2019 American Control Conference (ACC)
Philadelphia, PA, USA, July 10-12, 2019

Compositional planning in Markov decision processes: Temporal abstraction meets generalized logic composition

Xuan Liu and Jie Fu

***Abstract*— In hierarchical planning for Markov decision processes (MDPs), temporal abstraction allows planning with macro-actions that take place at different time scale in form of sequential composition. In this paper, we propose a novel approach to compositional reasoning and hierarchical planning for MDPs under co-safe temporal logic constraints. In addition to sequential composition, we introduce a composition of policies based on generalized logic composition: Given sub-policies for sub-tasks and a new task expressed as logic compositions of subtasks, a semi-optimal policy, which is optimal in planning with only sub-policies, can be obtained by simply composing sub-polices. Thus, a synthesis algorithm is developed to compute optimal policies efficiently by planning with primitive actions, policies for sub-tasks, and the compositions of sub-policies, for maximizing the probability of satisfying constraints specified in the fragment of co-safe temporal logic. We demonstrate the correctness and efficiency of the proposed method in stochastic planning examples with a single agent and multiple task specifications.**

I. INTRODUCTION

Temporal logic is an expressive language to describe desired system properties: safety, reachability, obligation, stability, and liveness [18]. The algorithms for planning and probabilistic verification with temporal logic constraints have developed, with both centralized [2], [7], [17] and distributed methods [10]. Yet, there are two main barriers to practical applications: 1) The issue of scalability: In temporal logic constrained control problems, it is often necessary to introduce additional memory states for keeping track of the evolution of state variables with respect to these temporal logic constraints. The additional memory states grow exponentially (or double exponentially depending on the class of temporal logic) in the length of a specification [11] and make synthesis computational extensive. 2) The lack of flexibility: With a small change in the specification, a new policy may need to be synthesized from scratch.

To improve scalability for planning given complex tasks, composition is an idea exploited in temporal abstraction and hierarchical planning in Markov decision processes (MDPs) [24], [1]. To accomplish complex tasks, temporal abstraction allows planning with macro-actions—policies for simple subtasks—with different time scales. A well-known hierarchical planner is called the options framework [20], [26], [22]. An option is a pre-learned policy for a subtask given the original task that can be completed by temporally abstracting subgoals and sequencing the subtasks' policies.

X. Liu and J. Fu are with the Department of Electrical and Computer Engineering, Robotics Engineering Program, Worcester Polytechnic Institute, Worcester, MA. USA. xliu9, jfu2@wpi.edu

This material is based upon work supported by the National Science Foundation under Grant No. #1728412.

Once an agent learns the set of options from an underlying MDP, it can use conventional reinforcement learning to learn the global optimal policy with the original action set augmented with the set of options, also known as sub-policies or macro-actions. In light of the options framework, hierarchical planning in MDPs is evolving rapidly, with both model-free [15] and model-based [24], [25], and with many practical applications in robotic systems [13], [14]. The option-critic method [1] integrates approximate dynamic programming [3] with the options framework to further improve its scalability.

Since temporal logic specifications describe temporally extended goals and the options framework uses temporally abstracted actions, it seems that applying the options framework to planning under temporal logic constraints is straightforward. However, a direct application does not take full advantages of various compositions observed in temporal logic. The options framework captures the *sequential composition*. However, it does not consider *composition for conjunction or disjunction in logic*. In this paper, we are interested in answering two questions: Given two options that maximize the probabilities of φ_1 and φ_2, is there a way to compose these two options to obtain a "good enough" policy for maximizing the probability of $\varphi_1 \wedge \varphi_2$, or $\varphi_1 \vee \varphi_2$? If there exists a way to compose, what shall be the least set of options that one needs to generate? Having multiple ways of composition enables more flexible and modular planning given temporal logic constraints. For example, consider a specification $\Diamond((R_1 \vee R_2) \wedge \Diamond R_3)$, i.e., eventually reaching the region R_3 after visiting any of the two regions R_1 and R_2. With composition for sequential tasks only, we may generate an option that maximizes the probability of reaching $R_1 \vee R_2$ and an option that maximizes the probability of reaching R_3. With both compositions of sequencing, conjunction, and disjunction of tasks, we may generate options that maximize the probabilities of reaching R_1, R_2, and R_3, respectively, and compose the first two to obtain the option for $\Diamond R_1 \vee \Diamond R_2$. In addition, we can compose options to not only for $\Diamond R_1 \vee \Diamond R_2$ but also have $\Diamond R_1 \vee \Diamond R_3$, $\Diamond R_1 \vee \Diamond R_2 \vee \Diamond R_3$, etc. When the task changes to $\Diamond((R_1 \vee \boldsymbol{R_3}) \wedge \Diamond \boldsymbol{R_2})$, the new option for $\Diamond R_1 \vee \Diamond R_3$ needs not to be learned or computed, but composed.

In a pursuit to answering these two questions, the contribution of this paper is two-fold: We develop an automatic decomposition procedure to generate a *small set* of primitive options from a given co-safe temporal logic specification. We formally establish an equivalence relation between Generalized Conjunction/Disjunction (GCD) functions [8] in quantitative logic and composable solutions of MDPs

978-1-5386-7928-9/$31.00 ©2019 AACC

using entropy-regulated Bellman operators [24], [19]. This equivalence enables us to compose policies for simple formulas/tasks to maximize the probability for satisfying formulas obtained via GCD composition of these simple formulas. Last, we use these novel composition operations to develop a hierarchical planning method for MDPs under co-safe temporal logic constraints. We demonstrate the efficiency and correctness of the proposed method with several examples.

II. Preliminaries

Notation: Let $\mathbb{N}$ be the set of nonnegative integers. Let Σ be an alphabet (a finite set of symbols). Given $k \in \mathbb{N}$, Σ^k indicates a set of strings with length k, $\Sigma^{\leq k}$ indicates a set of finite strings with length smaller or equal to k, and $\Sigma^0 = \lambda$ is the empty string. Σ^* is the set of all finite strings (also known as Kleene closure of Σ). Given a set X, let $\text{Dist}(X)$ be a set of probabilistic distributions with X as the support.

In this paper, we consider temporal logic formulas for specifying desired properties in a stochastic system. Given a set $\mathcal{AP}$ of atomic propositions, a syntactically co-safe linear temporal logic (sc-LTL) [16] formula over $\mathcal{AP}$ is inductively defined as follows:

$$\varphi \coloneqq \mathsf{true} | p | \neg p | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \bigcirc \varphi | \varphi_1 \,\mathsf{U}\, \varphi_2.$$

The above formula is composed of unconditional true true, state predicates p and its negation $\neg p$, conjunction ($\wedge$) and disjunction ($\vee$), temporal operators "Next" ($\bigcirc$), and "Until" (U). Temporal operator "Eventually" ($\Diamond$) is defined by: $\Diamond \phi \coloneqq \mathsf{true}\, \mathsf{U}\, \phi$. However, temporal operator "Always" cannot be expressed in sc-LTL. A detailed description of the syntax and semantics of sc-LTL can be found in [21]. An sc-LTL formula φ is evaluated over finite words. In addition to the above notation, we use a backslash ($\backslash$) between two propositions to represent the logic exclusion, i.e., rewriting $\varphi_1 \wedge \neg \varphi_2$ to $\varphi_1 \backslash \varphi_2$.

