Hierarchical Temporal Logic Guided Reinforcement Learning

Anonymous Authors[1]

Abstract

We present a framework that allows specification of tasks in terms of temporal logic (TL). Particularly, we allow tasks to be specified at multiple levels of spatio-temporal abstractions. Using the transformation between a TL formula and a finite state automaton, our method constructs a reward function from the hierarchical task specification along with a hierarchical policy that can be trained end-to-end. We show that our method is able to learn complex logical behaviors in both discrete and continuous state and action spaces.

1. Introduction

Reinforcement learning (RL) agents are capable of exploiting faulty reward functions (commonly referred to as reward hacking (Amodei et al., 2016)). As a result, undesirable behaviors that maximizes the given reward are produced (Clark & Amodei, 2016) (Lehman et al., 2018). Being able to accurately specify tasks in terms of reward functions is critical in reinforcement learning.

There are various means a reward function can be obtained. In addition to manually shaping a real valued function, the reward can be learned from human demonstrations as is commonly performed in inverse reinforcement learning (Abbeel & Ng, 2004). Learning rewards from human preferences (Christiano et al., 2017) has been shown to produce agent behaviors that are otherwise difficult to manually specify (simulated robotic agent learning to backflip). Additional methods include learning via agent debate (Irving et al., 2018) and natural language feedback (Chevalier-Boisvert et al., 2018), (Singh, 2017). The larger problem of designing evaluation metrics (rewards in the case of reinforcement learning) that align with the user's intentions is referred to as value alignment.

In this work, we focus on utilizing temporal logic (TL) as the task specification language to facilitate control policy learning. Temporal logic is a formal language capable of expressing logical relationships of propositions or predicates over time. It is suitable for defining high level tasks with complex logical structure (such as a traffic rule, a cooking recipe, etc). We find TL to be an effective means of incorporating high level prior knowledge into the learning agent.

A difficulty in defining tasks in temporal logic is the gap between symbolic high-level task representations and continuous low-level state features. For example, when defining the task of making a burger, we usually divide the task into a set of sub-tasks such as preheating the grill, oil the grate, etc. Then we go on to provide more detailed instructions for each of the sub-tasks. The robot's state space most likely consists of sensor readings and extracted features, while its action space is likely the motor commands. Making a direct connection between the symbolic tasks and the state and action space can be challenging. It would be much simpler to be able to define tasks in multiple levels of abstractions.

Our contribution in this work is to provide a hierarchical approach for task definition such that high-level tasks and low-level controls can be connected with layers of task abstractions. Even though the task may be time and history dependent (non-Markovian), the resulting system can be trained end-to-end with Markovian RL algorithms and a multi-level hierarchical policy can be obtained. Our method extends that of (Li et al., 2018) and shows that adding specification hierarchy not only reduces the complexity in bookkeeping for complex tasks, but also improves the sample efficiency during learning.

2. Related Work

Reward shaping (Ng et al., 1999) is a popular method that transforms sparse rewards to potential-based rewards while ensuring invariance of the optimal policy. Recent efforts in value alignment include iterated amplification (Christiano et al., 2018), cooperative inverse reinforcement learning (Hadfield-Menell et al., 2016), etc where the agent and human work together to learn the desired policy (possibly in an iterative process). Authors of (Leike et al., 2018) provide an up-to-date overview of recent progress in value alignment. Compared to these efforts, the method presented here

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

focuses on a simpler, more structured and deterministic way of specifying complex behaviors.

The combination of temporal logic with MDP has been studied by authors of (Giacomo et al., 2018) and (Camacho et al., 2017). The aim of their work is to solve the non-Markovian reward decision process (NMRD) by using temporal logics and automata. Our work differs mainly in the added benefit of handling hierarchical task specifications. We also introduce and use a notion of robustness as a continuous measure to relate the agent's state features with its progress towards satisfying the temporal logic specification as oppose to manually design the measure.

There is also a connection between our work and hierarchical learning. Here we distinguish between policy hierarchy and task hierarchy. The options framework (Sutton et al., 1999) learns a temporally abstracted hierarchical policy given a task. The task itself has no hierarchical structure. On the other hand, task hierarchy (such as the task graph in (Dietterich, 2000)) aims to divide complex tasks into simpler sub-tasks and either gradually increase the task difficulty as the agent learns or combine a library of simple policies to form more capable policies. Task hierarchy is commonly realized using curriculum learning (Bengio et al., 2009), multi-task learning (Andreas et al., 2017) and policy composition (van Niekerk et al., 2018). Our method combines both task and policy hierarchy by allowing the user to specify tasks at different levels of spatio-temporal abstraction (much like writing classes and functions in programming) and the resulting system generates a hierarchical policy that can be trained end-to-end. Policies for sub-tasks can be easily extracted at test time.

3. Preliminaries

We start with the definition of a Markov Decision Process and the general formulation of the reinforcement learning problem (Section 3.1). Then we introduce the syntax and semantics of scTLTL (Section 3.2.1) along with the definition of its equivalent FSA (Section 3.2.2). Section 3.2.3 shows how two FSAs can be combined together to form a product FSA. Finally, Section 3.3 presents a way to combine a MDP with a FSA such that a policy learned from the combined MDP is able to satisfy the scTLTL formula.

3.1. Reinforcement Learning

Definition 1. *An MDP is defined as a tuple $\mathcal{M} = \langle S, A, p(\cdot|\cdot,\cdot), r(\cdot,\cdot,\cdot)\rangle$, where $S \subseteq \mathbb{R}^n$ is the state space ; $A \subseteq \mathbb{R}^m$ is the action space (S and A can also be discrete sets); $p : S \times A \times S \to [0,1]$ is the transition function with $p(s'|s,a)$ being the conditional probability density of taking action $a \in A$ at state $s \in S$ and ending up in state $s' \in S$; $r : S \times A \times S \to \mathbb{R}$ is the reward function with $r(s,a,s')$ being the reward obtained by executing action a at state s and transitioning to s'.*

We define a task to be the process of finding the optimal policy $\pi^\star : S \to A$ (or $\pi^\star : S \times A \to [0,1]$ for stochastic policies) that maximizes the expected return, i.e.

$$\pi^\star = \arg\max_{\pi} \mathbb{E}^\pi[\sum_{t=0}^{T-1} r(s_t, a_t, s_{t+1})], \tag{1}$$

The horizon of a task (denoted T) is defined as the maximum allowable time-steps of each execution of π and hence the maximum length of a trajectory. In Equation (1), $\mathbb{E}^\pi[\cdot]$ is the expectation following π. The state-action value function is defined as

$$Q^\pi(s,a) = \mathbb{E}^\pi[\sum_{t=0}^{T-1} r(s_t, a_t, s_{t+1})|s_0 = s, a_0 = a] \tag{2}$$

i.e. it is the expected return of choosing action a at state s and following π onwards. For off-policy actor critic methods such as deep deterministic policy gradient (Lillicrap et al., 2015), Q^π is used to evaluate the quality of policy π. Parameterized $Q^\pi_{\theta_Q}$ and π_{θ_π} (θ_Q and θ_π are learnable parameters) are optimized alternately to obtain $\pi^\star_{\theta_\pi}$.

