Safe Model-based Control from Signal Temporal Logic Specifications Using Recurrent Neural Networks

Wenliang Liu[1], Mirai Nishioka[2], Calin Belta[1]
[1]Boston University, MA, USA, [2]Commonwealth School, MA, USA

Introduction

Temporal logics are widely used as specification languages for robotic systems due to their rich expressivity and similarity to natural language. In this work, we apply model-based Reinforcement Learning (RL) to simultaneously learn the system dynamics and a control policy that satisfies a given Signal Temporal Logic (STL) task. Compared with existing approaches, our algorithm:

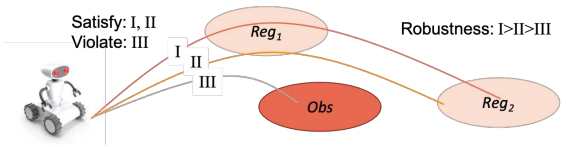
- requires no demonstrations (as opposed imitation learning);
- is data-efficient compared to model-free RL approaches;
- enables real-time execution after training;
- utilizes Control Barrier Function (CBF) to improve safety.

Signal Temporal Logic

STL, which is defined over real-valued signals, can express complex, time-related task specifications. Informally, an STL formula ϕ is made of:

- predicates: $l(x_t) \geq 0$, $x_t \in \mathbb{R}^n$;
- logical operators: $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT);
- temporal operators: $\mathbf{U}_I$ (until), $\mathbf{F}_I$ (eventually), $\mathbf{G}_I$ (always), where I is a time interval.

Qualitative semantics: whether a formula is satisfied;
Quantitative semantics (robustness): how strongly a formula is satisfied.
Example: $\mathbf{F}_{[0,5]} Reg_1 \wedge \mathbf{F}_{[5,10]} Reg_2 \wedge \mathbf{G}_{[0,10]} \neg Obs$.



Problem Formulation

Given:

- an STL formula φ with time horizon T;
- a control affine system with **unknown dynamcis** $x_{t+1} = f(x_t) + g(x_t)u_t$;
- a distribution p for the initial condition x_0,

find optimal parameters W^* for a Recurrent Neural Network (RNN) control policy $u_t = \pi(x_{0:t}, W)$ that maximizes the expected STL robustness $\rho(x_{0:T}, \varphi)$ over distribution p:

$$\begin{aligned} W^* &= \arg\max_W E_{p(x_0)}[\rho(\varphi, x_{0:T})] \\ \text{s.t. } & x_{t+1} = f(x_t) + g(x_t)\pi(x_{0:t}; W), t = 0, \ldots, T-1 \end{aligned} \tag{1}$$

Remark: We use RNN because memory is necessary for the controller [1].

Model-Based Safe Policy Search

System Model Learning:
We use a probabilistic model to reduce the effect of model bias, which is implemented as neural networks $\mathcal{F}$ and $\mathcal{G}$ with **dropout** layers [2]:

$$\hat{x}_{t+1} = \mathcal{F}(x_t; W_f, Z_f) + \mathcal{G}(x_t; W_g, Z_g)u_t,$$

- W_f, W_g: neural network parameters;
- Z_f, Z_g: dropout masks.

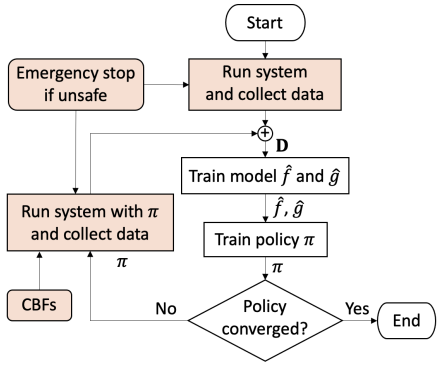
Run the system and collect transition data to simutaneously train the two neural networks. When running the system, **CBFs** are used **under the consideration of model uncertainty** to improve safety.

Control Policy Improvement:
Substitute the learned model to (1) and solve the optimization problem. In specific, we

- use the model to roll out the trajectory;
- compute the STL robustness and its gradient;
- update the policy using the gradient.

At each optimization step, sample a new set of initial states and system dropout masks. Use the mean value to approximate the expectation.

