AUTOMATA GUIDED SKILL COMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Skills learned through reinforcement learning often generalize poorly across tasks and re-training is necessary when presented with a new task. We present a framework that combines techniques in formal methods with reinforcement learning (RL) that allows for the construction of new skills from existing ones with no additional exploration necessary. Our method also allows for convenient specification of complex temporal dependent tasks using logical expressions. We provide theoretical results for our composition technique and evaluate on a simple grid world simulation as well as a robotic manipulation task.

1 INTRODUCTION

Policies learned using reinforcement learning aim to maximize the given reward function and are often difficult to transfer to other problem domains. Skill composition is the process of constructing new skills out of existing ones (policies) with little to no additional learning. In stochastic optimal control, this idea has been adopted by Todorov (2009) and Da Silva et al. (2009) to construct provably optimal control laws based on linearly solvable Markov decision processes.

Temporal logic (TL) is a formal language commonly used in software and digital circuit verification (Baier & Katoen, 2008) as well as formal synthesis (Belta et al., 2017). It allows for convenient expression of complex behaviors and causal relationships. TL has been used by Tabuada & Pappas (2004), Fainekos et al. (2006), Fainekos et al. (2005) to synthesize provably correct control policies. Aksaray et al. (2016) have also combined TL with Q-learning to learn satisfiable policies in discrete state and action spaces.

We make the distinction between skill composition and multi-task learning/meta-learning where the latter often requires a predefined set of tasks/task distributions to learn and generalize from, whereas the focus of the former is to construct new policies from a library of already learned policies that achieve new tasks (often some combination of the constituent tasks) with little to no additional constraints on task distribution at learning time. In this work, we focus on skill composition with policies learned using automata guided reinforcement learning (Li et al., 2018). We adopt the syntactically co-safe truncated linear temporal logic (scTLTL) as the task specification language. Compared to most heuristic reward structures used in the RL literature, formal specification language has the advantage of semantic rigor and interpretability.

In our framework, skill composition is accomplished by taking the product of finite state automata (FSA). Instead of interpolating/extrapolating among learned skills/latent features, our method is based on graph manipulation of the FSA. Therefore, the outcome is much more transparent. Compared with previous work on skill composition, we impose no constraints on the policy representation or the problem class. We validate our framework in simulation (discrete state and action spaces) and experimentally on a Baxter robot (continuous state and action spaces).

2 RELATED WORK

Recent efforts in skill composition have mainly adopted the approach of combining value functions learned using different rewards. Peng et al. (2018) constructs a composite policy by combining the value functions of individual policies using the Boltzmann distribution. With a similar goal, Zhu et al. (2017) achieves task space transfer using deep successor representations (Kulkarni et al., 2016). However, it is required that the reward function be represented as a linear combination of state-action features.
An MDP is defined as a tuple

\[ \mathcal{M} = (S, A, p(\cdot | \cdot, \cdot), r(\cdot, \cdot, \cdot)), \]

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); \( p : S \times A \times S \rightarrow [0, 1] \) is the transition function with \( p(s', a, s) \) being the conditional probability density of taking action \( a \in A \) at state \( s \in S \) and ending up in state \( s' \in S \); \( r : S \times A \times S \rightarrow \mathbb{R} \) is the reward function with \( r(s, a, s') \) being the reward obtained by executing action \( a \) at state \( s \) and transitioning to \( s' \).

\[ r(s, a, s') \]

\[ \pi(s, a) \]

\[ J(\pi) = \sum_{t=0}^{T-1} E[r_t + \alpha H(\pi(\cdot|s_t))] \],

where \( \pi : S \times A \rightarrow [0, 1] \) is a stochastic policy. \( E^\pi \) is the expectation following \( \pi \). \( H(\pi(\cdot|s_t)) \) is the entropy of \( \pi \). \( \alpha \) is the temperature parameter. In the limit \( \alpha \rightarrow 0 \), Equation (1) becomes the standard RL objective. The soft Q-learning algorithm introduced by \cite{haarnoja2017soft} optimizes the above objective and finds a policy represented by an energy-based model

\[ \pi^*(a_t|s_t) \propto \exp(-E(s_t, a_t)) \]

where \( E(s_t, a_t) \) is an energy function that can be represented by a function approximator.