3.2. scTLTL and Finite State Automata

3.2.1. Syntactically Co-Safe Truncated Linear Temporal Logic (scTLTL)

We consider tasks specified with *syntactically co-safe Truncated Linear Temporal Logic* (scTLTL) which is derived from truncated linear temporal logic(TLTL) (Li et al., 2018). The syntax of scTLTL is defined as

$$\phi := \top \mid f(s) < c \mid \neg\phi \mid \phi \wedge \psi \mid \Diamond\phi \mid \phi\,\mathcal{U}\,\psi \mid \phi\,\mathcal{T}\,\psi \mid \bigcirc\phi \tag{3}$$

where $\top$ is the True Boolean constant. $s \in S$ is a MDP state in Definition 1; $f(s) < c$ is a predicate over the MDP states where $c \in \mathbb{R}$; $\neg$ (negation/not) and $\wedge$ (conjunction/and) are Boolean connectives. $\Diamond$ (eventually), $\mathcal{U}$ (until), $\mathcal{T}$ (then), $\bigcirc$ (next), are temporal operators.$\Rightarrow$ (implication) and and $\vee$ (disjunction/or) can be derived from the above operators.

We denote $s_t \in S$ to be the MDP state at time t, and $s_{t:t+k}$ to be a sequence of states (state trajectory) from time t to $t+k$, i.e., $s_{t:t+k} = s_t s_{t+1} ... s_{t+k}$. The Boolean semantics of scTLTL is defined as:

$$
\begin{aligned}
s_{t:t+k} \models f(s) < c \quad &\Leftrightarrow \quad f(s_t) < c,\\
s_{t:t+k} \models \neg\phi \quad &\Leftrightarrow \quad \neg(s_{t:t+k} \models \phi),\\
s_{t:t+k} \models \phi \Rightarrow \psi \quad &\Leftrightarrow \quad (s_{t:t+k} \models \phi) \Rightarrow (s_{t:t+k} \models \psi),\\
s_{t:t+k} \models \phi \wedge \psi \quad &\Leftrightarrow \quad (s_{t:t+k} \models \phi) \wedge (s_{t:t+k} \models \psi),\\
s_{t:t+k} \models \phi \vee \psi \quad &\Leftrightarrow \quad (s_{t:t+k} \models \phi) \vee (s_{t:t+k} \models \psi),\\
s_{t:t+k} \models \bigcirc\phi \quad &\Leftrightarrow \quad (s_{t+1:t+k} \models \phi) \wedge (k > 0),\\
s_{t:t+k} \models \Diamond\phi \quad &\Leftrightarrow \quad \exists t' \in [t, t+k)\ s_{t':t+k} \models \phi,\\
s_{t:t+k} \models \phi\ \mathcal{U}\ \psi \quad &\Leftrightarrow \quad \exists t' \in [t, t+k)\ s.t.\ s_{t':t+k} \models \psi\\
&\qquad \wedge (\forall t'' \in [t, t')\ s_{t'':t'} \models \phi),\\
s_{t:t+k} \models \phi\ \mathcal{T}\ \psi \quad &\Leftrightarrow \quad \exists t' \in [t, t+k)\ s.t.\ s_{t':t+k} \models \psi\\
&\qquad \wedge (\exists t'' \in [t, t')\ s_{t'':t'} \models \phi).
\end{aligned}
$$

A trajectory $s_{0:T}$ is said to satisfy formula ϕ if $s_{0:T} \models \phi$.

The quantitative semantics (also referred to as robustness) is defined recursively as

$$
\begin{aligned}
\rho(s_{t:t+k}, \top) &= \rho_{max},\\
\rho(s_{t:t+k}, f(s_t) < c) &= c - f(s_t),\\
\rho(s_{t:t+k}, \neg\phi) &= -\rho(s_{t:t+k}, \phi),\\
\rho(s_{t:t+k}, \phi \Rightarrow \psi) &= \max(-\rho(s_{t:t+k}, \phi), \rho(s_{t:t+k}, \psi))\\
\rho(s_{t:t+k}, \phi_1 \wedge \phi_2) &= \min(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)),\\
\rho(s_{t:t+k}, \phi_1 \vee \phi_2) &= \max(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)),\\
\rho(s_{t:t+k}, \bigcirc\phi) &= \rho(s_{t+1:t+k}, \phi)\ (k > 0),\\
\rho(s_{t:t+k}, \Diamond\phi) &= \max_{t' \in [t, t+k)} (\rho(s_{t':t+k}, \phi)),\\
\rho(s_{t:t+k}, \phi\ \mathcal{U}\ \psi) &= \max_{t' \in [t, t+k)} (\min(\rho(s_{t':t+k}, \psi),\\
&\qquad \min_{t'' \in [t, t')} \rho(s_{t'':t'}, \phi))),\\
\rho(s_{t:t+k}, \phi\ \mathcal{T}\ \psi) &= \max_{t' \in [t, t+k)} (\min(\rho(s_{t':t+k}, \psi),\\
&\qquad \max_{t'' \in [t, t')} \rho(s_{t'':t'}, \phi))),
\end{aligned}
$$

where ρ_{max} represents the maximum robustness value. A robustness of greater than zero implies that $s_{t:t+k}$ satisfies ϕ and vice versa ($\rho(s_{t:t+k}, \phi) > 0 \Rightarrow s_{t:t+k} \models \phi$ and $\rho(s_{t:t+k}, \phi) < 0 \Rightarrow s_{t:t+k} \not\models \phi$). The robustness is used as a measure of the level of satisfaction of a trajectory $s_{0:T}$ with respect to a scTLTL formula ϕ.

3.2.2. Finite State Automata (FSA)

Definition 2. *An FSA corresponding to a scTLTL formula ϕ. is defined as a tuple $\mathcal{A}_\phi = \langle \mathbb{Q}_\phi, \Psi_\phi, q_{\phi,0}, p_\phi(\cdot|\cdot), \mathcal{F}_\phi \rangle$, where $\mathbb{Q}_\phi$ is a set of automaton states; Ψ_ϕ is the input alphabet which is a set of simple logic formulas without temporal operators; $q_{\phi,0} \in \mathbb{Q}_\phi$ is the initial state; $p_\phi : \mathbb{Q}_\phi \times \mathbb{Q}_\phi \to [0,1]$ is a conditional probability defined as*

$$
\begin{gathered}
p_\phi(q_{\phi,j}|q_{\phi,i}) = \begin{cases} 1 & \psi_{q_{\phi,i},q_{\phi,j}} \text{ is true} \\ 0 & \text{otherwise.} \end{cases}\\
or \\
p_\phi(q_{\phi,j}|q_{\phi,i}, s) = \begin{cases} 1 & \rho(s, \psi_{q_{\phi,i},q_{\phi,j}}) > 0 \\ 0 & \text{otherwise.} \end{cases}
\end{gathered}
\tag{4}
$$

$\mathcal{F}_\phi$ is a set of final automaton states.

