbf{x}) \geq 0$, ensuring that $B_\ell(\mathbf{x})$ remains nonnegative over the continuous flow. Given the non-negativity of $\gamma_e(\mathbf{x})$, condition (iii) ensures that $B_\ell(\mathbf{x})$ will not reach negative values at any discrete transition. Additionally, condition (iv) asserts that the reachable states of $\mathbf{H}$ and the unsafe region $\Xi(\ell)$ do not intersect. □

III. BMI for Barrier Certificate Generation

The problem of barrier certificate generation is an inherently infinite-dimensional problem. To make it amenable to polynomial optimization, the barrier certificate $B_\ell(\mathbf{x})$ should be constrained to a set of priori degree-bound polynomials. Investigating Theorem 1, verification conditions (i)-(iv) can be formulated as nonnegativity constraints for polynomials over the corresponding semi-algebraic sets. Given the degree bound, one may construct the template of the polynomial $B_\ell(\mathbf{x})$ with parameterized coefficients. To find real-valued coefficients for $B_\ell(\mathbf{x})$ involves solving a typical quantifier elimination problem with polynomial equalities and inequalities constraints, which involves high computational complexity.

To alleviate the computational challenge, the sum-of-square (SOS) relaxation-based approach [3] can be utilized to obtain $B_\ell(\mathbf{x})$. This approach starts with sufficient verification conditions expressed through SOS representations and proceeds by tackling semidefinite programming (SDP). Putinar's Positivstellensatz [12] provides a powerful representation for polynomial positivity on semi-algebraic sets. According to the (ii) and (iii) conditions of Theorem 1, where the parameters of $B_\ell(\mathbf{x})$ appear in the antecedent sides, the relevant SOS representation using Putinar's Positivstellensatz form non-convex BMI constraints due to the polynomial product between the barrier certificate and its polynomial multipliers. Subsequently, we present a detailed review of the procedure for transforming barrier certificate generation into BMI problem solving.

Initially, SOS relaxation is employed to represent the entailment checking in BC conditions as an SOS program. Indeed, all the conditions in Theorem 1 can be encoded as non-negative constraints on polynomials over relevant semi-algebraic sets, which relies on Putinar's Positivstellensatz. To achieve it, given a nonempty basic semi-algebraic set $\mathcal{K}(g)$ as:

$$\mathcal{K}(g) = \{\mathbf{x} \in \mathbb{R}^n \,|\, g_1(\mathbf{x}) \geq 0, \ldots, g_s(\mathbf{x}) \geq 0\}, \tag{2}$$

where $g_j \in \mathbb{R}[\mathbf{x}], 1 \leq j \leq s$. The *quadratic module* with respect to g is then defined as:

$$\mathcal{M}(g) := \Sigma[\mathbf{x}] + g_1\Sigma[\mathbf{x}] + \cdots + g_s\Sigma[\mathbf{x}].$$

Given a positive integer d, the $d-$th truncation of $M(g)$ is $M(g)_{2d} := \Sigma[\mathbf{x}]_{2d} + g_1\Sigma[\mathbf{x}]_{2d-\deg(g_1)} + \cdots + g_s\Sigma[\mathbf{x}]_{2d-\deg(g_s)}$. $M(g)$ is Archimedean if there exists a polynomial $b(\mathbf{x}) \in M(g)$ such that the set $\{\mathbf{x} \in \mathbb{R}^n \,|\, b(\mathbf{x}) \geq 0\}$ is compact.

Theorem 2: **[Putinar's Positivstellensatz** [12]**]** Let $\mathcal{K}(g) \subset \mathbb{R}[\mathbf{x}]$ be as in (2). Assume that the quadratic module $M(\mathbf{g})$ is archimedean. If $f(\mathbf{x}) > 0$ on $\mathcal{K}(g)$, then we have $f(\mathbf{x}) \in M(g)$, i.e., $f(\mathbf{x})$ can be represented as

$$f(\mathbf{x}) = \sigma_0(\mathbf{x}) + \sum_{i=1}^{s} \sigma_i(\mathbf{x}) g_i(\mathbf{x}), \tag{3}$$

where $\sigma_i \in \Sigma[\mathbf{x}], 0 \leq i \leq s$.

Following Theorem 2, the existence of the representation (3) provides a sufficient and necessary condition for the strict positivity of $f(\mathbf{x})$ on the compact set $\mathcal{K}(g)$. However, the high degrees of polynomials in (3) increase computational complexity, presenting more challenge for the SOS representation generation. To overcome this, we introduce a degree bound $2d$ and select the polynomial products in (3) accordingly. This strategy allows the construction of a sufficient condition for the nonnegativity of the given polynomial $f(\mathbf{x})$ on the semi-algebraic set $\mathcal{K}(g)$, that is, $f(\mathbf{x}) \in M(g)_{2d}$. Concretely,

$$f(\mathbf{x}) = \sigma_0(\mathbf{x}) + \sum_{i=1}^{s} \sigma_i(\mathbf{x}) g_i, \tag{4}$$

where $\sigma_0(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}$ and $\sigma_i \in \Sigma[\mathbf{x}]_{2d-\deg(g_i)},\ 1 \leq i \leq s$. Thus, the expression in (4) for the non-negative polynomial $f(\mathbf{x})$ on $\mathcal{K}(g)$ can be obtained by solving an SDP problem.

Notably, representation (4) ensures that a polynomial is nonnegative on a given semi-algebraic set. At this point, all conditions in Theorem 1 can be deduced as a unified type, namely, non-negativity of the polynomial over a semi-algebraic set. It implies the verification criteria can be simplified into more tractable form by the representation in (4).

Theorem 3: Let us examine the hybrid system $\mathbf{H} : \langle L, X, F, \Psi, E, \mathcal{G}, \mathcal{R}, \Theta, \ell_0 \rangle$ with the unsafe set $\Xi(\ell)$ defined previously. Consider the polynomials $\{B_\ell(\mathbf{x})\}$, and let d be a positive integer. If there exist $\lambda_\ell(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ and nonnegative polynomial $\gamma_e(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ with $\deg(\lambda) \leq 2d$, positive values $\epsilon_{\ell,1}, \epsilon_{\ell,2}$ and vectors of sums-of-squares $\sigma(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}$, $\phi_\ell(\mathbf{x}) \in \Sigma_{2d}[\mathbf{x}]$, $\eta_\ell(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}$, $\delta_\ell(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}$, such that the following conditions are satisfied:

1) $B_{\ell_0}(\mathbf{x}) - \sigma(\mathbf{x})\theta(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}$,
2) $\mathcal{L}_f B_\ell(\mathbf{x}) - \lambda_\ell(\mathbf{x})B_\ell(\mathbf{x}) - \phi_\ell(\mathbf{x})\psi_\ell(\mathbf{x}) - \epsilon_{\ell,1} \in \Sigma[\mathbf{x}]_{2d}$,
3) $B_{\ell'}(\mathcal{R}_e(\mathbf{x})) - \gamma_e(\mathbf{x})B_\ell(\mathbf{x}) - \eta_\ell(\mathbf{x})g_\ell(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}$,
4) $-B_\ell(\mathbf{x}) - \delta_\ell(\mathbf{x})\xi_\ell(\mathbf{x}) - \epsilon_{\ell,2} \in \Sigma[\mathbf{x}]_{2d}$,

for each $\ell \in L$ and $e = (\ell, \ell') \in E$. Then $\{B_\ell(\mathbf{x})\}$ satisfy the conditions in Theorem 1, thereby serving as the barrier certificates of $\mathbf{H}$, and the safety of $\mathbf{H}$ is ensured.

Proof 2: As stated in equation (3), the sum-of-square representation demonstrates that the fulfillment of conditions (1-4) leads correspondingly to the satisfaction of conditions (i-iv) in Theorem 1. Therefore, the claim is proved. □

Considering the unknown multipliers $\lambda_\ell(\mathbf{x})$ and $\gamma_e(\mathbf{x})$, along with the unknown barrier certificates $\{B_\ell(\mathbf{x})\}$, the constraints in Theorems 3 introduce nonlinear terms that involve products of coefficients from unknown polynomials. This results in a non-convex BMI problem, which is essentially NP-hard. To simplify the problem, a common approach is to pre-specify the multipliers $\lambda_\ell(\mathbf{x})$ and $\gamma_e(\mathbf{x})$, thereby reducing the BMI constraint into associated LMI one. Unfortunately, the

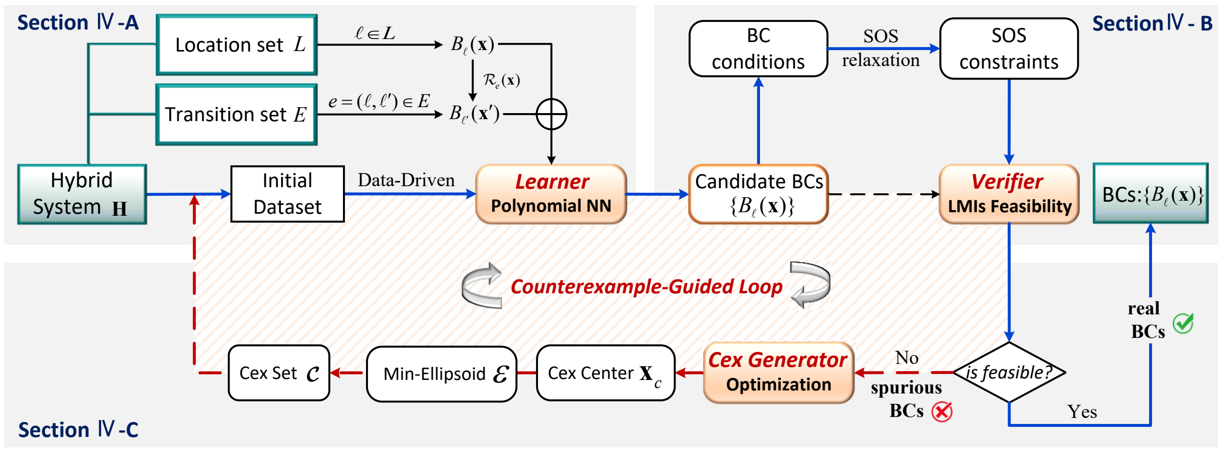


Fig. 1: The framework of *SynHbc*.

relaxed LMI problem tends to be more conservative than the original BMI problem. In other words, the overly conservative verification conditions exclude barrier certificates that satisfy non-convex conditions but not the stronger convex conditions.