Given an sc-LTL formula φ, there exists a deterministic finite-state automaton (DFA) that accepts all strings that satisfy the formula φ [11]. The DFA is a tuple $\mathcal{A}_\varphi = \langle Q, \Sigma, \delta, q_0, F \rangle$, where Q is a finite set of states, $\Sigma = 2^{\mathcal{AP}}$ is a finite alphabet, $\delta : Q \times \Sigma \to Q$ is a *deterministic* transition function such that when the symbol $\sigma \in \Sigma$ is read at state q, the automaton makes a deterministic transition to state $\delta(q, \sigma) = q'$, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of final, *accepting* states. The transition function is extended to a sequence of symbols, or a *word* $w = \sigma_0 \sigma_1 \ldots \in \Sigma^*$, in the usual way: $\delta(q, \sigma_0 v) = \delta(\delta(q, \sigma_0), v)$ for $\sigma_0 \in \Sigma$ and $v \in \Sigma^*$. A finite word w satisfies φ if and only if $\delta(q_0, w) \in F$. The set of words satisfying φ is the *language* of the automaton $\mathcal{A}_\varphi$, denoted $\mathcal{L}(\mathcal{A}_\varphi)$.

We consider stochastic systems modeled by MDPs. The specification is given by an sc-LTL formula and related to paths in an MDP via a labeling function.

Definition 1 (Labeled MDPs)**.** *A labeled MDP is a tuple $M = \langle S, A, \mu_0, P, \mathcal{AP}, L \rangle$ where S and A are finite state and action sets. $\mu_0 \in \text{Dist}(S)$ is the initial state distribution. The transition probability function $P : S \times A \times S \to [0, 1]$ is defined such that $\sum_{s' \in S} P(s, a, s') \in \{0, 1\}$ for any state $s \in S$ and any action $a \in A$. $\mathcal{AP}$ is a finite set of atomic propositions and $L : S \to 2^{\mathcal{AP}}$ is a labeling function which assigns to each state $s \in S$ a set of atomic propositions $L(s) \subseteq \mathcal{AP}$ that are valid at the state s. L can be extended to state sequences in the usual way, i.e., $L(\rho_1 \rho_2) = L(\rho_1) L(\rho_2)$ for $\rho_1, \rho_2 \in S^*$.*

A finite-memory, stochastic policy in the MDP is a function $\pi : S^* \to \text{Dist}(A)$. A Markovian, stochastic policy in the MDP is a function $\pi : S \to \text{Dist}(A)$. Given an MDP M and a policy π, the policy induces a Markov chain $M^\pi = \{s_t | t = 1, \ldots, \infty\}$ where s_k as the random variable for the k-th state in the Markov chain M^π and it holds that $s_0 \sim \mu_0$ and $s_{i+1} \sim P(\cdot | s_i, a_i)$ and $a_i \sim \pi(\cdot | s_i)$.

Given a finite (resp. infinite) path $\rho = s_0 s_1 \ldots s_N \in S^*$ (resp. $\rho \in S^\omega$), we obtain a sequence of labels $L(\rho) = L(s_0) L(s_1) \ldots L(s_N) \in \Sigma^*$ (resp. $L(\rho) \in \Sigma^\omega$). A path ρ satisfies the formula φ, denoted $\rho \models \varphi$, if and only if $L(\rho) \in \mathcal{L}(\mathcal{A}_\varphi)$. Given a Markov chain induced by policy π, the probability of satisfying the specification, denoted $\text{Prob}(M^\pi \models \varphi)$ is the sum of the probabilities of paths satisfying the specification.

$$\text{Prob}(M^\pi \models \varphi) \coloneqq \mathbf{E}\left[\sum_{t=0}^{\infty} \mathbf{1}(\boldsymbol{\rho}_t \models \varphi)\right].$$

where $\boldsymbol{\rho}_t = s_0 s_1 \ldots s_t$ is a path of length t in M^π.

We relate each subset $\sigma \in 2^{\mathcal{AP}}$ of atomic propositions to a propositional logic formula $\wedge_{p \in \sigma} p \wedge (\vee_{p' \in \mathcal{AP} \backslash \sigma} \neg p')$. A set of states satisfying the propositional logic formula for σ is denoted $[\![\sigma]\!]$. Given a subset of proposition set $C \subseteq 2^{\mathcal{AP}}$, let $[\![C]\!] = \bigcup_{c \in C} [\![c]\!]$. Slightly abusing the notation, we use σ to refer to the propositional logic formula that σ corresponds to. The optimal planning problem for MDPs under sc-LTL constraints is defined as follows.

Problem 1. *Given an MDP and an sc-LTL formula φ, design a policy π that maximizes the probability of satisfying the specification, i.e., $\pi \leftarrow \arg\max_\pi$ Prob$(M_\pi \models \varphi)$.*

Problem 1 can be solved with dynamic programming methods in a product MDP. The idea is to augment the state space of the MDP with additional memory states—the states in the automaton $\mathcal{A}_\varphi$, and reformulate the problem into a stochastic shortest path problem in the product MDP with the augmented state space. The reader is referred to [7] for more details. In this paper, our goal is to develop an efficient and hierarchical planner for solving Problem 1.

Remark 1. *The extension from sc-LTL to the class of LTL formulas can be made by expressing the specification formula using a deterministic Rabin automaton [9], [12] and perform two-step synthesis approach: The first step is to compute the maximal accepting end components, and the second step is to solve the Stochastic Shortest Path (SSP) MDP in the product MDP (assigning reward 1 to reaching a state in any maximal accepting end component). The details of the method can be found in [4], [7], [6]. Particularly, the tools facilitate*

symbolic computation of maximal accepting end components have been developed [5]. In the scope of this paper, we only consider sc-LTL formulas. Yet, the generalization can be made to handle planning for general LTL formulas with a similar two-step approaches.

III. Hiererachical and compositional planning under sc-LTL constraints

In this section, we present a compositional planning method for solving Problem 1. First, we propose a task decomposition method to identify a set of modular and reusable *primitive options*. Second, we establish a relation between logical conjunction/disjunction and composition of primitive options. Building on the options framework, we develop a hierarchical and compositional planning method for temporal logic constrained stochastic systems.

A. Automata-guided generation of primitive options

We present a procedure to decompose the task φ in sc-LTL into a set of primitive tasks. These primitive tasks will be composed in Sec III-B to generate the set of options in hierarchical planning.

Given a specification automaton $\mathcal{A}_\varphi = (Q, \Sigma, \delta, q_0, F)$, let the rank of a state be the minimal number of transitions to the set F of final states. Let L_k be the set of states of rank k. Thus, we have

- $L_0 = F$, and
- $L_k = \{q \mid \exists w \in \Sigma^k, \delta(q,w) \in F \text{ and } \forall \ell < k, \forall w \in \Sigma^\ell, \delta(q,w) \notin F\}$.