Here $q_{\phi,i}$ is the i^{th} automaton state of $\mathcal{A}_\phi$. $\psi_{q_{\phi,i},q_{\phi,j}} \in \Psi_\phi$ is the predicate guarding the transition from $q_{\phi,i}$ to $q_{\phi,j}$. Because $\psi_{q_{\phi,i},q_{\phi,j}}$ is a predicate without temporal operators, the robustness $\rho(s_{t:t+k}, \psi_{q_{\phi,i},q_{\phi,j}})$ is only evaluated at s_t. Therefore, we use the shorthand $\rho(s_t, \psi_{q_{\phi,i},q_{\phi,j}}) = \rho(s_{t:t+k}, \psi_{q_{\phi,i},q_{\phi,j}})$. The translation from a TLTL formula to a FSA can be done automatically with available packages like Lomap (Vasile, 2017). Examples of scTLTL formula and their corresponding FSAs are provided in Section 6.1.1.

3.2.3. Product Automata

It is possible to combine multiple FSAs into one by taking their product. Transitioning on the combined FSA can lead to satisfaction of conjunction or disjunction (depending on the termination condition) of the corresponding formula. We take inspiration from (Molnár & Vörös) and provide the definition of the product of two FSAs.

Definition 3. *Given $\mathcal{A}_{\phi_1} = \langle \mathbb{Q}_{\phi_1}, \Psi_{\phi_1}, q_{\phi_1,0}, p_{\phi_1}, \mathcal{F}_{\phi_1} \rangle$ and $\mathcal{A}_{\phi_2} = \langle \mathbb{Q}_{\phi_2}, \Psi_{\phi_2}, q_{\phi_2,0}, p_{\phi_2}, \mathcal{F}_{\phi_2} \rangle$ corresponding to formulas ϕ_1 and ϕ_2, the FSA of $\phi_\wedge = \phi_1 \wedge \phi_2$ is the product automaton of $\mathcal{A}_{\phi_1}$ and $\mathcal{A}_{\phi_1}$, i.e. $\mathcal{A}_{\phi_1 \wedge \phi_2} = \mathcal{A}_{\phi_1} \times \mathcal{A}_{\phi_2} = \langle \mathbb{Q}_{\phi_1 \wedge \phi_2}, \Psi_{\phi_1 \wedge \phi_2}, q_{\phi_1 \wedge \phi_2,0}, p_{\phi_1 \wedge \phi_2}, \mathcal{F}_{\phi_1 \wedge \phi_2} \rangle$ where $\mathbb{Q}_{\phi_1 \wedge \phi_2} = \mathbb{Q}_{\phi_1} \times \mathbb{Q}_{\phi_2}$ is the set of product automaton states, $q_{\phi_1 \wedge \phi_2,0} = (q_{\phi_1,0}, q_{\phi_2,0})$ is the product initial state, $\mathcal{F}_{\phi_1 \wedge \phi_2} = \mathcal{F}_{\phi_1} \cap \mathcal{F}_{\phi_2}$ are the final accepting states. Following Definition 2, for states $q_{\phi_1 \wedge \phi_2} = (q_{\phi_1}, q_{\phi_2}) \in \mathbb{Q}_{\phi_1 \wedge \phi_2}$ and $q'_{\phi_1 \wedge \phi_2} = (q'_{\phi_1}, q'_{\phi_2}) \in \mathbb{Q}_{\phi_1 \wedge \phi_2}$, the transition probability $p_{\phi_1 \wedge \phi_2}$ is defined as*

$$
p_{\phi_1 \wedge \phi_2}(q'_{\phi_1 \wedge \phi_2}|q_{\phi_1 \wedge \phi_2}) = \begin{cases} 1 & p_{\phi_1}(q'_{\phi_1}|q_{\phi_1}) \times \\ & p_{\phi_2}(q'_{\phi_2}|q_{\phi_2}) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5}
$$

Reaching $\mathcal{F}_{\phi_1 \wedge \phi_2}$ means satisfying $\phi_1 \wedge \phi_2$. We can also use $\mathbb{A}_{\phi_1 \wedge \phi_2}$ to find the termination state that satisfies $\phi_1 \vee \phi_2$ i.e. $\mathcal{F}_{\phi_1 \vee \phi_2} = \mathcal{F}_{\phi_1} \cup \mathcal{F}_{\phi_2}$. The product of multiple FSAs can be obtained in the same fashion iteratively.

3.3. FSA Augmented MDP

The FSA augmented MDP $\mathcal{M}_\phi$ (Li et al., 2018) establishes a connection between the TL specification and the standard reinforcement learning problem. A policy learned using $\mathcal{M}_\phi$ has implicit knowledge of the FSA through the automaton state $q_\phi \in \mathcal{Q}_\phi$.

Definition 4. *(Li et al., 2018) An FSA augmented MDP corresponding to FSA $\mathcal{A}_\phi = \langle \mathbb{Q}_\phi, \Psi_\phi, q_{\phi,0}, p_\phi(\cdot|\cdot), \mathcal{F}_\phi\rangle$ (constructed from scTLTL formula ϕ) and MDP $\langle S, A, p(\cdot|\cdot,\cdot), r(\cdot,\cdot,\cdot)\rangle$) is defined as $\mathcal{M}_\phi = \langle \tilde{S}_\phi, A, \tilde{p}_\phi(\cdot|\cdot,\cdot), \tilde{r}_\phi(\cdot,\cdot), \mathcal{F}_\phi\rangle$ where $\tilde{S}_\phi \subseteq S \times \mathbb{Q}_\phi$, $\tilde{p}_\phi(\tilde{s}'_\phi|\tilde{s}_\phi, a)$ is the probability of transitioning to $\tilde{s}'_\phi$ given $\tilde{s}_\phi$ and a,*

$$\begin{aligned}\tilde{p}_\phi(\tilde{s}'_\phi|\tilde{s}_\phi, a) &= p_\phi\big((s', q'_\phi)|(s, q_\phi), a\big)\\ &= \begin{cases} p(s'|s,a) & p_\phi(q\phi'|q_\phi, s) = 1\\ 0 & \textit{otherwise}.\end{cases}\end{aligned} \tag{6}$$

p_ϕ is defined in Equation (4). $\tilde{r}_\phi : \tilde{S}_\phi \times \tilde{S}_\phi \to \mathbb{R}$ is the FSA augmented reward function, defined by

$$\tilde{r}(\tilde{s}_\phi, \tilde{s}'_\phi) = \rho(s', D_\phi^{q_\phi}), \tag{7}$$

where $D_\phi^{q_\phi} = \bigvee_{q'_\phi \in \Omega_{q_\phi}} \psi_{q_\phi, q'_\phi}$ represents the disjunction of all predicates guarding the transitions that originate from q_ϕ (Ω_{q_ϕ} is the set of automata states that are connected with q through outgoing edges).

