An Iterative Scheme of Safe Reinforcement Learning for Nonlinear Systems via Barrier Certificate Generation

Zhengfeng Yang[1], Yidan Zhang[1], Wang Lin[2](✉), Xia Zeng[3], Xiaochao Tang[1], Zhenbing Zeng[4], and Zhiming Liu[3,5]

[1] Shanghai Key Lab of Trustworthy Computing, East China Normal University, Shanghai, China
zfyang@sei.ecnu.edu.cn,{ydzhang,xctang}@stu.ecnu.edu.cn

[2] School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou, China
linwang@zstu.edu.cn

[3] School of Computer and Information Science, Southwest University, Chongqing, China
xzeng0712@swu.edu.cn

[4] Department of Mathematics, Shanghai University, Shanghai, China
zbzeng@shu.edu.cn

[5] Centre for Intelligent and Embedded Software, Northwestern Polytechnical University, Suzhou, China
zliu@nwpu.edu.cn

Abstract. In this paper, we propose a safe reinforcement learning approach to synthesize deep neural network (DNN) controllers for nonlinear systems subject to safety constraints. The proposed approach employs an iterative scheme where a *learner* and a *verifier* interact to synthesize safe DNN controllers. The *learner* trains a DNN controller via deep reinforcement learning, and the *verifier* certifies the learned controller through computing a maximal safe initial region and its corresponding barrier certificate, based on polynomial abstraction and bilinear matrix inequalities solving. Compared with the existing verification-in-the-loop synthesis methods, our iterative framework is a sequential synthesis scheme of controllers and barrier certificates, which can learn safe controllers with adaptive barrier certificates rather than user-defined ones. We implement the tool SRLBC and evaluate its performance over a set of benchmark examples. The experimental results demonstrate that our approach efficiently synthesizes safe DNN controllers even for a nonlinear system with dimension up to 12.

This work was partially supported by the Scientific and Technological Innovation 2030 Major Projects under Grant 2018AAA0100902, the National Natural Science Foundation of China under Grant 61772203, 61902325, 62032019, 61732019, the Zhejiang Provincial Natural Science Foundation of China under Grant LY20F020020, the Capacity Development Grant of Southwest University under Grant SWU116007, the Fundamental Research Funds for the Central Universities under Grant SWU117058.

A. Silva and K. R. M. Leino (Eds.) CAV 2021, LNCS 12759, pp. 467–490, 2021.
https://doi.org/10.1007/978-3-030-81685-8_22

Keywords: Formal verification · Safe reinforcement learning · Barrier certificates · Continuous dynamical systems

1 Introduction

The design and synthesis of controllers for dynamical systems is a fundamental problem in the field of control. In recent years, with the boom of deep learning, there has been considerable research activities in the use of deep neural networks (DNNs) for control of cyber-physical systems such as unmanned aerial vehicles, self-driving cars, etc. [33]. For these safety-critical systems, one of the most important and challenging problems is safe controller synthesis, that is, to synthesize a controller guaranteeing that the system's trajectory will never intersect with an undesired region.

A number of techniques included under the umbrella of Deep Reinforcement Learning (DRL) have been used to effectively learn controllers from user-defined reward functions encoding desired system behavior [17,36]. A majority of these works lack formal reasoning about the safety of such DNN-controlled dynamical systems from such learning process. To guarantee the safety property of synthesized DNN controllers, considerable works focus on the safety verification of DNN-controlled closed-loop systems, which is a really hard problem because it is tangled with highly nonlinear DNN expressions. The main research on this topic is through reachable set estimation of DNN-controlled systems, which can only deal with time bounded safety property [11,12,18,19,37]. On the other hand, other than formally verifying synthesized DNN controllers, more recent works have been proposed to learn DNN controllers for dynamical systems with safety guarantees [8,39,40]. For example, a verification-in-the-loop DNN controller training algorithm is presented in [8], which integrates RL framework with user-provided control barrier functions (CBFs) for reward function encoding, combined with SMT based formal CBF checking; a correctness-by-design method is proposed in [39] that first learns DNN controllers and barrier certificates simultaneously using supervised learning, and then performs posterior formal verification of barrier certificates via SMT solvers.

In this paper, we propose a safe reinforcement learning approach to synthesize DNN controller for nonlinear systems subject to safety constraints via barrier certificate generation. The proposed approach employs an iterative scheme where a *learner* and a *verifier* interact to synthesize safe DNN controllers. Firstly, the *learner* applies DRL method to train a DNN controller by encoding the safety requirement (and the barrier certificate requirement, if applicable) into reward function. For the learned controller, the *verifier* computes a Maximal Safe Input Region (MSIR) and the corresponding barrier certificate. Once the MSIR is a superset of the prescribed initial set Θ, it is easy to see that the safety of the closed-loop system under the learned controller with Θ is verified. Otherwise, the computed barrier certificate needs to be adjusted and fed to guide the *learner* to retrain a new controller. The above inductive loop repeats until an MSIR enclosing Θ is computed.

Compared with [8], a user-provided barrier certificate is adopted for reward function encoding and the barrier certificate is fixed through the learning process, whereas in this paper the controllers and the barrier certificates are synthesized simultaneously and yielded in a larger state space, which increases the diversity and flexibility of barrier certificates. Meanwhile, the barrier certificates in our approach are computed by numerical optimization method, which is more efficient than the SMT based method in [8]. Compared with [39], our method is based on RL framework and thus has better data sampling efficiency than the meshing-based data set generation in [39] for supervised learning. Besides, our method is iterative so that can utilize intermediate learned results to guide learning in the next iteration, rather than restarting from scratch as in [39] when a learned barrier certificate failed formal checking. Thanks to these advantages, our method has really good performance in efficiency and scalability even for problems with dimension up to 12.

The main contributions of this paper are summarized as follows:

– We propose a safe reinforcement learning via barrier certificate generation to synthesize DNN controller, which can guarantee the unbounded-time safety of the closed-loop systems.
– Our synthesis approach employs a sequential iterative scheme, where DNN controllers and the corresponding barrier certificates are synthesized alternatively, and in each iteration, barrier certificates are slightly adjusted to guide retraining safe DNN controllers quickly.
– We provide a detailed experimental evaluation on a set of benchmarks, which shows the efficiency and effectiveness of our approach.

The paper is organized as follows. Section 2 gives a brief introduction to the safe controller synthesis problem. Section 3 describes an iterative scheme of safe reinforcement learning for safe DNN controller synthesis. In Sect. 4, we provide an overall algorithm with a detailed example attached to depict how the algorithm works. In Sect. 5, we present an experimental evaluation of our algorithm over a set of benchmark examples. We compare with related works in Sect. 6 before concluding in Sect. 7.

2 Preliminaries

Notations. Let $\mathbb{R}$ and $\mathbb{N}$ be the field of real number and natural number, respectively. $\mathbb{R}[\mathbf{x}]$ denotes the ring of polynomials with coefficients in $\mathbb{R}$ over variables $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, and $\mathbb{R}[\mathbf{x}]^n$ denotes the n-dimensional polynomial ring vector. Let $R[\mathbf{x}]_d \subset \mathbb{R}[\mathbf{x}]$ be the vector space of polynomials of degree at most d. Let $\mathbb{N}_d^n := \{\alpha \in \mathbb{N}^n : \sum_i \alpha_i \leq d\}$. Denote by $\Sigma[\mathbf{x}] \subset \mathbb{R}[\mathbf{x}]$ (resp. $\Sigma[\mathbf{x}]_d \subset \mathbb{R}[\mathbf{x}]_{2d}$) the space of sums of squares (SOS) polynomials.

Consider a continuous dynamical system of the form

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

where $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ and $\mathbf{f} = (f_1, \ldots, f_n)^T \in \mathbb{R}[\mathbf{x}]^n$ is the vector field defined on the state space $D \subset \mathbb{R}^n$. We assume that $\mathbf{f}$ satisfies the local Lipschitz condition, so that (1) has a unique solution $\mathbf{x}(t, \mathbf{x}_0)$ in D for every initial state $\mathbf{x}_0 \in D$ at time $t = 0$.

In many contexts, a dynamical system is equipped with a domain $\Psi \subset D$ and an initial set $\Theta \subset \Psi$, represented as a triple $\mathcal{C} \doteq (\mathbf{f}, \Theta, \Psi)$. Given a prespecified unsafe region $X_u \subset D$, we say that the system $\mathcal{C}$ is *safe* if all system trajectories starting from Θ can not evolve into any state specified by X_u, which has been widely investigated in safety critical applications.

Definition 1 (Safety). *For a constrained continuous dynamical system (CCDS) $\mathcal{C} = (\mathbf{f}, \Psi, \Theta)$ and a given unsafe region X_u, the system is safe if for all $\mathbf{x}_0 \in \Theta$, there does not exist $t_1 > 0$ such that*

$$\forall t \in [0, t_1].\mathbf{x}(t, \mathbf{x}_0) \in \Psi \quad \text{and } \mathbf{x}(t_1, \mathbf{x}_0) \in X_u,$$

that is, the system's trajectory never reaches X_u from Θ as long as it remains in Ψ.