Let \( r_t^{\pi|\pi} = r_t + \alpha H(\pi(\cdot|s_t)) \), the state-action value function (Q-function) following \( \pi \) is defined as

\[ Q^{\pi}(s, a) = E^\pi[\sum_{t=0}^{T-1} r_t^{\pi|\pi}|s_0 = s, a_0 = a] \].

Suppose we have a set of \( n \) tasks indexed by \( i, i \in \{0, ..., n\} \), each task is defined by an MDP \( \mathcal{M}_i \) that differs only in their reward function \( r_i \). Let \( Q^{\pi^*_i}_\alpha \) be the optimal entropy-regularized Q-function. Authors of \cite{vanNiekerk2018} provide the following results

\[ Q^{\pi^*_i}_\alpha = \alpha \log(||\exp(\pi^{\pi^*_i}/\alpha)||_w) \]

where \( ||\cdot||_w \) is the weighted 1-norm.

The authors proceed to provide the following corollary

\[ \max Q^{\pi^*_i}_\alpha \uparrow Q^{\pi^*_0}_0 \text{ as } \alpha \rightarrow 0, \text{ where } Q^{\pi^*_0}_0 \text{ is the optimal Q-function for the objective } J(\pi) = \sum_{t=0}^{T-1} E^\pi[r_t]. \]

Corollary (1) states that in the low temperature limit, the maximum of the optimal entropy-regularized Q-functions approaches the standard optimal Q-function.
where $s$ 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), $\land$ (conjunction/and) are Boolean connectives. $\Rightarrow$ (eventually), $\bigcup$ (until), $\bigcap$ (then), $\bigcirc$ (next), are temporal operators. $\Rightarrow$ (implication) and $\lor$ (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., $s_{t:t+k} = s_t s_{t+1} \ldots s_{t+k}$. The Boolean semantics of scTLTL is defined as:

$$s_{t:t+k} \models f(s) < c \iff f(s_t) < c,$$

$$s_{t:t+k} \models \neg \phi \iff \neg(s_{t:t+k} \models \phi),$$

$$s_{t:t+k} \models \phi \Rightarrow \psi \iff (s_{t:t+k} \models \phi) \Rightarrow (s_{t:t+k} \models \psi),$$

$$s_{t:t+k} \models \phi \land \psi \iff (s_{t:t+k} \models \phi) \land (s_{t:t+k} \models \psi),$$

$$s_{t:t+k} \models \phi \lor \psi \iff (s_{t:t+k} \models \phi) \lor (s_{t:t+k} \models \psi),$$

$$s_{t:t+k} \models \bigcup \phi \iff \exists t' \in [t, t + k) s_{t':t+k} \models \phi,$$

$$s_{t:t+k} \models \bigcup \psi \iff \exists t' \in [t, t + k) s_{t':t+k} \models \psi$$

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

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

$$\rho(s_{t:t+k}, \top) = \rho_{\text{max}},$$

$$\rho(s_{t:t+k}, f(s_t) < c) = c - f(s_t),$$

$$\rho(s_{t:t+k}, \neg \phi) = -\rho(s_{t:t+k}, \phi),$$

$$\rho(s_{t:t+k}, \phi \Rightarrow \psi) = \max(-\rho(s_{t:t+k}, \phi), \rho(s_{t:t+k}, \psi)),$$

$$\rho(s_{t:t+k}, \phi_1 \land \phi_2) = \min(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)),$$

$$\rho(s_{t:t+k}, \phi_1 \lor \phi_2) = \max(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)),$$

$$\rho(s_{t:t+k}, \bigcup \phi) = \rho(s_{t+1:t+k}, \phi) (k > 0),$$

$$\rho(s_{t:t+k}, \bigcap \phi) = \max_{t' \in [t, t+k]} \rho(s_{t':t+k}, \phi),$$

$$\rho(s_{t:t+k}, \bigcup \psi) = \max_{t' \in [t, t+k]} \left( \min_{t'' \in [t, t')] \rho(s_{t'':t+k}, \phi) \right),$$

$$\rho(s_{t:t+k}, \bigcap \psi) = \max_{t' \in [t, t+k]} \left( \min_{t'' \in [t, t')] \rho(s_{t'':t+k}, \phi) \right).$$

where $\rho_{\text{max}}$ represents the maximum robustness value. A robustness of greater than zero implies that $s_{t:t+k}$ satisfies $\phi$ 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 $\phi$. 

