Apprenticeship Learning with Temporal Logic Skills

Aniruddh G. Puranic, Jyotirmoy V. Deshmukh and Stefanos Nikolaidis

Appendix A
Signal Temporal Logic

Signal Temporal Logic (STL) is a real-time logic, generally interpreted over a dense-time domain for signals whose values are from a continuous metric space (such as $\mathbb{R}^n$). The basic primitive in STL is a *signal predicate* μ that is a formula of the form $f(\mathbf{x}(t)) > 0$, where $\mathbf{x}(t)$ is the tuple $(state, action)$ of the demonstration $\mathbf{x}$ at time t, and f maps the signal domain $\mathcal{D} = (S \times A)$ to $\mathbb{R}$. STL formulas are then defined recursively using Boolean combinations of sub-formulas, or by applying an interval-restricted temporal operator to a sub-formula. The syntax of STL is formally defined as follows: $\varphi ::= \mu \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{G}_I\varphi \mid \mathbf{F}_I\varphi \mid \varphi\mathbf{U}_I\varphi$. Here, $I = [a, b]$ denotes an arbitrary time-interval, where $a, b \in \mathbb{R}^{\geq 0}$. The semantics of STL are defined over a discrete-time signal $\mathbf{x}$ defined over some time-domain $\mathbb{T}$. The Boolean satisfaction of a signal predicate is simply *True* ($\top$) if the predicate is satisfied and *False* ($\bot$) if it is not, the semantics for the propositional logic operators $\neg, \wedge$ (and thus $\vee, \rightarrow$) follow the obvious semantics. The following behaviors are represented by the temporal operators:

- At any time t, $\mathbf{G}_I(\varphi)$ says that φ must hold for all samples in $t + I$.
- At any time t, $\mathbf{F}_I(\varphi)$ says that φ must hold *at least once* for samples in $t + I$.
- At any time t, $\varphi\mathbf{U}_I\psi$ says that ψ must hold at some time t' in $t + I$, and in $[t, t')$, φ must hold at all times.

Definition A.1 (Quantitative Semantics for Signal Temporal Logic)**.** Given an algebraic structure $(\oplus, \otimes, \top, \bot)$, we define the quantitative semantics for an arbitrary signal $\mathbf{x}$ against an STL formula φ at time t as in Table I.

TABLE I
Quantitative Semantics of STL

φ	$\rho(\varphi, \mathbf{x}, t)$
$true/false$	$\top/\bot$
μ	$f(\mathbf{x}(t))$
$\neg\varphi$	$-\rho(\varphi, \mathbf{x}, t)$
$\varphi_1 \wedge \varphi_2$	$\otimes(\rho(\varphi_1, \mathbf{x}, t), \rho(\varphi_2, \mathbf{x}, t))$
$\varphi_1 \vee \varphi_2$	$\oplus(\rho(\varphi_1, \mathbf{x}, t), \rho(\varphi_2, \mathbf{x}, t))$
$\mathbf{G}_I(\varphi)$	$\otimes_{\tau \in t+I}(\rho(\varphi, \mathbf{x}, \tau))$
$\mathbf{F}_I(\varphi)$	$\oplus_{\tau \in t+I}(\rho(\varphi, \mathbf{x}, \tau))$
$\varphi\mathbf{U}_I\psi$	$\oplus_{\tau_1 \in t+I}(\otimes(\rho(\psi, \mathbf{x}, \tau_1), \otimes_{\tau_2 \in [t, \tau_1)}(\rho(\varphi, \mathbf{x}, \tau_2)))$

A signal satisfies an STL formula φ if it is satisfied at time $t = 0$. Intuitively, the quantitative semantics of STL represent the numerical distance of "how far" a signal is away from the signal predicate. For a given requirement φ, a demonstration or policy d that satisfies it is represented as $d \models \varphi$ and one that doesn't is represented as $d \not\models \varphi$. In addition to the Boolean satisfaction semantics for STL, various researchers have proposed quantitative semantics for STL, [1], [2] that compute the degree of satisfaction (or *robust satisfaction values*) of STL properties by traces generated by a system. In this work, we use the following interpretations of the STL quantitative semantics: $\top = +\infty$, $\bot = -\infty$, and $\oplus = \max$, and $\otimes = \min$, as per the original definitions of robust satisfaction proposed in [1], [3].