Remark 1. If the trajectory $\mathbf{x}(t, \mathbf{x}_0)$ first leaves Ψ and then enters Ψ again, then by Definition 1, the part of the trajectory from the first exit point is excluded from our concern and is not relevant to the safety of the considered CCDS.

In this paper, we consider a *controlled CCDS* $\mathcal{C} = (\mathbf{f}, \Psi, \Theta)$ with continuous dynamics defined by

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{u} = \mathbf{k}(\mathbf{x}), \end{cases} \tag{2}$$

where $\mathbf{x} \in \Psi \subseteq \mathbb{R}^n$ are the system states, $\mathbf{u} \in U \subseteq \mathbb{R}^m$ are the control inputs, and $\mathbf{f} : \Psi \times U \to \mathbb{R}^n$ and $\mathbf{k} : \Psi \to U$ are the locally Lipschitz continuous vector field and feedback controller function, respectively. The problem we considered in this paper is defined as follows.

Definition 2 (Safe Controller Synthesis). *For a controlled CCDS $\mathcal{C} = (\mathbf{f}, \Psi, \Theta)$ with $\mathbf{f}$ defined by (2) and a given unsafe region X_u, design a locally Lipschitz continuous feedback control law $\mathbf{k}$ such that the closed-loop system $\mathcal{C}$ with $\mathbf{f} = \mathbf{f}(\mathbf{x}, \mathbf{k}(\mathbf{x}))$ is safe as per Definition 1.*

The concept of *barrier certificates* plays an important role in safety verification of continuous systems. The essential idea is to use the zero level set of a barrier certificate $B(\mathbf{x})$ as a barrier to separate all the reachable states from the unsafe region. The following theorem states the conditions that must be satisfied by a barrier certificate.

Theorem 1 *[26]. Given a continuous system $\mathcal{C} = (\mathbf{f}, \Psi, \Theta)$, and the unsafe region X_u. Suppose there exists a real-valued function $B : \Psi \to \mathbb{R}$ satisfying the following conditions:*

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

(ii) $B(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in X_u$,
(iii) $B(\mathbf{x}) = 0 \Rightarrow \mathcal{L}_f B(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in \Psi$,

where $\mathcal{L}_f B(\mathbf{x})$ *denotes the Lie-derivative of* $B(\mathbf{x})$ *along the vector field* $\mathbf{f}(\mathbf{x})$*, i.e.,* $\mathcal{L}_f B(\mathbf{x}) = \sum_{i=1}^{n} \frac{\partial B}{\partial x_i} \cdot f_i(\mathbf{x})$*, then* $B(\mathbf{x})$ *is a barrier certificate, and the safety of system* $\mathcal{C}$ *is guaranteed.*

Corollary 1. *For a controlled CCDS* $\mathcal{C} = (\mathbf{f}, \Psi, \Theta)$ *with* $\mathbf{f}$ *defined by (2), a feedback control law* $u = \mathbf{k}(\mathbf{x})$ *can be used to ensure the safety control of* $\mathcal{C}$*, if there exists a barrier certificate for the closed-loop system under the control law* $\mathbf{k}(\mathbf{x})$*.*

Throughout this paper, we assume that the initial set Θ, the domain Ψ and the unsafe set X_u are compact semi-algebraic sets, defined by polynomial equations and inequalities. Concretely, the semi-algebraic sets Θ, Ψ and X_u are represented as follows:

$$\begin{cases} \Theta : = \{\mathbf{x} \in \mathbb{R}^n \mid g_i(\mathbf{x}) \geq 0, i = 1, \ldots, m_1\}, \\ \Psi : = \{\mathbf{x} \in \mathbb{R}^n \mid h_j(\mathbf{x}) \geq 0, j = 1, \ldots, m_2\}, \\ X_u : = \{\mathbf{x} \in \mathbb{R}^n \mid q_k(\mathbf{x}) \geq 0, k = 1, \ldots, m_3\}, \end{cases}$$

for some polynomials $g_i, h_j, q_k \in \mathbb{R}[\mathbf{x}]$.

3 Synthesis of Safe Controller via Learning and Verification

In this section, we introduce an iterative framework for synthesizing a deep neural network (DNN) controller for a CCDS subject to safety constraints. As shown in Fig. 1, the procedure is structured as an inductive loop between a *learner* and a *verifier*. The *learner* trains a DNN controller using reinforcement learning. The trained DNN controller is passed to the *verifier*, which checks the safety of the closed-loop system under the trained controller via barrier certificate generation.

Observing Fig. 1, we first apply the reinforcement learning method to train a neural network controller $u = k(\mathbf{x})$ in terms of the target of the safety satisfiability, and then try to yield a barrier certificate $B(\mathbf{x})$ based on the bilinear matrix inequalities (BMI) solving, to guarantee the safety of the closed-loop system with the controller $k(\mathbf{x})$.

However, for the system with the controller $k(\mathbf{x})$, such barrier certificate $B(\mathbf{x})$ may not exist. The reasons are twofold: (i) the controller $k(\mathbf{x})$ is trained through the trajectories starting from finite points in the initial set Θ; (ii) the existence of the barrier certificate is just a sufficient condition of the safety of the given system.

In this situation, for the learned controller $k(\mathbf{x})$, one may compute a Maximal Safe Input Region (MSIR) Θ_γ and the corresponding barrier certificate $B(\mathbf{x})$, which can guarantee the safety of the continuous system with respect to the initial set Θ_γ. Once Θ_γ is a superset of the prescribed initial set Θ, i.e., $\Theta \subseteq \Theta_\gamma$,

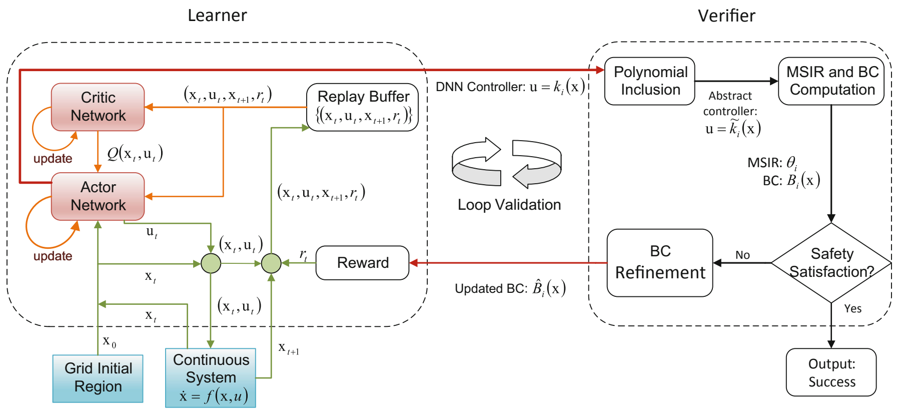

Fig. 1. The framework of safe neural network controller synthesis.

it is easy to see that the safety of the system with Θ is verified. Otherwise, we need adjust the barrier certificate $B(\mathbf{x})$ and the controller $k(\mathbf{x})$ sequentially. This operation is able to build an iterative framework, wherein each iteration proceeds in two stages:

– **Update the neural network controller.** We apply deep reinforcement learning method to obtain the updated controller $k_i(\mathbf{x})$ by feeding $\hat{B}_{i-1}(\mathbf{x})$, which is the barrier certificate yielded from the above iteration (See the *learner* in Fig. 1).
– **Compute the barrier certificate with the maximal safe input region.** With the updated controller $k_i(\mathbf{x})$, we transfer the problem of barrier certificate generation into a bilinear matrix inequalities (BMI) solving, and then compute the maximal region Θ_i with the corresponding barrier certificate $B_i(\mathbf{x})$. Namely, the existence of $B_i(\mathbf{x})$ suffices to prove the safety of the system with respect to the initial set Θ_i. Once Θ_i encloses the original initial set Θ, i.e., $\Theta \subseteq \Theta_i$, the current controller $k_i(\mathbf{x})$ is the desired safe one. Otherwise, we need refine $B_i(\mathbf{x})$, and then go to the next iteration (See the *verifier* in Fig. 1).

3.1 Training of Safe Controller

In the following, we focus on the *learner* component of Fig. 1 and show how to train a safe controller using deep deterministic policy gradient (DDPG) [23], which is a popular reinforcement learning approach suited for continuous control applications. The DDPG combines the value-based and policy-based method, and is made up of two parts: actor and critic. The critic uses the off-policy data to learn the action-value function, which evaluates how good the action k taken is in the given state $\mathbf{x}$. The actor can learn the continuous action policy by using the action-value function. In practice, it is difficult to obtain the exact action-value function and policy function. Thus, two deep neural networks are introduced to solve this problem, i.e. the critic network $Q(\mathbf{x}, \mathbf{u}|\beta^Q)$ and actor network $k(\mathbf{x}|\beta^k)$ with weights β^Q and β^k, respectively.