The reward in Equation (7) encourages transitioning out of the current q state and by repeatedly doing so eventually reach the final state. As a quick example, for the FSA in Figure (2c), $D_{\phi_2^f}^{q_{2,0}} = (\psi_C \wedge \neg\psi_D) \vee (\psi_C \wedge \psi_D) = \psi_C$, $D_{\phi_2^f}^{q_{2,1}} = \psi_D$. A policy $\pi^\star_\phi$ is said to satisfy ϕ if

$$\pi^\star_\phi = \arg\max_{\pi_\phi} \mathbb{E}^{\pi_\phi}[\mathbb{1}(\rho(s_{0:T}, \phi) > 0)]. \tag{8}$$

where $\mathbb{1}(\rho(s_{0:T}, \phi) > 0)$ is an indicator function with value 1 if $\rho(s_{0:T}, \phi) > 0$ and 0 otherwise.

4. Problem Formulation and Approach

We start by defining the following terms

- A scTLTL formula $\phi^f = f^f(\boldsymbol{\psi})$ is a flat formula if it is a TL formula over predicates ($f^f(\cdot)$ map a set of predicates to a scTLTL formula using the Boolean and temporal operators in Equation (3))
- The FSA $\mathcal{A}_{\phi^f} = \langle \mathbb{Q}_{\phi^f}, \Psi_{\phi^f}, q_{\phi^f,0}, p_{\phi^f}, \mathcal{F}_{\phi^f}\rangle$ is referred to as flat FSA (as in Definition 2)
- A scTLTL formula $\phi^h = f^h(\boldsymbol{\phi}^f)$ is a hierarchical formula if it is a TL formula over flat formula ($f^h(\cdot)$ maps a set of scTLTL formula to a new formula, $\boldsymbol{\phi}^f = \{\phi^f_1, ..., \phi^f_n\}$ is a set of flat formula).
- The FSA $\mathcal{A}_{\phi^h}$ is referred to as hierarchical FSA and is defined by $\mathcal{A}_{\phi^h} = \langle \mathbb{Q}_{\phi^h}, \Phi_{\phi^h}, q_{\phi^h,0}, p_{\phi^h}, \mathcal{F}_{\phi^h}\rangle$

We can see that compared to $\mathcal{A}_{\phi^f}$, the input alphabet of $\mathcal{A}_{\phi^h}$ are scTLTL formula instead of first order formula. In the definitions above, ϕ^f can be seen as sub-tasks and ϕ^h is a high level task. One can construct multiple layers of hierarchy (such as $\phi^{h_2} = f^h(\boldsymbol{\phi}^{h_1})$ where $\phi^{h_1} = f^f(\boldsymbol{\psi})$). Without loss of generality, we will focus on developing our method on 2 layers of hierarchy in this work.

Problem 1. *Given an MDP $\mathcal{M} = \langle S, A, p(\cdot|\cdot,\cdot), r(\cdot,\cdot,\cdot)\rangle$ with unknown transition dynamics $p(\cdot|\cdot,\cdot)$, a set of flat scTLTL formula $\boldsymbol{\phi}^f = \{\phi^f_1, ..., \phi^f_n\}$ and a hierarchical scTLTL formula $\phi^h = f^h(\boldsymbol{\phi}^f)$, find a policy $\pi^\star_{\phi^h}$ such that*

$$\pi^\star_{\phi^h} = \arg\max_{\pi_{\phi^h}} \mathbb{E}^{\pi_{\phi^h}}[\mathbb{1}(\rho(s_{0:T}, \phi^h) > 0)]. \tag{9}$$

where $\mathbb{1}(\rho(s_{0:T}, \phi^h) > 0)$ is an indicator function with value 1 if $\rho(s_{0:T}, \phi^h) > 0$ and 0 otherwise.

$\pi^\star_{\phi^h}$ in Equation (9) is said to satisfy ϕ^h. Problem 1 defines a policy search problem. Following the optimal policy $\pi^\star_{\phi^h}$ should result in trajectories that satisfy ϕ^h in expectation. We propose to solve Problem 1 with a hierarchical extension of the FSA augmented MDP (Definition 4). On a high level, because Φ_{ϕ^h} - the input alphabet of $\mathcal{A}_{\phi^h}$, is a set of scTLTL formula consisting of base formula ϕ^f connected by Boolean operators. Instead of only learning on the hierarchical FSA $\mathcal{A}_{\phi^h}$, we additionally construct a set of sub-FSA from Φ_{ϕ^h} by taking the product of the corresponding flat FSA $\mathcal{A}_{\phi^f}$ using Definition 3. We then add the sub-FSA to the original FSA augmented MDP. We show that by adding a small number of discrete dimensions to the state space, we can significantly facilitate learning of complicated tasks. More details will be provided in the following sections.

5. Hierarchical Automata Guided Reinforcement Learning

Given an MDP and a hierarchical scTLTL formula ϕ^h with its constituent flat formula $\boldsymbol{\phi}^f$, we can construct a FSA augmented MDP $\mathcal{M}_{\phi^h}$ using Definition 4. However, because Definition 4 is defined for a flat formula, we can no longer use the step-based Markovian reward in Equation (7). More specifically, $D_{\phi^h}^{q_{\phi^h}} = \bigvee_{q'_{\phi_h} \in \Omega_{q_{\phi^h}}} \phi_{q_{\phi^h}, q'_{\phi^h}}$ is no longer a flat formula of predicates. Comparing to the Definition of the D operator in Equation (7), we now have an scTLTL formula $\phi_{q_{\phi^h}, q'_{\phi^h}}$ guarding the outgoing edge $(q_{\phi^h}, q'_{\phi^h})$ instead of a predicate $\psi_{q_{\phi^h}, q'_{\phi^h}}$. If we were to use Equation (7) directly, we would end up with a history-dependent reward $\tilde{r}_t(\tilde{s}_{0:t}, \tilde{s}_{0:t+1}) = \rho(s_{0:t+1}, D^{q_{\phi^h}}_{\phi^h})$. This reward function can not be used with most reinforcement learning algorithms under the Markovian assumption.

Instead of trying to learn directly on $\mathcal{M}_{\phi^h}$, we design a Markovian reward by augmenting $\mathcal{M}_{\phi^h}$ with a set of sub-FSAs constructed from $\boldsymbol{\phi}^f$. Specifically, we introduce the hierarchical FSA augmented MDP as follows