3.2 scTLTL AND FINITE STATE AUTOMATA

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 recursively as:

$$\phi := \top | f(s) < c | \neg \phi | \phi \land \psi | \Diamond \phi | \diamond \phi \land \psi | \phi \bigcup \psi | \phi \bigcap \psi | \bigcup \phi$$

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), $\land$ (conjunction/and) are Boolean connectives. $\Rightarrow$ (eventually), $\bigcup$ (until), $\bigcap$ (then), $\bigcirc$ (next), are temporal operators. $\Rightarrow$ (implication) and $\lor$ (disjunction/or) can be derived from the above operators.
Definition 2. An FSA corresponding to a scTLTL formula $\phi$ is defined as a tuple $A_\phi = (Q_\phi, \Psi, \phi_0, p_\phi(\cdot, \cdot), \mathcal{F}_\phi)$, where $Q_\phi$ is a set of automaton states; $\Psi$ is the input alphabet (a set of first order logic formula); $\phi_0 \in Q_\phi$ is the initial state; $p_\phi : Q_\phi \times Q_\phi \rightarrow [0, 1]$ is a conditional probability defined as

$$p_\phi(q_\phi,j | q_\phi,i) = \begin{cases} 1 & \psi_{q_\phi,i, q_\phi,j} \text{ is true} \\ 0 & \text{otherwise.} \end{cases}$$ (5)

or

$$p_\phi(q_\phi,j | q_\phi,i, s) = \begin{cases} 1 & \rho(s, \psi_{q_\phi,i, q_\phi,j}) > 0 \\ 0 & \text{otherwise.} \end{cases}$$ (6)

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

Here $q_\phi,i$ is the $i^{th}$ automaton state of $A_\phi$. $\psi_{q_\phi,i, q_\phi,j} \in \Psi$ 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 $\rho(s; t+1, \psi_{q_\phi,i, q_\phi,j})$ is only evaluated at $s$. Therefore, we use the shorthand $\rho(s, \psi_{q_\phi,i, q_\phi,j}) = \rho(s; t+1, \psi_{q_\phi,i, q_\phi,j})$. The translation from a TLTL formula to a FSA can be done automatically with available packages like Lomap \cite{Vasile2017}.

3.3 FSA augmented MDP

The FSA augmented MDP is defined as follows

Definition 3. \cite{Li2018} An FSA augmented MDP corresponding to scTLTL formula $\phi$ (constructed from FSA $\langle Q_\phi, \Psi, \phi_0, p_\phi(\cdot, \cdot), \mathcal{F}_\phi \rangle$ and MDP $\langle S, A, p(\cdot, \cdot), r(\cdot, \cdot), \mathcal{F}_\phi \rangle$) is defined as $M_\phi = (\tilde{S}, \tilde{A}, \tilde{p}(\cdot, \cdot), \tilde{r}(\cdot, \cdot), \mathcal{F}_\phi)$ where $\tilde{S} \subseteq S \times Q_\phi$, $\tilde{p}(s', s, a) = \frac{p(s', s, a)}{p_\phi(q_\phi, | q_\phi, s, a) = 1}$ otherwise.

$p_\phi$ is defined in Equation (5). $\tilde{r} : \tilde{S} \times \tilde{S} \rightarrow \mathbb{R}$ is the FSA augmented reward function, defined by

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

where $D_\phi^q = \bigvee_{q' \in \Omega_{\phi_0}} \psi_{q_\phi, q'}$ represents the disjunction of all predicates guarding the transitions that originate from $q_\phi$ ($\Omega_{\phi_0}$ is the set of automata states that are connected with $q$ through outgoing edges).

A policy $\pi_\phi^*$ is said to satisfy $\phi$ if

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

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

As is mentioned in the original paper, there can be multiple policies that meet the requirement of Equation (5), therefore, a discount factor is used to find a maximally satisfying policy - one that leads to satisfaction in the least amount of time.