The reward function should be appropriately designed to achieve the goal of safety controller synthesis via reinforcement learning. For safe controller synthesis, the task is to synthesize a DNN controller such that all the trajectories of the closed-loop system starting from Θ can not evolve into the unsafe region X_u. Thus, the reward function is preliminarily defined as

$$\hat{r}_t = \beta_1 \cdot \mathrm{dist}(X_u, \mathbf{x}_t)$$

where $\beta_1 > 0$ is the scale factor, and $\mathrm{dist}(X_u, \mathbf{x}_t)$ denotes the distance between the state $\mathbf{x}_t$ and the unsafe region X_u. In addition, according to the third condition of Theorem 1, once the trajectory hit the zero level set of barrier certificate it must satisfy $\mathcal{L}_f B(\mathbf{x}_t) > 0$; otherwise, the system behavior should be penalized. For this purpose, the reward function is updated as

$$r_t = \begin{cases} \hat{r}_t - \min(\beta_2 |\mathcal{L}_f B(\mathbf{x}_t)|, \Delta r_{\min}), & |B(\mathbf{x}_t)| < \delta \text{ and } \mathcal{L}_f B(\mathbf{x}_t) \leq 0 \\ \hat{r}_t, & \text{otherwise} \end{cases} \tag{3}$$

where $\mathcal{L}_f B(\mathbf{x}_t) = \sum_{i=1}^{n} \frac{\partial B(\mathbf{x}_t)}{\partial x_i} f_i(\mathbf{x}_t, u)$, $\beta_2 > 0$ is the scale factor, δ is a small positive value characterizing the zero-level set of B, and $\Delta r_{\min} > 0$ is the threshold avoiding too large fluctuations of reward value. In this work, we set $\beta_1 = 1.0$, $\beta_2 = 1.0$, $\delta = 0.1$, $\Delta r_{\min}$ denotes the size of Ψ. Since $0 \leq \hat{r}_t \leq \Delta r_{\min}$, the setting r_t (3) can be kept within a certain range, making the convergence effect better.

Algorithm 1. Barrier Certificate Guided Reinforcement Learning

```
Input: CCDS $\mathcal{C}$; unsafe region $X_u$; barrier certificate $B(\mathbf{x})$
Output: DNN Controller $k$
 1: Initialize critic $Q$ and actor $k$, corresponding target networks $Q' = Q$ and $k' = k$
 2: Initialize barrier certificate $B(\mathbf{x}) = \perp$ and replay buffer $R = \emptyset$
 3: Sample initial states from $\Theta$ and store them to $\Omega_\Theta$
 4: for $\mathbf{x}_0 \in \Omega_\Theta$ do
 5:     for $t = 1, \cdots, T$ do
 6:         calculate $\mathbf{u}_t = k(\mathbf{x}_t)$
 7:         calculate $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$
 8:         calculate $r_t = r(\mathbf{x}_{t+1}, X_u, B(\mathbf{x}))$
 9:         store $(\mathbf{x}_t, \mathbf{x}_{t+1}, u_t, r_t)$ to $R$
10:         Sample random minibatch of transitions from $R$
11:         Update critic $Q$ and actor $k$
12:     end for
13:     Update the target networks $Q'$ and $k'$
14: end for
15: return $k$
```

To synthesize the safety controller using reinforcement learning, a dataset of sampled trajectories is needed. To sample trajectories, we first generate a set of initial states from Θ. Let $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ be the vectors of the lower and upper bounds of Θ, i.e., $\Theta \subseteq [\mathbf{l}, \mathbf{u}]$. We first sample from each dimension of $[\mathbf{l}, \mathbf{u}]$ equidistantly with a fixed mesh size. For a sampled initial state $\mathbf{x}_0$, its trajectory is generated, and the transition tuples $(\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{u}_t, r_t)$ are collected to form a replay buffer to update the action and critic networks. Concretely, the action network receives a state $\mathbf{x}_t$ in time step t as input, and directly outputs a continuous action $\mathbf{u}_t = k(\mathbf{x}_t|\beta^k)$. The critic network takes the state $\mathbf{x}_t$ and the action $\mathbf{u}_t$ as input, and outputs a scalar Q-value $Q(\mathbf{x}_t, \mathbf{u}_t|\beta^Q)$. For every m simulated time steps, we sample a batch of tuples from the buffer as the training data to update the actor and critic networks, until a certain prescribed termination condition is met for the learning process. The resulting actor network is the synthesized controller. All training related parameters, such as smoothing factor, are set as default. Our DDPG implementation is based on an open-source package DDPG [23]. The algorithm is outlined in Algorithm 1.

Remark 2. The barrier certificate is initialized to be $\perp$, which means that the *learner* initially trains a DNN controller via standard reinforcement learning, without the aid of barrier certificates.

3.2 Safety Verification with Barrier Certificates

In the following, we focus on the *verifier* component of the proposed safe DNN synthesis framework, as described in Fig. 2, and show how to verify the safety of the closed-loop system under the DNN controller yielded from the *learner*.

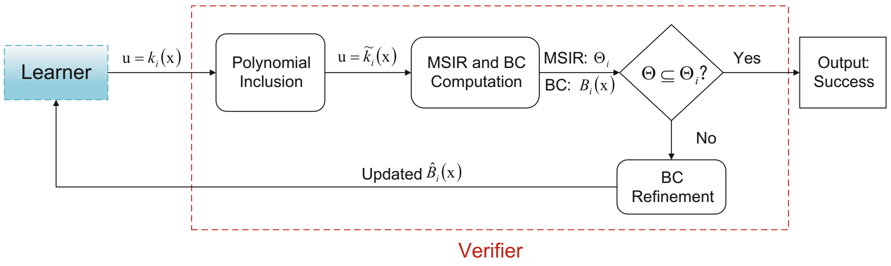

Fig. 2. The framework of the *verifier*.

Shown in Fig. 2, the *learner* produces a DNN controller $k_i(\mathbf{x})$. In order to make the problem of generating barrier certificates amenable to polynomial optimization problem, the *verifier* first employs Bernstein polynomial approximation to abstract the learned DNN controller as a polynomial one $\widetilde{k}_i(\mathbf{x})$, with the associated abstract error ϵ modeled as a bounded parameter, that is, $\mathbf{u} = \widetilde{k}_i(\mathbf{x}) + \epsilon$.

By doing it, the safety of the closed-loop system under the DNN controller can be guaranteed via the existence of barrier certificates for the closed-loop system under the abstract controller. The *verifier* then performs bilinear matrix inequalities (BMI) solving technique, to obtain a maximal safe initial region (MSIR) Θ_i and the corresponding barrier certificate $B_i(\mathbf{x})$. Once the computed MSIR Θ_i contains the given initial set Θ, then the safety of the closed-loop system under the DNN controller $\mathbf{u} = k_i(\mathbf{x})$ is verified. Otherwise, the *verifier* slightly adjusts the barrier certificate $B_i(\mathbf{x})$, based on quadratic programming solving, to gain an updated one $\widetilde{B}_i(\mathbf{x})$, which can separate the unsafe region from the initial set. Then, the refined BC is fed to guide the *learner* to retrain a new DNN controller.

Polynomial Abstraction of DNN Controllers. In the following, we consider the DNN controller with a single output, and for multiple-output cases, an extension is to approximate each output respectively. Formally, for a DNN controller $k(\mathbf{x})$, we seek to compute an approximate polynomial $p(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ with a verified bound $\mu \in \mathbb{R}_+$, such that

$$|k(\mathbf{x}) - p(\mathbf{x})| < \mu, \forall \mathbf{x} \in \Psi,$$

and the bound μ is as small as possible.

Weierstrass approximation theorem [7] asserts that a continuous function on a closed and bounded interval can be uniformly approximated on the interval by polynomials to any degree of accuracy. In this paper, we will compute the approximate polynomial based on the theory of Bernstein polynomials [9]. Let $\mathbf{d} = (d_1, \cdots, d_n) \in \mathbb{N}^n$ and $f : [0,1]^n \to \mathbb{R}$. The polynomial

$$B_{f,\mathbf{d}}(\mathbf{x}) = \sum_{\substack{0 \le c_j \le d_j \\ j \in \{1,\cdots,n\}}} f\Big(\frac{c_1}{d_1}, \cdots, \frac{c_n}{d_n}\Big) \prod_{j=1}^{n} \binom{d_j}{c_j} x_j^{c_j} (1 - x_j)^{d_j - c_j}$$

is called the multivariate Bernstein polynomial of f. Theoretically, the Bernstein polynomial $B_{f,\mathbf{d}}(\mathbf{x})$ converges uniformly to f for $d_1, \cdots, d_n \to \infty$. In practice, the estimation of the approximation error bound is needed. As stated in [9], assume f is a Lipschitz continuous function over $I : [0,1]^n$ with a Lipschitz constant L, then we have

$$\|B_{f,\mathbf{d}}(\mathbf{x}) - f(\mathbf{x})\| \le \frac{L}{2}\left(\sum_{j=1}^{n}\Big(\frac{1}{d_j}\Big)\right)^{\frac{1}{2}}, \forall \mathbf{x} \in I.$$

Now, for the DNN controller $k(\mathbf{x})$ over a domain Ψ, we can apply the above method to obtain a Bernstein polynomial with a valid approximate error bound as its abstraction. Concretely, we first construct an interval enclosure for Ψ, and apply a linear transformation to map the interval enclosure onto the unit box I, then utilize Bernstein polynomial approximation to obtain an abstract polynomial controller $\widetilde{k}(\mathbf{x}) + \epsilon$ with $\epsilon \in [-\mu, \mu]$, where $\widetilde{k}(\mathbf{x})$ is a Bernstein polynomial of $k(\mathbf{x})$ and μ is its valid approximate error bound. Note that the fully-connected neural networks with sigmoid and tanh activation functions are Lipschitz continuous, and the estimation of Lipschitz constants for deep neural networks has been studied in [14,31,34].