By definition, if DFA $\mathcal{A}_\varphi$ is coaccessible, i.e., for every state $q \in Q$ there is a word w that takes us from q to a final state, then for any state $q \in Q$, there exists L_k with a finite rank k that includes state q. Any DFA can be made coaccessible by trimming [23]. Finally, for a coaccessible DFA, we introduce a sink state to make it complete: For a state q and symbol $\sigma \in \Sigma$, if $\delta(q,\sigma)$ is undefined, then let $\delta(q,\sigma) = \mathsf{sink}$.

Based on the ranking, for each state $q \in Q$, we distinguish two types of transitions from the state:

- A transition is *progressing*: $q \xrightarrow{\sigma} q'$ and if $q \in L_k$ then $q' \in L_{k-1}$.
- A transition is *unsafe*: $q \xrightarrow{\sigma} \mathsf{sink}$, where sink is a non-accepting state with self-loops on all symbols.

Note that the DFA may have self-loops which are not included in either progressing transition or unsafe transitions. However, we shall see later that ignoring these self-loops will not affect the optimality of the planning algorithm.

A state may have multiple progressing and unsafe transitions. Let $\mathsf{Unsafe}(q)$ be the set of labels for unsafe transitions on q. Let $\mathsf{Prog}(q)$ be the set of labels for progressing transitions on q. A conditional reachability formula is defined for q as:

$$\neg\varphi_{\mathsf{Unsafe}(q)} \,\mathsf{U}\, \varphi_{\mathsf{Prog}(q)},$$

where $\varphi_{\mathsf{Unsafe}(q)} = \wedge_{\sigma\in\mathsf{Unsafe}(q)}\sigma$ and $\varphi_{\mathsf{Prog}} = \vee_{\sigma\in\mathsf{Prog}(q)}\sigma$ and $\sigma \in \Sigma$. This subformula is further decomposed into:

$$\varphi_i^q := \neg\varphi_{\mathsf{Unsafe}(q)} \,\mathsf{U}\, \sigma_i, \qquad \text{for each } \sigma_i \in \mathsf{Prog}(q).$$

We define the decomposition of φ as the collection of conditional reachability formulas

$$\Phi^{cr} = \{\varphi_i^q, q \in Q \mid i = 1, \ldots, |\mathsf{Prog}(q)|\}.$$

Next, we prune Φ^{cr} to obtain the set $\Phi \subseteq \Phi^{cr}$ of *primitive tasks*: $\phi \in \Phi \cap \Phi^{cr}$ if and only if there does not exist a set of formulas $\phi_i := \neg\psi \,\mathsf{U}\, \sigma_i \;\; i = 1, \ldots, k$, such that $\phi = \neg\psi \,\mathsf{U}\, \wedge_{i=1}^k \sigma_i$.

For each primitive task, the policy for maximizing the probability of satisfying a conditional reachability formula $\varphi_i^q \in \Phi$ can be solved through stochastic shortest path problem in the MDP M, referred to as an SSP MDP, with a formal definition follows.

Definition 2. *A (discounted) SSP MDP is defined as a tuple* $M = (S, A, P, r, \gamma, \mathsf{Goal}, \mathsf{Unsafe}, s_0)$ *where* $\mathsf{Goal} \subseteq S$ *is a set of* absorbing goal *states and* $\mathsf{Unsafe} \subseteq S$ *is a set of* absorbing unsafe *states. The transition probability function* P *satisfies* $P(s|s,a) = 1$ *for all* $s \in \mathsf{Goal} \cup \mathsf{Unsafe}$*, for all* $a \in A$*. The planning problem is to maximize the (discounted) probability of reaching* Goal *while avoiding* Unsafe*. This objective is equivalent to maximizing the total (discounted) reward with the reward function* $r : S \times A \to \mathbf{R}$ *defined as: For each* $s \in \mathsf{Goal} \cup \mathsf{Unsafe}$*,* $r(s,a) = 0$ *for all* $a \in A$*,* $r(s,a) = \mathbf{E}_{s'}\, \mathbf{1}_{\mathsf{Goal}}(s')$ *for* $s \notin$ Goal*.* $\gamma \in (0,1]$ *is the discounting factor.*

Given $\varphi_i^q = \neg\varphi_{\mathsf{Unsafe}(q)} \,\mathsf{U}\, \sigma_i$, the corresponding SSP MDP shares the same state and action sets with the underlying MDP M that models the system. The transition function is revised from the transition function in the original MDP M by making $\mathsf{Goal} = [\![\sigma_i]\!]$ and $\mathsf{Unsafe} = [\![\varphi_{\mathsf{Unsafe}(q)}]\!]$ absorbing states. Recall that $[\![\phi]\!]$ is a set of states satisfying the propositional logic formula ϕ. Note when $\gamma \neq 1$, the solution of SSP MDP is the solution of a discounted stochastic shortest path problem. The expected total reward becomes the discounted probability of satisfying the conditional reachability formula.

For stochastic shortest path problems, there exists a deterministic, optimal, Markov policy. However, to compose policies, we use a class of policies called *entropy-regulated policies*, where softmax Bellman operator is used instead of hardmax Bellman operator. Given τ as the temperature parameter, the optimal value function with softmax Bellman operator satisfies:

$$V^*(s) = \tau \log \sum_{a\in A} \exp\{(r(s,a) + \mathbf{E}_{s'\sim P(\cdot|s,a)}\, V^*(s'))/\tau\}.$$

The Q-function is:

$$Q^*(s,a) = r(s,a) + \mathbf{E}_{s'\sim P(\cdot|s,a)}\, V^*(s'),$$

and the entropy-regulated optimal policy is:

$$\begin{aligned}\pi^*(a|s) &= \exp\left((Q^*(s,a) - V^*(s))/\tau\right) \\ &= \frac{\exp(Q^*(s,a)/\tau)}{\sum_{a'}\exp(Q^*(s,a')/\tau)}.\end{aligned}$$

In the following, by optimal policy/value function, we mean the entropy-regulated optimal policy/value function unless otherwise specified.

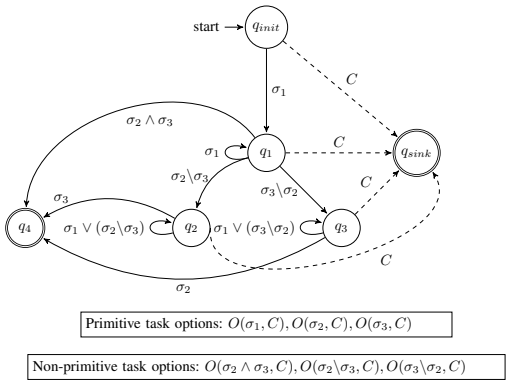

Fig. 1: The simplified DFA translation of the task $\neg C \,\mathsf{U}\, (\diamond (\sigma_1 \wedge (\diamond \sigma_2 \wedge \diamond \sigma_3)))$.