The FSA augmented MDP $M_\phi$ establishes a connection between the TL specification and the standard reinforcement learning problem. A policy learned using $M_\phi$ has implicit knowledge of the FSA through the automaton state $q_\phi \in Q_\phi$. We will take advantage of this characteristic in our skill composition framework.
Problem Formulation

Problem 1. Given two scTLTL formula $\phi_1$ and $\phi_2$ and their optimal Q-functions $Q^*_{\phi_1}$ and $Q^*_{\phi_2}$, obtain the optimal policy $\pi^*_{\phi_1 \land \phi_2}$ that satisfies $\phi_1 \land \phi_2$ and $\pi^*_{\phi_1 \lor \phi_2}$ that satisfies $\phi_1 \lor \phi_2$.

Here $Q^*_{\phi_1}$ and $Q^*_{\phi_2}$ can be the optimal Q-functions for the entropy-regularized MDP or the standard MDP. Problem 1 defines the problem of skill composition: given two policies each satisfying a scTLTL specification, construct the policy that satisfies the conjunction ($\land$) or disjunction ($\lor$) of the given specifications. Solving this problem is useful when we want to break a complex task into simple and manageable components, learn a policy that satisfies each component and "stitch" all the components together so that the original task is satisfied. It can also be the case that as the scope of the task grows with time, the original task specification is amended with new items. Instead of having to re-learn the task from scratch, we can learn only policies that satisfies the new items and combine them with the old policy.

Automata Guided Skill Composition

In this section, we provide a solution for Problem 1 by constructing the FSA of $\phi_1 \land \phi_2$ and using $\phi_1 \land \phi_2$ to synthesize the policy for the combined skill. We start with the following definition.

Definition 4. Given $A_{\phi_1} = \langle Q_{\phi_1}, \Psi_{\phi_1}, q_{\phi_1,0}, p_{\phi_1}, F_{\phi_1}\rangle$ and $A_{\phi_2} = \langle Q_{\phi_2}, \Psi_{\phi_2}, q_{\phi_2,0}, p_{\phi_2}, F_{\phi_2}\rangle$ corresponding to formulas $\phi_1$ and $\phi_2$, the FSA of $\phi_1 \land \phi_2$ is the product automaton of $A_{\phi_1}$ and $A_{\phi_2}$, i.e., $A_{\phi_1 \land \phi_2} = A_{\phi_1} \times A_{\phi_2} = \langle Q_{\phi_1 \land \phi_2}, \Psi_{\phi_1 \land \phi_2}, q_{\phi_1 \land \phi_2,0}, p_{\phi_1 \land \phi_2}, F_{\phi_1 \land \phi_2}\rangle$ where $Q_{\phi_1 \land \phi_2} \subseteq Q_{\phi_1} \times Q_{\phi_2}$ is the set of product automaton states, $q_{\phi_1,0} = (q_{\phi_1,0}, q_{\phi_2,0})$ is the product initial state, $F_{\phi_1 \land \phi_2} \subseteq F_{\phi_1} \cap F_{\phi_2}$ are the final accepting states. Following Definition 2 for states $q_{\phi_1} = (q_{\phi_1}, q_{\phi_2}) \in Q_{\phi_1 \land \phi_2}$ and $q'_{\phi_1} = (q'_{\phi_1}, q'_{\phi_2}) \in Q_{\phi_1 \land \phi_2}$, the transition probability $p_{\phi_1 \land \phi_2}$ is defined as

$$p_{\phi_1 \land \phi_2}(q'_{\phi_1}, q_{\phi_2}) = \begin{cases} 1 & p_{\phi_1}(q'_{\phi_1}, q_{\phi_1})p_{\phi_2}(q'_{\phi_2}, q_{\phi_2}) = 1 \\ 0 & \text{otherwise}. \end{cases}$$
Example 1. Figure [1] illustrates the FSA of $A_\phi$ and $A_{\phi_2}$ and their product automaton $A_{\phi_1}$. Here $\phi_1 = \Diamond r \land q \Diamond g$ which entails that both $r$ and $g$ needs to be true at least once (order does not matter), and $\phi_2 = \Diamond b$. The resultant product corresponds to the formula $\phi = \Diamond r \land q \Diamond g \land \Diamond b$.

