FoSRL: A Tool for Formally-Guaranteed Safe Reinforcement Learning

Luigi Berducci[1a], Shuo Yang[2], Mirco Giacobbe[3], Rahul Mangharam[2], Radu Grosu[1]

***Abstract*— Reinforcement learning shows promise for robot learning in complex environments but lacks of safety guarantees. This paper introduces FoSRL, a framework for training formally-guaranteed safe RL policies. Leveraging Control Barrier Functions and policy-gradient algorithms, FoSRL systematically integrates safety considerations in the exploration process. Demonstrations highlight FoSRL's efficacy in ensuring safety and robustness to environmental uncertainties, and usability to custom examples. Participants engage in interactive sessions, exploring FoSRL's practical usability and benefits in robotics applications through Python notebooks and simulations.**

I. INTRODUCTION

Reinforcement learning (RL) is pivotal in advancing autonomous robotics, offering adaptability to complex environments. However, ensuring safety in RL faces challenges due to real-world uncertainties and adoption of deep neural networks (DNNs). Despite advancements in Safe RL, the absence of formal guarantees hinders widespread deployment in critical domains like autonomous-driving or human-robot interaction. Without safety assurance, concerns persist regarding unpredictable behavior and risks to humans and the environment, hampering RL adoption. Addressing these challenges is crucial to unlocking the full potential of autonomous robotics and its transformative impact across various applications.

To address the above challenges, we present FoSRL in this paper, which is a novel framework designed to facilitate safe learning in robotics applications. By leveraging a set of rigorous assumptions on environment dynamics and uncertainties, our framework trains policies ensuring safe exploration.

Building upon Control Barrier Functions (CBF), FoSRL integrates modern RL algorithms while maintaining safety. Unlike prior works that rely on expert-defined CBFs, FoSRL employs a learner-verifier approach to construct a valid CBF.

The framework operates in two distinct phases (Fig. 1):

- **Counter-example Guided Pretraining**: First, FoSRL learns a CBF model from a symbolic abstraction of the environment. Through counter-example guided training, the framework iteratively refines the CBF to find a minimally-conservative function with provable validity.
- **Safe Interactive Learning**: Following CBF pretraining, FoSRL starts an iterative exploration process to collect interactions and reward from the environment. This phase optimizes policy for performance, while ensuring by construction to remain within the safe set of states.

[1]Institute of Computer Engineering, TU Wien
[2]Dept. of Electrical and Systems Engineering, University of Pennsylvania
[3]School of Computer Science, University of Birmingham
[a]Correspondance to: `luigi.berducci@tuwien.ac.at`

II. FoSRL: Theoretical Underpinnings

A. Safe RL through Robust CBF

In the study of safety-critical systems, a CBF h provides sufficient conditions for forward invariance of a safe set $\mathcal{C} = \{s \in S \backslash S_u : h(s) \geq 0\}$. However, uncertainty in other agents' behavior may deteriorate the safety guarantees of the CBF h. To address this problem, it is common in the literature to add a compensation term σ in the CBF condition.

Definition II.1. A continuous differentiable function h is a robust CBF (RCBF) if there exists a function $\sigma : S \to \mathbb{R}$ and α (where α is an extended class-$\mathcal{K}$ function) such that

$$\sup_a \dot{h}_n(s,a) - \sigma(s) + \alpha(h(s)) \geq 0 \qquad \forall s \in \mathcal{C} \tag{1}$$

$$\sigma(s) + \dot{h}(s,a,z) - \dot{h}_n(s,a) \geq 0 \qquad \forall s \in \partial\mathcal{C} \tag{2}$$

where $\dot{h}_n$ denotes the Lie derivative of h over the nominal dynamics f, g; the first condition ($g_f(s) \geq 0$) ensures the existence of an action to remain within the set $S \backslash S_u$; and the second condition ($g_r(s,a,z) \geq 0$) is sufficient to ensure σ to cancel the effect of uncertainties when approaching the safe-set boundary $\partial\mathcal{C}$. The availability of h and σ enables safe exploration in RL by the satisfaction of Condition (1). To this aim, we restrict the policy space to the convex policies solving the following QP at runtime for a safe action

$$\arg\min_a \; a^T Q a + p(s)^T a \tag{3}$$

$$\text{s.t.} \;\; \dot{h}_n(s,a) - \sigma(s) \geq -\alpha(s) h(s) \tag{4}$$

where Q is a fixed cost matrix and the state-dependent term $p(s) \sim \mathcal{N}(\mu_\theta, 1)$ is produced by a stochastic policy with learned mean μ_θ, trainable by policy gradient algorithms.

