Proceedings of Machine Learning Research vol xxx:1–14, 2020

Tractable Reinforcement Learning of Signal Temporal Logic Specifications

Harish Venkataraman KUMAA001@UMN.EDU
Derya Aksaray DAKSARAY@UMN.EDU
Peter Seiler SEILE017@UMN.EDU
Aerospace Engineering and Mechanics, University of Minnesota, Minneapolis, MN, USA

Abstract

Signal temporal logic (STL) is an expressive language to specify most real world robotics tasks and safety specifications. Recently, there has been a great interest in learning optimal policies to satisfy STL specifications via reinforcement learning (RL). Learning to satisfy STL specifications often needs a sufficient length of state history to compute reward and next action. The need for history results in exponential state-space growth (Curse of history) for the learning problem. Thus the learning problem becomes computationally intractable for most real world problems. In this paper, we propose a compact means to capture state history in a new augmented state-space representation. An approximation to the objective of maximizing probability of satisfying STL specifications is proposed and solved for in the new augmented state-space. We show the performance bound of the approximate solution and compare the proposed technique with an existing technique via simulations.

Keywords: Reinforcement Learning (RL), Q-Learning, Signal Temporal Logic (STL)

1. Introduction

Reinforcement learning (RL) for controlling unknown or partially known stochastic dynamical systems to satisfy complex time bound objectives has gained good momentum in the robotics community. An example is learning least energy control policy for a partially known aerial vehicle to continually inspect a public infrastructure, like a bridge at load bearing regions of interest in some user specified fashion, while maintaining a safe distance and state of charge through out the mission to avoid any potential catastrophic collisions.

Followed by success of model-based control synthesis for satisfaction of temporal logic (TL) specifications (e.g., Aksaray et al. (2015); Ding et al. (2014); Sadigh et al. (2014); Lahijanian et al. (2015); Fu and Topcu (2014)), model-free learning paradigm to satisfy TL specifications has gained momentum. RL has been used to find policies that maximizes the probability of satisfying a given linear temporal logic (LTL) (e.g.,Brazdil et al. (2014); Sadigh et al. (2014); Fu and Topcu (2014); Li et al. (2017); Li and Belta (2019); Xu and Topcu (2019)). Q-learning, a variant of RL has also been shown to be successful in learning policies satisfying signal temporal logic specifications.[Aksaray et al. (2016)].

Signal temporal logic (STL), is a rich predicate logic that can specify bounds on physical parameters and time intervals Donzé and Maler (2010). For example, an autonomous UAV needs to recharge itself periodically (every 15 minutes in 2 hour mission) to avoid crashing onto the ground.

Such a time bound objective can be expressed by STL as : $G_{[0,105\ min]}F_{[0,15\ min]}f(x,y) \in \mathscr{C}$ where $f(x,y) \in \mathscr{C}$ refers to system states x and y to be inside recharging stations. STL is also endowed with a continuous quantitative metric of satisfaction known as robustness degree Fainekos et al. (2009). The robustness degree metric is a real number whose sign indicates the satisfaction/violation. Moreover, its magnitude quantifies the degree of satisfaction/violation. This metric enables formulating control synthesis problems that maximizes robustness degree.

STL has no naturally elegant graph theoretical way to book keep history and thus Aksaray et al. (2016) had to resort to learning on a higher dimensional MDP model known as τ-MDP, which stores finite window state history of length τ inclusive of the current state. τ-MDP state enables computing the reward and action at each time step using current state and history. For instance, if the specification requires visiting region B after region A within 10 time steps, then STL satisfaction can only be verified with the knowledge of last 10 time steps also know as the horizon. The decision to move towards B also depends on whether A was visited within the horizon. The number of states in the τ-MDP model grows exponentially with the size of τ. For example, if the original MDP has m states, τ-MDP has m^{τ} states. This state-space explosion renders learning on τ-MDP model impractical for real world robotics problems with large state-space and long STL horizons.

The primary focus of this paper is to provide a new augmented system on which learning to satisfy STL specifications is more computationally tractable and thus could scale to problems with longer STL horizons. The basic idea is that both rewards and actions can be computed without exact state history. The reward and the next action can be computed based on the current state and newly defined notion of flags. The flags capture the historic knowledge of the partial satisfaction with respect to each STL sub-formula constituting the STL specification. The new augmented system is defined as a new MDP known as F-MDP, which holds the actual system states and the flag states.

We solve the problem of enforcing the STL specifications by maximizing the probability of satisfaction. This problem can be solved approximately, by using F-MDP in place of τ-MDP[Aksaray et al. (2016)]. The reward and the action are computed using the current system state and the flag values constituting the F-MDP state. The proposed technique is shown to have polynomial space complexity as a function of τ. Empirical results also support faster learning due to compactness of F-MDP.

The rest of this paper is organized as follows: Sections 2 introduces the key concepts, Section 3 defines the problem formally, Section 4 describes the proposed technique in detail, Section 5 analyzes the performance and computational complexity of the proposed technique, Section 6 provides simulation results and finally Section 7 concludes with future prospects.

2. Preliminaries

2.1. Signal Temporal Logic (STL)

In this paper, the desired system behavior is described by an STL fragment with the following *syntax*

$$\begin{aligned}
\Phi &:= F_{[a,b]}\phi | G_{[a,b]}\phi \\
\phi &:= \phi \wedge \phi | \phi \vee \phi | F_{[c,d]}\varphi | G_{[c,d]}\varphi \\
\varphi &:= \psi | \neg\varphi | \varphi \wedge \varphi | \varphi \vee \varphi,
\end{aligned} \tag{1}$$

where $a,b,c,d \in \mathbb{R}_{\geq 0}$ are finite non-negative time bounds; Φ, ϕ, and φ are STL formulae; ψ is predicate in the form of $f(\mathbf{s}) < d$ where $\mathbf{s} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is a signal, $f : \mathbb{R}^n \to \mathbb{R}$ is a function, and

$d \in \mathbb{R}$ is a constant. The Boolean operators $\neg$, $\wedge$, and $\vee$ are negation, conjunction (i.e., *and*), and disjunction (i.e., *or*), respectively. The temporal operators F and G refer to *Finally* (i.e., eventually) and *Globally* (i.e., always), respectively.