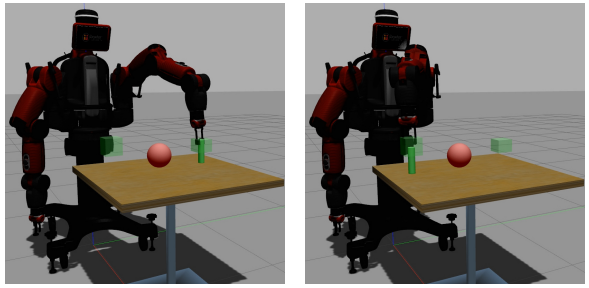
Model-Based Safe Policy Search:
Repeatedly train the system model and the control policy (called a cycle) until convergance.



Simulation Results

We simulated a Baxter robot, which has a 7 DOF manipulator. $x_t \in \mathbb{R}^{14}$ is the joint angles and velocities, $u_t \in \mathbb{R}^7$ is the torques. Consider the following STL formula for a Pick and Place task with obstacle avoidance:

$$\begin{aligned} \varphi_1 = & \left(\mathbf{F}_{[0,\frac{2.5}{\Delta t}]}(\mathbf{G}_{[0,\frac{0.5}{\Delta t}]}\mathbf{p} \in R_{pick})\right) \wedge \left(\mathbf{F}_{[\frac{3}{\Delta t},\frac{5.5}{\Delta t}]}(\mathbf{G}_{[0,\frac{0.5}{\Delta t}]}\mathbf{p} \in R_{place})\right) \\ & \wedge (\mathbf{G}_{[0,\frac{6}{\Delta t}]}\mathbf{q} \in V_{30}) \wedge (\mathbf{G}_{[0,\frac{6}{\Delta t}]}p_x > x_{body}) \wedge (\mathbf{G}_{[0,\frac{6}{\Delta t}]}p_z > h_{tb}) \wedge (\mathbf{G}_{[0,\frac{6}{\Delta t}]}\mathbf{p} \notin Obs) \end{aligned}$$



(a) Pick **(b)** Place

Testing results including success rate γ, unsafe rate β, mean robustness $\bar{\rho}$:

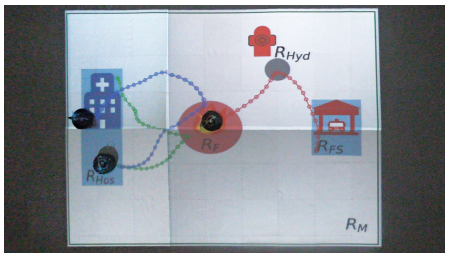
	w/ CBF			w/o CBF		
	γ	β	$\bar{\rho}$	γ	β	$\bar{\rho}$
Cycle 1	0%	10%	-0.4807	0%	24%	-0.7224
Cycle 2	2%	0%	-0.0973	14%	5%	-0.0880
Cycle 3	14%	0%	-0.0235	22%	0%	-0.0071
Cycle 4	83%	0%	0.0169	87%	0%	0.0192
Cycle 5	100%	0%	0.0401	100%	0%	0.0419

Testing results show that our algorithm achieves a high success rate, and the use of CBFs greatly reduces unsafe rate. Training costs 50 minutes, with 180 system runs (a small number compared with model-free methods)

Experiment Results

We used 3 ground vehicles in a fire fighting scenario:

$$\begin{aligned} \varphi_2 = & \wedge_{k=1}^{2}\left(\mathbf{F}_{[0,\frac{8}{\Delta t}]}(p^k \in R_F) \wedge \mathbf{F}_{[\frac{8}{\Delta t},\frac{15}{\Delta t}]}(p^k \in R_{Hos})\right) \wedge \mathbf{F}_{[0,\frac{10}{\Delta t}]}(p^3 \in R_{Hyd}) \\ & \wedge \mathbf{G}_{[\frac{10}{\Delta t},\frac{15}{\Delta t}]}(p^3 \in R_F) \wedge_{i,j\in\{1,2,3\},i\neq j} \mathbf{G}_{[0,\frac{15}{\Delta t}]}(\|p^i - p^j\|^2 \geq d^2) \wedge_{k=1}^{3} \mathbf{G}_{[0,\frac{15}{\Delta t}]}(p^k \in R_M). \end{aligned}$$



Scan for the video of the simulation and experiment:

- Collected 2372 transition data in 4 cycles of training (20 minutes), with the system running for about 13 minutes.
- Policy execution times: $0.01s$, which enables **real time control**.
- Success rate: 100%.

References

[1] W. Liu, N. Mehdipour, and C. Belta, "Recurrent neural network controllers for signal temporal logic specifications subject to safety constraints," *IEEE Control Systems Letters*, vol. 6, pp. 91–96, 2021.

[2] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.