Maximal Safe Initial Region Computation. Since $\widetilde{k}(\mathbf{x})+\epsilon$ enclosures $k(\mathbf{x})$, the safety of the closed-loop system under the DNN controller $k(\mathbf{x})$ can be guaranteed via the existence of barrier certificates for the closed-loop system under the abstract controller $\widetilde{k}(\mathbf{x})+\epsilon$. From this observation, we try to compute an MSIR Θ_γ and its corresponding barrier certificate $B_\gamma(\mathbf{x})$, which can guarantee the safety of the closed-loop system under the abstract controller $\widetilde{k}(\mathbf{x})+\epsilon$ with respect to Θ_γ.

Firstly, we consider how to predefine a suitable initial state set template Θ_γ from the given initial set Θ. In what follows, we provide some parametric initial state sets for two typical representations: Boxes and Euclidean ellipsoids (balls).

Box Template. Suppose that the box initial set Θ is represented as

$$\Theta = \{\mathbf{x} \in \mathbb{R}^n \| |x_i - c_i| \le b_i\},$$

where $\mathbf{x}_c = (c_1, \cdots, c_n)^T$ is the center of the box, and $b_i \in \mathbb{R}_{>0}$. Then, the parametric initial set can be expressed as

$$\Theta_\gamma = \{\mathbf{x} \in \mathbb{R}^n | \|D^{-1}(\mathbf{x} - \mathbf{x}_c)\|_\infty \le \gamma\},$$

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

Ellipsoid Template. Suppose that the ellipsoid initial set Θ is expressed as a common representation:

$$\Theta = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} = \mathbf{x}_c + A\mathbf{v},\ \|\mathbf{v}\|_2 \le 1\},$$

where $\mathbf{x}_c$ is the center of the ellipsoid, and the matrix A is nonsingular. Then the parametric initial set can be expressed as

$$\begin{aligned}\Theta_\gamma &= \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} = \mathbf{x}_0 + \gamma\, A\, \mathbf{v}, \|\mathbf{v}\|_2 \le 1\} \\ &= \{\mathbf{x} \in \mathbb{R}^n | \|A^{-1}\,(\mathbf{x} - \mathbf{x}_0)\|_2 \le \gamma\}.\end{aligned}$$

Without loss of generality, we can select the template of the parametric initial sets by taking the form $\Theta_\gamma := \{\mathbf{x} \in \mathbb{R}^n | g(\mathbf{x}) \le \gamma, i = 1, \ldots, m_1\}$ with $\gamma \in \mathbb{R}_{>0}$, where $g(\mathbf{x})$ is the polynomial used to defined the prescribed initial set Θ.

In order to enlarge the safe initial region by choice of Θ_γ, we maximize γ while imposing the constraints for the existence of barrier certificates. Assume that the barrier certificate $B(\mathbf{x})$ is a polynomial of degree at most d, whose coefficients form a vector space of dimension $s(d) = \binom{n+d}{d}$ with the canonical basis $(\mathbf{x}^\alpha)$ of monomials. Suppose the coefficients are unknown, and denote by $\mathbf{b} = (b_\alpha) \in \mathbb{R}^{s(d)}$ the coefficient vector of $B(\mathbf{x})$, and write

$$B(\mathbf{x}, \mathbf{b}) = \sum_{\alpha \in \mathbb{N}_d^n} b_\alpha \mathbf{x}^\alpha = \sum_{\alpha \in \mathbb{N}_d^n} b_\alpha\, x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n},$$

in the canonical basis. Thus, the problem of computing an MSIR Θ_γ of the closed-loop system under the abstract controller $\widetilde{k}(\mathbf{x})+\epsilon$ can be represented as an optimization problem

$$\left.\begin{array}{l}\gamma_{opt}^{*} = \max_{\mathbf{b},\gamma} \gamma \\ \text{s.t.} \;\; B(\mathbf{x},\mathbf{b}) \geq 0, \;\; \forall \mathbf{x} \in \Theta_{\gamma}, \\ \qquad \mathcal{L}_{\mathbf{f}} B(\mathbf{x},\mathbf{b}) > 0, \;\; \forall \mathbf{x} \in \Psi \text{ and } B(\mathbf{x},\mathbf{b}) = 0, \\ \qquad B(\mathbf{x},\mathbf{b}) < 0, \;\; \forall \mathbf{x} \in X_u. \end{array}\right\} \tag{4}$$

Then, Sum-of-Squares (SOS) relaxation technique is applied to encode the optimization problem (4) as a SOS program. In fact, given a basic semi-algebraic set $\mathbb{K}$ defined by:

$$\mathbb{K} = \{\mathbf{x} \in \mathbb{R}^n \mid p_1(\mathbf{x}) \geq 0, \ldots, p_s(\mathbf{x}) \geq 0\},$$

where $p_j \in \mathbb{R}[\mathbf{x}], 1 \leq j \leq s$, a sufficient condition for the nonnegativity of the given polynomial $f(\mathbf{x})$ on the semi-algebraic set $\mathbb{K}$ is provided as

$$f(\mathbf{x}) = \sigma_0(\mathbf{x}) + \sum_{i=1}^{s} \sigma_i(\mathbf{x}) p_i(\mathbf{x}), \tag{5}$$

where $\sigma_i \in \Sigma[\mathbf{x}]_d$, $1 \leq i \leq s$. Thus, the representation (5) ensures that the polynomial $f(\mathbf{x})$ is nonnegative on the given semi-algebraic set $\mathbb{K}$.

Observing (4), the polynomial $\mathcal{L}_{\mathbf{f}} B(\mathbf{x},\mathbf{b})$ is involved with the uncertain variable ϵ in the range $[-\mu, \mu]$, which can be written as the constraint, $\hat{h}(\epsilon) \geq 0$ with

$$\hat{h}(\epsilon) := (\epsilon + \mu)(\mu - \epsilon).$$

Thus, the problem (4) can be transformed into the following optimization problem

$$\left.\begin{array}{l}\gamma^{*} = \max_{\mathbf{b},\gamma} \gamma \\ \text{s.t.} \;\; B(\mathbf{x},\mathbf{b}) - \sigma(\mathbf{x})(\gamma - g(\mathbf{x})) \in \Sigma[\mathbf{x}], \\ \qquad \mathcal{L}_{\mathbf{f}} B(\mathbf{x},\mathbf{b}) - \lambda(\mathbf{x}) B(\mathbf{x},\mathbf{b}) - \sum_j \phi_j(\mathbf{x}) h_j(\mathbf{x}) - \nu(\mathbf{x},\varepsilon)\hat{h}(\varepsilon) - \epsilon_1 \in \Sigma[\mathbf{x}], \\ \qquad -B(\mathbf{x},\mathbf{b}) - \epsilon_2 - \sum_j \kappa_j(\mathbf{x}) q_j(\mathbf{x}) \in \Sigma[\mathbf{x}], \end{array}\right\} \tag{6}$$

where $\epsilon_1, \epsilon_2 > 0$, the entries of $\sigma(\mathbf{x})$, $\phi_j(\mathbf{x})$ $\kappa(\mathbf{x}) \in \Sigma[\mathbf{x}]$, and $\nu(\mathbf{x},\varepsilon) \in \Sigma[\mathbf{x},\varepsilon]$, and $\lambda(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$. Note that ϵ_1, ϵ_2 are needed to ensure positivity of polynomials as required in the second and third constraints in (4). Clearly, the feasibility of the constraints in (6) is sufficient to imply the feasibility of the constraints in (4), thus the optimum of (6) is a lower bound of the optimum of (4), i.e., $\gamma^* \leq \gamma_{opt}^*$.

The SOS program (6) is bilinear due to the product of the unknown coefficients of $(B(\mathbf{x},\mathbf{b}), \lambda(\mathbf{x}))$ and $(\sigma(\mathbf{x}), \gamma)$, yielding a non-convex bilinear matrix inequalities (BMI) problem. Fortunately, a Matlab package PENBMI solver [22], which combines the (exterior) penalty and (interior) barrier method with the augmented Lagrangian method, can be applied directly to obtain a numerical solution of the problem (6). The solution $\gamma^*, \mathbf{b}^*$ to problem (6) yields an MSIR Θ_{γ^*} and its corresponding barrier certificate $B(\mathbf{x},\mathbf{b}^*)$. It means that the closed-loop system under the abstract controller $\widetilde{k}(\mathbf{x}) + \epsilon$ is safe, with respect to Θ_{γ^*}. Moreover, if the given initial set Θ is a subset of Θ_{γ^*}, then the safety of the closed-loop system under the DNN controller $k(\mathbf{x})$ with respect to Θ is verified. Otherwise, $B(\mathbf{x},\mathbf{b}^*)$ will be further refined via quadratic programming method.