The authors are with the Department of Computer Science, University of Southern California, USA. Email: {puranic, jdeshmuk, nikolaid}@usc.edu

Appendix B
Derivations and Proofs

Lemma B.1. *The optimal policy is invariant to affine transformations in the reward function.*

Proof Sketch. From [4], we have the definition of the Q function as follows, for the untransformed reward function R:

$$Q(s, a) \doteq \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \cdot R(s, a)_{t+k+1} | S_t = s, A_t = a\right] \quad (1)$$

$$Q(s, a) \doteq R(s, a) + \gamma \sum_{s'} P(s, a, s') \max_{a'} Q(s', a') \quad (2)$$

We consider two cases of reward function affine transformations in our work: (a) scaling by a positive constant and (b) shifting by a constant. In both these cases, our objective is to express the new Q function in terms of the original. Note that we abbreviate $R(s, a)$ to just R for simplicity.

Case (a): Scaling R by a positive constant: Let the scaled reward function be defined as $R' = c \cdot R, c > 0$. The new Q function is then

$$Q'(s, a) \doteq \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \cdot R'_{t+k+1} | S_t = s, A_t = a\right]$$
$$Q'(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \cdot c \cdot R_{t+k+1} | S_t = s, A_t = a\right]$$
$$Q'(s, a) = c \cdot \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \cdot R_{t+k+1} | S_t = s, A_t = a\right]$$
$$Q'(s, a) = c \cdot Q(s, a)$$

Thus we see that the new Q function scales with the scaling

From Equation 2 and by later substituting for Q' from the above result, we have,

$$Q'(s,a) \doteq R'(s,a) + \gamma \sum_{s'} P(s,a,s') \max_{a'} Q'(s',a')$$
$$c \cdot Q(s,a) = c \cdot R(s,a) + \gamma \sum_{s'} P(s,a,s') \max_{a'} (c \cdot Q(s',a'))$$
$$c \cdot Q(s,a) = c \cdot R(s,a) + c\gamma \sum_{s'} P(s,a,s') \max_{a'} \cdot Q(s',a')$$
$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s,a,s') \max_{a'} \cdot Q(s',a')$$

Thus the Bellman equation holds indicating that the policy is invariant to scaling by a positive constant.

Case (b): Shifting R by a constant: Let the shifted reward function be defined as $R' = R + c$. The new Q function is then

$$Q'(s,a) \doteq \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \cdot R'_{t+k+1} | S_t = s, A_t = a\right]$$
$$Q'(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \cdot (R_{t+k+1} + c) | S_t = s, A_t = a\right]$$
$$Q'(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \cdot R_{t+k+1} | S_t = s, A_t = a\right] + \sum_{k=0}^{\infty} \gamma^k c$$
$$Q'(s,a) = Q(s,a) + \frac{c}{1-\gamma}$$

Thus we see that the new Q values get shifted by the constant.

From Equation 2 and by later substituting for Q' from the above result, we have,

$$Q'(s,a) \doteq R'(s,a) + \gamma \sum_{s'} P(s,a,s') \max_{a'} Q'(s',a')$$
$$\begin{aligned} Q(s,a) + \frac{c}{1-\gamma} &= R(s,a) + c \\ &\quad + \gamma \sum_{s'} P(s,a,s') \max_{a'} \left(Q(s',a') + \frac{c}{1-\gamma}\right) \\ Q(s,a) + \frac{c}{1-\gamma} &= R(s,a) + c \\ &\quad + \gamma \sum_{s'} P(s,a,s') \max_{a'} Q(s',a') \\ &\quad + \gamma \sum_{s'} P(s,a,s') \frac{c}{1-\gamma} \\ Q(s,a) + \frac{c}{1-\gamma} &= R(s,a) + \gamma \sum_{s'} P(s,a,s') \max_{a'} Q(s',a') \\ &\quad + c + \frac{c\gamma}{1-\gamma} \end{aligned}$$
$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s,a,s') \max_{a'} \cdot Q(s',a')$$

Thus the Bellman equation holds indicating that the policy is invariant to shifting by a constant.

□

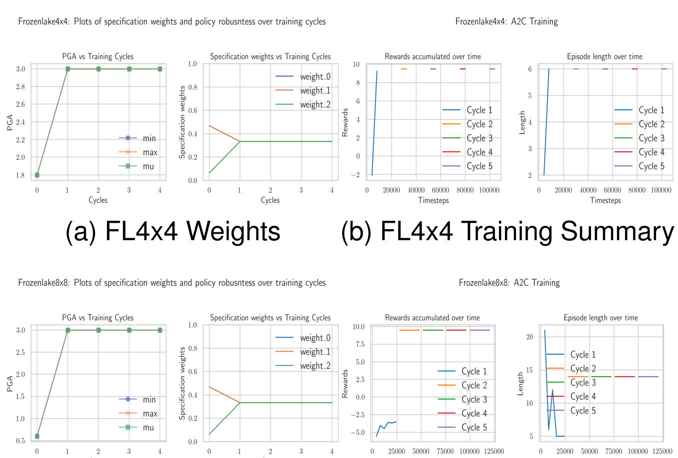
Therefore, any combination of scaling or shifting does not affect the optimal policy in our work. Similarly, the optimal policy is shown to be invariant towards reward shaping with potential functions [5].