It is noted that the softmax Bellman operator is also proved equivalent to the following form [19]:

$$V^*(s) = \sum_{a \in A} \pi^*(a|s)[r(s,a) - \tau \log \pi^*(a|s) + \gamma \, \mathbf{E}_{s' \sim P(\cdot|s,a)} \, V^*(s')],$$

which means in softmax optimal planning the objective has to trade off maximizing the reward and minimizing the total entropy of the stochastic policy—such a trade off is reflected in the choice of τ. In this case, the construction of reward function needs to be different from the reward defined in Def. 2 to reduce the value diminishing problem, which means the entropy of the stochastic policy outweights the total reward for small reward signals in softmax optimal planning. In this paper, we define the reward function for the entropy-regulated MDP as:

$$r(s,a) = \alpha \cdot \mathbf{E}_{s'} \, \mathbf{1}_{\mathsf{Goal}}(s'), \quad \forall s \notin \mathsf{Goal}, \tag{1}$$

where $\alpha > 0$ is a large constant.

For each conditional reachability subtask φ_i^q, the optimal policy π in the corresponding stochastic shortest path MDP is an option $o \coloneqq \langle I, \pi, \beta \rangle$, following the definition in [26], [22] in which $I \subseteq S$ is an initiation set and is the domain of π and $\beta : S \to [0,1]$ is the termination function, defined by $\beta(s) = 1$ only if $s \in \mathsf{Unsafe} \cup \mathsf{Goal}$. We refer the option for task $\neg \psi \,\mathsf{U}\, \sigma$ as $O(\sigma, \psi)$.

Definition 3. *The option $O(\sigma, \psi)$ for subtask $\neg \psi \,\mathsf{U}\, \sigma$ is a primitive option if and only if σ is an atomic proposition and ψ is the co-safe constraint in the given task DFA.*

Example 1. *Consider the DFA in Fig. 1 and the corresponding sc-LTL task specification is $\neg C \,\mathsf{U}\, (\diamond (\sigma_1 \wedge (\diamond \sigma_2 \wedge \diamond \sigma_3)))$ (reach $[\![\sigma_1]\!]$ and then reach regions $[\![\sigma_2]\!]$ and $[\![\sigma_3]\!]$, always avoid $[\![C]\!]$). The set of atomic propositions are $\{\sigma_1, \sigma_2, \sigma_3, C\}$. The level sets are $L_0 = \{q_4\}$, $L_1 = \{q_1, q_2, q_3\}$, and $L_2 = \{q_{\mathsf{Init}}\}$. For a given state, for example, q_1, the set of labels for progressing transitions are $\{q_1 \xrightarrow{\{\sigma_2, \sigma_3\}} q_4\}$. According to Def. 3, the set of primitive tasks are $\neg C \,\mathsf{U}\, \sigma_1, \neg C \,\mathsf{U}\, \sigma_2, \neg C \,\mathsf{U}\, \sigma_3$. Therefore, three primitive options are computed as: $O(\sigma_i, C)$, for $i = 1, 2, 3$. In addition, $\neg C \,\mathsf{U}\, (\sigma_2 \wedge \sigma_3)$, $\neg C \,\mathsf{U}\, (\sigma_2 \backslash \sigma_3)$ and $\neg C \,\mathsf{U}\, (\sigma_3 \backslash \sigma_2)$ are not primitive tasks.*

B. Composition of options with disjunction and conjunction

For a given state $q \in Q$, we have obtained a set of conditional reachability formulas φ_i^q, for $i = 1, \ldots, n$ where n is the total number of primitive tasks generated from q. However, a progress transition can be made by satisfying any of the conditional reachability formula. That is to say, we may be interested in synthesizing option that maximizes $\vee_{i=1}^n \varphi_i^q$, or potentially the conjunction/disjunction of a subset of all primitive tasks Φ. A naive approach is to take the new specification $\vee_{i=1}^n \varphi_i^q$, construct a DFA, and synthesize the optimal policy using methods [7], [2] for MDPs under temporal logic constraints. However, we are interested in finding a "good enough" policy given the new specification via composing existing policies. The problem is formally stated as follows.

Problem 2. *Given two conditional reachability formulas $\varphi_1 \coloneqq \neg \varphi_{\mathsf{Unsafe}} \,\mathsf{U}\, \sigma_1$, and $\varphi_2 \coloneqq \neg \varphi_{\mathsf{Unsafe}} \,\mathsf{U}\, \sigma_2$, construct a* good enough *policy given the goal of maximizing the probability of satisfying the disjunction: $\varphi_1 \vee \varphi_2$; or b) the conjunction: $\varphi_1 \wedge \varphi_2$; or c) the exclusion $\varphi_1 \backslash \varphi_2$ or $\varphi_2 \backslash \varphi_1$.*

The definition of "good enough" policies will be provided later. Here, we consider the case when φ_1 and φ_2 share the same set of unsafe states. Particularly, if $\varphi_1 = \varphi_i^q$, and $\varphi_2 = \varphi_j^q$ for some $i \neq j$ and $q \in Q$, then it is always the case that φ_1 and φ_2 share the same set of unsafe states. Next, we propose a method for policy composition based on generalized logic conjunction/disjunction [8], which is briefly introduced below.

Generalized conjunction/disjunction: Generalized Conjunction-Disjunction (GCD) was introduced in [8] for quantitative reasoning with logic formulas. GCD is a mapping $\lambda : [0,1]^n \to [0,1]$, $n > 1$, that has properties similar to logic conjunction and disjunction. The level of similarity is adjustable using a parameter η, called the conjunction degree (andness). Formally, let $x_1, \ldots, x_n$ be variables representing the *level of truthfulness* for a set of logic formulas $\varphi_1, \ldots, \varphi_n$, the GCD formula $\lambda(x_1, \ldots, x_n|\eta)$, which unifies conjunction and disjunction, is defined as,

$$\lambda(x_1, \ldots, x_n|\eta) = \frac{1}{\eta} \log(\sum_{i=1}^{n} W_i \exp(\eta x_i)), \quad 0 < |\eta| < +\infty, \tag{2}$$

where when $\eta \to \infty$, $\lim_{\eta \to \infty} \lambda(x_1, x_2, ..., x_n|\eta) = x_1 \vee x_2 \vee ... \vee x_n$ recovers the conventional disjunction, and when $\eta \to -\infty$, $\lim_{\eta \to -\infty} \lambda(x_1, x_2, ..., x_n|\eta) = x_1 \wedge x_2 \wedge ... \wedge x_n$ recovers the conventional conjunction. For any $\eta \in (-\infty, +\infty)$, $\lambda(x_1, x_2, ..., x_n|\eta)$ returns a *level of truthfulness* of a GCD. In addition, parameter W_i is the corresponding weight (or relative importance) of the i-th formula, for $i = 1, \ldots, n$. In this paper, we select $W_i = 1$ by assuming that all formulae are equally important.

We use GCD to compose a "good enough" policy, that is, the optimal policy in semi-MDP planning.