B. Abstraction for Learning RCBF

So far we assumed h and σ to be given and valid, which is difficult to prove in general for complex systems such as the multi-agent setting we consider. Here, we define the system abstraction necessary to learn and verify a valid RCBF.

Definition II.2. The abstraction $(S, S_i, S_u, A, Z, f, g, f_z)$ is a tuple which ensures to robustly include the actual dynamics. Formally, we assume that for all s, a:

$$\dot{s} \in f(s) + g(s)a + f_z(s,z), \;\; z \in Z$$

so that a valid RCBF remains valid in the actual system.

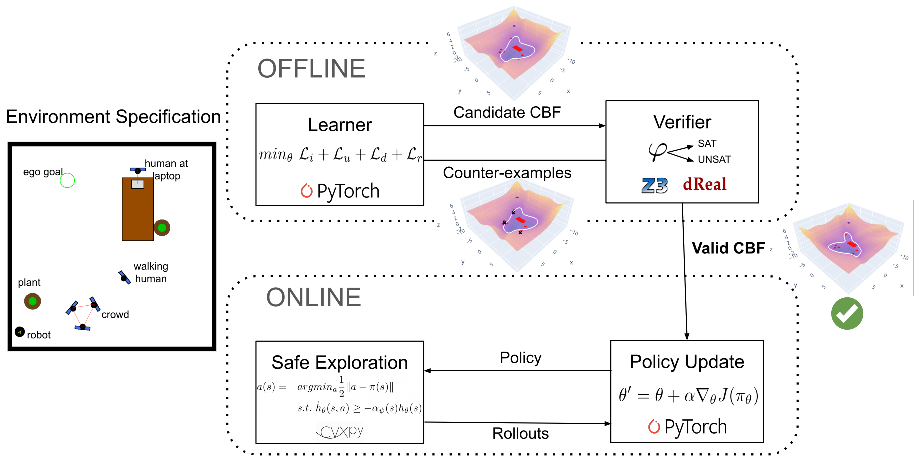
Environment Specification
ego goal
human at laptop
walking human
plant
crowd
robot
OFFLINE
Learner
$min_\theta\ \mathcal{L}_i + \mathcal{L}_u + \mathcal{L}_d + \mathcal{L}_r$
PyTorch
Candidate CBF
Counter-examples
Verifier
SAT
UNSAT
Z3
dReal
Valid CBF
ONLINE
Safe Exploration
$a(s) = \quad argmin_a \frac{1}{2}\|a - \pi(s)\|$
$s.t.\ \dot{h}_\theta(s,a) \geq -\alpha_\psi(s) h_\theta(s)$
CVXPY
Policy
Rollouts
Policy Update
$\theta' = \theta + \alpha \nabla_\theta J(\pi_\theta)$
PyTorch

Fig. 1. Framework architecture.

C. *Verifying RCBF*

The availability of the multi-agent system abstraction allows us to verify the validity conditions of RCBF with a formal verifier. To this aim, we translate the candidate neural models for h_ψ, σ_ϕ into symbolic representations as in [1].

Using a verifier, we look for any of these counterexamples:

- An initial state with negative CBF value.
- An unsafe state with non-negative CBF value.
- An state with non-negative CBF value for which the CBF condition (1) is unfeasible.
- An state and agent parameter for which the compensator robustness condition (2) is violated.

Each of these conditions can be formalized in a tractable quantifier-free formulation, which enables the use of efficient SMT solvers such as `Z3` and `dReal`. If no counterexample is found, then h, σ define a valid RCBF. Otherwise, we add the counterexamples to the training data and refine the models.

D. *Learning RCBF*

Using the multi-agent system abstraction, we can build an initial training distribution by randomly sampling the set of states, input and uncertain variables. These data are generated with little cost because we do not execute the actual systems. The training dataset D is composed by 4 batches of k samples: samples of initial states D_i, unsafe state D_u, unfeasible transitions D_f and non-robust transitions D_r.