For any signal $\mathbf{s}$, let s_t denote the value of $\mathbf{s}$ at time t and let $(\mathbf{s},t)$ be the part of the signal that is a sequence of $s_{t'}$ for $t' \in [t,\infty)$. Accordingly, the *Boolean semantics* of STL is recursively defined as follows:

$$\begin{array}{llllll}
(\mathbf{s},t) \models (f(\mathbf{s}) < d) & \Leftrightarrow & f(s_t) < d & \Leftrightarrow & f(s_t) < d = 1, \\
(\mathbf{s},t) \models \neg(f(\mathbf{s}) < d) & \Leftrightarrow & \neg\big((\mathbf{s},t) \models (f(\mathbf{s}) < d)\big) & \Leftrightarrow & f(s_t) < d = 0, \\
(\mathbf{s},t) \models \phi_1 \wedge \phi_2 & \Leftrightarrow & (\mathbf{s},t) \models \phi_1 \text{ and } (\mathbf{s},t) \models \phi_2 & \Leftrightarrow & \min((\mathbf{s},t) \models \phi_1, (\mathbf{s},t) \models \phi_2) = 1, \\
(\mathbf{s},t) \models \phi_1 \vee \phi_2 & \Leftrightarrow & (\mathbf{s},t) \models \phi_1 \text{ or } (\mathbf{s},t) \models \phi_2 & \Leftrightarrow & \max((\mathbf{s},t) \models \phi_1, (\mathbf{s},t) \models \phi_2) = 1, \\
(\mathbf{s},t) \models G_{[a,b]}\phi & \Leftrightarrow & (\mathbf{s},t') \models \phi \quad \forall t' \in [t+a,t+b] & \Leftrightarrow & \min\limits_{t' \in [t+a,t+b]} ((\mathbf{s},t') \models \phi) = 1, \\
(\mathbf{s},t) \models F_{[a,b]}\phi & \Leftrightarrow & \exists t' \in [t+a,t+b] \text{ s.t. } (\mathbf{s},t') \models \phi & \Leftrightarrow & \max\limits_{t' \in [t+a,t+b]} ((\mathbf{s},t') \models \phi) = 1.
\end{array}$$

where boolean 0 corresponds to $false$ and 1 corresponds to $true$. We assume 0 and 1 are real numbers and thus use all real number operations on them. e.g. max(0,1) = 1, min(0,0) = 0 are used assuming $0 < 1$.

For a signal $(\mathbf{s},0)$, i.e., the whole signal starting from time 0, satisfying $F_{[a,b]}\phi$ means that "there exists a time within $[a,b]$ such that ϕ will eventually be true", and satisfying $G_{[a,b]}\phi$ means that "ϕ is true for all times between $[a,b]$".

As in Dokhanchi et al. (2014), let $hrz(\phi)$ denote the *horizon* of an STL formula ϕ, which is the required number of samples to resolve any (future or past) requirements of ϕ. The horizon can be computed recursively as

$$\begin{array}{rcl}
hrz(\psi) & = & 0, \\
hrz(\phi) & = & b \qquad \text{if } \phi = G_{[a,b]}\psi \text{ or } F_{[a,b]}\psi, \\
hrz(F_{[a,b]}\phi) & = & b + hrz(\phi), \\
hrz(G_{[a,b]}\phi) & = & b + hrz(\phi), \\
hrz(\neg\phi) & = & hrz(\phi), \\
hrz(\phi_1 \wedge \phi_2) & = & \max(hrz(\phi_1), hrz(\phi_2)), \\
hrz(\phi_1 \vee \phi_2) & = & \max(hrz(\phi_1), hrz(\phi_2)),
\end{array}$$

where $a,b \in \mathbb{R}_{\geq 0}$, ψ is a predicate, and ϕ, ϕ_1, ϕ_2 are STL formulae.

STL formula Φ as defined in (1) allows at most three layers of nesting. Top layer Φ constitutes of just $F_{[a,b]}\phi$ or $G_{[a,b]}\phi$. Second layer ϕ constitutes of more than one STL fragment ϕ_i in conjunction using logical operators and their order of precedence of operation. Third layer φ_i in ϕ_i allows more than one predicate in conjunction using logical operators and their order of precedence of operation. i is index variable used to specify each STL fragment in the second layers of Φ. This formulation allows for specifying most complex time bound objectives or safety specifications, involving asymptotic and/or periodic behaviour. We call Φ as STL formula and its constituent ϕ_i as i^{th} STL sub-formula throughout the paper.

Example 1 *Consider the regions A and B illustrated in Fig. 1 and a specification as "visit regions A and B every 3 minutes along a mission horizon of 10 minutes". Note that the desired specification* *can be formulated in STL as*

$$\begin{aligned} \Phi = &\quad G_{[0,7]}\phi \\ \phi = &\quad F_{[0,3]}(s > 5 \wedge s < 6) \wedge F_{[0,3]}(s > 1 \wedge s < 2). \end{aligned} \tag{2}$$

The horizon of Φ *and* ϕ *are* $hrz(\Phi) = 10$ *and* $hrz(\phi) = 3$*, respectively. Let* $\phi = \phi_1 \wedge \phi_2$ *be made of* $\phi_1 = F_{[0,3]}\varphi_1$*, where* $\varphi_1 = (s > 5) \wedge (s < 6)$ *and* $\phi_2 = F_{[0,3]}\varphi_2$*, where* $\varphi_2 = (s > 1) \wedge (s < 2)$*. Then satisfying* Φ *implies satisfying* $\bigwedge_{t\in[0,7]} (F_{[t,t+3]}\varphi_1 \wedge F_{[t,t+3]}\varphi_2)$*.* Φ *is the STL formula with constituent sub-formulae* ϕ_1 *and* ϕ_2*. Let* $\mathbf{s^1}$ *and* $\mathbf{s^2}$ *be two signals as illustrated in Fig. 1. The signal* $\mathbf{s^1}$ *satisfies* Φ *because A and B are visited within* $[t, t+3]$ *for every* $t \in [0,7]$*. However, the signal* $\mathbf{s^2}$ *violates* Φ *because region B is not visited within* $[0,3]$*. Moreover, the robustness degree of* $\mathbf{s}$ *with respect to* Φ *can be computed via the quantitative semantics as follows:*

$$\min_{t\in[0,7]} \min\left\{ \max_{t'\in[t,t+3]} \{(\mathbf{s},t') \models \varphi_1\}, \max_{t'\in[t,t+3]} \{(\mathbf{s},t') \models \varphi_2\} \right\} \tag{3}$$