Appendix C
Experiment Details

A. Task: Discrete-Space Frozenlake

We make use of the $Frozenlake$ (FL) deterministic environments from OpenAI Gym [6] that consist of a grid-world of sizes 4x4 or 8x8 with a reach-avoid task. Informally, the task specifications are (i) eventually reaching the goal, (ii) always avoid unsafe regions and (iii) take as few steps as possible. In these small environments $m = 5$ demonstrations of varying optimality are manually generated. We use A2C as the RL agent and show the training results in Fig. 1. The left figures show the statistics of the rollout PGAs and the evolution of weights over time. The right figures show the rewards accumulated and episode lengths.

We see from the left figures, that initially, the non-uniform weights of specifications correspond to the suboptimal demonstrations. And over time, the weights all converge to $1/3$ indicating that there are no edges in the final DAG, while the PGAs of rollouts from the final policy are maximum, as hypothesized. Since the environments are deterministic, the final policy achieve a 100% success rate. Since the task can be achieved even with IRL-based methods, we compare the amount of demonstrations required. Under identical conditions, the minimum number of demonstrations used by MCE-IRL are 50 for 4x4 grid and 300 for 8x8 grid. The algorithm in [7] uses over 1000 demonstrations in the 8x8 grid, even though they use temporal logic specifications similar to ours. *This clearly suggests that the choice of the reward inference algorithm plays a significant role in sample complexity*. This is due to the unsafe regions being scattered over the map, requiring the desirable *dense* features to appear very frequently.



(a) FL4x4 Weights (b) FL4x4 Training Summary

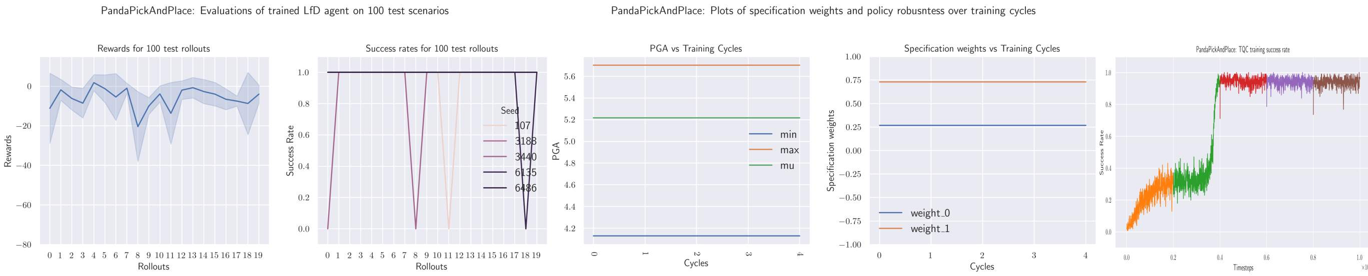
(c) FL8x8 Weights (d) FL8x8 Training Summary

Fig. 1. Results for the 4x4 and 8x8 Frozenlake environments.

Here, we provide details about on the hyperparameters used for reward and policy inference.

B. Task: Reaching Pose

The hyperparameters for both tasks: Panda-Reach and Needle-Reach, were nearly identicalTable II. The specifications for both these tasks are:



(a) Success rates on test trials (b) PGA and Weights (c) RL training summary

Fig. 2. Summary of training and evaluations for the Cube-Placing task.

1) Reaching the target pose: $\varphi_1 := \mathbf{F}(\|ee_{pose} - target_{pose}\| \leq \delta)$, where ee indicates the end-effector and δ is the threshold used to determine success. For Panda-Reach, $\delta = 0.2$ and for Needle-Reach, $\delta = 0.025$.
2) Reaching the target as quickly as possible: $\varphi_2 := \mathbf{G}(t <= 50)$, where t is the time when the end-effector reaches the target.

TABLE II
REACH TASK HYPERPARAMETERS.