Remark 3. The gap between the optima of problems (4) and (6) decreases as increasing of degrees for the multiplier polynomials. The degree bound for the multiplier polynomials is exponential with the number of variables $\mathbf{x}$ and the degrees of the polynomials appearing in the semi-algebraic sets. In practice, we set up a truncated SOS programming for (6) by fixing a *priori* (much smaller) degree bound of all the unknown multiplier polynomials, to avoid high computational complexity.

Barrier Certificate Refinement. Consider the case in which the initial set Θ is not a subset of the MSIR Θ_{γ^*}. In this case, the barrier certificate $B(\mathbf{x}, \mathbf{b}^*)$ can succeed to separate the unsafe region X_u from Θ_{γ^*}, but it may fail to separate from Θ. In other words, $B(\mathbf{x}, \mathbf{b}^*)$ can not be regarded as a truly candidate barrier certificate with respect to Θ and X_u. Therefore, we will utilize the information of $B(\mathbf{x}, \mathbf{b}^*)$ to refine it, in order to obtain a new candidate barrier certificate that can separate Θ from X_u. Consider the change in $B(\mathbf{x}, \mathbf{b}^*)$ is expected as small as possible, the step of the barrier certificate refinement can be represented as

$$\left.\begin{array}{ll} \min \|\hat{\mathbf{b}} - \mathbf{b}^*\|_2^2 & \\ \text{s.t.} \;\; B(\mathbf{x}, \hat{\mathbf{b}}) \geq 0 \;\forall \mathbf{x} \in \Theta, & \\ \qquad B(\mathbf{x}, \hat{\mathbf{b}}) < 0 \;\forall \mathbf{x} \in X_u. & \end{array}\right\} \tag{7}$$

By investigating (7), the constraints are the ones involving universal quantifiers. To avoid eliminating universal quantifiers directly, here we provide a relaxation technique to deal with (7), which is based on selecting sampling points. For Θ and X_u, let us first construct rectangular meshes in Θ and X_u respectively, with a mesh spacing $r \in \mathbb{R}_+$ (say $r = 0.05$). The resulting mesh point sets are denoted as Ω_Θ and Ω_{X_u}, respectively.

It is known that for a continuously differentiable function $\phi(\mathbf{x})$ over a compact domain D, the mean value theorem yields that

$$|\phi(\mathbf{x} + \Delta\mathbf{x}) - \phi(\mathbf{x})| \leq n\eta \|\Delta\mathbf{x}\|_\infty,$$

where $\mathbf{x}, \mathbf{x} + \Delta \in \Omega$ are chosen randomly, and $\eta = \sup_{\mathbf{x} \in D} \|\nabla\phi(\mathbf{x})\|_\infty$. Based on the above observation, the following implications are satisfied:

$$\left.\begin{array}{l} B(\mathbf{x}_j, \hat{\mathbf{b}}) - \delta_1 \geq 0, \;\forall \mathbf{x}_j \in \Omega_\Theta \Longrightarrow B(\mathbf{x}, \hat{\mathbf{b}}) \geq 0 \;\forall \mathbf{x} \in \Theta, \\ B(\mathbf{x}_j, \hat{\mathbf{b}}) + \delta_2 < 0, \;\forall \mathbf{x}_j \in \Omega_{X_u} \Longrightarrow B(\mathbf{x}, \hat{\mathbf{b}}) < 0 \;\forall \mathbf{x} \in X_u. \end{array}\right\}$$

where $\delta_i = n\eta_i r \in \mathbb{R}_{>0}, i = 1, 2$ with $\eta_1 = \sup_{\mathbf{x} \in \Theta} \|\nabla B(\mathbf{x}, \mathbf{b}^*)\|_\infty$ and $\eta_2 = \sup_{\mathbf{x} \in X_u} \|\nabla B(\mathbf{x}, \mathbf{b}^*)\|_\infty$.

By using the above relaxation technique based on sampling points, (7) can be relaxed as the following problem

$$\left.\begin{array}{ll} \min \|\hat{\mathbf{b}} - \mathbf{b}^*\|_2^2 & \\ \text{s.t.} \;\; B(\mathbf{x}_j, \hat{\mathbf{b}}) - \delta \geq 0, \;\forall \mathbf{x}_j \in \Omega_\Theta, & \\ \qquad B(\mathbf{x}_j, \hat{\mathbf{b}}) + \delta < 0, \;\forall \mathbf{x}_j \in \Omega_{X_u}, & \end{array}\right\} \tag{8}$$

which is a typical quadratic programming problem and can be solved by state-of-the-art solvers with great efficiency.

Now, the refined $\widehat{B}(\mathbf{x}) = B(\mathbf{x}, \hat{\mathbf{b}})$ can separate Θ from X_u, but may still not satisfy the Lie derivative condition for barrier certificates. According to Theorem 1, $\widehat{B}(\mathbf{x})$ is not a truly barrier certificate for the closed-loop system under the abstract controller $\widetilde{k}(\mathbf{x}) + \epsilon$ with respect to Θ and X_u. Next, the refined $\widehat{B}(\mathbf{x})$ will be further fed to guide the *learner* to retrain a new controller. To do it, we first consider the additional constraint for the Lie derivative of $\widehat{B}(\mathbf{x})$, and apply barrier certificate guided reinforcement learning to compute a new DNN controller.

4 Algorithm

In Sect. 3, we have elaborated on the iteration-based safe controller synthesis method that iteratively co-synthesizes a DNN controller within the RL framework and a polynomial barrier certificate via BMI solving. Briefly, we describe the main implementation steps of our approach in the following Algorithm 2.

Algorithm 2. SRLBC: Safe Reinforcement Learning with Barrier Certificate

```
Input: The CCDS $\mathcal{C}$; unsafe region $X_u$; maximum number of iterations $maxIter$
Output: Safe DNN Controller $k$
1: $iter \leftarrow 0$
2: $B \leftarrow \perp$
3: while $iter < maxIter$ do
4:     $k \leftarrow \text{Learning}(f, \Theta, X_u, B)$
5:     $\widehat{k}, \mu \leftarrow \text{PolyInclusion}(k)$
6:     $\Theta^*_\gamma, B(\mathbf{x}, \mathbf{b}^*) \leftarrow \text{MaxSafeSet}(f, \widehat{k}, \mu, \Theta, X_u)$
7:     if $\Theta \subseteq \Theta^*_\gamma$ then
8:         return k
9:     end if
10:    $B \leftarrow \text{RefineBarrier}(B(\mathbf{x}, \mathbf{b}^*), \Theta, X_u)$
11: end while
```

Algorithm 2 shows the iteration scheme of our safe controller synthesis, which guides the experiment implementation. The procedure takes as inputs a CCDS $\mathcal{C}$, an unsafe region X_u, a maximum number of iterations *maxIter*, and returns a safe DNN controller of a given architecture. In a pass of the iteration, the implementation process has four steps as follows.

(i) Apply the RL method to train a DNN controller. The *learner* introduced in Sect. 3.1 is implemented by Line 4 in Algorithm 2, and the barrier certificate is initialized to be $\perp$, which means that the *learner* trains a DNN controller via classical reinforcement learning, without the aid of barrier certificates in the initial pass;

(ii) For the closed-loop system under the DNN controller learned in Step (i), compute a maximal safe initial region (MSIR), with which a barrier certificate exists. We use Bernstein polynomial approximation to compute a polynomial abstraction for the learned DNN controller by Line 5, and then compute an MSIR Θ_{γ^*} and the corresponding barrier certificate $B(\mathbf{x}, \mathbf{b}^*)$ by Line 6;
(iii) Check the condition wether the MSIR Θ_{γ^*} in Step (ii) contains the given initial set Θ. If $\Theta \subseteq \Theta_{\gamma^*}$, then we terminate the loop with a verified safe DNN controller; otherwise go to Step (iv). This process refers to Lines 7–9;
(iv) Slightly modify the barrier certificate from Step (iii) so that it separates the initial set and the unsafe region, and then go to Step (i) to learn a new controller by encoding the refined barrier certificate into the reward function. For this task, the barrier certificate B is refined via quadratic programming by Line 10.

This inductive loop repeats until an MSIR enclosing Θ_γ and its corresponding barrier certificate are computed or until a timeout is reached.

Remark 4. Our procedure is sound, i.e. a valid output from the *verifier* is provably correct. However, we cannot claim any completeness, since our procedure might in general not terminate because the existence of the barrier certificate is just a sufficient condition of the safety of the system, and such a barrier certificate may not exist indeed. Once the procedure fails, we may improve the relaxation precision and then increase the possibility to find the barrier certificate by increasing the degree bound for the multiplier polynomials in the SOS program (6).

Furthermore, an example is used to depict how our safe controller synthesis algorithm works.