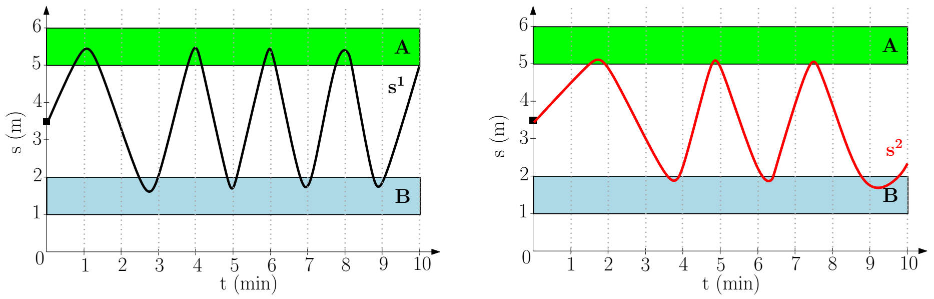

Figure 1: The specification is "visit regions A and B every 3 minutes along a mission horizon of 10 minutes", i.e., $\Phi = G_{[0,7]}(F_{[0,3]}(s > 5 \wedge s < 6) \wedge F_{[0,3]}(s > 1 \wedge s < 2))$, which is satisfied by signal $\mathbf{s^1}$ and violated by signal $\mathbf{s^2}$.

2.2. Markov Decision Process

Markov Decision Process (MDP) is a mathematical framework for representing discrete time, finite state and action space stochastic dynamical systems as $M = \langle \Sigma, A, P, R\rangle$, where Σ denotes the state-space, A denotes the action-space, $P : \Sigma \times A \times \Sigma \to [0,1]$ is the probabilistic transition relation, and $R : \Sigma \to \mathbb{R}$ is the reward function. If $\sigma \in \Sigma$ is the current state, then the next state $\sigma' \in \Sigma$ on taking an action $a \in A$ is determined using the probabilistic transition function P, also known as the model of the system. Given an initial state σ_0 and action sequence $a_{0:T-1}$, we donate the state sequence generated by M for T time steps as $\sigma_{0:T}$. Every state $\sigma \in \Sigma$ has a scalar measure of value given by the reward function R.

2.3. Reinforcement Learning: Q-learning

For systems with unknown stochastic dynamics, reinforcement learning can be used to design optimal control policies, i.e., the system learns how to take actions by trial and error interactions with the environment Sutton and Barto (1998).

Q-learning is an off-policy model-free reinforcement learning method [Watkins and Dayan (1992)], which can be used to find the optimal policy for a finite MDP. In particular, the objective of an agent at state σ_t is to maximize $V(\sigma_t)$, its expected (discounted) cumulative reward in finite or infinite horizon, i.e.,

$$E\Big[\sum_{k=0}^{T} R(\sigma_{k+t+1})\Big] \quad \text{or} \quad E\Big[\sum_{k=0}^{\infty} \gamma^k R(\sigma_{k+t+1})\Big] \tag{4}$$

where $R(\sigma)$ is the reward obtained at state σ, and γ is the discount factor. Also, $V^*(\sigma) = \max_a Q^*(\sigma,a)$, where $Q^*(\sigma,a)$ is the optimal Q-function for every state-action pair (σ,a).

Starting from state σ, the system chooses an action a, which takes it to state σ' and results in a reward $R(\sigma)$. Then, the Q-learning rule is defined as below:

$$Q(\sigma,a) := (1-\alpha)Q(\sigma,a) + \alpha[R(\sigma) + \gamma \max_{a^* \in A} Q(\sigma',a^*)], \tag{5}$$

where $\gamma \in (0,1)$ is the discount factor and $\alpha \in (0,1]$ is the learning rate. Accordingly, if each action $a \in A$ is repetitively implemented in each state $\sigma \in \Sigma$ for infinite number of times and α decays appropriately, then Q converges to Q^* with probability 1 (Tsitsiklis (1994)). Thus, we can find the optimal policy $\pi^* : \Sigma \to A$ as $\pi^* = \arg\max_a Q^*(\sigma,a)$. Algorithm 1 shows the steps of Q-learning.

Algorithm 1: Q-learning Sutton and Barto (1998)

Input: σ - current state
Output: π- control policy maximizing the sum of (discounted) rewards

```
1 : initialization: Arbitrary $Q(\sigma,a)$ and $\pi$;
2 :  for  $k = 1 : N_{episode}$
3 :     Initialize $\sigma$
3 :     for $t = 1 : T$
4 :         Select an action $a$ (via $\varepsilon$-greedy or $\pi$);
5 :         Take action $a$, observe $r$ and $\sigma'$;
6 :         $Q(\sigma,a) \leftarrow (1-\alpha_k)Q(\sigma,a) + \alpha_k\big[r + \gamma \max_{a'} Q(\sigma',a')\big]$;
7 :         $\pi(s) \leftarrow \arg\max_{a} Q(\sigma,a)$;
8 :         $\sigma \leftarrow \sigma'$;
9 :     end for
10 : end for
```

3. Problem Statement

In this paper, we assume that the dynamical system is abstracted as an MDP $M = \langle \Sigma, A, P, R \rangle$, where Σ denotes the set of partitions in the state-space, A is the set of motion primitives, and each motion primitive $a \in A$ drives the system from the centroid of a partition to the centroid of an adjacent partition. In real-world applications, many systems (e.g., robotic platforms) have uncertainty in their dynamics (e.g., uncertainty in actuation, gusts in the environment, noises in sensors). In this aspect, we assume that the transition probability function P is unknown in MDP M. This implies that when the system takes an action a at time t, it is not certain where it will be at time $t+1$. In other words, the probability distribution for σ_{t+1} is unknown. In this regard, Q-learning can be used to learn an optimal policy to satisfy an STL specification. Hence, a learning problem can be defined as follows:

Problem 1 (Maximizing Probability of Satisfaction) *Given an STL specification $\Phi = F_{[0,T]}\phi$ or $G_{[0,T]}\phi$ with a horizon $hrz(\Phi) = T$, a stochastic system model $M = \langle \Sigma, A, P, R \rangle$ with unknown P and an initial partial state trajectory $\sigma_{0:\tau}$ for $\tau = \left\lceil \frac{hrz(\phi)}{\Delta t} \right\rceil + 1$, find a control policy*

$$\pi_1^* = \arg\max_\pi Pr^\pi[\sigma_{0:T} \models \Phi] = \begin{cases} \arg\max_\pi E^\pi \left[\max_{t \in [\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi \right], & \text{if } \Phi = F_{[0,T]}\phi \\ \arg\max_\pi E^\pi \left[\min_{t \in [\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi \right], & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{6}$$

where $Pr^\pi[\sigma_{0:T} \models \Phi]$ is the probability of $\sigma_{0:T}$ satisfying Φ under policy π. Pr, E are short hand notations for probability and expectation respectively.