Definition 5. *Given a hierarchical scTLTL formula $\phi^h = f^h(\boldsymbol{\phi}^f)$, the corresponding FSAs $\mathbb{A}_{\phi^h} = \langle \mathbb{Q}_{\phi^h}, \Psi_{\phi^h}, q_{\phi^h,0}, p_{\phi^h}(\cdot|\cdot), \mathcal{F}_{\phi^h}\rangle$ and $\mathbb{A}_{\boldsymbol{\phi}^f} = \{\mathbb{A}_{\phi^f_1}, ..., \mathbb{A}_{\phi^f_n}\}$ where $\mathbb{A}_{\phi^f_i} = \langle \mathbb{Q}_{\phi^f_i}, \Psi_{\phi^f_i}, q_{\phi^f_i,0}, p_{\phi^f_i}(\cdot|\cdot), \mathcal{F}_{\phi^f_i}\rangle$, $i \in \{1, 2, ..., n-1, n\}$. A hierarchical FSA augmented MDP is a tuple $\mathbb{M}^h_{\phi^h} = \langle \tilde{S}^h_{\phi^h}, A, \tilde{p}^h_{\phi^h}(\cdot|\cdot,\cdot), \tilde{r}^h_{\phi^h}(\cdot,\cdot), \mathcal{F}^h_{\phi^h}\rangle$ where $\tilde{S}^h_{\phi^h} \subseteq S \times \mathbb{Q}_{\phi^h} \times \mathbb{Q}_{\phi^f_1} \times \cdots \times \mathbb{Q}_{\phi^f_n}$, $\tilde{p}^h_{\phi^h}(\tilde{s}^{h'}_{\phi^h}|\tilde{s}^h_{\phi^h}, a)$ is the probability of transitioning to $\tilde{s}^{h'}_{\phi^h}$ given $\tilde{s}^h_{\phi^h}$ and a,*

$$\tilde{p}^h_{\phi^h} = \begin{cases} p(s'|s,a) & p_{\phi^h}(q'_{\phi^h}|q_{\phi^h}, s)\times \\ & \prod_{i=0}^{n} p_{\phi^f_i}(q'_{\phi^f_i}|q_{\phi^f_i}, s) = 1 \\ 0 & \textit{otherwise}. \end{cases} \tag{10}$$

p_ϕ is defined in Equation (4). Let $D^{q_{\phi^h}}_{\phi^h} = \bigvee_{q'_{\phi^h} \in \Omega_{q_{\phi^h}}} \phi_{q_{\phi^h}, q'_{\phi}}$ be similar to that in Definition 4 (instead of guarding predicates $\psi_{q_{\phi^h}, q'_{\phi}}$, we have guarding formula $\phi_{q_{\phi^h}, q'_{\phi}}$). Let $C^{q_{\phi^h}}_{\phi^h} = \bigwedge_{q'_{\phi^h} \in \Omega_{q_{\phi^h}}} \phi_{q_{\phi^h}, q'_{\phi}}$ be the conjunction alternative to $D^{q_{\phi^h}}_{\phi^h}$ (referred to as C^{q^h} to avoid clutter). Let $\boldsymbol{\phi}^f_{c^{q^h}} \subseteq \boldsymbol{\phi}^f$ be the set of flat formula in C^{q^h}, we define $\mathbb{A}_{C^{q^h}}$ to be the product automaton constructed from the FSAs of $\boldsymbol{\phi}^f_{c^{q^h}}$. Let $q^f_{C^{q^h}}$ be the q state of $\mathbb{A}_{C^{q^h}}$. $\tilde{r}^h_{\phi^h} : \tilde{S}^h_{\phi^h} \times \tilde{S}^h_{\phi^h} \to \mathbb{R}$ is the hierarchical FSA augmented reward function, defined by

$$\tilde{r}^h_{\phi^h}(\tilde{s}^h_{\phi^h}, \tilde{s}^{h'}_{\phi^h}) = \mathbb{1}\big(\rho(s', D^{q_{\phi^h}}_{\phi^h}) > 0\big) + w\rho(s', D^{q^f_{C^{q^h}}}_{C^{q^h}}). \tag{11}$$

Note that C^{q^h} is simply a logic formula of $\boldsymbol{\phi}^f_{c^{q^h}}$ with only Boolean operations. $\mathbb{A}_{C^{q^h}}$ can be obtained by taking the product of $\mathbb{A}_{\boldsymbol{\phi}^f_{c^{q^h}}}$ using Definition 3. The q state of $\mathbb{A}_{C^{q^h}}$ is therefore $\mathbb{Q}_{C^{q^h}} : \bigtimes_{\phi_i \in \boldsymbol{\phi}^f_{C^{q_h}}} \mathbb{Q}_{\phi_i}$ ($q^f_{C^{q^h}} \in \mathbb{Q}_{C^{q^h}}$).

Equation (11) is a weighted reward. Typically the weight w is set between 0 and 1 so to encourage progress in the high-level specification more than that in the lower-level spec (since the ultimate goal is to satisfy ϕ^h).

To train using the hierarchical FSA augmented MDP, simply provide the scTLTL task specifications ϕ^h (high-level spec) and $\boldsymbol{\phi}^f$ (low-level specs), the parameterized policy $\pi(\cdot|\theta_\pi)$ and value function $Q(\cdot|\theta_Q)$ (if necessary). Construct the $\mathbb{M}^h_{\phi^h}$ using Definition 5 and train using any suitable RL algorithm. During training and testing, transitions in the FSAs are tracked using Equation (10).

An optimal policy found using Definition 5 is guaranteed to satisfy the hierarchical specification $\phi^h = f^h(\boldsymbol{\phi}^f)$. The advantages of using the hierarchical FSA augmented MDP over the non-hierarchical version (Definition 4) includes the ease of specifying complex tasks using abstraction and improved learning performance (shown in the next section).

6. Experiments

In this section, we introduce our simulation experiment setup. Particularly, our method is evaluated on a discrete grid environment and a continuous particle environment. The environments are presented in Figure 1.

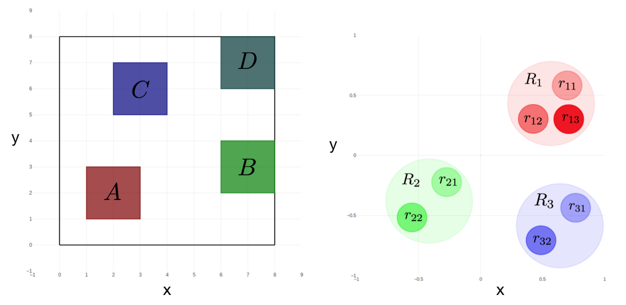


Figure 1. Simulation environments. (left) 2D grid environment with discrete state and action spaces. (right) 3D particle environment with continuous state and action spaces.

6.1. Experiment Setup

In the following subsections, we describe the environment setup as well as the evaluation task used for each environment.

6.1.1. Grid Environment

As shown in Figure 1 (left), the grid environment consists of four regions. An agent navigates in this environment with a set of discrete actions $A = \{\text{stay, left, right, up, down}\}$. The MDP state of the agent is simply its 2D coordinates $s = (x, y)$ where x and y take discrete values from 0 to 8.

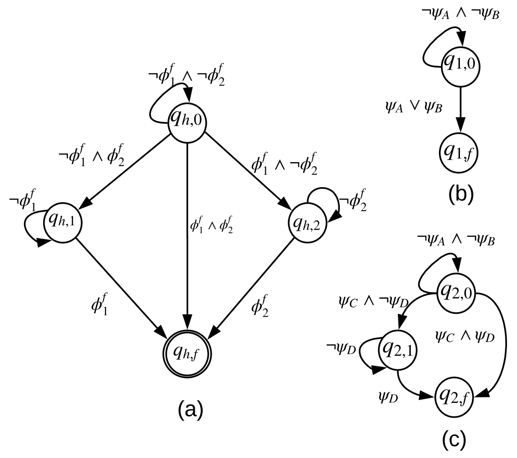
We define the hierarchical task for this environment as follows:

- $\phi^h_G = \Diamond\phi^f_1 \wedge \Diamond\phi^f_2$.
  Description: eventually accomplish the tasks described by ϕ_1 and ϕ_2.