Definition 4. *[26], [22] Given an MDP $M = (S, A, P, \gamma, r)$ and a set $O = \{o_i = \{\mathcal{I}_i, \pi_i, \beta_i\}, i = 1, \ldots, n\}$ of options where $\mathcal{I}_i$ is a set of initial states, $\beta_i : S \to [0, 1]$ is a termination condition and $\pi_i : S \to Dist(A)$ is a policy in the MDP M. An* option policy *in M is a function $\pi^o : S \to Dist(O)$. let Π^o be the set of option policies in M. Given a reward function $r : S \times A \to \mathbf{R}$, an option policy is optimal if and only if it maximizes the total discounted reward:*

$$\pi^{o,*} = \arg\max_{\pi^o \in \Pi^o} \mathbf{E}_{\pi^o} \sum_{t=1}^{\infty} r(s_t, o_t) \tag{3}$$

where $r(s_t, o_t) = \mathbf{E}_{\pi_{o_t}} \left[r_{t+1} + \gamma r_{t+2} + \ldots \gamma^{k-1} r_{t+k} \right]$ is the total accumulated rewards when the policy π_{o_t} of option o_t is applied to the MDP for the duration of k steps.

Assumption 1. *Given a conditional reachability subtask φ_i, the optimal policy π that maximizes the probability of satisfying φ_i induces in M an absorbing Markov chain M^{π_i}.*

In other words, with probability one, the system will visit an absorbing state.

Lemma 1. *Assuming 1, given a set $O = \{o_i = \{\mathcal{I}_i, \pi_i, \beta_i\}, i = 1, \ldots, n\}$ of options where $\mathcal{I}_i = S$, π_i is the softmax optimal policy for maximizing the probability of satisfying $\varphi_i = \neg\varphi_{\mathsf{Unsafe}} \,\mathsf{U}\, \sigma_i$, i.e., the SSP MDP $M_i = (S, A, P, r, \gamma, [\![\sigma_i]\!], [\![\varphi_{\mathsf{Unsafe}}]\!], s_0)$ $\beta_i(s) = \mathbf{1}_X(s)$ where $X = \bigcup_{j=1}^{n} [\![\sigma_j]\!] \cup \mathsf{Unsafe}$ is the termination function. In the MDP M, the optimal option policy for maximizing the GCD $\lambda(\varphi_1, \ldots, \varphi_n \mid \eta; s)$ for any $s \in S$, with unit weights, i.e., $W_i = 1$, $i = 1, \ldots, n$, is*

$$\pi^o(s)[j] = \frac{\sum_{i=1}^{n} \exp(\eta Q_i(s, o_j))}{\sum_{o_k \in O} \sum_{i=1}^{n} \exp(\eta Q_i(s, o_k))}, \text{ for } j = 1, \ldots, n. \tag{4}$$

where $Q_i(s, o_j)$ is the evaluation of policy π_j with respect to specification φ_i.

Proof. For state s and an option o_j, by definition of GCD, we have

$$\lambda(\varphi_1, \ldots, \varphi_n \mid \eta; s, o_j) = \frac{1}{\eta} \log(\sum_{i=1}^{n} \exp(\eta\, \mathbf{E}_j[\mathbf{1}(s \models \varphi_i)])).$$

where the expectation $\mathbf{E}_j[\cdot]$ is taken over paths in the Markov chain M^{π_j}. Given the option o_j, $\mathbf{E}_j\, \mathbf{1}(s \models \varphi_i) = Q_i(s, o_j)$—the evaluation of policy π_j with respect to specification φ_i and thus $\lambda(\varphi_1, \ldots, \varphi_n \mid \eta; s, o_j) = \frac{1}{\eta} \log(\sum_{i=1}^{n} \exp(\eta Q_i(s, o_j)))$.

Next, we distinguish two cases between conjunction and disjunction:

Case I (Disjunction): $\eta > 0$: To maximize $\lambda(\varphi_1, \ldots, \varphi_n \mid \eta; s)$ given an option-only decision rule $\pi^o(s) : O \to [0, 1]$, based on the softmax operator, when $\eta > 0$, $\pi^o(s)[j] \propto \exp(\lambda(\varphi_1, \ldots, \varphi_n \mid \eta; s, o_j)/\tau)$ where $\tau > 0$ is a temperature parameter. When $\tau = 1/\eta$, then

$$\pi^o(s)[j] = \frac{\sum_{i=1}^{n} \exp(\eta Q_i(s, o_j))}{\sum_{o_k \in O} \sum_{i=1}^{n} \exp(\eta Q_i(s, o_k))}, \quad \text{for } j = 1, \ldots, n. \tag{5}$$

which is the same as in (4).

Case II(Conjunction): $\eta < 0$: In this case, we have

$$\lambda(\varphi_1, \ldots, \varphi_n \mid \eta; s, o_j) = -\frac{1}{|\eta|} \log(\sum_{i=1}^{n} \exp(-|\eta|\, \mathbf{E}_j[\mathbf{1}(s \models \varphi_i)])). \tag{6}$$

To maximize $\lambda(\varphi_1, \ldots, \varphi_n \mid \eta; s)$ is equivalent to minimize $\frac{1}{|\eta|} \log(\sum_{i=1}^{n} \exp(-|\eta| x_i))$ where x_i is the level of truthfulness for formula φ_i. Further, minimizing $\frac{1}{|\eta|} \log(\sum_{i=1}^{n} \exp(-|\eta| x_i))$ is equivalent to minimizing $\log(\sum_{i=1}^{n} \exp(\eta x_i))$ as $\eta = -|\eta|$, which is exactly the opposite case to that of disjunction. Thus, the optimal option policy satisfies $\pi^o(s)[j] \propto \exp(-\lambda(\varphi_1, \ldots, \varphi_n \mid \eta; s, o_j)/\tau)$ (softmin operator) where $\tau > 0$ is a temperature parameter. When $\tau = -1/\eta$, then given $\lambda(\varphi_1, \ldots, \varphi_n \mid \eta; s, o_j) = Q_i(s, o_j)$, for $j = 1, \ldots, n$,

$$\begin{aligned} \pi^o(s)[j] &= \frac{\sum_{i=1}^{n} \exp(-Q_i(s, o_j)/\tau)}{\sum_{o_k \in O} \sum_{i=1}^{n} \exp(-Q_i(s, o_k)/\tau)} \\ &= \frac{\sum_{i=1}^{n} \exp(\eta Q_i(s, o_j))}{\sum_{o_k \in O} \sum_{i=1}^{n} \exp(\eta Q_i(s, o_k))}, \end{aligned}$$

which is the same as in (4). Thus the proof is completed. □

Intuitively, for the case of disjunction, this policy makes sense because given π_j is the optimal policy for satisfying φ_j, $\pi^o(s)$ selects policy j with a likelihood proportional to $\exp(\eta Q_j(s, o_j))$ plus some bonus $\sum_{i \neq j} \exp(\eta Q_j(s, o_i))$ obtained by satisfying other specifications. Given two specifications φ_1 and φ_2, since the disjunction can be satisfied by satisfying only one of these two, then this policy exponentially prefers π_1 to π_2 if φ_1 has a higher probability to be satisfied.