Example 1. Consider the Van der Pol system

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_1 + \frac{1}{3}x_1^3 - x_2 + u \end{bmatrix}$$

with the domain $\Psi = \{\mathbf{x} \in \mathbb{R}^2 \mid -3 \leq x_1, x_2 \leq 3\}$. Our goal is to design a control law k such that all trajectories of the system under $u = k(x_1, x_2)$ starting from the initial set

$$\Theta = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 1.5)^2 + x_2^2 \leq 1.1^2\}$$

will never enter the unsafe set

$$X_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 1)^2 + (x_2 + 1)^2 \leq 1\}.$$

We complete our goal by Algorithm 2, and provide the details here. At first, we apply the reinforcement learning method to train the initial neural network controller $u = k_0(\mathbf{x})$ in terms of the target of safety satisfiability, which is Step

(i) and refers to Line 4 in Algorithm 2, and then try to yield the barrier certificate $B(\mathbf{x})$. We compute polynomial abstraction of DNN Controller $k_0(\mathbf{x})$ via Bernstein polynomials which is Step (ii), where

$$\begin{aligned}\tilde{k}_0(\mathbf{x}) = 0.0142x_1 + 0.0092x_2 - 0.0205x_1^2 + 0.0077x_1x_2 + 0.0340x_2^2 \\ + 0.0246x_1^3 + 0.0018x_1^2x_2 - 0.0820x_1x_2^2 + 0.0435x_2^3 + \epsilon.\end{aligned} \tag{9}$$

with $\epsilon \in [-0.05, 0.05]$, which is implemented by Line 5. Thus, the polynomial abstraction technique can yield an abstract polynomial system.

Go on Step (ii) to compute a maximal safety region Θ_γ and the corresponding barrier certificate $B(\mathbf{x})$. In this case, we parameterize the initial set:

$$\Theta_\gamma = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 1.5)^2 + x_2^2 \le \gamma\}.$$

For the given abstract polynomial system with the parameterized initial set Θ_γ, our goal is to maximize the radius γ subject to the existence of a barrier certificate. By calling the PENBMI solver [22] we can obtain a barrier certificate $B_0(\mathbf{x})$ with the maximal safe initial region Θ_0 (Line 6 in our Algorithm 2), i.e.,

$$\begin{aligned}&\Theta_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 1.5)^2 + x_2^2 \le 0.8132\}, \\ &B_0(\mathbf{x}) = 11.716 + 22.8064x_1 + 21.5368x_2 - 4.5273x_1^2 + 13.8084x_1x_2 + 3.0453x_2^2.\end{aligned} \tag{10}$$

Thus, the safety of the system with the controller $k_0(\mathbf{x})$ with respect to the set Θ_0 is guaranteed. Now the present controller $k_0(\mathbf{x})$ can not be safe for whole initial set Θ, we continue to update controller and barrier certificate (Line 7–9).

Let $k_0(\mathbf{x})$ and $B_0(\mathbf{x})$ be the initial controller and the initial barrier certificate, we perform the iterative framework to synthesize the controller subject to the safety constraint. As shown in Fig. 3(a), the zero level set of $B_0(\mathbf{x})$ is the blue dashed line. Observing Fig. 3(a), $B_0(\mathbf{x})$ can succeed to separate the unsafe region X_u (the red circle) from Θ_0 (the green dashed circle), but not separate from the initial set Θ, which means that $B_0(\mathbf{x})$ can not be regarded as the truly barrier certificate. Therefore, one may perturb the coefficients of $B_0(\mathbf{x})$ to obtain $\hat{B}_0(\mathbf{x})$ which can separate Θ and X_u. And this process corresponds to Step (iv) and Line 10 of our Algorithm 2. The perturbed polynomial is represented as

$$\hat{B}_0(\mathbf{x}) = 10.5590 + 22.9401x_1 + 18.2448x_2 - 0.8954x_1^2 + 14.4971x_1x_2 + 1.1060x_2^2.$$

As shown in Fig. 1(b), the zero level set of the barrier $\hat{B}_0(\mathbf{x})$ (the blue dash) separates X_u (the red circle) from Θ (the green circle). According to the concept of barrier certificate and Theorem 1, $\hat{B}_0(\mathbf{x})$ is not a truly barrier certificate, since the condition of the Lie derivative of the barrier certificate is not satisfied. Accordingly, by using the $\hat{B}_0(\mathbf{x})$ and the initial controller $k_0(\mathbf{x})$, we then try to retrain a control law with an additional constraint of the lie derivative for the barrier certificate $\hat{B}_0(\mathbf{x})$. Calling the *learner* module (Line 4), we update a new control law $k_1(\mathbf{x})$ represented as a two-hidden layer sigmoid-based DNN with 20 neurons per layer by RL approach.

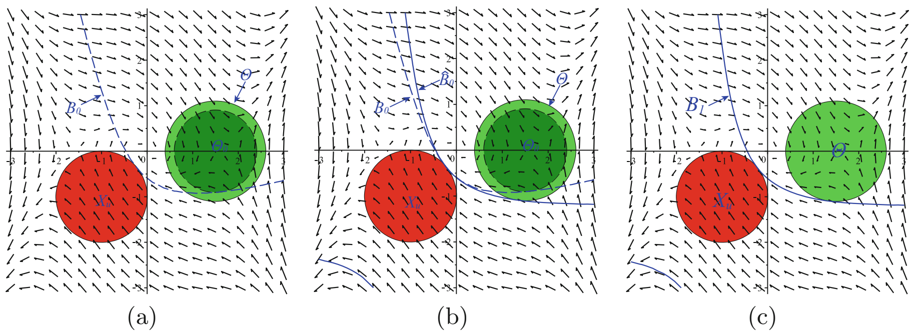

Fig. 3. This picture shows the iteration process of barrier certificate updating when we learn the safe controller. The red circles stand for unsafe regions, the blue curves stand for the zero level set of barrier certificates, and the green circles stand for the initial sets and safe initial sets. Subfigure (a) describes the intermediate results of maximal safe initial set Θ_0 (the green dashed circle) with its associate barrier certificate B_0 obtained from Line 6 in Algorithm 2 at the first iteration. We slightly modify the barrier function B_0 to separate Θ and X_u by Line 10 and obtain $\hat{B}_0$ which is the blue solid curve shown in Subfigure (b). Using $\hat{B}_0$ as a guide, a new controller is learned, from which a barrier certificate B_1 is generated as shown in Subfigure (c). It can be shown that B_1 is the real barrier certificate of the system. (Color figure online)

Repeating the above abstraction technique and solving the BMI problem for finding the maximal safety initial set Θ_1, we obtain the barrier certificate $B_1(\mathbf{x})$ with respect to Θ_1, i.e.,

$$\begin{aligned} &\Theta_1 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 1.5)^2 + x_2^2 \le 1.2201\}, \\ &B_1(\mathbf{x}) = 10.3661 + 22.6569x_1 + 17.7852x_2 - 0.9037x_1^2 + 14.1832x_1x_2 + 0.9471x_2^2. \end{aligned} \tag{11}$$

It is easy to check that the original initial set Θ is now a subset of Θ_1, which means that $B_1(\mathbf{x})$ is a truly barrier certificate.

5 Experiments

In this section, we first depict an example of three dimension nonlinear continuous system to show our algorithm by synthesizing a safe DNN controller for it, and then present an experimental evaluation of our algorithm over a set of benchmark examples by comparing with a DNN controller learning framework called *nncontroller* in [39].

Example 2. Consider the continuous dynamical system

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} = \begin{bmatrix} x_3 + 8x_2 \\ -x_2 + x_3 \\ -x_3 - x_1^2 + u \end{bmatrix}$$

with the domain

$$\Psi = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1^2 + x_2^2 + x_3^2 \le 16\}.$$

Our goal is to design a control law k such that all trajectories of the closed-loop system under $u = k(x_1, x_2, x_3)$ starting from the initial set

$$\Theta = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1^2 + x_2^2 + x_3^2 \leq 1\}$$

will never enter the unsafe set

$$X_u = \{\mathbf{x} \in \mathbb{R}^3 \mid (x_1 - 2.1)^2 + (x_2 - 2.1)^2 + (x_3^2 - 2.1) \leq 1.8^2\}.$$

It suffices to synthesize a control law k and a barrier certificate $B(\mathbf{x})$ with the maximal safe initial region Θ_γ such that $\Theta \subseteq \Theta_\gamma$. Suppose that the DNN controller k is represented as a five-hidden layer sigmoid activated DNN with 30 neurons per layer. We first call the *learner* to train a DNN controller, and then call the *verifier* to compute the maximal safe initial region Θ_γ and its corresponding barrier certificate $B(\mathbf{x})$. After two iterations, we successfully obtain a safe DNN controller, and the following barrier certificate

$$\begin{aligned} B(\mathbf{x}) = {} & 220.1981 - 45.7322x_1 - 40.2831x_2 - 218.4765x_3 + 4.9575x_1^2 \\ & + 38.7288x_1x_2 - 9.8224x_1x_3 - 66.8398x_2^2 + 17.2562x_2x_3 + 18.3967x_3^2. \end{aligned} \tag{12}$$

As shown in Fig. 4, the zero level set of the barrier certificate $B(\mathbf{x})$ (the blue surface) separates X_u (the red ball) from all trajectories starting from Θ (the green ball). Therefore, the safety of the above system is verified.