- $\phi^f_1 = \Diamond\psi_A \vee \Diamond\psi_B$.
  Description: eventually visit region A or B.

- $\phi^f_2 = \Diamond(\psi_C \wedge \Diamond\psi_D)$.
  Description: eventually visit region C then D.

Here $\psi_i = |x^c_i - x| + |y^c_i - y| < th,\ i \in \{A, B, C, D\}$ is a set of predicates defined by the thresholded Manhattan distance between the agent and the regions. (x^c_i, y^c_i) is the center coordinate of region i. The FSA corresponding to ϕ, ϕ_1, ϕ_2 are shown in Figure 2. These FSAs are used to construct the products and rewards described in Definition 5. The states for the hierarchical FSA augmented MDP is therefore $\tilde{s}_{\phi^h_G} = (x, y, q_h, q_1, q_2)$ ($\tilde{S}_{\phi^h_G} \in \mathbb{Z}^5_{\geq 0}$).



$\neg\phi^f_1 \wedge \neg\phi^f_2$
$q_{h,0}$
$\neg\phi^f_1 \wedge \phi^f_2$
$\phi^f_1 \wedge \neg\phi^f_2$
$\neg\phi^f_1$
$q_{h,1}$
$\phi^f_1 \wedge \phi^f_2$
$q_{h,2}$
$\neg\phi^f_2$
ϕ^f_1
ϕ^f_2
$q_{h,f}$
(a)
$\neg\psi_A \wedge \neg\psi_B$
$q_{1,0}$
$\psi_A \vee \psi_B$
$q_{1,f}$
(b)
$\neg\psi_A \wedge \neg\psi_B$
$q_{2,0}$
$\psi_C \wedge \neg\psi_D$
$\psi_C \wedge \psi_D$
$\neg\psi_D$
$q_{2,1}$
ψ_D
$q_{2,f}$
(c)

Figure 2. FSA for (a) $\phi^h_G = \Diamond\phi^f_1 \wedge \Diamond\phi^f_2$, (b) $\phi^f_1 = \Diamond\psi_A \vee \Diamond\psi_B$, (c) $\phi^f_2 = \Diamond(\psi_C \wedge \Diamond\psi_D)$

6.1.2. Particle Environment

As shown in Figure 1 (right), the environment is made of three large circular regions R_1, R_2 and R_3. Each of the large region consists of two or three sub-regions. The agent navigating in this environment is controlled by its x and y forces ($A \in \mathbb{R}^2$). The agent's MDP state space consists of the its 2D positions, velocities, and the positions of all sub-regions ($S \in \mathbb{R}^{18}$).

We define the hierarchical task for this environment as follows:

- $\phi^h_P = \Diamond\phi^f_{R_1} \wedge \Diamond\phi^f_{R_2} \wedge \Diamond\phi^f_{R_3}$.
  Description: eventually service regions R_1, R_2 and R_3.

Table 1. Hyperparameters Used For Particle World

Hyperparm.	Value	Hyperparam	Value
learning rate	0.0001	horizon	150
discount	0.98	batch size	10
minibatch	64	num. epoches	10
clipping	0.2	value coeff.	0.1
entropy coeff.	0.14	GAE	0.98
max iterations	700	w	0.2

- $\phi^f_{R_1} = \Diamond\psi_{r_{11}} \vee (\Diamond\psi_{r_{12}} \wedge \Diamond\psi_{r_{13}})$.
  Description: to service R_1 means to eventually visit region r_{11} or eventually visit region r_{12} and $r_1 3$.

- $\phi^f_{R_2} = \Diamond(\psi_{r_{21}} \wedge \Diamond\psi_{r_{22}})$.
  Description: to service R_2 means to eventually first visit r_{21} and then r_{22}.

- $\phi^f_{R_3} = \Diamond\psi_{r_{31}} \wedge \Diamond\psi_{r_{32}}$.
  Description: to service R_3 means to eventually visit regions r_{31} and r_{32}.

In the above formula, $\psi_i = \sqrt{(x^c_i - x)^2 + (y^c_i - y)^2} < th,\ i \in \{r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{31}, r_{32}\}$ is a set of predicates defined by the thresholded distance between the agent and region centers. The FSA for ϕ^G ($\mathcal{A}_{\phi^h_G}$) has 8 nodes and 26 edges, $\mathcal{A}_{\phi^f_{R_1}}$ has 4 nodes and 8 edges, $\mathcal{A}_{\phi^f_{R_2}}$ has 3 nodes and 5 edges, $\mathcal{A}_{\phi^f_{R_3}}$ has 4 nodes and 8 edges. Due to space constraints, the FSAs are not explicitly provided. The state space for the hierarchical FSA augmented MDP is $\tilde{S}_{\phi^h_P} \in \mathbb{R}^{18} \times \mathbb{Z}^4_{\geq 0}$ where the last four discrete dimensions are the automata states. Code for this environment is adopted from (Lowe et al., 2017).

6.2. Implementation Details

For the grid environment, a greedy policy is obtained using the Q-Learning (Watkins, 1989) algorithm. We used a discount factor of 0.99, learning rate of 0.03 and episode horizon of 100 steps. For exploration, the ϵ-greedy strategy is used with ϵ decaying from 1.0 to 0.1 in 2×10^5 steps. The policy is updated for 150k steps.

For the particle environment, we use a feed-forward neural network to represent the policy. The network has 3 hidden layers each with 300, 200, 100 ReLu units respectively. The value function uses a network with the same architecture. The proximal policy optimization (Schulman et al., 2017) algorithm with general advantage estimation (Schulman et al., 2015) is used for training. The agent's position as well as the positions of all the regions are randomly initialized after each episode to facilitate generalization. The hyperparameters used in this experiment are provided in Table 1.

6.3. Comparison Cases

We compare the method presented in this paper with the regular FSA augmented MDP (Li et al., 2018). In order to do so, the hierarchical task specifications in Sections 6.1.1 and 6.1.2 are transformed to their flat form. Specifically $\phi^f_G = (\Diamond\psi_A \vee \Diamond\psi_B) \wedge (\Diamond(\psi_C \wedge \Diamond\psi_D))$ and the same is done to obtain ϕ^f_P. The state space then becomes $\tilde{S}_{\phi^f_G} \in \mathbb{Z}^3_{\geq 0}$ and $\tilde{S}_{\phi^f_P} \in \mathbb{R}^{18} \times \mathbb{Z}_{\geq 0}$ where only one automata state is appended to the original MDP states because only one FSA is constructed for each task. Even though doing so reduces the dimensionality of the augmented MDP state space, it increases the size of the resulting FSA ($\mathbb{A}_{\phi^f_G}$ has 6 nodes and 17 edges, $\mathbb{A}_{\phi^f_P}$ has 48 nodes and 486 edges). We show in the next section that incorporating abstractions in the TL formula improves learning performance particularly for large specifications. In these comparison cases, the learning algorithm is kept fixed changing only the augmented MDP structure.