The situation is complicated for conjunction. The conjunction of two formulas, $\varphi_1 \coloneqq \neg\varphi_{\mathsf{Unsafe}} \,\mathsf{U}\, \sigma_1$ and $\varphi_2 \coloneqq \neg\varphi_{\mathsf{Unsafe}} \,\mathsf{U}\, \sigma_2$, is $(\neg\varphi_{\mathsf{Unsafe}} \,\mathsf{U}\, \sigma_1) \wedge (\neg\varphi_{\mathsf{Unsafe}} \,\mathsf{U}\, \sigma_2)$. For any state, the planner will select the option i with a probability that is inverse proportional to $Q(s, o_i)$, i.e., if φ_1 has a lower probability to be satisfied, then option 1 has a higher probability to be chosen. Once it reaches $[\![\sigma_1]\!]$, it will select option 2 with a higher probability because $Q(s, o_2) < Q(s, o_1) = 1$ for $s \in [\![\sigma_1]\!]$ to force a visit to $[\![\sigma_2]\!]$. However, without memory, the planner will alternate between two options indefinitely, or until it reaches an unsafe region. Thus the conjunction on multiple memoryless options will require additional memory to manage the switching condition of terminating function among goals. However, when the intersection $[\![\sigma_1 \wedge \sigma_2]\!] \neq \emptyset$ and either option has a nonzero probability of reaching the intersection, a memoryless composed option may eventually reach a state in $[\![\sigma_1 \wedge \sigma_2]\!]$. Thus, we may approximate the solution of $\neg\varphi_{\mathsf{Unsafe}} \,\mathsf{U}\, (\sigma_1 \wedge \sigma_2)$ with a memoryless composed option for the conjunction. In this paper, we only focus on the memoryless option, further discussions on the additional memory method will be included in the future work.

Based on the proof of Lemma 1, we can further show that the GCD method is indeed invertible to compute the

exclusion $\varphi_1 \backslash \varphi_2$. Since $(\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \backslash \varphi_2) = \varphi_1$, we can apply generalized disjunction in Eq. (2) to the MDP M, then

$$\begin{aligned}\exp(\eta\, \mathbf{E}_1[\mathbf{1}(s \models \varphi_1)]) &\approx \exp(\eta\lambda(\varphi_1 \wedge \varphi_2, \varphi_1 \backslash \varphi_2 | \eta; s, o_1) \\ &= \exp(\eta\, \mathbf{E}_1[\mathbf{1}(s \models \varphi_1 \wedge \varphi_2)]) \\ &+ \exp(\eta\, \mathbf{E}_1[\mathbf{1}(s \models \varphi_1 \backslash \varphi_2)]).\end{aligned}$$

Therefore, the Q function of task exclusion can be computed by

$$\begin{aligned}Q(s, o_1) &= \mathbf{E}_1[\mathbf{1}(s \models \varphi_1 \backslash \varphi_2)] \\ &\approx \frac{1}{|\eta|} \log(\exp(\eta\, \mathbf{E}_1[\mathbf{1}(s \models \varphi_1)]) \\ &- \exp(\eta\, \mathbf{E}_1[\mathbf{1}(s \models \varphi_1 \wedge \varphi_2)])).\end{aligned}$$

Finally, the set of options $\mathcal{O}$ includes both primitive options—one for each primitive tasks and composed options using GCD. The set of actions A is now augmented with options $\mathcal{O}$, and the optimal policy can be obtained by solving the following planning problem in the product MDP with an augmented action space.

Definition 5. *Given a labeled MDP $M = (S, A, P, \mu_0, L)$ and an sc-LTL formula φ, represented by a DFA $\mathcal{A}_\varphi = (Q, \Sigma, \delta, q_0, F)$, a set $\mathcal{O}$ of options, the* product MDP with macro- and micro-actions *is defined by*

$$M \ltimes \mathcal{A}_\varphi = (S \times Q, A \cup \mathcal{O}, \bar{P}, \bar{\mu}_0),$$

where the probabilistic transition function is defined by: $\bar{P}((s', q')|(s, q), a) = P(s'|s, a)$ if $q' = \delta(q, L(s'))$, $a \in A$, and $\bar{P}((s', q')|(s, q), o) = \prod_{t=0}^{k} P(s_{t+1}|s_t, a_t)$ where $s_0 = s$ and $s_{k+1} = s'$, k is the number of steps that are taken under the (policy) of option o before being interrupted by triggering a discrete transition in the automata, $\{s_t, a_t, t = 0, \ldots\}$ is Markov chain induced by the policy of option o, and $q' = \delta(q, L(s'))$. The reward function is defined by $r((s, q), a) = \mathbf{E}_{(s',q')}\, \mathbf{1}(q' \in F)$, when $q \notin F$, and $r((s, q), o) = \mathbf{E}_{\pi_o}\left[\sum_{t=0}^{k-1} \gamma^t r(s_t, a_t)\right]$ is the total accumulated rewards when the policy π_o of option o is applied to the MDP for the duration of k steps before it is interrupted.

Note that when the chain is absorbing for any policies of options, then the discounting factor γ can be set to 1. By setting $\gamma \neq 1$, we will encourage the behavior of satisfying the specification in a less number of steps.

Remark 2. *The planning is performed in the product MDP with both actions and options. It is ensured to recover the optimal policy had only actions being used. Having options helps to speed up the convergence. Note that even if self-loops in the DFA have not been considered in generating primitive and composed options, the optimality of the planner will not be affected.*

Remark 3. *For the labeled MDP in Def. 5, let $N_S, N_Q, N_A, N_\mathcal{O}$ denote the size of $S, Q, A, \mathcal{O}$. In value iteration algorithm, the space complexity of the planning performed with actions only is $\mathcal{O}(N_S N_Q N_A)$, while the planning performed with both actions and options consumes $\mathcal{O}(N_S N_Q (N_A + N_O) + N_S N_A N_O)$ of memory, for storing the policy with both action and options and these option policies. If we assume $\mathcal{O}(N_Q) = \mathcal{O}(N_A) = \mathcal{O}(N_\mathcal{O}) = \mathcal{O}(\bar{N})$, and $\mathcal{O}(N_S) \gg \mathcal{O}(\bar{N})$, the two methods will have the same space complexity, that is $\mathcal{O}(\bar{N}^2 N_S)$.*

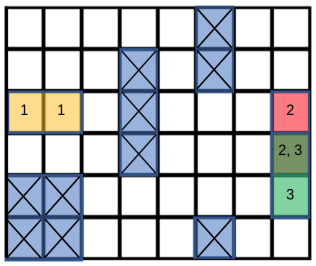

Fig. 2: The state space setting for the task cases in a 2-D grid world.