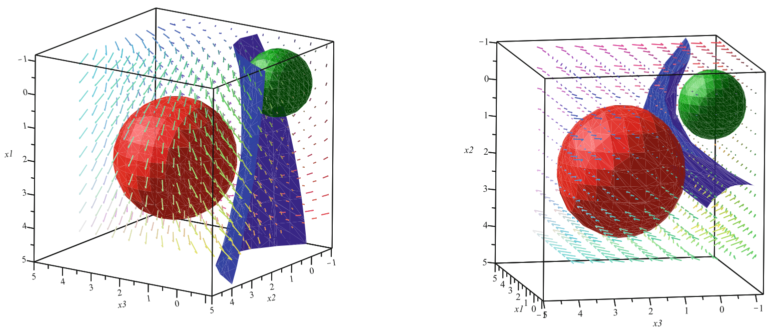

Fig. 4. Phase portrait of the system in Example 2. The zero level set of the barrier certificate $B(\mathbf{x})$ (the blue surface) separates X_u (the red ball) from all trajectories starting from Θ (the green ball). (Color figure online)

We have implemented a safe controller synthesis tool called *SRLBC* based on Algorithm 2, with Tensorflow 1.14 for the DNN controller synthesis and a Matlab package PENBMI [22] for barrier certificate generation. Table 1 shows the performance evaluation of our *SRLBC* and *nncontroller* in [39] on 12 continuous systems. All experiments are conducted on a machine running Windows 10 with 16 GB RAM, a 3.20 GHz AMD Ryzen 7 3700X CPU, and an NVIDIA GeForce GTX 1650 super GPU.

In Table 1, the origins of these 12 examples are provided in the first column; $d_{\mathbf{f}}$ denotes the maximal degree of the polynomials in the vector fields; $n_{\mathbf{x}}$ denotes the number of the state variables; L and N refer to the numbers of hidden layers and the neurons per each hidden layer, respectively; t_1 and t_2 denote the time spent by *SRLBC* and *nncontroller* in seconds, respectively; the symbol $'-'$ means that *nncontroller* was unable to return a safe DNN controller within 10,000 s.

Table 1. Performance evaluation

Examples	d_f	$n_{\mathbf{x}}$	NNstructure		*SRLBC*		*nncontroller*	
			L	N	deg$B(\mathbf{x})$	t(s)	NN-type BC	t(s)
C1 [28]	2	2	4	20	2	54.77	2-10-1	20.52
C2 [6]	3	2	4	20	2	37.54	2-10-1	8.46
C3 [6]	3	2	4	20	2	35.99	2-10-1	6.77
C4 [27]	3	2	4	20	4	38.68	2-10-1	6.88
C5 [39]	3	3	5	30	2	56.21	3-10-1	32.19
C6 [20]	3	4	5	30	2	45.54	4-10-1	78.52
C7 [6]	3	4	5	30	4	40.82	4-10-1	184.85
C8 [32]	2	5	5	30	2	423.11	5-20-1	2217.41
C9 [38]	2	6	5	30	2	383.26	–	–
C10 [4]	3	6	5	30	4	942.74	–	–
C11 [21]	2	7	5	30	2	1829.46	–	–
C12 [21]	2	9	5	30	2	6208.79	–	–

Table 1 shows that for the 12 examples, our *SRLBC* manages to handle all of them within 3 iterations, while *nncontroller* can only deal with 8 successfully. Especially for the four examples from C9 to C12 whose dimensions exceed 5, *nncontroller* fails to synthesize safe controllers within specified time bound after various attempt. We have tried different network structures with the number of hidden layers varies from 1 to 5 and the number of hidden neurons chosen among $\{10, 20, 30, 40\}$, the *nncontroller* fails to train candidate DNN controllers and barrier certificates within the time limit, whereas our *SRLBC* can yield safe controllers, represented as five-layer sigmoid activated neural networks.

Consider the efficiency of our *SRLBC* and *nncontroller* in terms of the time spent in synthesizing safe DNN controllers for shared examples. On average, our *SRLBC* takes 91.58 s to synthesize a safe DNN controller while *nncontroller* needs 323.2 s, which is about 3.53 times slower than our *SRLBC*. Despite the network structures used for *SRLBC* is more complex than that for *nncontroller*, and the number of neural network neurons of *SRLBC* is much more than that of *nncontroller*, we could synthesize more efficiently.

Obviously, our *SRLBC* scales better than *nncontroller* for the considered examples. Although our *SRLBC* consumes a little more time than *nncontroller* for the systems with dimension 2 or 3, our tool shows its advantage on time consuming when handling the systems with dimension higher than 3 (C6-C8) and its ability on examples C9-C12. Comparing with *nncontroller* which is also a data driven approach, *SRLBC* inherits the advantage in learning efficiency of reinforcement learning, whereas the size of the training data for *nncontroller* increases exponentially with the dimension of the considered systems, which greatly limits the scale of the problem to deal with. Beyond Table 1, we have tried an example of nonlinear polynomial system [16] with dimension up to 12, and *SRLBC* yields successfully a result in 54,314 s while *nncontroller* fails. It is clear that our approach is able to attack large-scale problems.

During the experiment, we have observed that *SRLBC* obtains the near-safe controllers at the first iteration for most examples, and the remaining work is to refine barrier certificates slightly and use them to guide and adjust the controllers. In fact, the numbers of the iterations in our experiments on the benchmarks did not exceed 3 for all cases. These observations show that our iterative scheme of safe reinforcement learning converges well in practice, because the refinement of the controllers could utilize the intermediate learned results before we get the final results. In addition, *SRLBC* could easily generalize to deal with non-polynomial systems and it has successfully solved the classical continuous Cartpole system [3], which would be presented in the future work.

6 Related Work

Our work on synthesizing DNN controllers for safety control of nonlinear systems is mainly related to two categories of research, i.e. *formal verification of nonlinear systems with DNN controller* and *safe DNN controller synthesis*. There has been considerable research conducted in these areas because of the applications in safety critical systems in recent years.

Formal Verification of Nonlinear Systems with DNN Controller. One of the mainstream methodologies is through constructing over-approximations to the reachable sets of the system trajectories under DNN controllers. And the core technique first focuses on output range analysis of the neural network components, then combines the output range with reachability analysis on the dynamical systems. For instance, based on the output range analysis in [13], Dutta et al. verified the feedback control systems with DNN controllers using mixed-integer linear programming [12]. And they implemented the prototype tool for the neural rule generation inside the tool termed as Sherlock, and used it together with Flow* for computing the reach sets of the systems [10].

The difference of works on this direction lies in what kind of abstract domains is adopted for output range analysis of the neural network components. A recent attempt involves the work of Xiang et al. that computes the output ranges as a union of convex polytopes [37]. For the piecewise linear systems with ReLU neural network as the controller, they compute the output range of ReLU neural network by a layer-by-layer approach. Dutta et al. propose an approach to abstract the DNN by a local polynomial approximation along with rigorous error bound, and then integrate it with a Taylor model-based flow pipe construction scheme for continuous differential equations to derive the over-approximation of the real reachable set [11]. Likely, Huang et al. present an approach to constructing a polynomial approximation for a DNN controller using Bernstein polynomials, and then integrate result with the plant to get the over-approximated reachable set [18]. There is a different route for reachability of systems with neural network components proposed by Ivanov et al. and termed as Verisig [19]. It transforms the problem of verifying neural network controlled system into a hybrid system verification problem by first transforming a sigmoid-based neural network into an equivalent hybrid system and then composing it with the plant.

Instead of computing reachable sets, a different approach for verifying neural network controlled systems is through barrier certificate synthesis. Tuncali et al. synthesize candidate barrier certificates using simulation-guided techniques, and then verify the overall system safety by checking the validity of the barrier certificate conditions for the candidate [35]. The safety property was proofed, or a counterexample was returned to updated candidate barrier certificates.

Safety Critical Controller Generation. Research works in this category differ in: (1) the overall learning framework, e.g. reinforcement learning (RL) or supervised learning; (2) the kind of safety certificate, e.g., control Lyapunov function (CLF) or control barrier function (CBF) [2].

For CLFs or CBFs synthesis, a demonstrator-learner-verifier framework was proposed in [29] to learn polynomial CLFs for polynomial nonlinear dynamical systems; a special type of neural network was designed in [30] as candidates for learning Lyapunov functions; a supervised learning approach was proposed in [5] to learn neural network Lyapunov functions and linear control policies; data-driven model predictive control (MPC) exploiting neural Lyapunov function and neural network dynamics model was proposed in [12,25]. For multi-agent systems, barrier function has recently been applied for safe policy synthesis on POMDP models [1]. The computer science community has dealt with the issue of safe controller learning in different ways from above: for example, a logical-proof based approach was proposed in [15] towards safe RL; a synthesis framework capable of synthesizing deterministic programs from neural network policies was proposed in [41] and so formal verification techniques for traditional software systems can be applied. Compared with these works, [39] learn controllers based on neural networks. To certify the safety property they utilize barrier certificates, which are represented by DNNs as well. In this way, they train DNN controllers and DNN barrier certificates simultaneously, achieving a verification-in-the-loop synthesis. Liu et al. proposed a Recurrent Neural Network (RNN) framework to synthesize feedback control policies for a system under STL specifications [24]. The CBF was used to modify the control policies predicted by the RNN to guarantee safety.