This paper introduces a novel approach to directly tackle the challenging BMI problem outlined above. Our method integrates counterexample-guided NN learning and SOS-based verification to effectively generate barrier certificates. By training NNs with polynomial expansions, we derive explicit scalar polynomial barrier functions and their corresponding multipliers. This enables us to utilize known $\{B_\ell(\mathbf{x})\}$ in Theorem 3, thereby simplifying the non-trivial BMI solving verification phase to merely testing the feasibility of LMI problems for identifying real BCs. Notably, our method ensures that learned multipliers, as opposed to arbitrary ones, encompass a broader spectrum of feasible BCs solutions, thus enhancing effectiveness and practicality of barrier certificate generation compared to traditional BMI or LMI-based approaches. A comprehensive overview of our proposed framework is provided in the subsequent section.

IV. The Counterexample-Guided Framework for Neural Barrier Certificate Synthesis

In this section, we present an iterative framework named *SynHbc* for synthesizing polynomial neural barrier certificates for hybrid systems. The framework, depicted in Fig. 1, comprises three components: *Learner*, *Verifier*, and counterexample generator (short for *Cex Generator*). The *Learner* component trains neural networks to generate barrier certificate candidates, while the *Cex Generator* computes counterexample set to refine the spurious candidates identified by the *Verifier*, ultimately yielding real ones with formal safety guarantee.

Completely, the *Learner* initially trains NNs in a data-driven approach to generate polynomial-type BC candidates that satisfy the verification conditions within finite sampled datasets (more details are in Section IV-A). To ensure the learned candidates correctness over entire dense state domains, the *Verifier* adopts a formal method of establishing Linear Matrix Inequalities (LMIs) constraints based on SOS relaxation and identifies real BCs by solving LMI feasibility testing problems (outlined in Section IV-B). In the case where the verification process fails, the *Cex Generator* constructs a counterexample set derived from computing a minimum-volume ellipsoid through polynomial optimization (refer to Section IV-C) and feeds it back to the *Learner* for refining the spurious BC candidates. The BC synthesis process outlined above adheres to an inductive procedure under counterexample guidance (illustrated by the red lines in Fig. 1), ultimately yielding real barrier certificates $\{B_\ell(\mathbf{x})\}$ (followed the blue lines in Fig. 1) for the hybrid system $\mathbf{H}$.

A. The Learner

For a hybrid system $\mathbf{H} : \langle L, X, F, \Psi, E, \mathcal{G}, \mathcal{R}, \Theta, \ell_0\rangle$, the first component, *Learner*, seeks to generate candidate barrier certificate $B_\ell(\mathbf{x})$ for each location ℓ. This involves structuring a neural network, with reference to polynomial NN [9], for forming a polynomial-type $B_\ell(\mathbf{x})$ and concurrently training it alongside the learnable parameters of the multipliers $\lambda_\ell(\mathbf{x})$ and $\gamma_e(\mathbf{x})$. The training procedure incorporates datasets sampled in batches from various regions within $\mathbf{H}$, resulting in ℓ scalar functions, $\{B_\ell(\mathbf{x})\}$, that closely approximate the conditions in Theorem 1 and serve as barrier certificate candidates.

1) NN Architecture: Polynomial neural networks (e.g. Π-Nets [13]) have been demonstrated to possess strong expressive power, even yielding favorable outcomes across a plethora of tasks and signals (such as images, graphs, and differential equations [14], [15]) without employing nonlinear activation functions (which may lead to challenging neural network verification problems). Inspired by the polynomial NNs, we aim to train polynomial BCs that can be directly applied to effective SOS-based verification, thereby avoiding the non-trivial verification of the nonlinear neural BCs (e.g. ReLU NN forms). In the following, we propose a neural BC network based on polynomial expansions to approximate polynomial-type barrier functions.

Given a neural network associated with a location $\ell \in L$ within the hybrid system $\mathbf{H}$, which comprises input layer with n neurons (where n denotes the dimension of $\mathbf{f}_\ell$), h hidden layers, and a single-neuron output layer (for scalar output). More precisely, our objective is to learn a polynomial function

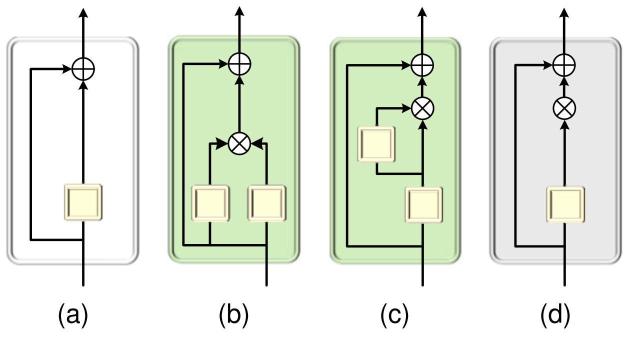


Fig. 2: A diagram of polynomial expansion modes from different architectures. The yellow box means an affine function, $\oplus$ and $\otimes$ denote the addition and Hadamard product (element-wise) operations, respectively. Subfigure 2a denotes the fundamental first-order polynomial. Subfigure 2b describes for the d-degree BC. Subfigure 2c and Subfigure 2d show the $2d$-degree BC composed of polynomial expansion and traditional *Square* polynomial activation function.

approximator, where each element of the output is represented as a polynomial with variables $\mathbf{x} = [x_1, \ldots, x_n]^T$ in the input layer. Let $p(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ be a parametric polynomial with total degree d, which can be expressed as

$$p(\mathbf{x}) = \sum_{\alpha} b_\alpha \mathbf{x}^\alpha,$$

where $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ is a monomial with $|\alpha| = \sum_{i=1}^{n} \alpha_i \leq d$, and b_α is the unknown coefficient corresponding to $\mathbf{x}^\alpha$.

[Polynomial Expansion]. Below, polynomials express a relationship between input variables and (learnable) coefficients; this relationship only involves the two core operations of addition and multiplication. Then the polynomial captures the relationships between the different elements of the input vector. The polynomial can either capture the interactions across every element of the matrix with every other element.

Let us express the output $p(\mathbf{x})$ as a polynomial expansion with degree d and a n-dimensional input $\mathbf{x} \in \mathbb{R}^n$, that is,

$$p(\mathbf{x}) = b + \sum_{|\alpha|=1} w_\alpha^{[1]} \mathbf{x}^\alpha + \sum_{|\alpha|=2} w_\alpha^{[2]} \mathbf{x}^\alpha + \cdots + \sum_{|\alpha|=d} w_\alpha^{[d]} \mathbf{x}^\alpha,$$

where $b \in \mathbb{R}$ is the constant term, and w_α^i is the coefficient corresponding to the monomial $\mathbf{x}^\alpha$ with degree i, $1 \leq i \leq d$. We adopt various modes as shown in Fig. 2 to consider the product of low-degree polynomials as final result, aiming to efficiently derive polynomial function $p(\mathbf{x})$ serves as BC candidate with improved expressive power and ease of verification.

As shown in Fig. 2, training a first-order polynomial referenced as Fig. 2a yields a straightforward linear-type BC. Fig. 2b provides a polynomial expression to form d-dimensional BC by multiplying arbitrary network layer output with first-order polynomial of input. Comparing Fig. 2c with the *Square* activation function in Fig. 2d, the former polynomial expansion mode enables the discovery of more expressive $2d$-degree BCs. That is attributed to *Square* activation function [13] performs a single sum-of-square operation with the same input parameters, whereas Fig. 2c introduces more learnable parameters, mitigating network divergence from poor initialization and enhancing the likelihood of successful BC network training.

[Empirical loss function]. The BC learning process involves adjusting the NN parameters by minimizing a loss function that acts as a proxy for the conditions outlined in Theorem 1, thereby progressively satisfying the $B_\ell(\mathbf{x})$ requirements. To avoid potential challenges posed by conventional Lie-derivative condition on measure-zero set where $B_{\ell_m}(\mathbf{x}) = 0$ [16], we introduce auxiliary multipliers $\lambda_m(\mathbf{x})$. Additionally, we incorporate similarly learneable nonnegative polynomial multipliers $\gamma_e(\mathbf{x})$ to effectively manage transitions $e = (\ell_m, \ell'_m)$. These considerations lead to a customized empirical loss function, $L_{hyb} = L_I + L_U + L_D + L_G$, namely:

$$\begin{cases} L_I = \sum\limits_{\mathbf{x}\in\Theta} L_I(\mathbf{x}) = \sum\limits_{\mathbf{x}\in S_I} \max\{\tau, -B_{\ell_0}(\mathbf{x})\}, \\ L_U = \sum\limits_{\mathbf{x}\in\Xi(\ell)} L_{U_{(\ell)}}(\mathbf{x}) = \sum\limits_{m=1}^{|\ell|} \sum\limits_{\mathbf{x}\in S_{U_m}} \max\{\tau, B_m(\mathbf{x})\}, \\ L_D = \sum\limits_{\mathbf{x}\in\Psi(\ell)} L_{D_{(\ell)}}(\mathbf{x}) \\ \quad = \sum\limits_{m=1}^{|\ell|} \sum\limits_{\mathbf{x}\in S_{D_m}} \max\{\tau, -(\mathcal{L}_f B_m(\mathbf{x}) - \lambda_m(\mathbf{x}) \cdot B_m(\mathbf{x}))\}, \\ L_G = \sum\limits_{\mathbf{x}\in\mathcal{G}_e} L_{G_e}(\mathbf{x}) \\ \quad = \sum\limits_{e=(\ell,\ell')} \sum\limits_{\mathbf{x}\in S_{G_e}} \max\{\tau, -(B_{\ell'}(\mathbf{x}') - \gamma_e(\mathbf{x}) \cdot B_\ell(\mathbf{x}))\} \\ \qquad + \sum\limits_{e=(\ell,\ell')} \sum\limits_{\mathbf{x}\in S_{G_e}} \max\{\tau, -\gamma_e(\mathbf{x})\}, \end{cases}$$

where terms L_I, L_U, L_D, and L_G denote sub-loss functions associated with the constraints of the initial, unsafe, Lie-derivative, and guard conditions in Theorem 1, respectively. S_I, S_U, S_D, and S_G are initial training datasets sampled from regions Θ, Ξ, Ψ, $\mathcal{G}$, which are iteratively enhanced by incorporating counterexamples fed back from the inductive loops. To minimize L_{hyb}, we employ the $\max\{\tau, *\}$ function to impose the penalties on each portion, where τ is a small positive tolerances to ensure strict positive or negative of the inequalities. We compute the above loss function in a parallel fashion, and similarly, the generated counterexamples are added to the relevant batch for further NN refinement.