However, the authors in Aksaray et al. (2016) has shown that the objective function in (6) is not in the standard form of Q-learning. Hence, they have proposed an approximation of Problem 1.

Problem 2 (Maximizing Approximate Probability of Satisfaction, Aksaray et al. (2016)) *Given an STL specification $\Phi = F_{[0,T]}\phi$ or $G_{[0,T]}\phi$ with a horizon $hrz(\Phi) = T$, a stochastic system model $M = \langle \Sigma, A, P, R \rangle$ with unknown P, known reward function R, and an initial partial state trajectory $\sigma_{0:\tau}$ for $\tau = \left\lceil \frac{hrz(\phi)}{\Delta t} \right\rceil + 1$, find a control policy*

$$\pi_2^* = \begin{cases} \arg\max_\pi E^\pi \left[\sum_{t=\tau-1}^{T} e^{\beta(\sigma_{t-\tau+1:t} \models \phi)} \right], & \text{if } \Phi = F_{[0,T]}\phi \\ \arg\max_\pi E^\pi \left[-\sum_{t=\tau-1}^{T} e^{-\beta(\sigma_{t-\tau+1:t} \models \phi)} \right], & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{7}$$

Solving Problem 2 required two changes to the original problem. First change was to define and learn on a new higher dimensional state-space. The new system is defined using a τ-MDP model $M^\tau = \langle \Sigma^\tau, A, P^\tau, R^\tau \rangle$, with states $\sigma_t^\tau = \sigma_{t-\tau+1:t}$. The second change was to use log-exp-sum[Chen and Chiang (2012)] approximation to reformulate the objective from its min/max form to summation form in accordance with the Q-learning.

τ-MDP model used in Aksaray et al. (2016) suffers from exponential state space growth with horizon τ, which limits its application to problems with small STL horizon (τ). Hence, we intend to solve the same log-sum-exp approximated objective function as shown in (7), in a new compact system representation.

4. Proposed Technique

The proposed technique is based on the definition of a new compact system representation for Q-learning, which is rich enough to capture the necessary history within a finite window horizon. The new representation should be sufficient to compute reward and action at each time step.

Let Φ be $G_{[0,T]}\phi$ or $F_{[0,T]}\phi$, where Φ is the STL formula with the syntax as given in (1). Suppose Φ has n STL sub-formulae ϕ_i with horizon $hrz(\phi_i) = \tau_i, \forall i \in [1,n]$. Then, we assign one discrete valued flag variable ($f_i \in \mathfrak{F}_i := \{k/(\tau_i - 1), k \in [0, \tau_1 - 1]\}$) for each ϕ_i and proceed with defining a flag state augmented MDP know as F-MDP, which captures current state and flags to test for satisfaction of each STL sub-formula ϕ_i. $\mathfrak{F}_i$ is the flag state set of the flag $f_i, i \in [1,n]$

Definition 1 (F-MDP) *Given MDP $M = (\Sigma, A, P, R)$ and flag state sets $\mathfrak{F}_i, \forall i \in [1,n]$, an F-MDP is a tuple $M^F = (\Sigma^F, A, P^F, R^F)$, where*

- $\Sigma^F \subseteq (\Sigma \times \prod_{i=1}^{n} \mathfrak{F}_i)$ *is the set of finite states, obtained by the cartesian product between state set and all n flag state sets. Each state $\sigma^F \in \Sigma^F$ holds the current $\sigma \in \Sigma$ and $f_i \in \mathfrak{F}_i, \forall i \in [1,n]$.*
- $P^F : \Sigma^F \times A \times \Sigma^F \to [0,1]$ *is a probabilistic transition relation. Let $\sigma_k^F = \sigma_k, f_1, f_2, .., f_n$ and $\sigma_l^F = \sigma_l, f_1', f_2', .., f_n'$. $P^F(\sigma_k^F, a, \sigma_l^F) > 0$ if and only if $P(\sigma_k, a, \sigma_l) > 0$ and $f_i' = update(f_i, \sigma_k), \forall i \in [1,n]$. where $update(f_i, \sigma)$ is the flag update rule (8).*
- $R^F : \Sigma^F \to \mathbb{R}$ *is a reward function.*

Let $(\sigma_k^F = \sigma_k, f_1, f_2, .., f_n) \in (\Sigma^F = \Sigma, \mathfrak{F}_1, \mathfrak{F}_2..\mathfrak{F}_n)$ transition to a new state $(\sigma_l^F = \sigma_l, f_1', f_2', .., f_n') \in (\Sigma^F = \Sigma, \mathfrak{F}_1, \mathfrak{F}_2..\mathfrak{F}_n)$ on taking an action $a \in A$. σ_l depends on σ_k and the choice of a according to the transition relation P, but the flags f_i' are obtained from f_i and σ_k independent of the choice of action a as shown in the update rule:

$$f_i' = \begin{cases} 1 & \text{if } \phi_i = F_{[0,t]}\varphi_i \text{ and } \sigma_k \models \varphi_i \\ min(f_i - 1/(\tau_i - 1), 0) & \text{if } \phi_i = F_{[0,t]}\varphi_i \text{ and } \sigma_k \not\models \varphi_i \\ max(f_i + 1/(\tau_i - 1), 1) & \text{if } \phi_i = G_{[0,t]}\varphi_i \text{ and } \sigma_k \models \varphi_i \\ 0 & \text{if } \phi_i = G_{[0,t]}\varphi_i \text{ and } \sigma_k \not\models \varphi_i \end{cases} \quad \forall i \in [1,n] \tag{8}$$

where $max(a,b)$ represents the maximum of a and b, $min(a,b)$ represents the minimum of a and b and τ_i represents the horizon of ϕ_i. φ_i could be a predicate or composition of predicates using logical operators as defined in (1).

From the update rule, we can see how each flag f_i can only take on discrete values between 0 and 1 with a step size of $1/(\tau_i - 1)$. Number of states the flag f_i can take is τ_i and thus $\mathfrak{F}_i$, has a size equal to horizon τ_i of the STL sub-formula $\phi_i, \forall i \in [1,n]$.