7 Conclusion

In this paper, we have developed a novel scheme for synthesizing safe controllers of nonlinear systems with control against safety constraints. It employs an iterative architecture, where a *learner* trains DNN controllers using reinforcement learning and a *verifier* checks them via computation of maximal safe initial regions and the corresponding barrier certificates, based on polynomial abstraction and bilinear matrix inequalities solving. The key idea in this paper is to use an alternating co-synthesis scheme of controllers and barrier certificates to generate safe controllers, which could refine barrier certificates during iteration. On the one hand, this synthesis scheme has inherited the higher learning efficiency from RL technique than other data driven methods. On the other hand, this iterative architecture could modify barrier certificates to obtain an adaptive one along with DNN controller retraining, and other verification-in-the-loop synthesis methods are usually based on user-defined barrier functions. Furthermore, our BMI solving based barrier certificate generation is more efficient than SMT based verification. The experimental results demonstrate that our method is more scalable and effective than the existing DNN controller synthesis method *nncontroller*.

References

1. Ahmadi, M., Singletary, A., Burdick, J.W., Ames, A.D.: Safe policy synthesis in multi-agent POMDPs via discrete-time barrier functions. In: Proceedings of the IEEE 58th Conference on Decision and Control (CDC), pp. 4797–4803. IEEE (2019)
2. Ames, A.D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., Tabuada, P.: Control barrier functions: theory and applications. In: Proceedings of the 17th European Control Conference, (ECC), pp. 3420–3431 (2019)
3. Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Trans. Syst. Man Cybern. **13**(5), 834–846 (1983)
4. Bouissou, O., Chapoutot, A., Djaballah, A., Kieffer, M.: Computation of parametric barrier functions for dynamical systems using interval analysis. In: Proceedings of the 53rd IEEE Conference on Decision and Control (CDC), pp. 753–758. IEEE (2014)
5. Chang, Y.C., Roohi, N., Gao, S.: Neural Lyapunov control. In: Proceedings of the Annual Conference on Advances in Neural Information Processing Systems (NeurIPS), pp. 3245–3254 (2019)
6. Chesi, G.: Computing output feedback controllers to enlarge the domain of attraction in polynomial systems. IEEE Trans. Autom. Control **49**(10), 1846–1853 (2004)
7. Davis, P.J.: Interpolation and Approximation. Dover Books on Mathematics. Dover Publications, New York (1975)
8. Deshmukh, J.V., Kapinski, J., Yamaguchi, T., Prokhorov, D.: Learning deep neural network controllers for dynamical systems with safety guarantees: Invited paper. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1–7 (2019)

9. Duchoň, M.: A generalized bernstein approximation theorem. Tatra Mt. Math. Publ. **49**(1), 99–109 (2011)
10. Dutta, S., Chen, X., Jha, S., Sankaranarayanan, S., Tiwari, A.: Sherlock - a tool for verification of neural network feedback systems: demo abstract. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC), pp. 262–263 (2019)
11. Dutta, S., Chen, X., Sankaranarayanan, S.: Reachability analysis for neural feedback systems using regressive polynomial rule inference. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC), pp. 157–168 (2019)
12. Dutta, S., Jha, S., Sankaranarayanan, S., Tiwari, A.: Learning and verification of feedback control systems using feedforward neural networks. IFAC-PapersOnLine **51**(16), 151–156 (2018)
13. Dutta, S., Jha, S., Sankaranarayanan, S., Tiwari, A.: Output range analysis for deep feedforward neural networks. In: Dutle, A., Muñoz, C., Narkawicz, A. (eds.) NFM 2018. LNCS, vol. 10811, pp. 121–138. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77935-5_9
14. Fazlyab, M., Robey, A., Hassani, H., Morari, M., Pappas, G.J.: Efficient and accurate estimation of lipschitz constants for deep neural networks. arXiv preprint arXiv:1906.04893 (2019)
15. Fulton, N., Platzer, A.: Safe reinforcement learning via formal methods: toward safe control through proof and learning. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI), pp. 6485–6492 (2018)
16. Gao, S.: Quadcopter model. https://github.com/dreal/benchmarks
17. García, J., o Fernández, F., et al.: A comprehensive survey on safe reinforcement learning. J. Mach. Learn. Res. **16**(42), 1437–1480 (2015)
18. Huang, C., Fan, J., Li, W., Chen, X., Zhu, Q.: ReachNN: reachability analysis of neural-network controlled systems. ACM Trans. Embedded Comput. Syst. **18**(5s), 106:1-106:22 (2019)
19. Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig: verifying safety properties of hybrid systems with neural network controllers. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC), pp. 169–178 (2019)
20. Jarvis-Wloszek, Z.: Lyapunov based analysis and controller synthesis for polynomial systems using sum-of-squares optimization. Ph.D. thesis, University of California (2003)
21. Klipp, E., Herwig, R., Kowald, A., Wierling, C., Lehrach, H.: Systems Biology in Practice: Concepts. Implementation and Application, Wiley-Blackwell (2005)
22. Kočvara, M., Stingl, M.: PENBMI user's guide (version 2.0) (2005). http://www.penopt.com
23. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. In: Proceedings of the 4th International Conference on Learning Representations (ICLR) (2016)
24. Liu, W., Mehdipour, N., Belta, C.: Recurrent neural network controllers for signal temporal logic specifications subject to safety constraints (2020). https://arxiv.org/abs/2009.11468
25. Mittal, M., Gallieri, M., Quaglino, A., Salehian, S.S.M., Koutník, J.: Neural Lyapunov model predictive control (2020). https://arxiv.org/abs/2002.10451
26. Prajna, S., Jadbabaie, A., Pappas, G.J.: A framework for worst-case and stochastic safety verification using barrier certificates. IEEE Trans. Autom. Control **52**(8), 1415–1429 (2007)

27. Prajna, S., Parrilo, P.A., Rantzer, A.: Nonlinear control synthesis by convex optimization. IEEE Trans. Autom. Control **49**(2), 310–314 (2004)
28. Pylorof, D., Bakolas, E.: Analysis and synthesis of nonlinear controllers for input constrained systems using semidefinite programming optimization. In: Proceedings of the 2016 American Control Conference (ACC), pp. 6959–6964 (2016)
29. Ravanbakhsh, H., Sankaranarayanan, S.: Learning control Lyapunov functions from counterexamples and demonstrations. Auton. Rob. **43**(2), 275–307 (2019)
30. Richards, S.M., Berkenkamp, F., Krause, A.: The Lyapunov neural network: adaptive stability certification for safe learning of dynamic systems (2018). http://arxiv.org/abs/1808.00924
31. Ruan, W., Huang, X., Kwiatkowska, M.: Reachability analysis of deep neural networks with provable guarantees. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI), pp. 2651–2659 (2018)
32. Sassi, M.A.B., Sankaranarayanan, S.: Stabilization of polynomial dynamical systems using linear programming based on bernstein polynomials (2015). arXiv preprint arXiv:1501.04578
33. Squires, E., Pierpaoli, P., Egerstedt, M.: Constructive barrier certificates with applications to fixed-wing aircraft collision avoidance. In: Proceedings of the IEEE Conference on Control Technology and Applications (CCTA), pp. 1656–1661 (2018)
34. Szegedy, C., et al.: Intriguing properties of neural networks. In: Proceedings of the 2nd International Conference on Learning Representations (ICLR) (2014)
35. Tuncali, C.E., Kapinski, J., Ito, H., Deshmukh, J.V.: Reasoning about safety of learning-enabled components in autonomous cyber-physical systems. In: Proceedings of the 55th Annual Design Automation Conference (DAC), pp. 30:1–30:6 (2018)
36. Turchetta, M., Kolobov, A., Shah, S., Krause, A., Agarwal, A.: Safe reinforcement learning via curriculum induction. In: Proceedings of the Annual Conference on Advances in Neural Information Processing Systems (NeurIPS), pp. 12151–12162 (2020)
37. Xiang, W., Tran, H.D., Rosenfeld, J.A., Johnson, T.T.: Reachable set estimation and safety verification for piecewise linear systems with neural network controllers. In: Proceedings of the Annual American Control Conference (ACC), pp. 1574–1579 (2018)
38. Zeng, X., Lin, W., Yang, Z., Chen, X., Wang, L.: Darboux-type barrier certificates for safety verification of nonlinear hybrid systems. In: Proceedings of the 2016 International Conference on Embedded Software (EMSOFT), pp. 1–10 (2016)
39. Zhao, H., Zeng, X., Chen, T., Liu, Z., Woodcock, J.: Learning safe neural network controllers with barrier certificates. In: Proceedings of the International Symposium on the Dependable Software Engineering. Theories, Tools, and Applications (SETTA), pp. 177–185 (2020)
40. Zhao, H., Zeng, X., Chen, T. Liu, Z., Woodcock, J.: Learning safe neural network controllers with barrier certificates. Formal Aspects Comput., 1–19 (2021). https://doi.org/10.1007/s00165-021-00544-5
41. Zhu, H., Xiong, Z., Magill, S., Jagannathan, S.: An inductive synthesis framework for verifiable reinforcement learning. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), pp. 686–701 (2019)