Parameters	Values	
	Panda-Reach	Needle-Reach
# Demos	5	
Reward Model	Neural Network $[200 \rightarrow 200]$	
RL		
Model	SAC+HER	
Training Timesteps	$2 \cdot 10^5$	$2.5 \cdot 10^5$
# AL-STL Cycles	5	5
Policy Network	Shared $[64 \rightarrow 64]$	
Learning Rate	$3 \cdot 10^-4$	
Discount Factor γ	0.95	
Learning Starts	100	
Batch Size	256	
Polyak Update τ	0.005	
Training Success Rate	100%	
Test Success Rate	100%	

C. Task: Placing Cube

The hyperparameters are given in Table III. The specifications for both these tasks are:

1) Placing the cube at the target pose: $\varphi_1 := \mathbf{F}(\|cube_{pose} - target_{pose}\| \leq 0.05)$.
2) Reaching the target as quickly as possible: $\varphi_2 := \mathbf{G}(t <= 50)$, where t is the time when the end-effector reaches the target.

The statistics of the PGA shows that is maximum value is ≈ 6 since there are 2 specifications, each scaled by a factor of 3.

D. Task: Opening Door

The Panda robot uses operational space control to control the pose of the end-effector. The horizon for this task is 500

TABLE III
HYPERPARAMETERS FOR CUBE-PLACING TASK.

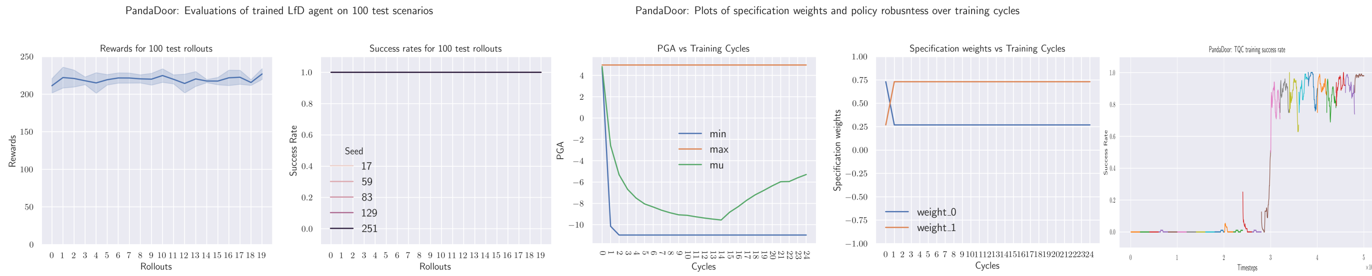
Parameters	Value
# Demos	5
Reward Model	Gaussian Process (Scale+RBF kernels)
RL	
Model	TQC+HER
Training Timesteps	10^7
# AL-STL Cycles	5
Policy Network	Shared $[512 \rightarrow 512 \rightarrow 512]$
Learning Rate	$1 \cdot 10^{-3}$
Discount Factor γ	0.95
Learning Starts	1000
Batch Size	2048
Polyak Update τ	0.05
Training Success Rate	98%
Test Success Rate	96%
Training Time	10.75 hours (2.15 hours/cycle)

and the control frequency is 20 Hz. The hyperparameters are given in Table IV. The specifications for both these tasks are:

1) Opening the door: $\varphi_1 := \mathbf{F}(\angle door\_hinge \geq 0.3)$. Angle is measured in radians.
2) Reaching the door handle: $\varphi_2 := \mathbf{F}(\|ee - door\_handle\| < 0.2)$; end-effector should be within $2cm$ of the door handle.

TABLE IV
HYPERPARAMETERS FOR DOOR-OPENING TASK.

Parameters	Value
# Demos	5
Reward Model	Neural Network $[16 \rightarrow 16 \rightarrow 16]$
RL	
Model	TQC
Training Timesteps	$5 \cdot 10^6$
# AL-STL Cycles	25
Policy Network	Shared $[256 \rightarrow 256]$
Learning Rate	$1 \cdot 10^{-3}$
Discount Factor γ	0.97
Learning Starts	100
Batch Size	256
Polyak Update τ	0.5
Training Success Rate	98%
Test Success Rate	100%
Training Time	6.5 hours (0.26 hours/cycle)



(a) Success rates on test trials (b) PGA and Weights (c) RL training summary

Fig. 3. Summary of training and evaluations for the Door-Opening task.

References

[1] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, 2009.

[2] S. Jaksic, E. Bartocci, R. Grosu, T. Nguyen, and D. Nickovic, "Quantitative monitoring of STL with edit distance," *Formal Methods in System Design*, 2018.

[3] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *FORMATS*, 2010.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., ser. Adaptive Computation and Machine Learning Series. Cambridge, MA: The MIT Press, 2018.

[5] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*. Morgan Kaufmann, 1999, pp. 278–287.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016.

[7] W. Zhou and W. Li, "Safety-aware apprenticeship learning," in *CAV*. Springer, 2018.