Given a $\sigma^F = \sigma, f_1, f_2..f_n$, the satisfaction function $sat(\sigma^F, \phi)$ used to test for satisfaction of STL formula ϕ and it's constituent STL sub-formulae ϕ_i, is recursively defined as below:

$$sat(\sigma^F, \phi_i) = \begin{cases} 1 & \text{if } f_i > 0 \text{ or } \sigma \models \varphi_i \text{ for } \phi_i = F_{[0,t]}\varphi_i \\ 0 & \text{if } f_i = 0 \text{ and } \sigma \not\models \varphi_i \text{ for } \phi_i = F_{[0,t]}\varphi_i \\ 1 & \text{if } f_i = 1 \text{ and } \sigma \models \varphi_i \text{ for } \phi_i = G_{[0,t]}\varphi_i \\ 0 & \text{if } f_i < 1 \text{ or } \sigma \not\models \varphi_i \text{ for } \phi_i = G_{[0,t]}\varphi_i \end{cases} \quad \text{where } i \in [1,n]$$
$$sat(\sigma^F, \phi_j \wedge \phi_k) = min(sat(\sigma^F, \phi_j), sat(\sigma^F, \phi_k))$$
$$sat(\sigma^F, \phi_j \vee \phi_k) = max(sat(\sigma^F, \phi_j), sat(\sigma^F, \phi_k)) \tag{9}$$

where ϕ_j, ϕ_k can be STL sub-formulae or their conjunction using logical operators $\wedge, \vee$. e.g. If $\phi = ((\phi_1 \wedge \phi_2) \vee \phi_3)$, first current state σ^F is evaluated with respect to sub-formulae $\phi_i, \forall i \in [1,3]$. Then σ^F is evaluated with respect to $\phi_j \wedge \phi_k, j = 1, k = 2$. Finally σ^F is evaluated with respect to new $\phi_j \vee \phi_k$, where $\phi_j = \phi_1 \wedge \phi_2$ and $k = 3$.

The reward function R^F of the problem in M^F is defined as below:

$$r = \begin{cases} e^{\beta sat(\sigma^F,\phi)} & \text{if } \Phi = F_{[0,T]}\phi \\ -e^{-\beta sat(\sigma^F,\phi)} & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{10}$$

where sat(σ^F, ϕ) is the satisfaction function as defined in (9) and $\beta > 0$ is the log-sum-exp approximation constant.

The overview of the complete technique to solve (7) using F-MDP is shown below:
1) For any STL formula Φ in accordance with (1)(i.e., $G_{[0,T]}\phi$ or $F_{[0,T]}\phi$), create one flag per STL sub-formula ϕ_i in Φ and redefine the learning problem in a new flag state augmented state-space Σ^F which has new state dimensions corresponding to the flags.
2) Define the objective function such that the agent observes an immediate reward as a function of current state in Σ^F. After executing these steps, one can use standard Q-learning algorithm to find the optimal policy $\pi_3^* : \Sigma^F \to A$.

4.1. Theoretical Results

Let π_3^* be the optimal solution obtained by solving the following problem with Q-learning.

Problem 3 (Maximizing Approximate Probability of Satisfaction with F-MDP) *Let Φ be STL formula with the syntax in (1), made up of STL sub-formulae $\phi_i, \forall i \in [1,n]$. Let $T = hrz(\Phi)$, $\tau_i = \left\lceil \frac{hrz(\phi_i)}{\Delta t} \right\rceil + 1, \forall i \in [1,n]$ and $\tau = \max_{i \in [1,n]}(\tau_i)$. Given an unknown MDP M, and flag state sets $\mathfrak{F}_i, \forall i \in [1,n]$, F-MDP $M^F = \langle \Sigma^F, A, P^F, R^F \rangle$ can be constructed. Assume that initial τ-states $\sigma_{0:\tau-1}$ are given from which $\sigma^F_{\tau-1}$ can be obtained. Let $\beta > 0$ be a known approximation parameter. Find a control policy $\pi_3^* : \Sigma^F \to A$ such that*

$$\pi_3^* = \begin{cases} \arg\max_{\pi} E^{\pi}\left[\sum_{t=\tau-1}^{T} e^{\beta sat(\sigma_t^F,\phi)}\right], & \text{if } \Phi = F_{[0,T]}\phi \\ \arg\max_{\pi} E^{\pi}\left[-\sum_{t=\tau-1}^{T} e^{-\beta sat(\sigma_t^F,\phi)}\right], & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{11}$$

where $sat(\sigma^F, \phi)$ is the satisfaction function as defined in (9).

Given a STL formula Φ and ϕ with the syntax in (1), with $hrz(\Phi) = T$ and a state sequence $\sigma_{0:T}$ in $M = \langle \Sigma, A, P, R \rangle$. State sequence $\sigma^F_{\tau-1:T}$ in $M^F = \langle \Sigma^F, A, P^F, R^F \rangle$ can be computed using flag update rule (8). Then we can show

$$(\sigma_{t-\tau+1:t} \models \phi) = sat(\sigma_t^F, \phi) \quad \forall t \in [\tau-1, T]$$

which means that the optimal policy π_3^* of Problem 3 is the same as π_2^* of Problem 2.

The optimal policy π_1^* of Problem 1, can be related to π_3^* of Problem 3 by the following theorem.

Theorem 2 *Let Φ and ϕ be STL formula with the syntax in (1) such that $\Phi = F_{[0,.]}\phi$ or $\Phi = G_{[0,.]}\phi$. Let $hrz(\Phi) = T$. Assume that a partial state trajectory $s_{0:\tau-1}$ is initially given where $\tau = \left\lceil \frac{hrz(\phi)}{\Delta t} \right\rceil + 1$. For some $\beta > 0$ and $\Delta t = 1$[1], let π_1^* and π_3^* be the optimal policies obtained by solving Problems 1 and 3 respectively. Then,*

$$Pr^{\pi_1^*}[s_{0:T} \models \Phi] - \frac{1}{\beta}\log(T-\tau+2) \;\leq\; Pr^{\pi_3^*}[s_{0:T} \models \Phi] \;\leq\; Pr^{\pi_1^*}[s_{0:T} \models \Phi] \tag{12}$$

Proof Proof 1 in Appendix ■

Now we analyze the space complexity and learning rate of the proposed technique and compare it with that of the known τ-MDP technique as proposed in Aksaray et al. (2016) for solving (7).