2) Initial dataset sampling: The initial dataset plays a crucial role in NN training, directly influencing the quality of barrier certificate candidates generated in the first iteration and potentially impacting the overall efficiency of the procedure. To strike a balance between uniform samples in the inner space and high-value boundary samples (which poses challenges for continuous functions with predetermined properties), we implement an innovative initial dataset sampling technique for the following two typical state-space shapes:

(i) Box state-space: Considering a box sampling state-space $\mathcal{B}$, which is represented as $\mathcal{B} = \{\mathbf{x} \in \mathbb{R}^n | \ |x_i - c_i| \leq b_i\}$, where $\mathbf{x}_c = [c_1, \cdots, c_n]^T$ is the center of the box, and $b_i \in \mathbb{R}_{>0}$, i.e.

$$\mathcal{B} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} = \mathbf{x}_c + D\mathbf{u}, \ \ ||\mathbf{u}||_\infty \leq 1\},$$

where $D = \mathrm{diag}(b_1, \cdots, b_n)$.

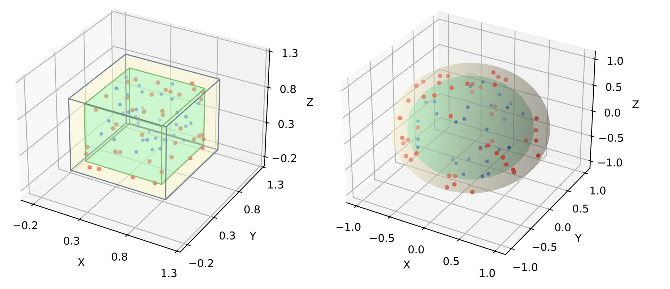


Fig. 3: Initial near-boundary samples (the red points) and inner samples (the blue points) from box and ellipsoid state-spaces.

(ii) Ellipsoid state-space: Considering an ellipsoid sampling state-space $\mathcal{E}$, which is expressed as:

$$\mathcal{E} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} = \mathbf{x}_c + A\mathbf{u}, \|\mathbf{u}\|_2 \leq 1\},$$

where $\mathbf{x}_c$ is the center of the ellipsoid, and A is nonsingular.

Then, the near-boundary and inner sampling areas of the box and ellipsoid state-spaces can be divided into:

$$\begin{cases} \mathcal{B}_b = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} = \mathbf{x}_c + D\,\mathbf{u},\ 1-\epsilon \leq ||\mathbf{u}||_\infty \leq 1\}, \\ \mathcal{B}_i = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} = \mathbf{x}_c + D\,\mathbf{u},\ ||\mathbf{u}||_\infty < 1-\epsilon\}, \end{cases}$$

$$\begin{cases} \mathcal{E}_b = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} = \mathbf{x}_c + A\,\mathbf{u},\ 1-\epsilon \leq ||\mathbf{u}||_2 \leq 1\}, \\ \mathcal{E}_i = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} = \mathbf{x}_c + A\,\mathbf{u},\ \|\mathbf{u}\|_2 < 1-\epsilon\}, \end{cases}$$

respectively, where ϵ is a small positive parameter used to adjust the internal-external proportion, and the initial dataset sampling results are illustrated in Fig. 3.

B. The Verifier

In the previous section, we demonstrated how the *Learner* generates $B_\ell(\mathbf{x})$ candidates for each location ℓ within a given hybrid system $\mathbf{H}$ and unsafe region $\Xi(\ell)$. However, it is worth mentioning that these candidates may not strictly satisfy the conditions in Theorem 1 across the entire state space. Thus, our focus shifts to identifying real barrier certificates that fulfill all BC requirements. Concretely, given that the BC terms in the BMI problem from Theorems 3 are now determined, the verification process can be reformulated as the following LMI feasibility testing problems:

$$\begin{cases} \text{find} \quad \sigma(\mathbf{x}), \delta_\ell(\mathbf{x}), \phi_\ell(\mathbf{x}), \eta_\ell(\mathbf{x}), \lambda_\ell(\mathbf{x}), \gamma_{\mathrm{e}}(\mathbf{x}) \\ \text{s.t.} \quad B_{\ell_0}(\mathbf{x}) - \sigma(\mathbf{x})\theta(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}, \\ \qquad \mathcal{L}_f B_\ell(\mathbf{x}) - \lambda_\ell(\mathbf{x})B_\ell(\mathbf{x}) \\ \qquad\qquad -\phi_\ell(\mathbf{x})\psi_\ell(\mathbf{x}) - \epsilon_{\ell,1} \in \Sigma[\mathbf{x}]_{2d}, \\ \qquad B_{\ell'}(\mathbf{x}') - \gamma_{\mathrm{e}}(\mathbf{x})B_\ell(\mathbf{x}) - \eta_\ell(\mathbf{x})g_\ell(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}, \\ \qquad -B_\ell(\mathbf{x}) - \delta_\ell(\mathbf{x})\xi_\ell(\mathbf{x}) - \epsilon_{\ell,2} \in \Sigma[\mathbf{x}]_{2d}, \end{cases} \tag{5}$$

where $\epsilon_{\ell,1}, \epsilon_{\ell,2} \in \mathbb{R}_{>0}$ are prespecified small positive numbers, parameters $\sigma(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}$, $\delta(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}$, $\eta_\ell(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}$, $\phi_\ell(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}$, and $\lambda_\ell(\mathbf{x})$ is an arbitrary polynomial with degree $\deg(\lambda) \leq 2d$ whereas $\gamma_e(\mathbf{x})$ is a positive one.

Furthermore, the large SOS programming (5) is equivalent to several SOS programming sub-problems with fewer multipliers, which can be tackled sequentially. More precisely, (5) corresponds to the following (6-9) problems:

$$\begin{cases} \text{find} \quad \sigma(\mathbf{x}) \\ \text{s.t.} \quad B_{\ell_0}(\mathbf{x}) - \sigma(\mathbf{x})\theta(\mathbf{x}) \in \Sigma_{2d}[\mathbf{x}]_{2d}, \end{cases} \tag{6}$$

$$\begin{cases} \text{find} \quad \phi_\ell(\mathbf{x}), \lambda_\ell(\mathbf{x}) \\ \text{s.t.} \quad \mathcal{L}_f B_\ell(\mathbf{x}) - \lambda_\ell(\mathbf{x})B_\ell(\mathbf{x}) - \phi_\ell(\mathbf{x})\psi_\ell(\mathbf{x}) \\ \qquad -\epsilon_{\ell,1} \in \Sigma[\mathbf{x}]_{2d}, \end{cases} \tag{7}$$

$$\begin{cases} \text{find} \quad \eta_\ell(\mathbf{x}), \gamma_{\mathrm{e}}(\mathbf{x}) \\ \text{s.t.} \quad B_{\ell'}(\mathbf{x}') - \gamma_e(\mathbf{x})B_\ell(\mathbf{x}) - \eta_\ell(\mathbf{x})g_\ell(\mathbf{x}) \in \Sigma[\mathbf{x}]_{2d}, \end{cases} \tag{8}$$

$$\begin{cases} \text{find} \quad \delta_\ell(\mathbf{x}) \\ \text{s.t.} \quad -B_\ell(\mathbf{x}) - \delta_\ell(\mathbf{x})\xi_\ell(\mathbf{x}) - \epsilon_{\ell,2} \in \Sigma[\mathbf{x}]_{2d}. \end{cases} \tag{9}$$

Due to the equivalence between (5) and (6-9), solving the sub-problems (6-9) allows us to determine the validity of the BC candidates $\{B_\ell(\mathbf{x})\}$ more efficiently. In case all sub-problems are feasible, $\{B_\ell(\mathbf{x})\}$ serve as real BCs. Otherwise, $\{B_\ell(\mathbf{x})\}$ may be considered spurious. Many SDP solvers, such as *MOSEK* [17], *SOSTOOLS* [18] and *YALMIP* [19], are available to solve the aforementioned SDP problems.

C. The Counterexample Generator

Expanding on the previous section, the failure of any problems in (6-9) to find a feasible solution may indicate the existence of counterexample (Cex) that renders the BC candidates spurious. Therefore, if candidates $\{B_\ell(\mathbf{x})\}$ do not pass the verification, we will capture Cexs fed to the *Learner* for retraining new BC candidates. Furthermore, recognizing that a single Cex may not always offer insightful information to NN updating, we aim to construct a Cex set for guidance.