It is worth emphasizing that a fair comparison of our work with other RL algorithms would require running different algorithms on the same MDP. But given our main focus here is in the construction of the MDP itself, it would be difficult to interpret the comparison results of different algorithms operating in different MDPs (with rewards constructed from other means).

Directly designing a real valued function that achieves similar semantics to the TL specifications provided here is a challenge in itself. Authors of (Vecerik et al., 2017) have shaped a reward for a two step clip insertion task and this reward is already exhibiting a nested min/max structure similar to the robustness definition. We expect that if enough effort is put into reward shaping, one will likely end up with a function close to the robustness degree for the tasks that we are looking at.

7. Results And Discussion

Figure 3 shows two sub-policies extracted from the optimal policy learned in the grid environment. Specifically, Figure 3 (left) shows the policy $\pi^\star_{\phi^h_G}(q_{h,0}, q_{1,0}, q_{2,0}, \cdot, \cdot)$. The FSA in Figure 1(a) indicates that in this configuration, the agent can either visit regions A or B (satisfying ϕ^f_1) or visit region C (triggering a transition in $\mathbb{A}_{\phi^f_2}$). However, the resulting policy only favors visiting regions A and B. This is because visiting one of these two regions triggers a transition in the high level FSA $\mathbb{A}_{\phi^h_G}$ which gives a larger reward than a transition in the lower level ones (Equation (11) with $0 < w < 1$). Figure 1 (right) shows the policy $\pi^\star_{\phi^h_P}(q_{h,2}, q_{1,f}, q_{2,0}, \cdot, \cdot)$. This configuration indicates that the agent has already satisfied ϕ^f_1. Therefore, this sub-policy leads the agent towards region C which must be visited first in order to satisfy ϕ^f_2.

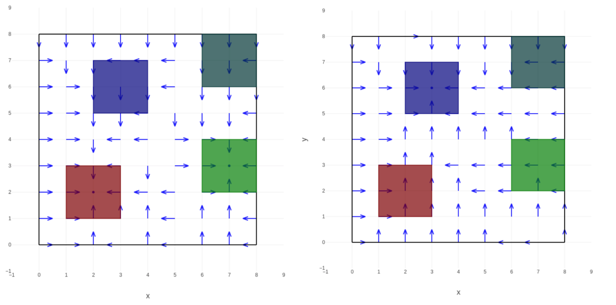


Figure 3. (left) Sub-policy $\pi^\star_{\phi^h_G}(q_{h,0}, q_{1,0}, q_{2,0}, \cdot, \cdot)$. (right) Sub-policy $\pi^\star_{\phi^h_P}(q_{h,2}, q_{1,f}, q_{2,0}, \cdot, \cdot)$

Figure 4 shows two sample trajectories from executing $\pi^\star_{\phi^h_G}$. The dot represents the agent at each time-step. Darker color indicates more recent in time. First we can see that both trajectories visit regions C and D in sequence as required by ϕ^f_2. However, visitation of regions A or B depend on which is closer to the starting position. This is dictated by the robustness definition of the $\vee$ operator (i.e. $\rho(s_{t:t+k}, \phi_1 \vee \phi_2) = \max(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2))$).

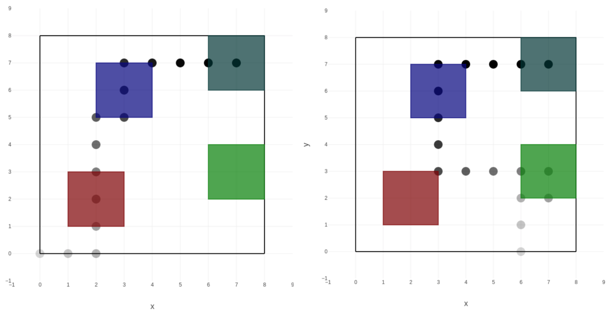


Figure 4. Sample trajectories from executing the optimal policy. Darker color indicates more recent in time

Figure 5 illustrates two sample trajectories from executing $\pi^\star_{\phi^h_P}$ of the particle environment. Figure 5 (left) shows that the agent is able to take approximately the shortest path to satisfying the specification. In trying to satisfy $\phi^f_{R_1}$, the agent mostly chooses to visit region r_{11} as oppose to visiting regions r_{12} and r_{13} (Figure 5 (left)) as this is the simpler and less time consuming path of the two. However, this is not the case in Figure 5 (right) because visiting r_{11} would mean a larger detour. These examples show that our method allows agent to make logical decisions that would be difficult to specify otherwise.

Because our method uses a different reward structures from the comparison method, the discounted return learning curve

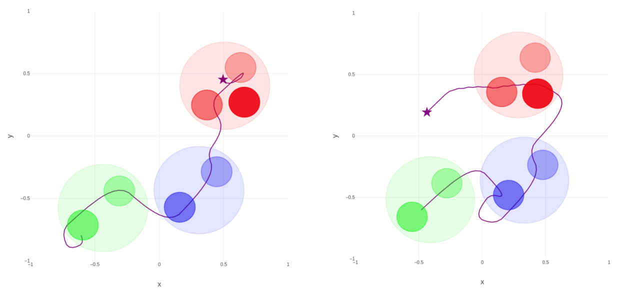


Figure 5. Sample trajectories in the particle environment. Star represent initial agent position.

can not be used here for comparison. Instead, we use the episode length as a measure of task completion and learning progress. An episode terminates by either reaching the horizon or satisfying the specification. A shorter episode length indicates faster satisfaction making it a suitable comparison measure. Figure 6 and Figure 7 show the episode length distribution as a function of policy updates. Here we denote our method by *hierarchical* and the comparison by *flat*.

Figure 6 shows that for the simple task ϕ_G^h, our method exhibits less variance during training but otherwise does not show a clear advantage over the comparison method. Both methods are able to reach the optimal policy (the fluctuation in the learning curve is due to exploration). The reason for this is likely that since the FSAs in this task are relatively small, the benefit of abstraction is not apparent. The slightly slower convergence of our method may be caused by the added dimensions to the state space.

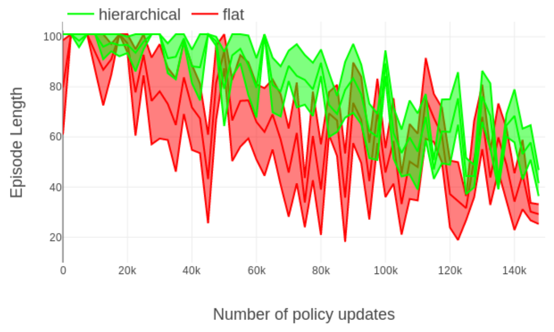


Figure 6. Learning curve in terms of episode length for the grid environment. The episode length distribution is calculated from 5 episodes and shown with 1 standard deviation.