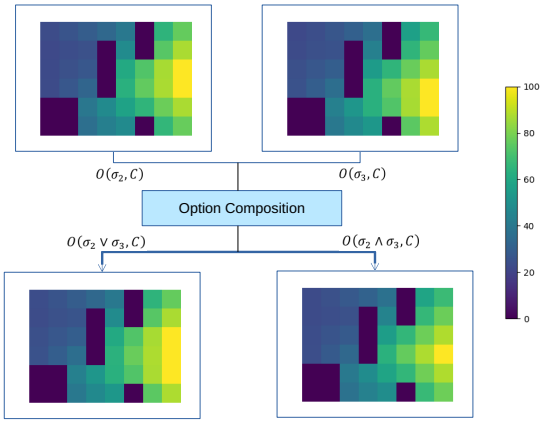

Fig. 3: The figure shows the converging distribution of entropy regularized value functions of primitive options $O(\sigma_2, C)$, $O(\sigma_3, C)$ and their compositions to approximate general conjunction and disjunction.

IV. Case Study

This section illustrates our compositional planning method using robotic motion planning problems. All experiments in this section are performed on a computer equipped with an Intel® Core™ i7-5820K and 32GB of RAM running a python 3.6 script on a 64-bit Ubuntu® 16.04 LTS.

The environment is modeled as a 2D grid world, shown in Fig. 2. The robot has actions: Up, Down, Left, and Right. With probability 0.7, the robot arrives at the cell it intended with the action and has a probability of $(1-0.7)/(|Adj|-1)$ to transit to other adjacent locations, where $|Adj|$ is the number of adjacent cells, including the current cell when the action is applied. Especially, when the transition hits the boundary of the grid world, the probability under that transition adds to the probability of staying put. The discounting factor γ is fixed 0.9. Fig. 2 shows a grid world of size 6×8. In this grid world, there are a set of obstacles (unsafe grids) marked with cross signs and several regions of interests, marked with numbers. The cell marked with number i is

labeled with symbol σ_i, for $i = 1, 2, 3$. Region marked with number $2, 3$ is the nonempty intersection satisfying $\sigma_2 \wedge \sigma_3$.

We consider three sc-LTL tasks $\varphi = \{\varphi_1, \varphi_2, \varphi_3\}$ where

$$\varphi_1 \coloneqq \neg C \,\mathsf{U}\, (\Diamond (\sigma_1 \wedge (\Diamond \sigma_2 \wedge \Diamond \sigma_3))),$$

(reach $[\![\sigma_1]\!]$ and then reach regions $[\![\sigma_2]\!]$ and $[\![\sigma_3]\!]$, while avoiding $[\![C]\!]$.)

$$\varphi_2 \coloneqq \neg C \,\mathsf{U}\, (\Diamond ((\sigma_1 \vee \sigma_3) \wedge \Diamond \sigma_2)),$$

(reach either $[\![\sigma_1]\!]$ or $[\![\sigma_3]\!]$ and then reach $[\![\sigma_2]\!]$, while avoiding $[\![C]\!]$.)

$$\varphi_3 \coloneqq \neg C \,\mathsf{U}\, (\Diamond ((\sigma_1 \vee \sigma_2) \wedge \Diamond (\sigma_2 \wedge \sigma_3))),$$

(reach either $[\![\sigma_1]\!]$ or $[\![\sigma_3]\!]$ and then reach either $[\![\sigma_2]\!]$ or $[\![\sigma_3]\!]$, while avoiding $[\![C]\!]$.)

Figure 1.a shows the DFA of φ_1. We omit the DFAs for φ_2 and φ_3 given the limited space. Based on the set of tasks, the following set of primitive tasks are generated:

$$\phi_1 \coloneqq \neg C \,\mathsf{U}\, \sigma_1, \quad \phi_2 \coloneqq \neg C \,\mathsf{U}\, \sigma_2, \quad \phi_3 \coloneqq \neg C \,\mathsf{U}\, \sigma_3.$$

For each conditional reachability specifications, we formulate the SSP MDP and compute the softmax optimal policy using temperature parameter $\tau = 1$ and the reward function defined in Eq. 1, where parameter α is selected to be 100 to prevent the reward being outweighted by the entropy term.

Our first experiment is to demonstrate the composition of policies based on GCD.

Policy composition: We use composition of $\phi_2 = \neg C \,\mathsf{U}\, \sigma_2$, $\phi_3 = \neg C \,\mathsf{U}\, \sigma_3$ to generate the options $\neg C \,\mathsf{U}\, (\sigma_2 \oplus \sigma_3)$ where $\oplus \in \{\wedge, \vee\}$. To validate that the composed policies are "good enough", we compare the values of the optimal policies for these two formulas, computed using standard value iteration, and the values of the composed policies obtained via Lemma 1. For comparison, we consider relative errors $e_{2,\oplus} = \|V^{\pi^\oplus} - V^{\pi^{o,\oplus}}\|_2 / \|V^{\pi^{o,\oplus}}\|_2$ and $e_{\infty,\oplus} = \|V^{\pi^\oplus} - V^{\pi^{o,\oplus}}\|_\infty / \|V^{\pi^{o,\oplus}}\|_\infty$. We have $e_{2,\vee} \approx 10^{-4}$, $e_{\infty,\vee} \approx 10^{-4}$, $e_{2,\wedge} \approx 10^{-3}$, $e_{\infty,\wedge} \approx 10^{-3}$.

Figure 3 shows heat maps comparing two option value functions for the case of disjunction or conjunction. The shaded areas represent globally unsafe regions with V values always fixed zero during the iteration. In Fig. 3, all value distributions are in the range between 0 to 100 because we scaled the reward of 1 by 100 to avoid entropy term outweighing the total reward. From Fig. 3 (c), it is shown that the value of regions marked by either 2 or 3 is highest, corresponding to the disjunction. In the case of conjunction in Fig. 3 (d), the intersection of regions marked by 2 and 3 has the highest value.

Next, we compare the convergence between three different planning methods for three task specifications: planning with only micro-action (action), planning with macro-actions (primitive and composed options), and planning with both micro- and macro-actions (mixed). In addition, we compared the optimality of these planners with optimal planning with only micro-action using hardmax Bellman operator as the baseline. The results to be compared are the speed of convergence and the optimality of the converged policy.

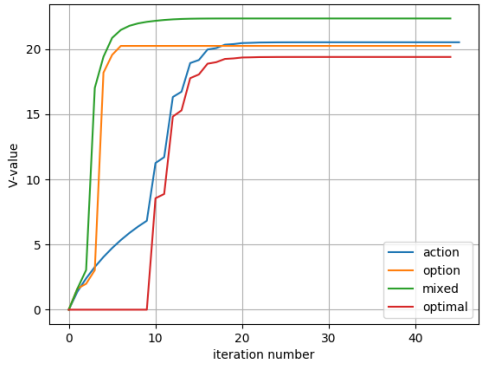

Fig. 4: Value iteration of four different planners for task φ_1.