We present an effective technique for computing a Cex set based on polynomial optimization. This approach increases the likelihood of finding more potential Cexs and improves their overall quality. As a result, it is possible to reduce the number of iterations in the inductive framework, ultimately enhancing the efficiency of synthesizing neural barrier certificates. Informally, the construction of Cex set proceeds in two stages:

- **[Computing the Cex center $\mathbf{x}_c$]:** solve optimal solution of polynomial optimization problem that most violates the BC verification conditions as the worst Cex $\hat{\mathbf{x}}$. Then, the midpoint $\mathbf{x}_c$ (depicted in Fig. 4a) of the shortest distance between $\hat{\mathbf{x}}$ and the candidate $B_\ell(\mathbf{x})$ quadratic function is computed as the Cex center.
- **[Computing a minimum-volume ellipsoid $\mathcal{E}$]:** randomly sample points from a δ-ball centered at $\mathbf{x}_c$, and then compute a minimum-volume ellipsoid $\mathcal{E}$ (shown in Fig. 4c) that encloses all filtered real Cexs (red points depicted in Fig. 4b). Consequently, we can construct a high-accuracy Cex set $\mathcal{C}$ from region $\mathcal{E}$ with an elevated probability of encountering more real Cexs.

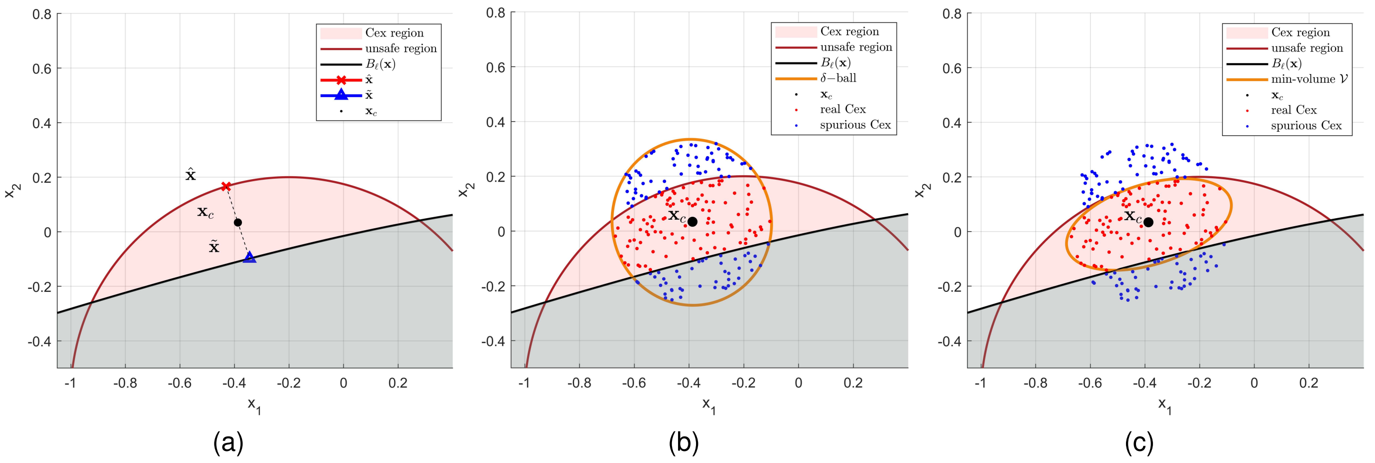


Fig. 4: Schematic diagram of Cex set construction. The gray shadow shields part of the unsafe region separated by quadratic function $B_\ell(\mathbf{x})$, while the uncovered area is the Cex region. Subfigure 4a involves the Cex center $\mathbf{x}_c$ between the computed worst Cex $\hat{\mathbf{x}}$ and the point $\tilde{\mathbf{x}}$ at $B_\ell(\mathbf{x})$ nearest to $\hat{\mathbf{x}}$. Subfigure 4b shows random samples from a ball with center $\mathbf{x}_c$ and radius δ, filtering into real Cexs (the red points) and spurious ones (the blue points). Subfigure 4c illustrates the computed minimum-volume ellipsoid $\mathcal{E}$, covering larger Cex region with more real Cexs.

1) Computing the Cex center $\mathbf{x}_c$: Given the impracticality of exploring all potential Cexs, our goal is first to identify a worst Cex, denoted as $\hat{\mathbf{x}}$, which notably violates the BC condition. For instance, consider the unsafe condition (iv) in Theorem 1. If points exist in $\Xi(\ell)$ where $B_\ell(\mathbf{x}) \geq 0$, then it may be assumed there are Cexs. We compute the worst Cex $\hat{\mathbf{x}}_{\Xi(\ell)}$ by solving the following optimization problem:

$$\begin{cases} \underset{\mathbf{x}}{\text{maximize}} & B_\ell(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in \Xi(\ell). \end{cases} \tag{10}$$

If the yielded $B_\ell(\mathbf{x})$ optimal value $P^* < 0$, then there is no Cex; otherwise, the variable value $\mathbf{x}$ at P^* is exactly the worst Cex $\hat{\mathbf{x}}_{\Xi(\ell)}$ we need.

To collect more accurate additional Cexs than those randomly sampled outward from the worst Cex $\hat{\mathbf{x}}$ [8], and to cover the largest possible real Cex region for constructing a Cex set, we search for a point $\tilde{\mathbf{x}}$ at $B_\ell(\mathbf{x}) = 0$ nearest to $\hat{\mathbf{x}}$. It can be achieved by solving optimization problem

$$\begin{cases} \underset{\mathbf{x}}{\text{minimize}} & \|\mathbf{x} - \hat{\mathbf{x}}\|_2 \\ \text{s.t.} & B_\ell(\mathbf{x}) = 0. \end{cases} \tag{11}$$

Then, we find the optimizer of (11), denoted by $\tilde{\mathbf{x}}$, and define the Cex center $x_c = \frac{\hat{\mathbf{x}}+\tilde{\mathbf{x}}}{2}$ and sample within a δ-ball neighborhood as shown in Fig. 4b.

After filtering out spurious Cexs, it can be observed that more random samples are falling into the Cex region and those are the real Cexs (red points in Fig. 4c). Therefore, our objective is to utilize this portion of samples to compute a minimum-volume ellipsoid $\mathcal{E}$, which covers more Cex region than a simple ball with radius of $\frac{\|\tilde{\mathbf{x}}-\hat{\mathbf{x}}\|_2}{2}$. This facilitates rapid sampling from within $\mathcal{E}$ to obtain more real Cexs for constructing a high-quality Cex set.

2) Computing a minimum-volume ellipsoid $\mathcal{E}$: Without loss of generality, we denote the filtered real Cexs as a finite set $S = \{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$, then we consider the problem of finding a minimum-volume ellipsoid $\mathcal{E} = \{\mathbf{x} \in \mathbb{R}^n | \|A\mathbf{x} - \mathbf{b}\|_2 \leq 1\}$:

$$\begin{cases} \text{minimize} & \log\det A^{-1} \\ \text{s.t.} & \|A\mathbf{x}_i - \mathbf{b}_i\|_2 \leq 1, \ \ i = 1, \dots, m, \\ & A \succ 0, \end{cases} \tag{12}$$

with variables $A \in \mathbb{S}^n$ and $\mathbf{b} \in \mathbb{R}^n$. Since the objective and constraints are both convex, the optimization problem (12) is convex, making it efficiently tractable. Once we compute $\tilde{A}$ and $\tilde{\mathbf{b}}$ from (12) and determine minimum-volume ellipse as

$$\mathcal{E} = \{\mathbf{x} \in \mathbb{R}^n | \, \|\tilde{A}\mathbf{x} - \tilde{\mathbf{b}}\|_2 \leq 1\}.$$

For ease of sampling, the parameterized representation of $\mathcal{E}$ is given as

$$\mathcal{E} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} = (\tilde{A})^{-1}\tilde{\mathbf{b}} + (\tilde{A})^{-1}\mathbf{u}, \|\mathbf{u}\|_2 \leq 1\}.$$

Then we construct a high-quality Cex set $\mathcal{C} = \{\mathbf{x}_1, \dots, \mathbf{x}_k\} \subseteq \mathbb{R}^n$, where the sampling points $\mathbf{x}_1, \dots, \mathbf{x}_k$ are chosen from $\mathcal{E}$ randomly by setting $\|\mathbf{u}\|_2 \leq 1$, making them more likely to be real Cexs. Thus the Cex set $\mathcal{C}$ may more effectively guide the barrier certificate refinement.

D. Algorithm

In this section, we present Algorithm 1 to summarize the procedure called *SynHbc* of synthesizing polynomial neural barrier certificates for hybrid systems. In Line 6, $\mathit{Learner}(\mathcal{NN}, \lambda_\ell(\mathbf{x}), \gamma_e(\mathbf{x}), \mathcal{S})$ denotes the training process of BC candidates which is elaborated in Subsection IV-A; In Line 8, $\mathit{Verifier}(B_\ell(\mathbf{x}), \mathbf{H}, \Xi(\ell))$ shows the verification process for checking validity of the learned candidates by solving SOS-based LMIs feasibility testing problems described in Subsection IV-B; Line 11 to line 14 means the Cex set construction by $\mathit{Cex\ Generator}$ provided in Subsection IV-C.

*Theorem 4 (***Soundness***):* If $B_\ell(\mathbf{x})$ is the function produced by Algorithm 1 for each $\ell \in L$ and $e \in E$, then $\{B_\ell(\mathbf{x})\}$ satisfies all conditions of barrier certificates of hybrid system $\mathbf{H}$ in Theorem 1, and the safety of $\mathbf{H}$ is guaranteed.