We provide the following theorem on automata guided skill composition

**Theorem 2.** Let $Q_{\phi}^\pi = [Q_{\phi_1}^\pi, \ldots, Q_{\phi_n}^\pi]$ be a vector with entries $Q_{\phi_i}^\pi$, being the optimal Q-function for the FSA augmented MDP $M_{\phi_i}$. The optimal Q-function for $M_{\phi_1}$ where $\phi_1 = \bigwedge_i \phi_i$ is $Q_{\phi_1}^\pi = \max_i (Q_{\phi_i}^\pi)$.

**Proof.** For Theorem 2. We provide the following theorem on automata guided skill composition and in the product automaton $A_{\phi_1}$, $\phi_1$ is $Q_{\phi_1}^\pi = \max_i (Q_{\phi_i}^\pi)$. Therefore, we have

$$D_{\phi_1} = \bigvee q_{\phi_1} \psi_{\phi_2}$$

where $q_{\phi_1}, q_{\phi_2}$ don’t equal to $q_{\phi_1}, q_{\phi_2}$ at the same time (to avoid self looping edges). Using the fact that $\psi_{\phi_1}, \psi_{\phi_2}$ exist in $\phi_1$ and repeatedly applying the distributive laws $(\Delta \land \Omega_1) \lor (\Delta \land \Omega_2) = \Delta \land (\Omega_1 \lor \Omega_2)$ and $(\Delta \lor \Omega_1) \land (\Delta \lor \Omega_2) = \Delta \lor (\Omega_1 \lor \Omega_2)$ to $D_{\phi_1}$, we arrive at

$$D_{\phi_1} = \bigvee q_{\phi_1} \psi_{\phi_1} \land \bigvee q_{\phi_2} \psi_{\phi_2} = D_{\phi_1} \lor D_{\phi_2}.$$  

(11)

Let $r_{\phi_1}, r_{\phi_2}$ and $s_{\phi_1}, s_{\phi_2}$ be the reward functions and states for FSA augmented MDP $M_{\phi_1}, M_{\phi_2}$ respectively, $s_{\phi_1}, s_{\phi_2}$ are the states for the corresponding MDPs. Plugging Equation (11) into Equation (10) and using the robustness definition for disjunction results in

$$\bar{r}_{\phi_1}(s_{\phi_1}, s_{\phi_2}) = \rho(s_{\phi_1}, D_{\phi_1}) = \rho(s_{\phi_1}, D_{\phi_1} \lor D_{\phi_2}) = \max \rho(s_{\phi_1}, D_{\phi_1}), \rho(s_{\phi_2}, D_{\phi_2})$$

(12)

Looking at Theorem[1] the log-sum-exp of the composite reward $r = \alpha \log(|| \exp(r/\alpha)||_\omega)$ is in fact an approximation of the maximum function. In the low temperature limit we have $r \rightarrow \max(r)$ as $\alpha \rightarrow 0$. Applying Corollary[1] results in Theorem[2].

Having obtained the optimal Q-function, a policy can be constructed by taking the greedy step with respective to the Q-function in the discrete action case. For the case of continuous action space where the policy is represented by a function approximator, the policy update procedure in actor-critic methods can be used to extract a policy from the Q-function.

In our framework, $-AND-$ and $-OR-$ task compositions follow the same procedure (Theorem 2). The only difference is the termination condition. For $-AND-$ task, the final state $F_{\phi_1} = \bigcap F_{\phi_i}$ in Definition 2 needs to be reached (i.e. all the constituent FSAs are required to reach terminal state, as in state $q_{\phi_0,j}$ in Figure 1). Whereas for the $-OR-$ task, only $F_{\phi_j} = \bigcup F_{\phi_i}$ needs to be reached (one of states $q_{\phi_0,2}, q_{\phi_0,4}, q_{\phi_0,5}, q_{\phi_0,6}, q_{\phi_0,j}$ in Figure 1). A summary of the composition procedure is provided in Algorithm 1.