From the validity conditions of RCBF, we define a smooth loss to quantify their violation over the training data:

$$\mathcal{L}_i = \sum_{s_i \in D_i} \mathrm{SP}(-h_\psi(s_i)),\ \mathcal{L}_u = \sum_{s_u \in D_u} \mathrm{SP}(h_\psi(s_i))$$
$$\mathcal{L}_f = \sum_{(s,a) \in D_f} \mathrm{SP}(-g_f(s)),\ \mathcal{L}_r = \sum_{(s,a,z) \in D_f} \mathrm{SP}(-g_r(s,a,z))$$

where $\mathrm{SP}(\cdot)$ is the Softplus. We then train candidate models of h_ψ and σ_ϕ using the joint loss: $\mathcal{L} = \mathcal{L}_i + \mathcal{L}_u + \mathcal{L}_f + \mathcal{L}_r$.

E. *Learning Adaptive Behaviors with RCBF*

Once we obtain a valid robust CBF from the previous offline stage, we can safely interact with the other actors and explore the environment. However, the learned CBF might be overly conservative in controlling the rate with which the agent is allowed to approach the safety boundary and might preclude performing agile maneuvers. To this aim, we jointly optimize a policy and an adaptive safety network as in [2] to allow (1) to overcome possible over-conservatism of CBF, and (3) to adapt the degree of conservatism based on the context. However, while in [2] we solve a cost-constrained optimization problem, here we ensure robust safety by construction of the RCBF and optimize for performance with a PPO algorithm.

III. FoSRL: Architecture

FoSRL is implemented in Python, and we provide Jupyter Notebook files for users to freely access and play around with the tool. Overall, there are two sections in FoSRL: one is to find a valid Control Barrier Function (called *Offline Counter-example guided Pretraining*), and another is to optimize a control policy subject to safety specifications through Reinforcement Learning using found CBF (called *Online Safe Interactive Learning*). Furthermore, FoSRL supports the verification and synthesis of robust CBF for uncertain systems. Here we introduce specific procedures of using FoSRL through a simple example.

different modules like system, uncertainty, how to extend them for users

Offline Counter-example guided Pretraining We first need to define a set of *symbolic assumptions* of the dynamics and uncertainty that our concerns system has. For all systems, the assumptions consist of the characterization of the state spaces, and domains for initial, unsafe and other states. Moreover, if we are considering an uncertain system, we need to define the uncertainty domain. To enable symbolic verification, each domain must be a symbolic set. We offer several implementations of common multi-dimensional sets, such as:

- Rectangle, Sphere,
- Union, Intersection, Complement,
- and some other special sets.

For instance, let's consider the following two-dimensional dynamical system, which we denote as *Single Integrator*:

$$\dot{x} = u_x, \dot{y} = u_y. \tag{5}$$

We consider an uncertain version of the above system with additive uncertainty as follows:

$$\dot{x} = u_x + z_x, \dot{y} = u_y + z_y. \tag{6}$$

These can be instantiated simply as follows:

```
# define control-affine dynamical system
system_id = "SingleIntegrator"
uncertainty_type = "AdditiveBounded"
system = make_system(system_id)()
system = add_uncertainty(uncertainty_type,
    system=system)
```

We can define the working space and domains of the system as the followings:

- State domain: Rectangle((-5.0, -5.0), (5.0, 5.0));
- Control input domain: Rectangle((-5.0, -5.0), (5.0, 5.0));
- Initial state domain: Complement(Rectangle((-4.0, -4.0), (4.0, 4.0)));
- Unsafe state domain: Sphere((0.0, 0.0), 1.0);
- Uncertainty domain: Sphere((0.0, 0.0), 1.0).

The above domains can be defined as follows:

```
# define state domains and input domains
# and initial, unsafe, lie state domains
domains = system.domains
```

Then we can define the numerical datasets (for the learning).

```
# data generator for datasets of initial
    state, unsafe state, lie state, and
    uncertainty variable
from fosco.common.consts import DomainName
    as dn

data_gen = {
   'init': lambda n:
       domains[dn.XI.value].generate_data(n),
   'unsafe': lambda n:
       domains[dn.XU.value].generate_data(n),
   'lie': lambda n: torch.concatenate([
      domains["lie"].generate_data(n),
      domains["input"].generate_data(n),
      domains["uncertainty"].generate_data(n),
   ], dim=1
   ),
   'uncertainty': lambda n:
       torch.concatenate([
      domains["lie"].generate_data(n),
      domains["input"].generate_data(n),
      domains["uncertainty"].generate_data(n),
   ],
   dim=1,
   )
}
```