Algorithm 1: *SynHbc*: Synthesizing hybrid system barrier certificates.

```
   Input: hybrid system $\mathbf{H} : \langle L, X, F, \Psi, E, \mathcal{G}, \mathcal{R}, \Theta, \ell_0 \rangle$;
          unsafe set $\Xi(\ell)$
   Output: $\{B_\ell(\mathbf{x})\}$: the real barrier certificates set
1  for each location $\ell \in L$ and transition $e \in E$ do
2      $bFeasible \leftarrow FALSE$
3      Initialize $\mathcal{NN}$ architecture of polynomial $B_\ell(\mathbf{x})$
4      Initialize learnable multipliers $\lambda_\ell(\mathbf{x})$ and $\gamma_e(\mathbf{x})$
5      Encode initial dataset $\mathcal{S} \leftarrow S_I$, $S_U$, $S_D$, $S_G$
6      $B_\ell(\mathbf{x})$← Learner($\mathcal{NN}, \lambda_\ell(\mathbf{x}), \gamma_e(\mathbf{x}), \mathcal{S}$)
7        // Training BC candidates
8      $bFeasible \leftarrow$ Verifier($B_\ell(\mathbf{x}), \mathbf{H}, \Xi(\ell)$)
9        // Identifying real BC
10     while $bFeasible = FALSE$ and not timeout do
11         Compute point $\mathbf{x}_c \leftarrow$ Cex Generator($B_\ell(\mathbf{x}), \mathbf{H}$)
12         Compute ellipsoid $\mathcal{E} \leftarrow$ Cex Generator($\mathbf{x}_c$)
13         Construct cex set $\mathcal{C} \leftarrow$ Sampling from $\mathcal{E}$
14           // Generating counterexamples
15         $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{C}$
16         $B_\ell(\mathbf{x}) \leftarrow$ Learner($B_\ell(\mathbf{x}), \lambda_\ell(\mathbf{x}), \gamma_e(\mathbf{x}), \mathcal{S}$)
17         $bFeasible \leftarrow$ Verifier($B_\ell(\mathbf{x}), \mathbf{H}, \Xi(\ell)$)
18     end
19     return real barrier certificate $B_\ell(\mathbf{x})$ of location $\ell$
20 end
21 return $\{B_\ell(\mathbf{x})\}$ of $L$ locations in $\mathbf{H}$
```

Proof 3: Consider a hybrid system $\mathbf{H}$ with locations ℓ, ℓ' and transitions e=(ℓ, ℓ'), e'=(ℓ', ℓ). The non-trivial quantifier elimination problem for BCs generation under conditions (i)-(iv) in Theorem 1 can be converted into an SDP problem by SOS relaxation method [3] with degree bounded polynomial products as (4) yield from Theorem 2. To tackle the resulting non-convex BMI problem in Theorem 3, in Line 6, we form $B_\ell(\mathbf{x})$ and $B_{\ell'}(\mathbf{x})$ as two polynomials by $\mathcal{NN}$ and trained their candidates with the polynomial multipliers $\lambda_\ell(\mathbf{x})$, $\lambda_{\ell'}(\mathbf{x})$ for condition (ii) and $\gamma_e(\mathbf{x})$, $\gamma_{e'}(\mathbf{x})$ over reset functions $\mathcal{R}_e(\mathbf{x})$ for condition (iii). Thus, the BMI problem is transformed as an LMI problem in (5) with the known $\{B_\ell(\mathbf{x})\}$ terms. In Line 8, we just need to solve the LMI feasibility of (6-9) to identify real BCs over the entire state-space and if the indicator $bFeasible$ is $TRUE$, then it is achieved. Otherwise, Lines 10 to 18 will repeat the counterexample-guided NN training with the Cex set constructed by (10-12), until the Line 17 is satisfied, and the Algorithm 1 will finally return real barrier certificates $\{B_\ell(\mathbf{x})\} = \{B_\ell(\mathbf{x}), B_{\ell'}(\mathbf{x})\}$ for location ℓ and ℓ'. $B_\ell(\mathbf{x})$ means that $\mathbf{H}$ is safe at initial location ℓ, and $B_{\ell'}(\mathbf{x})$ means after discrete transformations between ℓ and ℓ', $\mathbf{H}$ is still safe at location ℓ'. Therefore, it is established from Lines 8 and 17 that the algorithm is sound. □

V. Experiments

We have developed an automated tool named *SynHbc* for synthesizing polynomial neural barrier certificates based on Algorithm 1. In this section, we first demonstrate the application of *SynHbc* in cases of continous and hybrid systems, followed by ablation studies to evaluate our techniques. After that, we compare *SynHbc* with state-of-the-art neural BC synthesis tools and SOS-based BC generation tools. All experiments were conducted on a Windows 11 machine equipped with 32 GB RAM and an AMD Ryzen 9 7945HX CPU running at 2.5GHz. The source code and more experimental details (e.g. parameter settings, benchmarks) are available at https://github.com/blliu6/SynHbc.

A. Case studies

Example 1. Consider the following continuous system [20]:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -x_1 + x_2 - x_3 \\ -x_1(x_3+1) - x_2 \\ 0.76524x_1 - 4.7037x_3 \end{bmatrix}$$

with the state space $\Psi = \{\mathbf{x} \in \mathbb{R}^3 | -2 \le x_1, x_2, x_3 \le 2\}$. Our goal is to confirm that all trajectories starting from the initial set $\Theta = \{\mathbf{x} \in \mathbb{R}^3 \,|\, x_1^2 + x_2^2 + x_3^2 \le 1\}$, will never enter the unsafe region $\Xi = \{\mathbf{x} \in \mathbb{R}^3 | \|\mathbf{x} - \mathbf{x}_c\|_2 \le 0.5\}$, where $\mathbf{x}_c = [1.5, 1.5, 1.5]^T$.

SynHbc forms a degree-2 polynomial BC by a 3-10-1 structure $\mathcal{NN}$ with polynomial expansion mode in Fig. 2c, where the neurons of each layer is separated by "-". The initial training datasets S_I, S_U and S_D are constructed each with 500 samples to converge the loss function ($\tau = 0.01$) by random gradient descent. After first iteration, the learned BC candidate shown in Fig. 5 is as follows:

$$\begin{aligned} B'(\mathbf{x}) = &- 3.526x_1^2 - 3.746x_2x_1 - 2.392x_1x_3 + 1.959x_1 - 4.565x_2^2 \\ &- 2.797x_2x_3 + 0.310x_2 - 3.912x_3^2 + 0.551x_3 + 6.977. \end{aligned}$$

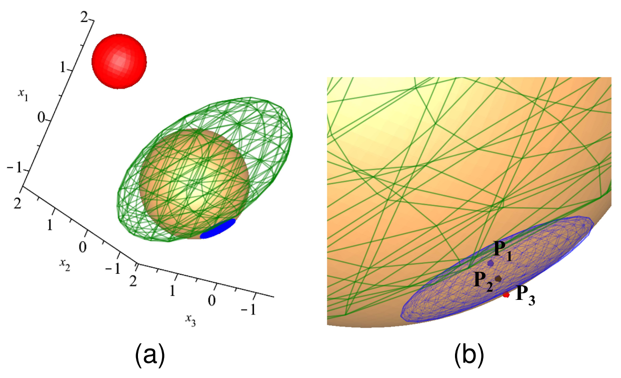


Fig. 5: Synthesis process of BC candidate for Example 1.

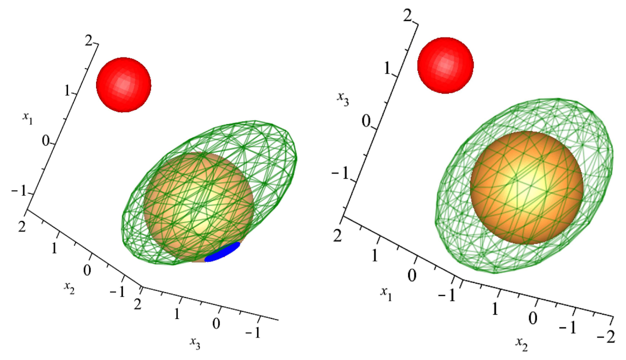
As shown in Fig. 5a, the zero level set of $B'(\mathbf{x})$ (the green spine surface) does not separate Θ (the yellow ball) from Ξ (the red ball), indicating the current candidate $B'(\mathbf{x})$ is not a real BC. Thus, we compute a minimum-volume ellipsoid $\mathcal{E}$ as shown in Fig. 5b to capture the Cexs of $B'(\mathbf{x})$. Specifically, we first locate the worst Cex $P_3 : (-0.628, -0.613, -0.480)$, the point $P_1 : (-0.555, -0.544, -0.426)$ on $B'(\mathbf{x})$ nearest to

P_3, and the Cex center $P_2 : (-0.591, -0.578, -0.453)$, then we sample points around a δ-ball centered at P_2 with radius $\delta = \gamma \cdot \frac{\|P_1 - P_3\|_2}{2}$ (scaling factor $\gamma = 6$) and filter real Cexs. Finally, we compute a minimum-volume ellipsoid $\mathcal{E}$ illustrated in Fig. 5 as $\mathcal{E} = \left\{\mathbf{x} \in \mathbb{R}^3 | \ \|A\mathbf{x} - b\|_2 \leq 1\right\}$, where

$$A = \begin{bmatrix} 6.580 & 3.723 & 2.769 \\ 3.723 & 6.528 & 2.693 \\ 2.769 & 2.693 & 4.956 \end{bmatrix}, \quad b = \begin{bmatrix} -7.038 \\ -6.991 \\ -5.267 \end{bmatrix},$$

$\mathbf{x} = [x_1, x_2, x_3]^T$. Next, we construct a Cex set $\mathcal{C}$ with 100 data points sampled from $\mathcal{E}$ to guide the further BC candidates retraining, ultimately obtaining a real BC after two iterations as shown in Fig. 6b, where

$$\begin{aligned} B(\mathbf{x}) = & -3.309x_1^2 - 1.737x_1x_2 + 0.012x_1x_3 + 1.616x_1 - 2.670x_2^2 \\ & - 2.055x_2x_3 - 0.861x_2 - 3.114x_3^2 - 0.676x_3 + 6.786. \end{aligned}$$



(a) (b)

Fig. 6: Phase portrait of the system in Example 1. Subfigure 6a describes the spurious BC candidate $B'(\mathbf{x})$ with the minimum-volume ellipsoid $\mathcal{E}$. Subfigure 6b describes the real BC $B(\mathbf{x})$.