In Algorithm 1 steps 3 and 4 seeks to obtain the optimal policy and Q-function using any off-policy actor critic RL algorithm. $B_{\phi_{i,j}}$ are the replay buffers collected while training for each skill. Step
Algorithm 1 Automata Guided Skill Composition

1: Inputs: scTLTL task specification φ₁ and φ₂, randomly initialized policies πφ₁, πφ₂ and action-value functions Qπφ₁, Qπφ₂. State and action spaces of the MDP.
2: Construct FSA augmented MDPs Mφ₁ and Mφ₂ ⊢ using Definition 3
3: πφ₁, Qπφ₁ Bφ₁ ← ActorCritic(Mφ₁) ⊢ learns the optimal policy and Q-function
4: πφ₂, Qπφ₂ Bφ₂ ← ActorCritic(Mφ₂)
5: Qπφ₁∧φ₂ = max(Qπφ₁, Qπφ₂) ⊢ construct the optimal composed Q-function using Theorem 2
6: Bφ, ← ConstructProductBuffer(Bφ₁, Bφ₂)
7: πφ₁∧φ₂ ← ExtractPolicy(Qπφ₁∧φ₂, Bφ,)
8: return πφ₁∧φ₂

6 constructs the product replay buffer for policy extraction. This step is necessary because each Bφ₁ contains state of form (s, q₁), i ∈ {1, 2} whereas the composed policy takes state (s, q₁, q₂) as input (as in Definition 4). Therefore, we transform each experience ((s, q₁), a₁, (s’, q₁’), r₁) to ((s, q₁, q₁≠i), a₁, (s’, q₁’, q₁≠i), r₁) where q₁≠i is chosen at random from the automaton states of Aφ₁ and q₁≠i is calculated from Equation (3). The reward r₁ will not be used in policy extraction as the Q-function will not be updated. Step 7 extracts the optimal composed policy from the optimal composed Q-function (this corresponds to running only the policy update step in the actor critic algorithm).

6 Case Studies

We evaluate our composition method in two environments. The first is a simple 2D grid world environment that is used as for proof of concept and policy visualization. The second is a robot manipulation environment.

6.1 Grid World

Consider an agent that navigates in a 8 × 10 grid world. Its MDP state space is S : X × Y where x, y ∈ X, Y are its integer coordinates on the grid. The action space is A : [up, down, left, right, stay]. The transition is such that for each action command, the agent follows that command with probability 0.8 or chooses a random action with probability 0.2. We train the agent on two tasks, φ₁ = ◊r ∧ ◊g and φ₂ = ◊b (same as in Example 1). The regions are defined by the predicates r = (1 < x < 3) ∧ (1 < y < 3) and g = (4 < x < 6) ∧ (4 < y < 6). Because the coordinates are integers, a and b define a point goal rather than regions. φ₂ expresses a similar task for b = (1 < x < 3) ∧ (6 < y < 8). Figure 1 shows the FSA for each task.

We apply standard tabular Q-learning [Watkins 1989] on the FSA augmented MDP of this environment. For all experiments, we use a discount factor of 0.95, learning rate of 0.1, episode horizon of 200 steps, a random exploration policy and a total number of 2000 update steps which is enough to reach convergence (learning curve is not presented here as it is not the focus of this paper).

Figure 2 (a) and (b) show the learned optimal policies extracted by πφ₁(x, y, qφ₁) = arg max a Qφ₁(x, y, qφ₁, a). We plot πφ₁(x, y, qφ₁) for each qφ₁ and observe that each represents a sub-policy whose goal is given by Equation 7.

Figure 2 (c) shows the composed policy of φ = φ₁ ∧ φ₂ using Theorem 2. It can be observed that the composed policy is able to act optimally in terms maximizing the expected sum of discounted rewards given by Equation 12. Following the composed policy and transitioning the FSA in Figure 1 (b) will in fact satisfy φ (¬AND¬). As discussed in the previous section, if the ¬OR-task is desired, following the same composed policy and terminate at any of the states qφ ∧, 2, qφ ∧, 4, qφ ∧, 5, qφ ∧, 6, qφ ∧, f will satisfy φ = φ₁ ∨ φ₂.