Figure 4 shows the convergence of value function evaluated at the initial state with $s_0 = (3, 3)$ given specification φ_1. It shows that among all three methods, the mixed planner converges the fastest. Both option and mixed planners converge much faster than action planner: The action planner converges after about 20 iterations, while the other two converges after 6-9 iterations. It is also interesting to notice that the policy obtained by the mixed planner achieves a higher value comparing to the action planner. This is because the entropy of policy weighs less in the policy obtained by the mixed planner comparing to that obtained by the action planner in softmax optimal planning. Moreover, the influence of entropy can also be observed between softmax action planner (action) and hardmax action planner (optimal), where the two planners converge almost at the same rate but to different values since softmax adds additional policy entropy to the total value.

Table I compares the performance of three planners in the given three tasks. The entry p refers to the probability of satisfying the specification from an initial state s_0 under the optimal policies obtained by three planners. Number n is the number of value iterations taken for each method to converge with a pre-defined error tolerance threshold 0.001. Number t shows the CPU time costs.

In converging iteration numbers and CPU times, the advantage of option planner outperforms the other two significantly. Considering the additional time cost from learning the primitive options, the experiment shows that every single option takes in average 30 iterations to converge in a 6×8 option state space, and in total costs 0.5 seconds to compute all the options for primitive tasks. However, these options only need to be solved for once and are reused across three tasks. The composition of options takes negligible computation time (0.001 seconds on average for each composition). The performance loss of option planner, comparing with the global optimal planner (using hardmax Bellman), is only 13% of the optimal value for task φ_1 and negligible for tasks φ_2 and φ_3. Composition makes temporal logic planning flexible: If we change a task from φ_1 to φ_2, then the option and mixed planner can quickly generate new, optimal policies without reconstructing primitive option.

Last, we evalute and compare policies generated by option,

TABLE I: Comparison: Performance, convergence rate, and computation cost

	$p(\varphi\|s_0)$			iteration(n)			runtime($t:sec$)		
φ	φ_1	φ_2	φ_3	φ_1	φ_2	φ_3	φ_1	φ_2	φ_3
$Optimal$	0.54	0.88	0.89	46	24	24	0.6	0.28	0.22
$Action$	0.35	0.87	0.87	45	25	23	0.6	0.32	0.21
$Option$	0.47	0.88	0.88	8	6	5	0.1	0.03	0.02
$Mixed$	0.41	0.88	0.87	34	18	15	0.4	0.17	0.12

TABLE II: The error (2-Norm and ∞-Norm) between policies obtained with different methods

	Option π_1 vs Action π_2			Mixed π_1 vs Action π_2		
φ	φ_1	φ_2	φ_3	φ_1	φ_2	φ_3
2-Norm	0.0076	0.0345	0.0394	0.0022	0.0009	0.001
∞-Norm	0.0249	0.1755	0.17	0.0088	0.0071	0.0067

mixed and action planners. Table II shows the relative errors in 2-norm and infinite-norm, i.e., $e(\pi_1, \pi_2) = \|V^{\pi_1} - V^{\pi_2}\|/\|V^{\pi_2}\|$. Both the option planner and mixed planner have negligible deviation to (less than 3%) to the action planner, while the option planner is clearly less similar to the action planner comparing to the mixed planner, especially on the ∞-norm error.

V. CONCLUSIONS

This paper presents a compositional method for MDP planning constrained by sc-LTL specifications. The method formally relates the composition of stochastic policies for logical task specifications and generalized conjunction/disjunction (GCD) in logic. We show that the composition based on GCD is equivalent to the semi-MDP planning under the softmax Bellman operator. The semi-MDP planning with both primitive options and composed options achieves much faster convergence, comparing to planning with actions, or a mixture of actions and options, with a relatively small performance loss. Besides, our compositional planning method brings in more flexibility in re-using and composing options for one task in a different task in the same stochastic system with the same labeling function.

Although composed options may not be optimal, the convergence to the global optimal policy is guaranteed as the options framework uses both macro-actions/options and micro-actions (actions in the original MDP). The future direction along this line of work is to exploit the compositional planning in model-free reinforcement learning and to improve the scalability of the planning method by replacing value/policy iteration with approximate dynamic programming [3]. We will further investigate finite-memory policy composition to handle the issue raised from conjunction-based composition.

REFERENCES

[1] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence*, 2017.

[2] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[3] Dimitri P. Bertsekas. *Neuro-dynamic programming*, pages 2555–2560. Springer, 2009.

[4] Andrea Bianco and Luca de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Foundations of Software Technology and Theoretical Computer Science*, pages 499–513, 1995.

[5] Krishnendu Chatterjee, Monika Henzinger, Manas Joglekar, and Nisarg Shah. Symbolic algorithms for qualitative analysis of Markov decision processes with Büchi objectives. *Formal Methods in System Design*, 42(3):301–327, Jun 2013.

[6] X. C. Ding, S. L. Smith, C. Belta, and D. Rus. LTL control in uncertain environments with probabilistic satisfaction guarantees*. *IFAC Proceedings Volumes*, 44(1):3515 – 3520, 2011.

[7] X. C. Ding, S. L. Smith, C. Belta, and D. Rus. MDP optimal control under temporal logic constraints. In *IEEE Conference on Decision and Control and European Control Conference*, pages 532–538, Dec 2011.

[8] Jozo J Dujmovic and Henrik Legind Larsen. Generalized conjunction/disjunction. *International Journal of Approximate Reasoning*, 46(3):423–446, 2007.

[9] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. Spot 2.0 – a framework for LTL and ω-automata manipulation. In *International Symposium on Automated Technology for Verification and Analysis*, pages 122–129. Springer, 2016.

[10] Jie Fu, Shuo Han, and Ufuk Topcu. Optimal control in markov decision processes via distributed optimization. In *IEEE Conference on Decision and Control (CDC)*, pages 7462–7469, 2015.

[11] Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In *International Conference on Computer Aided Verification*, pages 53–65. Springer, 2001.

[12] D. Giannakopoulou and K. Havelund. Automata-based verification of temporal properties on running programs. In *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, pages 412–416, Nov 2001.

[13] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Autonomous skill acquisition on a mobile manipulator. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pages 1468–1473, 2011.

[14] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

[15] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 3675–3683, 2016.

[16] Orna Kupferman and Moshe Y Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

[17] Morteza Lahijanian, Sean B Andersson, and Calin Belta. Temporal logic motion planning and control with probabilistic satisfaction guarantees. *IEEE Transactions on Robotics*, 28(2):396–409, 2012.

[18] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems: Specification*. Springer Science & Business Media, 2012.

[19] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2772–2782, 2017.

[20] Ronald Parr and Stuart J Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems*, pages 1043–1049, 1998.

[21] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[22] Doina Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

[23] Jacques Sakarovitch and Reuben Thomas. *Elements of automata theory*, volume 6. Cambridge University Press, 2009.

[24] D Silver and K Ciosek. Compositional planning using optimal option models. In *Proceedings of the 29th International Conference on Machine Learning*, volume 2, pages 1063–1070, 2012.

[25] Jonathan Sorg and Satinder Singh. Linear options. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 31–38, 2010.

[26] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.