Figure 3 shows a clear advantage of our method over the comparison. By augmenting 3 more dimensions to the state space, we are able to reduce the max number of automata states from 48 to 8, and the max number of edges from 486 to 26. This makes the structure of the reward for ϕ_P^h significant simpler than that of ϕ_P^f. The benefit of reducing the automata size outweighs that of the added state dimensions and hence the performance improvement. In 50 evaluation trials, the policy obtained with our method is able to complete the task for 46 trials (92%) while the comparison is completes only 10 trials (20%). A video of the learning process of the particle environment can be accessed at https://www.dropbox.com/s/csgijbufpu946ts/particle_env.mp4?dl=0.

One potential problem of our method is the explosion of state dimensions when the task consists of many layers of hierarchies. It is not clear whether this will considerably hinder learning as each added dimension (q state) is likely to take values from a small set of discrete integers. It will be meaningful to conduct such a study in future.

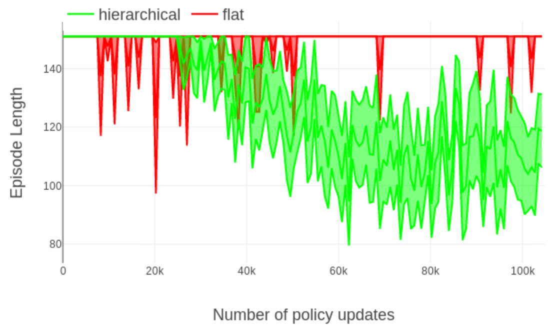


Figure 7. Learning curve in terms of episode length for the particle environment. The episode length distribution is calculated from 10 episodes and shown with 1 standard deviation.

8. Conclusions

We present in this paper a technique to specify complex tasks using temporal logic with multiple levels of abstraction. The hierarchical specification is translated into a set of FSAs and a dense step-based reward function that can be integrated with the original MDP. The resulting hierarchical FSA augmented MDP possesses both spatial (task level) and temporal (policy level) abstraction characteristics and can be trained end-to-end with off-the-shelf reinforcement learning algorithms. Our method is evaluated on a discrete grid environment and a continuous particle environment, and has shown success in generating trajectories that satisfy the given TL specifications. Future work includes extending this framework to full TLTL over both MDP states and actions as well as evaluation on real world robotic household tasks.

References

Abbeel, Pieter and Ng, Andrew Y. Apprenticeship learning via inverse reinforcement learning. In ICML, 2004.

Amodei, Dario, Olah, Chris, Steinhardt, Jacob, Christiano, Paul F., Schulman, John, and Mané, Dan. Concrete problems in ai safety. CoRR, abs/1606.06565, 2016.

Andreas, Jacob, Klein, Dan, and Levine, Sergey. Modular multitask reinforcement learning with policy sketches. In ICML, 2017.

Bengio, Yoshua, Louradour, Jérôme, Collobert, Ronan, and Weston, Jason. Curriculum learning. In ICML, 2009.

Camacho, Alberto, Chen, Oscar, Sanner, Scott, and McIlraith, Sheila A. Decision-making with non-markovian rewards: From ltl to automata-based reward shaping. In Proceedings of the Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM), pp. 279–283, 2017.

Chevalier-Boisvert, Maxime, Bahdanau, Dzmitry, Lahlou, Salem, Willems, Lucas, Saharia, Chitwan, Nguyen, Thien Huu, and Bengio, Yoshua. Babyai: First steps towards grounded language learning with a human in the loop. CoRR, abs/1810.08272, 2018.

Christiano, Paul F., Leike, Jan, Brown, Tom B., Martic, Miljan, Legg, Shane, and Amodei, Dario. Deep reinforcement learning from human preferences. In NIPS, 2017.

Christiano, Paul Francis, Abate, Marianna, and Amodei, Dario. Supervising strong learners by amplifying weak experts. CoRR, abs/1810.08575, 2018.

Clark, J. and Amodei, D. Faulty reward functions in the wild, 2016. URL https://blog.openai.com/faulty-reward-functions.

Dietterich, Thomas G. Hierarchical reinforcement learning with the maxq value function decomposition. J. Artif. Intell. Res., 13:227–303, 2000.

Giacomo, Giuseppe De, Iocchi, Luca, Favorito, Marco, and Patrizi, Fabio. Reinforcement learning for ltlf/ldlf goals. CoRR, abs/1807.06333, 2018.

Hadfield-Menell, Dylan, Dragan, Anca D., Abbeel, Pieter, and Russell, Stuart J. Cooperative inverse reinforcement learning. In NIPS, 2016.

Irving, Geoffrey, Christiano, Paul Francis, and Amodei, Dario. Ai safety via debate. CoRR, abs/1805.00899, 2018.

Lehman, Joel et al. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. CoRR, abs/1803.03453, 2018.

Leike, Jan, Krueger, David, Everitt, Tom, Martic, Miljan, Maini, Vishal, and Legg, Shane. Scalable agent alignment via reward modeling: a research direction. arXiv preprint arXiv:1811.07871, 2018.

Li, Xiao, Ma, Yao, and Belta, Calin. Automata guided reinforcement learning with demonstrations. arXiv preprint arXiv:1809.06305, 2018.

Lillicrap, Timothy P., Hunt, Jonathan J., Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. CoRR, abs/1509.02971, 2015.

Lowe, Ryan, Wu, Yi, Tamar, Aviv, Harb, Jean, Abbeel, Pieter, and Mordatch, Igor. Multi-agent actor-critic for mixed cooperative-competitive environments. Neural Information Processing Systems (NIPS), 2017.

Molnár, Vince and Vörös, András. Synchronous product automaton generation for controller optimization.

Ng, Andrew Y., Harada, Daishi, and Russell, Stuart J. Policy invariance under reward transformations: Theory and application to reward shaping. In ICML, 1999.

Schulman, John, Moritz, Philipp, Levine, Sergey, Jordan, Michael I., and Abbeel, Pieter. High-dimensional continuous control using generalized advantage estimation. CoRR, abs/1506.02438, 2015.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. CoRR, abs/1707.06347, 2017.

Singh, Satinder. Communicating hierarchical neural controllers for learning zero-shot task generalization. 2017.

Sutton, Richard S., Precup, Doina, and Singh, Satinder P. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artif. Intell., 112:181–211, 1999.

van Niekerk, Benjamin, James, Steven, Earle, Adam Christopher, and Rosman, Benjamin. Will it blend? composing value functions in reinforcement learning. CoRR, abs/1807.04439, 2018.

Vasile, C. Github repository, 2017.

Vecerik, Matej, Hester, Todd, Scholz, Jonathan, Wang, Fumin, Pietquin, Olivier, Piot, Bilal, Heess, Nicolas, Rothörl, Thomas, Lampe, Thomas, and Riedmiller, Martin A. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. CoRR, abs/1707.08817, 2017.

Watkins, Christopher John Cornish Hellaby. Learning From Delayed Rewards. PhD thesis, King's College, Cambridge, England, 1989.