7
Figure 2: Policies for (a) $\phi_1 = \Diamond a \land \Diamond b$, (b) $\phi_2 = \Diamond c$. (c) $\phi_3 = \phi_1 \land \phi_2$. The agent moves in a $8 \times 10$ gridworld with 3 labeled regions. The agent has actions [up, down, left, right, stay] where the directional actions are represented by arrows, stay is represented by the blue dot.

Figure 3: The FSA for (a): $\phi_{traverse} = \Diamond (\psi_r \land (\psi_g \land \Diamond \psi_b))$ b $\phi_{interrupt} = (\psi_{hand,in,sight} \Rightarrow \Diamond \psi_h) \cup \psi_b$. The subscripts $\phi_1$ and $\phi_2$ are dropped for clarity.

6.2 Robotic Manipulation

In this sub-section, we test our method on a more complex manipulation task.

6.2.1 Experiment Setup

Figure 4(a) presents our experiment setup. Our policy controls the 7 degree-of-freedom joint velocities of the right arm of a Baxter robot. In front of the robot are three circular regions (red, green,
blue plates) and it has to learn to traverse in user specified ways. The positions of the plates are tracked by motion capture systems and thus fully observable. In addition, we also track the position of one of the user’s hands (by wearing a glove with trackers attached). Our MDP state space is 22 dimensional that includes 7 joint angles, xyz positions of the three regions (denoted by $p^r$, $p^g$, $p^b$), the user’s hand ($p^h$) and the robot’s end-effector ($p^{ee}$). State and action spaces are continuous in this case.

We define the following predicates

1. $\psi_i = |p_i - p^{ee}| < \epsilon$, $i \in \{r, g, b, h\}$ where $\epsilon$ is a threshold which we set to be 5 centimeters. $\psi_i$ constrains the relative distance between the robot’s end-effector and the selected object.

2. $\psi_{\text{hand in sight}} = (x_{\text{min}} < p^h_x < x_{\text{max}}) \land (y_{\text{min}} < p^h_y < y_{\text{max}}) \land (z_{\text{min}} < p^h_z < z_{\text{max}})$. This predicate evaluates to true if the user’s hand appears in the cubic region defined by $[x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}, z_{\text{min}}, z_{\text{max}}]$. In this experiment, we take this region to be 40 centimeters above the table (length and width the same as the table).

We test our method on the following composition task

1. $\phi_{\text{traverse}} = \Diamond(\psi_r \land \Diamond(\psi_g \land \Diamond(\psi_b)))$
   Description: traverse the three regions in the order of red, green, blue.

2. $\phi_{\text{interrupt}} = (\psi_{\text{hand in sight}} \Rightarrow \Diamond \psi_h) \cup \psi_b$
   Description: before reaching the blue region, if the user’s hand appears in sight, then eventually reach for the user’s hand and the blue region, otherwise just reach for the blue region.

3. $\phi_{\Lambda} = \phi_{\text{traverse}} \land \phi_{\text{interrupt}}$
   Description: conjunction of the first two tasks.

The FSAs for $\phi_1$ and $\phi_2$ are presented in Figure 3. The FSA for $\phi_{\Lambda}$ (14 nodes and 72 edges) is not presented here due to space constraints.

6.2.2 Implementation Details

Our policy and Q-function are represented by a neural network (3 layers each with 300, 200, 100 ReLU units). For tasks $\phi_{\text{traverse}}$ and $\phi_{\text{interrupt}}$, the input state space is 23 dimensional (22 continuous dimensional MDP state and 1 discrete dimension for the automaton state). For task $\phi_{\Lambda}$, the state space is 24 dimensional (2 discrete dimensions for automaton states of $\phi_{\text{traverse}}$ and $\phi_{\text{interrupt}}$). We use soft Q-learning (SQL) [Haarnoja et al., 2017] to learn the optimal policies and Q-functions for $\phi_{\text{traverse}}$ and $\phi_{\text{interrupt}}$ and Theorem 2 to compose the Q-functions. The policy for task $\phi_{\Lambda}$ is extracted from the composite Q-function by running the same algorithm on already collected experience without updating the Q-function (similar procedure as [Haarnoja et al., 2018]). After each
episode, the joint angles, the FSA state, the position of the plates as well as the position of the hand (represented by the yellow sphere in Figure 4 (a)) are randomly reset (within certain boundaries) to ensure generalization across different task configurations. The robot is controlled at 20 Hz. Each episode is 100 time-steps (about 5 seconds). The episode restarts if the final automaton state $q_f$ is reached. During training, we perform 100 policy and Q-function updates every 5 episodes of exploration. All of our training is performed in simulation and for this set of tasks, the policy is able to transfer to the real robot without further fine-tuning.