For a given MDP M and STL formula Φ(1), the F-MDP state-space Σ^F over which the Q-learning is solved has $|\Sigma^F| = |\Sigma| \times \Pi_1^n |\mathfrak{F}_i|$ number of elements. Q-table used for Q-learning on F-MDP has $|\Sigma^F| \times |A|$ number of entries. τ-MDP on the other hand has $|\Sigma|^\tau$ elements in Σ^τ and $|\Sigma^\tau| \times |A|$ entries in it's Q-table. For most real world problems, it is safe to assume that both the number of states ($|\Sigma|$) and STL horizon ($hrz(\phi) = \tau$) is more than STL sub-formulae horizons ($hrz(\phi_i) = \tau_i, \forall i \in [1,n]$) and the number of STL sub-formulae (n) in ϕ respectively. The above reasoning shows how the proposed technique has polynomial growth of state-space with horizon τ, as compared to τ-MDP technique which has exponential growth of state-space with horizon τ.

Now let us look at the implication of smaller sized Q-table on learning rate. Q-learning on a smaller, compact Q-table is faster to converge to the optimal solution due to the need for the Q-learning algorithm to visit every state, action pair infinitely often, as a condition for convergence. Thus for problems with large STL horizon τ, the proposed technique has lesser Q-table entries and thus faster learning rate. Faster convergence to the optimal policy is observed in practice.

5. Simulation Results

In the following case studies, we consider a single agent moving in a discretized environment. The set of motion primitives at each state is $A = \{N, NW, W, SW, S, SE, E, NE, stay\}$. We model the motion uncertainty as in Fig. 2 where, for any selected feasible action in A, the agent follows the corresponding blue arrow with probability 0.93 or a red arrow with probability 0.023. Moreover, the resulting state after taking an infeasible action (i.e., the agent is next to a boundary and tries to move towards it) is the current state. All simulations were implemented in MATLAB and performed on a laptop with a quad core 2.4 GHz processor and 8.0 GB RAM.

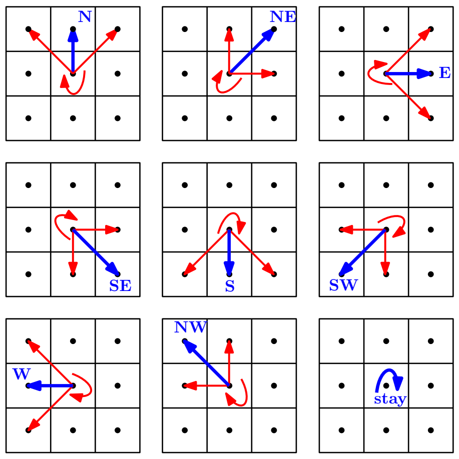

Figure 2: Motion uncertainty (red arrows) for an action (blue arrow).

5.1. Case Study 1: Reachability

1. $\Delta t = 1$ is selected due to clarity in presentation, but it can be any time step.

In this case study, the initial state of the agent is $s_0 = (1.5, 1.5)$ as shown in Fig. 3. We consider an STL formula defined over the environment as

$$\Phi_1 = F_{[0,6]} G_{[0,1]} (x > 4 \wedge y > 4), \tag{13}$$

which expresses "eventually visit the desired region within $[0,7]$. Note that $\Phi_1 = F_{[0,6]}\phi$ where $\phi = \phi_1 = G_{[0,1]}(x > 4 \wedge y > 4)$ and $hrz(\phi_1) = 1$. Moreover, we choose $\Delta t = 1$, thus $\tau_1 = 2$. The state-spaces of the system based on Fig. 3 are $|\Sigma| = 36$ and $|\Sigma^F| = 72$ since $\tau_1 = 2$.

To implement the Q-learning algorithm, the number of episodes is chosen as 2000 (i.e., $1 \le k \le 2000$), and we use the parameters $\beta = 50$, $\gamma = 0.9999$, and $\alpha_k = 0.95^k$. After 2000 trainings (episodes), which took approximately 2 minutes, the resulting policy π_3^* is used to generate 500 trajectories, which has

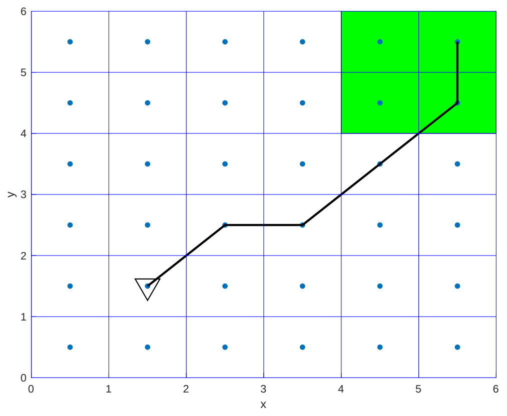

Figure 3: Sample trajectory generated by π_A^* eventually reaches the desired region in green.

$$Pr^{\pi_A^*}[s_{0:7} \models \Phi_1] = 0.996$$

Fig. 3 shows a sample trajectory generated by π_3^*. Φ_1 is satisfied in 498/500 sample trajectories.

5.2. Case Study 2: Patrolling

In the second case study, we consider an agent moving in an environment illustrated in Fig. 4(a). We consider an STL formula defined over the environment as

$$\Phi_2 = G_{[0,12]} \big(F_{[0,h]}(region\ A) \wedge F_{[0,h]}(region\ B) \big) \tag{14}$$

where *region A* represents $x > 1 \wedge x < 2 \wedge y > 3 \wedge y < 4$ and *region B* represents $x > 2 \wedge x < 3 \wedge y > 2 \wedge y < 3$. Note that Φ_2 expresses the following: "for all $t \in [0,12]$, eventually visit *region A* every $[t, t+h]$ and eventually visit *region B* every $[t, t+h]$". Note that $\Phi_2 = G_{[0,12]}\phi$ where $\phi = \phi_1 \wedge \phi_2$ with $\phi_1 = F_{[0,h]}(region\ A)$, $hrz(\phi_1) = h$ and $\phi_2 = F_{[0,h]}(region\ B)$, $hrz(\phi_2) = h$. Assuming that $\Delta t = 1$, $\tau = h + 1$ based on (**??**).