Example 2. Consider following hybrid system [20] illustrated in Fig. 7, where

$$\mathbf{f}_1 = \begin{bmatrix} -x_1 + x_1x_2 \\ -x_2 \end{bmatrix}, \quad \mathbf{f}_2 = \begin{bmatrix} -x_1 + 2x_1^2x_2 \\ -x_2 \end{bmatrix}.$$

The hybrid system is initiated at ℓ_1 with $\Theta = \{\mathbf{x} \in \mathbb{R}^2 | (x_1 + 2)^2 + (x_2 - 2)^2 \leq 0.25\}$. Our goal is to verify the system will never enter the unsafe set $\Xi(\ell_2) = \{\mathbf{x} \in \mathbb{R}^2 | (x_1 - 2)^2 + (x_2 - 2)^2 \leq 0.25\}$ in location ℓ_2.

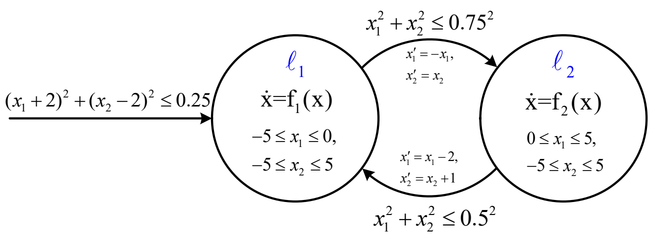
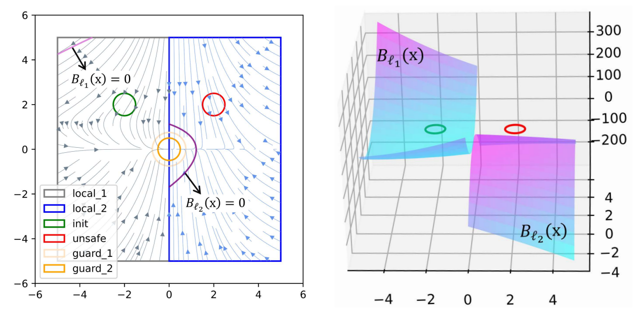


Fig. 7: The hybrid automata of the system in Example 2.

Let the polynomial $B_{\ell_1}(\mathbf{x})$ and $B_{\ell_2}(\mathbf{x})$ configured as two 2-10-1 $\mathcal{NN}$s with polynomial expansion mode in Fig. 2c. Applying *SynHbc*, we derive two polynomial neural BCs, both of degree 2, for the hybrid case after completing 3 iterations. The BCs results visually represented in Fig. 8 are as follows:

$$\begin{aligned} B_{\ell_1}(\mathbf{x}) = & \ 2.072x_1^2 - 5.567x_1x_2 - 1.401x_1 + 8.004x_2^2 \\ & - 5.241x_2 + 1.086, \\ B_{\ell_2}(\mathbf{x}) = & -0.647x_1^2 + 3.110x_1x_2 - 10.664x_1 - 7.298x_2^2 \\ & - 4.144x_2 + 13.916. \end{aligned}$$



(a) 2D (b) 3D

Fig. 8: Phase portrait of the hybrid system in Example 2.

B. Performance Evaluation of SynHbc

We conduct a series of ablation studies on representative examples with our tool *SynHbc*, to evaluate the performance of polynomial neural BC construction and the Cex generation.

1) Evaluation for the polynomial neural BC construction: We compare our method with non-polynomial *ReLU* neural BC synthesis based on SMT post-verification and the conventional polynomial *Square* NN learning with SOS-based verification. The experimental results are shown in Table I.

TABLE I: Performance Evaluation of NN Architecture

Ex.	$n_{\mathbf{x}}$	$d_{\mathbf{f}}$	*Ours*			*ReLU+SMT*			*Square+SOS*		
			I	T_l	T_v	I	T_l	T_v	I	T_l	T_v
H_9 [21]	3	3	3	13.709	3.335	×	×	×	×	×	×
C_3 [16]	2	2	2	2.287	0.306	1	0.631	0.003	×	×	×
C_8 [22]	3	2	2	1.293	0.566	2	0.066	0.232	2	3.146	1.366
C_{14} [16]	6	1	4	3.570	9.408	2	0.044	0.177	7	16.246	41.872
C_{15} [23]	6	2	3	4.226	8.420	5	1.883	6.097	3	16.818	20.231
C_{16} [24]	6	3	4	10.764	11.150	-	-	OT	3	18.827	17.866
C_{18} [16]	8	1	7	4.952	54.066	-	-	OT	×	×	×
C_{21} [25]	13	3	2	2.882	34.328	-	-	OT	2	5.237	68.179

In Table I, the number of the system variables is represented by $n_{\mathbf{x}}$; the maximal degree of the polynomials in the vector fields is denoted by $d_{\mathbf{f}}$; T_l denotes the time cost of NN training, T_v denotes the verification time; "OT" indicates out of time in 3600 seconds; $\times$ means 10 iterations failure.

As shown in Table I, while *ReLU* NN allows for faster convergence and SMT-based verification is quick for low-dimensional examples, the computational complexity of the SMT solver leads to timeout failures as the dimension increases. In contrast, the learned polynomial neural BCs, verified using the efficient SOS method, offer better scalability for high-dimensional examples. Additionally, our approach, leveraging various polynomial expansion modes, successfully handled more examples than the neural network using a single *Square* function activation.

2) Evaluation for the counterexamples generation method: We compare the performance of our minimize-volume ellipsoid $\mathcal{E}$ based Cex cet construction method with randomly sampling Cexs from a ball around worst Cex $\hat{\mathbf{x}}$. We also consider the influence of the Cexs number in the Cex set. The experimental results are shown in Table II.

TABLE II: Performance Evaluation of Cex Set Construction

Ex.	$n_{\mathbf{x}}$	$d_{\mathbf{f}}$	*Ours*						*Random*		
			1 cex			*100 cex*			*100 cex*		
			I	T_c	T_e	I	T_c	T_e	I	T_c	T_e
H_9	3	3	10	0.746	56.019	3	0.371	17.415	4	0.113	21.140
C_3	2	2	8	0.220	10.977	2	0.165	2.758	4	0.022	5.755
C_8	3	2	2	0.041	1.897	2	0.125	1.984	2	0.015	1.896
C_{14}	6	1	7	0.607	23.862	4	0.760	13.738	5	0.161	14.634
C_{15}	6	2	6	8.367	27.402	3	10.014	22.660	6	5.630	35.789
C_{16}	6	3	5	4.424	29.959	4	5.723	13.738	4	2.711	23.284
C_{18}	8	1	13	1.567	111.512	7	1.852	54.066	10	0.657	85.596
C_{21}	13	3	4	0.514	79.787	2	0.288	37.498	5	0.723	148.946

Table II shows that while our method may consume more time in finding Cexs compared to the random way due to some extra computation steps, it produces a higher-quality Cex set, as reflected in fewer iterations and a shorter overall synthesis time for the BC. Additionally, using a Cex set with multiple Cexs proves to be more efficient than relying solely on the worst-case Cex, which is consistent with our previous analysis that individual Cexs may offer limited information.

C. Comparison with Existing Approaches

Our *SynHbc* tool, implemented in Python, was subjected to a comparative evaluation with state-of-the-art CEGIS-based neural BC synthesis Python tools *FOSSIL* [5], [16] and *SynNBC* [8], which relies on the verification of δ-complete *dReal* solver (where δ=0.0001) and SOS-based *Verifier*, respectively. Additionally, to assess the performance of SOS relaxation method incorporated into our tool, we compared *SynHbc* against conventional SOS-based BMI problem solving (by tool *PENBMI* [26]) and SOS-based LMI problem solving (by tools *SOSTOOLS* [18] and *YALMIP* [19]) methods for BC generation. Among the experiment benchmarks, examples $H_1 \sim H_{10}$ are all hybrid systems, while $C_1 \sim C_{26}$ represent continuous systems. The performance is reported in Table III.

In Table III, the number of system variables is denoted by $n_{\mathbf{x}}$; the maximal degree of the polynomials in vector fields is denoted by $d_{\mathbf{f}}$, and the degree of BC is denoted by d_B. The time spent on BC learning and verification is recorded by T_l and T_v, respectively. And $T_c(S)$ in the *SynHbc* column means the time cost of Cex generation in Algorithm 1. For ease of comparison, we form the BC network architecture for *FOSSIL* and *SynNBC* as a unified one hidden layer NN with input neurons equal to $n_{\mathbf{x}}$ and output layer with a single neuron, and ϕ records the optimal NN activation function in *FOSSIL*. The total amount of time spent of each approach is denoted as $T_e(S)$, $T_e(F)$, $T_e(N)$, $T_e(P)$, or $T_e(L)$, where $T_e(S) = T_l(S)+T_c(S)+T_v(S)$ and $T_e(F) = T_l(F)+T_v(F)$. Moreover, I_s, I_f and I_n are the numbers of iterations. "OT" signifies a timeout after 3600 seconds. The symbol "$\times$" indicates the case cannot be handled, or that tools like *PENBMI*, *SOSTOOLS*, and *YALMIP* fail to provide a feasible solution within an empirical $\deg(B) \leq 6$.