6.2.3 Results and Discussion

In Figure 5 (left), we report the discounted return as a function of policy update steps for task $\phi_\wedge$. 5 evaluation episodes are collected after each set policy updates to calculate the performance statistics. As comparison, we learn the same task using SQL with FSA augmented MDP. We can see that our composition method takes less update steps to reach a policy that achieves higher returns with lower variance than the policy obtained from learning. Figure 5 (right) shows the episode length as a function of policy update (upper bound clipped at 100 steps). As mentioned in the previous section, a shorter episode length indicates faster accomplishment of the task. It can be observed that both the composition and learning method result in high variances likely due to the randomized task configuration (some plate/joint/hand configurations make the task easier to accomplish than others). However, the policy obtained from composition achieves a noticeable decrease in the average episode length.

It is important to note that the wall time for learning a policy is significantly longer than that from composition. For robotic tasks with relatively simple policy representations (feed-forward neural networks), learning time is dominated by the time used to collect experiences and the average episode length (recall that we update the policy 100 times with each 5 episodes of exploration). Since skill composition uses already collected experience, obtaining a policy can be much faster. Table 1 shows the mean training time and standard deviation (over 5 random seeds) for each task ($\phi_{traverse}$, $\phi_{interrupt}$ and $\phi_\wedge$ (learned) are trained for 80K policy updates, $\phi_\wedge$ (composed) is trained for 40K policy updates). In general, training time is shorter for tasks with higher episodic success rate and shorter episode length. We also show the task success rate evaluated on the real robot over 20 evaluation trials. Task success is evaluated by calculating the robustness of the trajectories resulting from executing each policy. A robustness of greater than 0 evaluates to success and vice versa. $\pi_{\phi_\wedge, \text{(learned)}}$ fails to complete the task even though a convergence is reached during training. This is likely due to the large FSA of $\phi_\wedge$ with complex per-step reward ($D^T_\phi$ in Equation (7)) which makes learning difficult. Figure 4 (b) shows an evaluation run of the composed policy for task $\phi_\wedge$.

7 Conclusion

We provide a technique that takes advantage of the product of finite state automata to perform deterministic skill composition. Our method is able to synthesize optimal composite policies for $-AND-$ and $-OR-$ tasks. We provide theoretical results on our method and show its effectiveness on a grid world simulation and a real world robotic task. For future work, we will adapt our
Table 1: Results on Training and Evaluation Performance

<table>
<thead>
<tr>
<th></th>
<th>$\phi_{\text{traverse}}$</th>
<th>$\phi_{\text{interrupt}}$</th>
<th>$\phi_\land$ (learned)</th>
<th>$\phi_\land$ (composed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean training time/std (Hour)</td>
<td>4.2/0.1</td>
<td>4.7/0.3</td>
<td>5.8/0.5</td>
<td>0.8/0.04</td>
</tr>
<tr>
<td>Mean task success rate/std</td>
<td>85.4%/2.3%</td>
<td>80.2%/2.5%</td>
<td>3.3%/1%</td>
<td>68.9%/3%</td>
</tr>
<tr>
<td>Mean discounted return/std</td>
<td>-13.2/2.3</td>
<td>-10.3/3.5</td>
<td>-40.6/10.5</td>
<td>-20.1/8.3</td>
</tr>
</tbody>
</table>

method to the more general case of task-space transfer - given a library of optimal policies (Q-functions) that each satisfies its own specification, construct a policy that satisfies a specification that’s an arbitrary (temporal) logical combination of the constituent specifications.

REFERENCES