In this case study, we chose three values for $h := 2, 4, 5$ with rest of the parameters remaining the same. The sizes of the state-spaces are $|\Sigma| = 36, 36, 36$ and $|\Sigma^F| = 324, 900, 1296$[2] for each $\tau = 3, 5, 6$ respectively. To implement the Q-learning algorithm, the number of episodes is chosen as 10000 (i.e., $1 \le k \le 10000$), with $\beta = 50$, $\gamma = 0.9999$, and $\alpha_k = 0.95^k$. After 10000 trainings, the resulting policies π_A^* is used to generate 500 trajectories, which lead to

$$\begin{aligned} Pr^{\pi_A^*}[s_{0:14} \models \Phi_2] &= 0.6794 \quad \text{for} \quad h = 2 \\ Pr^{\pi_A^*}[s_{0:16} \models \Phi_2] &= 0.8237 \quad \text{for} \quad h = 4 \\ Pr^{\pi_A^*}[s_{0:17} \models \Phi_2] &= 0.8432 \quad \text{for} \quad h = 5 \end{aligned}$$

Sample trajectories generated by π_A^* is displayed in Fig. 4.

2. This indicates that there are $36 \times (h+1) \times (h+1)$ F-MDP states, 36 system states, $(h+1)$ f_1 flag states and $(h+1)$ f_2 flag states.

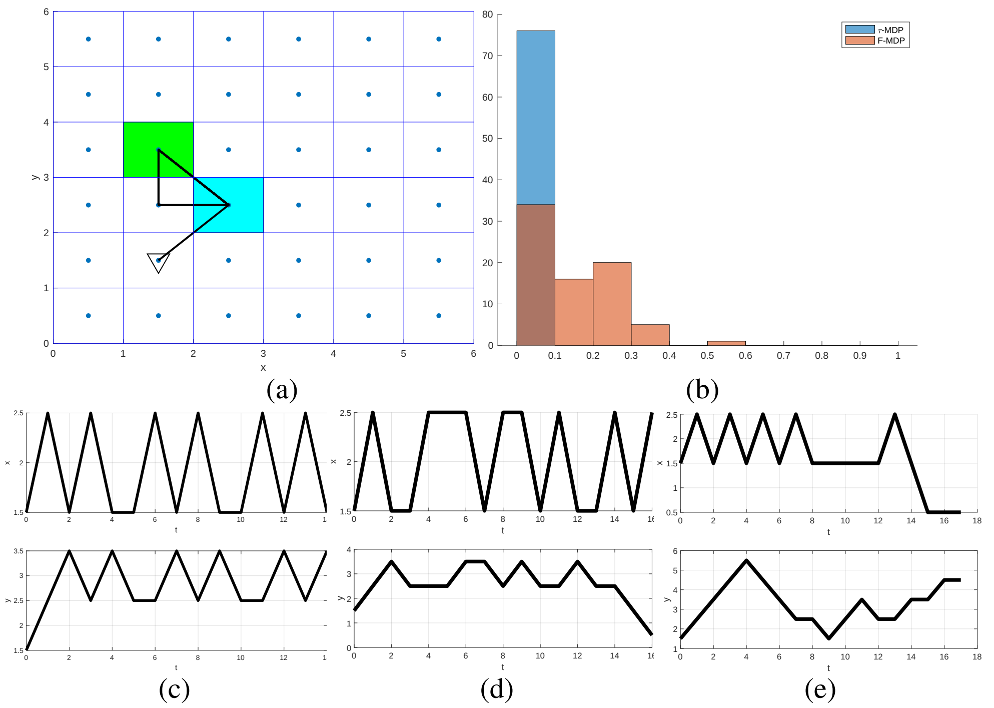

Figure 4: (a) Initial state and the desired regions (b) Distribution of probability of satisfaction of 75 different roll-outs with 10000 episodes each for $\tau = 3$ using F-MDP in orange and τ-MDP in blue (c) Sample trajectory generated by π_A^* for $\tau = 3$ (d) $\tau = 5$ (e) $\tau = 6$

The time and space requirements for 10000 episode trainings on F-MDP and τ-MDP for each of $\tau = 3, 5, 6$ are provided in the Table. 1. We can see that F-MDP state-space is very compact for large τ. Reduced space requirement also translates into faster execution time. Fig.4 part (b) is histogram plot of probability of satisfaction by 75 roll-outs with 10000 episodes each for $\tau = 3$ using F-MDP in orange and τ-MDP in blue. For this problem with $\tau = 3$, learning on τ-MDP required $10\times$ longer episode length to achieve similar return distribution as with the case of learning on F-MDP. Fig. 4 part (c), (d) and (e) shows sample trajectory generated by the optimal policy for $\tau = 3, 5$ and 6.

Table 1: Execution time (in minutes) and space requirements

Technique	$\tau = 3$	$\tau = 5$	$\tau = 6$
F-MDP	6	18	24
τ-MDP	21	290	$*$[†]

Technique	$\tau = 3$	$\tau = 5$	$\tau = 6$
F-MDP	$2.9\times e^{3}$	$8.1\times e^{3}$	$1.1\times e^{4}$
τ-MDP	$1.7\times e^{4}$[‡]	$1.0\times e^{6}$[‡]	$8.1\times e^{6}$[‡]

. †. Matlab run time error: preferred array size exceeded

. ‡. Pruned based on feasibility of transition

6. Conclusion

We have proposed a model-free control synthesis technique for stochastic dynamical systems with unknown model to satisfy complex time bound objectives specified as STL. We have used Q-

learning to find a control policy that enforces the desired STL formula by maximizing the approximate probability of satisfaction.

The proposed technique 1) remodels the system as a F-MDP where each state corresponds to current system state and flags which capture temporal distance of partial satisfaction with respect to each STL sub-formula, 2) approximates the objective of maximizing the probability of satisfaction by using log-sum-exp approximation rule and solves the learning problem using off the shelf Q-learning. We showed that the policy computed by the proposed technique can be sufficiently close to the optimal policy of the original problem when the approximation parameter is selected appropriately. Finally, we demonstrated the performance of the proposed technique on some case studies, and we observed that the proposed technique provided significant advantage in term of the space complexity and learning rate compared to existing techniques.

Future research includes incorporating complexity reduction techniques for the objective of maximizing expected robustness degree with respect to STL formula.

References

Derya Aksaray, Kevin Leahy, and Calin Belta. Distributed multi-agent persistent surveillance under temporal logic constraints. *IFAC-PapersOnLine*, 48(22):174–179, 2015.

Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6565–6570. IEEE, 2016.

Toma Brazdil, Krishnendu Chatterjee, Martin Chmelik, M.k, Vojtech Forejt, Jan Kretinsky, Marta Kwiatkowska, David Parker, and Mateusz Ujma. Verification of markov decision processes using learning algorithms. In Franck Cassez and Jean-Franois Raskin, editors, *Automated Technology for Verification and Analysis*, volume 8837 of *Lecture Notes in Computer Science*, pages 98–114. Springer International Publishing, 2014. ISBN 978-3-319-11935-9. doi: 10.1007/978-3-319-11936-6_8. URL http://dx.doi.org/10.1007/978-3-319-11936-6_8.