Table III shows that for the 36 examples, our *SynHbc* successfully handle 34 of them including all 10 hybrid systems, while the number of successful examples for *FOSSIL*, *SynNBC*, *BMI* and *LMI* are 17, 13, 16 and 17, respectively. This effectiveness highlights *SynHbc* as a valuable complement to existing tools. In Table III, although BMI-based and LMI-based methods can cover 22 examples, *SynHbc* is necessary to solve the remaining 14 ones. We use *PENBMI* directly tackles non-convex BMI problems to produce BCs, and evaluate *SOSTOOLS*&*YALMIP* with polynomial multipliers of $\deg(B) \leq 2$ in 10 tests for adopting a more conservative LMI approach, where the best results are recorded under the "LMI" column. Within our *SynHbc* tool, the verification of learned BC candidates involves solving feasibility testing problems for the LMIs established by SOS relaxation. Remarkably, *SynHbc* effectively addresses 20 examples beyond the scope of *BMI* and 17 ones where *LMI* struggles. The aforementioned analysis serves as evidence that our tool significantly enhances the effectiveness of BC generation compared to existing numerical computation methodologies.

We assess the efficiency of *SynHbc* compared to a pair of approaches by examining the time required for neural barrier certificate synthesis. Among the 16 examples handled by both *SynHbc* and *FOSSIL*, *SynHbc* takes an average of 12.94 seconds to synthesize barrier certificates, whereas the best performance between the two *FOSSIL* versions *FOSSIL 1.0 [16]*&*FOSSIL 2.0 [5]* requires 93.18 seconds, making it approximately 7.20 times slower than *SynHbc*. For the 13 examples managed by both *SynHbc* and *SynNBC*, *SynHbc* requires an average of 56.01 seconds to synthesize barrier certificates, compared to 183.15 seconds for *SynNBC*, making *SynHbc* over three times faster. These empirical findings establish that *SynHbc* performs notably well in terms of efficiency.

VI. Related Work

Theoretically, the generation of barrier certificates presents a similarity to other problems involving the elimination of quantifiers. Approaches based on the SOS relaxation have gained traction due to their reasonably reduced computational complexity. They convert quantified constraints into non-convex bilinear matrix inequalities solved by SDP solvers such as *PENBMI* [26] instead of directly handling them. For semi-algebraic hybrid systems, Prajna et al. created barrier certificates [1], [3]. Using semi-definite programming, which is described by exponential conditions, Kong et al. [11] proposed a method to generate barrier certificates for semi-algebraic hybrid systems. To address its computational intractability, a convex surrogate has been proposed that performs effectively. Specifically, once the multipliers are fixed, the BMI problem is transformed into an LMI problem, which can be quickly solved using convex optimization techniques. This approach was first introduced by S. Prajna et al., and there are many toolboxes capable of solving such problems by formed SOS representations, like *SOSTOOLS* [18] and *YALMIP* [19].

In addition, combining NN learning and verification to synthesize barrier certificates is an emerging research direction.

TABLE III: Performance Evaluation on Benchmark Examples (time in seconds)

Ex.	$n_{\mathbf{x}}$	$d_{\mathbf{f}}$	*SynHbc*						*FOSSIL 1.0 [16]&FOSSIL 2.0 [5]*					*SynNBC [8]*		*BMI*		*LMI*	
			d_B	I_s	$T_l(S)$	$T_c(S)$	$T_v(S)$	$T_e(S)$	ϕ	I_f	$T_l(F)$	$T_v(F)$	$T_e(F)$	I_n	$T_e(N)$	d_B	$T_e(P)$	d_B	$T_e(L)$
H_1 [27]	2	1	2	2	5.777	0.020	0.892	6.689	×	×	×	×	×	×	×	2	0.953	2	3.025
H_2 [28]	2	2	2	2	7.506	0.020	1.088	8.614	×	×	×	×	×	×	×	×	×	×	×
H_3 [29]	2	2	2	2	1.636	0.021	1.049	2.706	×	×	×	×	×	×	×	2	0.571	2	0.249
H_4 [29]	2	2	2	2	7.905	0.019	1.504	8.978	×	×	×	×	×	×	×	×	×	×	×
H_5 [30]	2	2	2	2	1.798	0	2.399	4.197	×	×	×	×	×	×	×	2	0.592	×	×
H_6 [16]	2	3	2	1	0.620	0	0.422	1.042	$Square$	1	2.731	0.074	2.805	×	×	×	×	×	×
H_7 [20]	2	3	2	3	14.729	0.062	1.100	15.891	×	×	×	×	×	×	×	4	1.216	×	×
H_8 [20]	2	3	2	1	1.151	0	0.481	1.632	×	×	×	×	×	×	×	×	×	×	×
H_9 [21]	3	3	2	3	13.709	0.371	3.335	17.415	×	×	×	×	×	×	×	2	1.365	×	×
H_{10} [31]	3	5	2	3	35.146	0.135	8.419	43.700	–	–	–	–	OT	×	×	×	×	×	×
C_1 [32]	2	1	3	1	0.284	0	0.194	0.478	$Poly\_2$	1	0.018	0.002	0.020	×	×	×	×	2	0.050
C_2 [33]	2	2	1	1	0.236	0	0.147	0.383	$ReLU$	1	0.027	0.005	0.032	×	×	2	1.252	2	0.057
C_3 [16]	2	2	2	2	2.287	0.165	0.306	2.758	$Sigmoid$	3	3.067	1.329	4.396	2	2.277	2	0.372	2	0.061
C_4 [34]	2	3	3	1	0.694	0	0.242	0.936	$Square$	4	0.918	0.007	0.925	1	0.897	×	×	2	0.056
C_5 [16]	2	3	2	1	0.520	0	0.122	0.642	$Tanh$	1	0.606	0.003	0.609	1	0.670	4	0.875	2	0.059
C_6 [35]	2	5	2	1	0.343	0	0.352	0.695	$ReLU$	1	0.036	0.004	0.040	×	×	4	0.458	2	0.055
C_7 [36]	3	2	2	3	0.643	0.314	0.931	1.888	$Poly\_2$	1	0.033	0.041	0.074	×	×	×	×	2	0.121
C_8 [22]	3	2	2	2	1.293	0.125	0.566	1.984	$ReLU$	2	0.066	0.232	0.298	×	×	2	0.409	2	0.069
C_9 [37]	3	2	–	–	–	–	–	OT	–	–	–	–	OT	×	×	2	1.891	×	×
C_{10} [24]	3	3	2	1	0.496	0	0.335	0.831	$Sigmoid$	3	0.855	0.132	0.987	×	×	4	1.598	2	0.075
C_{11} [16]	4	1	2	1	0.556	0	0.411	0.967	$Square$	3	0.057	1.387	1.444	3	8.790	2	1.258	2	0.197
C_{12} [38]	4	2	2	1	1.272	0	0.959	2.231	$Tanh$	2	0.052	0.476	0.528	2	20.908	×	×	×	×
C_{13} [38]	4	2	–	–	–	–	–	OT	$Poly\_2$	2	0.028	0.251	0.279	×	×	2	5.102	×	×
C_{14} [16]	6	1	2	4	3.570	0.760	9.408	13.738	$Linear$	4	9.233	32.574	41.807	4	13.544	2	8.139	2	0.679
C_{15} [23]	6	2	2	3	4.226	10.014	8.420	22.660	–	–	–	–	OT	3	10.578	×	×	×	×
C_{16} [24]	6	3	2	4	10.764	5.273	11.150	27.187	–	–	–	–	OT	4	18.845	×	×	2	1.205
C_{17} [39]	7	2	2	1	1.029	0	2.900	3.929	–	–	–	–	OT	×	×	×	×	×	×
C_{18} [16]	8	1	2	7	4.952	1.852	54.066	60.870	$Linear$	4	3.380	27.021	30.401	4	49.638	2	82.807	2	3.857
C_{19} [39]	9	2	2	6	6.659	110.588	35.975	153.222	–	–	–	–	OT	×	×	–	OT	×	×
C_{20} [40]	12	1	2	12	83.527	1845.418	240.087	2169.032	–	–	–	–	OT	×	×	–	OT	×	×
C_{21} [25]	13	3	2	2	2.882	0.288	34.328	37.498	$Linear$	16	0.208	149.097	149.306	2	83.038	–	OT	2	11.924
C_{22} [25]	15	3	2	2	6.503	0.567	73.078	80.148	$Linear$	20	0.282	1256.956	1257.238	6	355.969	–	OT	2	27.530
C_{23} [25]	17	3	2	2	4.221	0.373	159.616	164.210	–	–	–	–	OT	7	649.260	–	OT	×	×
C_{24} [25]	19	3	2	2	5.093	1.108	307.966	314.167	–	–	–	–	OT	6	1166.529	–	OT	×	×
C_{25} [25]	21	3	2	5	4.930	3.994	1600.066	1608.990	–	–	–	–	OT	–	OT	–	OT	×	×
C_{26} [25]	23	3	2	3	9.033	5.562	1980.354	1994.949	–	–	–	–	OT	–	OT	–	OT	×	×

Zhao et al. first proposed pioneering work to synthesize barrier certificates using neural networks for continuous dynamical systems [4]. Peruffo et al. utilized a Counterexample-Guided Inductive Synthesis (CEGIS) [6] program with *dReal* [7] SMT solver to generate neural barrier certificates by the developed *FOSSIL* tool [5], [16]. Although the aforementioned methods utilizing conventional networks (e.g. ReLU NN) can fit BC candidates with good expressiveness, they rely on SMT solvers for post-verification, which significantly limits the scale of verifiable systems. To improve the scalability of neural BC synthesis, Zhao et al. proposed a novel CEGIS-based method that trains BC directly applied for more efficient SOS verification [8], which greatly increases the dimension of synthesizable neural BC, but only handle continuous systems.

For the neural BC synthesis of hybrid system, the *FOSSIL* tool is the first to attempt synthesizing neural BC for hybrid system based on CEGIS framework, but only works with simple state-dependent hybrid systems defined by continuous variable ranges and lacking complex discrete event logic, producing a single unified barrier certificate. Inspired by the work, we propose a novel approach for synthesizing polynomial neural BC for each location of more general hybrid systems defined by hybrid automata with guard conditions and reset functions. We preserved the efficiency derived from counterexample guidance and introduced polynomial expansions from polynomial NNs to augment the expressive capabilities of learned neural polynomial BCs, which can be directly applied to effective SOS-based verification, thereby avoiding the non-trivial verification of the non-polynomial neural BCs. Furthermore, we substitute the *dReal*-based *Verifier* with an SOS-based LMIs feasibility testing verification approach and construct a minimum-volume ellipsoid based counterexample set by solving polynomial optimization problem for capturing more real counterexamples to equip our method for addressing instances with high dimensions.