Before we start training, some configurations of the pre-training and hyper-parameters should be setup:

```
   config = CegisConfig(
   SEED=seed,         # the seed for
       reproducibility
   CERTIFICATE="rcbf", # the type of
       certificate, either cbf or rcbf
   VERIFIER="z3",     # the type of verifier,
       either z3 or dreal
   ACTIVATION=["htanh"], # the activation of
       the i-th hidden layer
   N_HIDDEN_NEURONS=[20], # the nr of
       neurons of the i-th hidden layer
   CEGIS_MAX_ITERS=20, # the maximum number
       of iterations
   N_DATA=5000,       # the nr of samples in
       each training dataset
   RESAMPLING_N=100, # the nr of points to
       sample around each counter-example
   RESAMPLING_STDDEV=0.1, # the std
       deviation to sample around each
       counter-example
   LOSS_WEIGHTS={    # the weights for each
       loss term
      'init': 1.0,
      'unsafe': 1.0,
      'lie': 1.0,
      'robust': 1.0,
      'conservative_b': 1.0,
      'conservative_sigma': 0.1
   },
)
```

Finally we can find a valid (robust) CBF:

```
cegis = Cegis(
   system=system,
   domains=domains,
   config=config,
   data_gen=data_gen,
   verbose=verbosity
)

result = cegis.solve()
```

Given a candidate CBF h, we use a verifier to check its validity. In particular, we aim to find any counter-example to the following four conditions:
Initial condition

$$\exists s : s \in D_i \ \wedge \ h(s) < 0 \tag{7}$$

Unsafe condition

$$\exists s : s \in D_u \ \wedge \ h(s) > 0 \tag{8}$$

Feasibility condition

$$\exists s : s \in S \ \wedge \ \forall a : \dot{h}_n(s,a) - \sigma(s,a) + \alpha(h(s)) < 0 \tag{9}$$

Under the assumption that the input domain A is a convex-hull delimited by vertices $v_1, ..., v_k$, this condition can be rewritten in a more tractable formulation:

$$\exists s : s \in S \ \wedge \bigwedge_{v_1,\dots,v_k} \dot{h}_n(s,v) - \sigma(s,v) + \alpha(h(s)) < 0 \tag{10}$$

Robustness condition

$$\exists s, z, a : x \in S \ \wedge \ z \in Z \ \wedge \ a \in A \ \wedge \qquad (11)$$

$$\dot{h}_n(s, a) - \sigma(s, a) + \alpha(h(s)) \geq 0 \ \wedge \qquad (12)$$

$$\dot{h}(s, a, z) + \alpha(h(s)) < 0 \qquad (13)$$

Remark. It is worth noting that sometimes it is not easy to construct a valid CBF and the verifier (such as `Z3` and `dReal`) keeps finding counter-examples. In this case, the learner-verifier framework fails to learn a valid CBF and will stop after some iterations.

Online Safe Interactive Learning After obtaining a valid robust safety certificate, we can now proceed with the policy training. Specifically, we restrict the reinforcement learning exploration satisfying the CBF condition to ensure online safety. With FoSRL, the necessary code can be condensed to the followings:

```
from rl_trainer.safe_ppo.safeppo_trainer
    import SafePPOTrainer

trainer = SafePPOTrainer(envs=envs,
    config=config, barrier=result.barrier,
    compensator=result.compensator)

results = trainer.train(envs=envs)
```

where *envs* is any Gymnasium environment as commonly used in the RL community.

On the other hand, however, the learned CBF might be overly conservative in controlling the rate with which the agent is allowed to approach the safety boundary and might preclude performing agile maneuvers. To this aim, we jointly optimize a policy and an adaptive safety network as in [2] to

- overcome possible over-conservatism of CBF, and
- adapt the degree of conservatism based on the context.

The multi-head architecture for the agent has the following components:

$$\text{Representation model:} \qquad z_t = \phi_\eta(s_t) \qquad (14)$$

$$\text{Policy model:} \qquad p_t \sim \pi_\xi(\ \cdot \mid z_t) \qquad (15)$$

$$\text{Safety model:} \qquad \alpha_t \sim \alpha_\psi(\ \cdot \mid z_t) \qquad (16)$$

However, while in [2] we solve a cost-constrained optimization problem, here we ensure robust safety by construction of the RCBF and optimize for performance with a PPO algorithm.

IV. FoSRL: Library of Systems and Uncertainties

Dynamics We implemented single integrator and double integrator. We plan to include bicycle model in the future.

Uncertainties [Insert aaron ames's table?]

V. CONCLUSIONS

In this work, we presented an extension of the Adaptive Safe RL framework for multi-agent systems presented in [2]. We consider the setting with human-robot interaction and propose the first learner-verifier framework for RCBF to enable safe-exploration under other actors' uncertainty.