Minghua Chen and Mung Chiang. Distributed optimization in networking: Recent advances in combinatorial and robust formulations. In *Modeling and Optimization: Theory and Applications*, pages 25–52. Springer, 2012.

Xu Chu Ding, Stephen L Smith, Calin Belta, and Daniela Rus. Optimal control of markov decision processes with linear temporal logic constraints. *IEEE Trans. on Automatic Control*, 59(5):1244–1257, 2014.

Adel Dokhanchi, Bardh Hoxha, and Georgios Fainekos. On-line monitoring for temporal logic robustness. In *Runtime Verification*, pages 231–246. Springer, 2014.

Alexandre Donzé and Oded Maler. *Robust satisfaction of temporal logic over real-valued signals*. Springer, 2010. doi: 10.1007/978-3-642-15297-9_9. URL http://dx.doi.org/10.1007/978-3-642-15297-9_9.

Georgios E Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

Jie Fu and Ufuk Topcu. Probably approximately correct MDP learning and control with temporal logic constraints. *CoRR*, abs/1404.7073, 2014. URL http://arxiv.org/abs/1404.7073.

Morteza Lahijanian, Sean B. Andersson, and Calin Belta. Formal verification and synthesis for discrete-time stochastic systems. *IEEE Trans. on Automatic Control*, 6(8):2031–2045, 2015. doi: 10.1109/TAC.2015.2398883.

Xiao Li and Calin Belta. Temporal logic guided safe reinforcement learning using control barrier functions. *arXiv preprint*, 2019.

Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. *arXiv preprint*, 2017.

Dorsa Sadigh, Eric S Kim, Samuel Coogan, S Shankar Sastry, and Sanjit A Seshia. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *IEEE Conf. on Decision and Control*, pages 1091–1096. IEEE, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

John N Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine Learning*, 16 (3):185–202, 1994.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Zhe Xu and Ufuk Topcu. Transfer of temporal logic formulas in reinforcement learning. *arXiv preprint*, 2019.

7. Appendix

Log-Sum-Exp Approximation: Given n data points $x_1, x_2, ...x_n$ with the objective of finding maximum/minimum of the sequence. The maximum/minimum can be approximated as shown below:

$$\begin{aligned} \max(x_1,\dots,x_n) &\sim \frac{1}{\beta}\log\sum_{i=1}^{n} e^{\beta x_i}, \\ \min(x_1,\dots,x_n) &\sim -\frac{1}{\beta}\log\sum_{i=1}^{n} e^{-\beta x_i} \end{aligned} \tag{15}$$

where $\beta > 0$ is a constant and

$$\begin{aligned} \max(x_1,\dots,x_n) \le \frac{1}{\beta}\log\sum_{i=1}^{n} e^{\beta x_i} \le \max(x_1,\dots,x_n) + \frac{1}{\beta}\log n, \\ \min(x_1,\dots,x_n) - \frac{1}{\beta}\log n \le -\frac{1}{\beta}\log\sum_{i=1}^{n} e^{-\beta x_i} \le \min(x_1,\dots,x_n) \end{aligned} \tag{16}$$

with arbitrarily large β, the approximation becomes the exact solution.

Proof 1: Using Log-Sum-Exp approximation, we get

$$g(\boldsymbol{\sigma}) = \begin{cases} \max\limits_{t\in[\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi & \text{if } \Phi = F_{[0,T]}\phi \\ \min\limits_{t\in[\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \sim \hat{g}(\boldsymbol{\sigma}) = \begin{cases} \frac{1}{\beta}\log \sum\limits_{t=\tau-1}^{T} e^{\beta(\sigma_{t-\tau+1:t}\models\phi)} & \text{if } \Phi = F_{[0,T]}\phi \\ -\frac{1}{\beta}\log \sum\limits_{t=\tau-1}^{T} e^{-\beta(\sigma_{t-\tau+1:t}\models\phi)} & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{17}$$

Since $\log(.)$ is a strictly monotonous function and $1/\beta$ is a constant. $\widehat{g}(\boldsymbol{\sigma})$ with $\log(.)$ and $1/\beta$ dropped from $\hat{g}(\boldsymbol{\sigma})$ can substitute $\hat{g}(\boldsymbol{\sigma})$ in equation $\pi^* = \arg\max\limits_{\pi} E^{\pi}[\hat{g}(\boldsymbol{\sigma}))]$ to get equation (7).

Using (16) we can obtain the following inequality

$$E^{\pi}[\max_{t\in[\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi] \leq E^{\pi}\Big[\sum_{t=\tau-1}^{T} e^{\beta(I(\sigma_t^F,\phi))}\Big] \leq E^{\pi}[\max_{t\in[\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi] + \frac{1}{\beta}\log(T-\tau+2)$$
$$E^{\pi}[\min_{t\in[\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi] - \frac{1}{\beta}\log(T-\tau+2) \leq E^{\pi}\Big[-\sum_{t=\tau-1}^{T} e^{-\beta(I(\sigma_t^F,\phi))}\Big] \leq E^{\pi}[\min_{t\in[\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi] \tag{18}$$

In (18) we use policy $\pi = \pi_1^*, \pi_3^*$, which along with the following inequalities

$$E^{\pi_A^*}[\max_{t\in[\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi] \leq E^{\pi^*}[\max_{t\in[\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi] \tag{19}$$
$$E^{\pi_A^*}[\min_{t\in[\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi] \leq E^{\pi^*}[\min_{t\in[\tau-1,T]} \sigma_{t-\tau+1:t} \models \phi] \tag{20}$$
$$E^{\pi^*}\Big[\sum_{t=\tau-1}^{T} e^{\beta(I(\sigma_t^F,\phi))}\Big] \leq E^{\pi_A^*}\Big[\sum_{t=\tau-1}^{T} e^{\beta(I(\sigma_t^F,\phi))}\Big] \tag{21}$$
$$E^{\pi^*}\Big[-\sum_{t=\tau-1}^{T} e^{-\beta(I(\sigma_t^F,\phi))}\Big] \leq E^{\pi_A^*}\Big[-\sum_{t=\tau-1}^{T} e^{-\beta(I(\sigma_t^F,\phi))}\Big] \tag{22}$$

can be solved together to obtain (12).