VII. Conclusion

In this paper, we have proposed a novel counterexample-guided approach to synthesize polynomial neural BCs for the general hybrid systems consist of continuous systems and discrete transitions with reset function and guard conditions. We built an iterative framework using inductive loop made up of *Learner*, *Verifier*, and *Cex Generator*. The *Learner* generated potential polynomial-form BC candidates by numerically training NNs with the introduced polynomial expansion modes. The *Verifier* then assessed the validity of these learned candidates through LMI feasibility testing for the sub-programming LMIs formed through SOS relaxation and identified real BCs. When formal verification failed, the *Learner* retrained and updated the candidates guided by the

Cex set constructed from a minimum-volume ellipsoid, which was computed via polynomial optimization in the *Cex Generator*. The experimental results have demonstrated that our approach is more effective and practical compared to existing state-of-the-art CEGIS-based neural BC synthesis methods and SOS-based BC generation techniques.

Considering the limitations of polynomial form, in future work, we will also explore integrating our approach with non-polynomial dynamical systems and improve our method to achieve a balance between time efficiency and problem scale. Meanwhile, to improve the accuracy of SOS verification, we will consider incorporating moment-SOS hierarchy [41] and more rigorous theorem provers such as *Coq* [42], [43] into the BC generation process for the safety verification.

REFERENCES

[1] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Proceedings of the HSCC*, 2004, pp. 477–492.

[2] H. Sibai and S. Mitra, "State estimation of dynamical systems with unknown inputs: Entropy and bit rates," in *Proceedings of the 21st HSCC*, 2018, p. 217–226.

[3] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE TAC*, vol. 52, no. 8, pp. 1415–1428, 2007.

[4] H. Zhao, X. Zeng, T. Chen, and Z. Liu, "Synthesizing barrier certificates using neural networks," in *Proceedings of the HSCC*, 2020, pp. 1–11.

[5] A. Edwards, A. Peruffo, and A. Abate, "Fossil 2.0: Formal certificate synthesis for the verification and control of dynamical models," *arXiv preprint arXiv:2311.09793*, 2023.

[6] A. Solar-Lezama, L. Tancau, R. Bodik, S. Seshia, and V. Saraswat, "Combinatorial sketching for finite programs," in *Proceedings of the 12th ASPLOS*, 2006, pp. 404–415.

[7] S. Gao, S. Kong, and E. M. Clarke, "dreal: An smt solver for nonlinear theories over the reals," in *Automated Deduction–CADE-24*, 2013, pp. 208–214.

[8] H. Zhao, N. Qi, L. Dehbi, X. Zeng, and Z. Yang, "Formal synthesis of neural barrier certificates for continuous systems via counterexample guided learning," *ACM TECS*, vol. 22, no. 5s, pp. 1–21, 2023.

[9] S.-K. Oh, W. Pedrycz, and B.-J. Park, "Polynomial neural networks architecture: analysis and design," *Computers & Electrical Engineering*, vol. 29, no. 6, pp. 703–725, 2003.

[10] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical computer science*, vol. 138, no. 1, pp. 3–34, 1995.

[11] H. Kong, F. He, X. Song, W. N. Hung, and M. Gu, "Exponential-condition-based barrier certificate generation for safety verification of hybrid systems," in *Proceedings of CAV*, 2013, pp. 242–257.

[12] M. Putinar, "Positive polynomials on compact semi-algebraic sets," *Indiana University Mathematics Journal*, vol. 42, no. 3, pp. 969–984, 1993.

[13] G. G. Chrysos, S. Moschoglou, G. Bouritsas, J. Deng, Y. Panagakis, and S. Zafeiriou, "Deep polynomial neural networks," *IEEE TPAMI*, vol. 44, no. 8, pp. 4021–4034, 2021.

[14] T. Schneidereit and M. Breuß, "Polynomial neural forms using feedforward neural networks for solving differential equations," in *Proceedings of the 20th ICAISC*. Springer, 2021, pp. 236–245.

[15] Y. Wu, Z. Zhu, F. Liu, G. Chrysos, and V. Cevher, "Extrapolation and spectral bias of neural nets with hadamard product: a polynomial net study," *Advances in NeurIPS*, vol. 35, pp. 26 980–26 993, 2022.

[16] 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 HSCC*, 2021, pp. 1–11.

[17] E. D. Andersen and K. D. Andersen, "The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm," in *High performance optimization*. Springer, 2000, pp. 197–232.

[18] S. Prajna, "Sostools: Sum of squares optimization toolbox for matlab," *http://www. mit. edu/˜ parrilo/sostools/index. html*, 2004.

[19] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *Proceedings of IEEE ICRA*, 2004, pp. 284–289.

[20] X. Zeng, W. Lin, Z. Yang, X. Chen, and L. Wang, "Darboux-type barrier certificates for safety verification of nonlinear hybrid systems," in *Proceedings of the 13th EMSOFT*, 2016, pp. 1–10.

[21] Z. Yang, C. Huang, X. Chen, W. Lin, and Z. Liu, "A linear programming relaxation based approach for generating barrier certificates of hybrid systems," in *Proceedings of the 21st FM*. Springer, 2016, pp. 721–738.

[22] M. A. Ben Sassi and S. Sankaranarayanan, "Stability and stabilization of polynomial dynamical systems using bernstein polynomials," in *Proceedings of the 18th HSCC*, 2015, pp. 291–292.

[23] J. Llibre and C. Valls, "On the integrability of the einstein–yang–mills equations," *Journal of mathematical analysis and applications*, vol. 336, no. 2, pp. 1203–1230, 2007.

[24] 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, 2010.

[25] S. Ratschan, "Simulation based computation of certificates for safety of dynamical systems," in *Proceedings of the 15th FORMATS 2017*. Springer, 2017, pp. 303–317.

[26] M. Kocvara, M. Stingl, and P. GbR, "Penbmi user's guide (version 2.0)," *software manual, PENOPT GbR, Hauptstrasse A*, vol. 31, p. 91338, 2005.

[27] G. Wang, J. Liu, H. Sun, J. Liu, Z. Ding, and M. Zhang, "Safety verification of state/time-driven hybrid systems using barrier certificates," in *Proceedings of the 35th CCC*, 2016, pp. 2483–2489.

[28] C. Huang, X. Chen, W. Lin, Z. Yang, and X. Li, "Probabilistic safety verification of stochastic hybrid systems using barrier certificates," *ACM TECS*, vol. 16, no. 5s, pp. 1–19, 2017.

[29] Z. Yang, M. Wu, and W. Lin, "An efficient framework for barrier certificate generation of uncertain nonlinear hybrid systems," *Nonlinear Analysis: Hybrid Systems*, vol. 36, p. 100837, 2020.

[30] S. Ratschan and Z. She, "Safety verification of hybrid systems by constraint propagation-based abstraction refinement," *ACM TECS*, vol. 6, no. 1, pp. 8–es, 2007.

[31] Z. She and B. Xue, "Discovering multiple lyapunov functions for switched hybrid systems," *SIAM Journal on Control and Optimization*, vol. 52, no. 5, pp. 3312–3340, 2014.

[32] Q. Wang, M. Chen, B. Xue, N. Zhan, and J.-P. Katoen, "Encoding inductive invariants as barrier certificates: Synthesis via difference-of-convex programming," *Information and Computation*, vol. 289, p. 104965, 2022.

[33] J. Liu, N. Zhan, and H. Zhao, "Computing semi-algebraic invariants for polynomial dynamical systems," in *Proceedings of the ninth EMSOFT*, 2011, pp. 97–106.

[34] R. Rebiha, A. V. Moura, and N. Matringe, "Generating invariants for non-linear hybrid systems," *Theoretical Computer Science*, vol. 594, pp. 180–200, 2015.

[35] A. Sogokon, K. Ghorbal, and T. T. Johnson, "Non-linear continuous systems for safety verification (benchmark proposal)," in *Proceedings of the 3rd ARCH@ CPSWeek*, vol. 43. EasyChair, 2016, pp. 42–51.

[36] A. Djaballah, A. Chapoutot, M. Kieffer, and O. Bouissou, "Construction of parametric barrier functions for dynamical systems using interval analysis," *Automatica*, vol. 78, pp. 287–296, 2017.

[37] 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 *ACC*, 2014, pp. 3571–3578.

[38] A. Ferragut and A. Gasull, "Seeking darboux polynomials," *Acta Applicandae Mathematicae*, vol. 139, no. 1, pp. 167–186, 2015.

[39] E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach, *Systems biology in practice: concepts, implementation and application*. John Wiley & Sons, 2005.

[40] S. Gao, J. Kapinski, J. Deshmukh, N. Roohi, A. Solar-Lezama, N. Aréchiga, and S. Kong, "Numerically-robust inductive proof rules for continuous dynamical systems," in *Proceedings of the 31st CAV*. Springer, 2019, pp. 137–154.

[41] J. Wang, V. Magron, and J.-B. Lasserre, "Tssos: A moment-sos hierarchy that exploits term sparsity," *SIAM Journal on optimization*, vol. 31, no. 1, pp. 30–58, 2021.

[42] P. Roux, M. Iguernlala, and S. Conchon, "A non-linear arithmetic procedure for control-command software verification," in *Proceedings of the TACAS*, 2018, pp. 132–151.

[43] É. Martin-Dorel, G. Melquiond, and P. Roux, "Enabling floating-point arithmetic in the coq proof assistant," *Journal of Automated Reasoning*, vol. 67, no. 4, p. 33, 2023.