References

[1] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo, "Fossil: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pp. 1–11, 2021.

[2] L. Berducci, S. Yang, R. Mangharam, and R. Grosu, "Learning adaptive safety for multi-agent systems," in *2024 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2024. ArXiv:2309.10657.

		Method Summary	$\sigma(t,x,u)$
RCBF	cite	Bounded uncertainty: $\lVert\tilde{f}(t,x)\rVert \leq p$.	$\lVert\nabla h(x)\rVert p$
	cite	Bounded uncertainty: $\lVert\tilde{f}(t,x)\rVert \leq p$, continuous non-increasing κ with $\kappa(0)=1$	$\kappa(h(x))\lVert\nabla h(x)\rVert p$
	cite	$\tilde{f}$ is a convex hull of functions $\psi_i(x), i=1,\cdots,q$, $\tilde{g}$ is a convex hull of functions $\rho_i(x), i=1,\cdots,q$,	$-\min_{i\in\{1,..,q\}} \nabla h(x)\phi_i(x)$ $-\min_{i\in\{1,..,q\}} \nabla h(x)\rho_i(x)u$
	cite	$[\tilde{f}, (\tilde{g}\ \mathrm{diag}(u))^\top]^\top = \psi(x,u)\theta$, and $\exists A, b$ s.t. $A\theta \leq b$	$\inf\limits_{A\theta\leq b} \nabla h(x)\psi(x,u)\theta$
	cite	Sector bounded nonlinear perturbation at input, i.e., $\exists\alpha,\beta$ defining $g_\mathrm{s} = \frac{\alpha+\beta}{2}g$ and $\theta = \frac{\beta-\alpha}{\beta+\alpha}$.	$\left(L_g h(x) - L_{g_\mathrm{s}}h(x)\right)u + \theta\lVert u\rVert\lVert L_{g_\mathrm{s}}h(x)\rVert$
	[?]	$\hat{f}$ estimates $\tilde{f}$ and b_d defines an error band.	$-\nabla h(x)\hat{f}(x) + b_d(t)$
	cite	$b(t,x) = L_{\tilde{f}}h(t,x)$ is Lipschitz in x (constant L_b), $\hat{b}$ estimates b and k_b is estimation gain.	$-\hat{b}(t,x) + L_b/k_b$
	[?]	$L_{\tilde{f}}h(t,x) = \rho(t,x)\theta$, $\lVert\theta\rVert \leq \overline{\theta}$ $\hat{\theta}$ estimates θ and $\tilde{\theta}_U$ is upper error bound.	$\min\left\{\lVert\rho(t,x)\rVert\overline{\theta}, -\rho(t,x)\hat{\theta}(t) + \lVert\rho(t,x)\rVert\tilde{\theta}_U(t)\right\}$
	[?]	$\tilde{f}$ and $\tilde{g}$ are Lipschitz in x (with $L_{\tilde{f}}, L_{\tilde{g}}$), $\exists N$ data x_i, u_i and $\dot{x}_i$ so $\tilde{F}_i = \dot{x}_i - f(x_i) + g(x_i)u_i$	$\sum\limits_{i=1}^{N}\left(\lambda_i^T\tilde{F}_i - \lVert\lambda_i\rVert(L_{\tilde{f}} + L_{\tilde{g}}\lVert u_i\rVert)\lVert x-x_i\rVert\right)$, λ_i are Lagrange multipliers.
	[?]	$\dot{h}_\mathrm{n}(t,x,u)=0$, $\dot{h}$ is Lipschitz in x and u (with L_x, L_u), $\exists N$ data x_i, u_i and $\dot{h}_i$	$\min\limits_{i\in[1\cdots N]}\left[-\dot{h}_i + L_x\lVert x-x_i\rVert + L_u\lVert u-u_i\rVert\right]$
ISSf	**[?]**	Bounded uncertainty, continuously differentiable $\epsilon > 0$ with $\epsilon r \geq 0, \forall r$	$\dfrac{\lVert\nabla h(x)\rVert^2}{\epsilon(h(x))}$

TABLE I

A BRIEF SUMMARY OF ROBUST CONTROL BARRIER FUNCTION (RCBF) AND INPUT-TO-STATE SAFETY (ISSF) BASED METHODS FOR ROBUST SAFETY-CRITICAL CONTROL, WITH THE CORRESPONDING σ TERM TO PROVIDE SAFETY WITH ROBUSTNESS AGAINST THE UNCERTAINTIES .