Compositional Reinforcement Learning from Logical Specifications

Kishor Jothimurugan
University of Pennsylvania
kishor@seas.upenn.edu

Suguman Bansal
University of Pennsylvania
suguman@seas.upenn.edu

Osbert Bastani
University of Pennsylvania
obastani@seas.upenn.edu

Rajeev Alur
University of Pennsylvania
alur@seas.upenn.edu

Abstract

We study the problem of learning control policies for complex tasks given by logical specifications Typically, these approaches automatically generate a reward function from a given specification and use a suitable reinforcement learning algorithm to learn a policy that maximizes the expected reward. These approaches, however, scale poorly to complex tasks that require high-level planning. In this work, we develop a compositional learning approach, called DiRL, that leverages the specification to decompose the task into a high-level planning problem and a set of simpler reinforcement learning tasks. An evaluation of DiRL on a challenging control benchmark with continuous state and action spaces demonstrates that it outperforms state-of-the-art baselines.

Keywords Temporal Logic, Reinforcement Learning

1 Introduction

Reinforcement learning has emerged as a promising approach to automatically learning controllers for challenging robotics tasks. For instance, it has been applied to challenging control tasks such as walking [4] and grasping [3], control of multi-agent systems [6, 10], and control from visual inputs [8]. A key challenge facing reinforcement learning is the difficulty in specifying the goal. Typically, reinforcement learning algorithms require the user to provide a reward function that encodes the desired task. However, for complex, long-horizon tasks, providing a suitable reward function can be a daunting task, requiring the user to manually compose rewards for individual subtasks. Poor reward functions can make it hard for the reinforcement learning algorithm to achieve the goal; for instance, it can result in reward hacking [1], where the agent learns to optimize the reward function without achieving the goal.

Recent work has proposed a number of high-level languages for specifying RL tasks [2, 5, 7, 9, 12]. A key feature of these approaches is that they enable the user to specify tasks *compositionally*—i.e., the user can independently specify a set of short-term subgoals, and then ask the agent to perform a complex task that involves achieving some of these subgoals. Existing approaches for learning from high-level specifications typically generate a reward function, which is then used by an off-the-shelf RL algorithm to learn a policy. Recent works [5, 13] have proposed RL algorithms that exploit the structure of the specification to improve learning. However, these algorithms are based on model-free RL at both the high- and low-levels instead of model-based RL. Model-free RL is known to be effective at learning low-level control tasks; however, at the high-level, it is unable to exploit the large amount of available structure. Thus, these approaches scale poorly to long horizon tasks involving complex decision making.

We propose DiRL– a novel compositional reinforcement learning algorithm that reduces the policy synthesis problem into a high-level planning problem and a set of simpler reinforcement learning tasks. Then, it interleaves model-based high-level planning with model-free RL to compute a policy that tries to maximize the probability of satisfying the specification. In more detail, our algorithm begins by converting the user-provided specification into an abstract graph whose edges encode the subtasks, and whose vertices encode regions of the state space where each subtask is considered achieved. Then, it uses a Dijkstra-style forward graph search algorithm to compute a sequence of subtasks for achieving the specification, aiming to maximize the success probability. Rather than compute a policy to achieve each subtask beforehand, it constructs them on-the-fly for a subtask as soon as Dijkstra's algorithm requires the cost of that subtask.

We empirically evaluate our approach on an environment with continuous state and action spaces. We demonstrate that DiRL significantly outperforms state-of-the-art deep RL algorithms for learning policies from specifications.

2 Problem Formulation

MDP. We consider a *Markov decision process (MDP)* $\mathcal{M} = (S, A, P, \eta)$ with continuous states $S \subseteq \mathbb{R}^n$, continuous actions $A \subseteq \mathbb{R}^m$, transitions $P(s, a, s') = p(s' \mid s, a) \in \mathbb{R}_{\geq 0}$ (i.e., the probability density of transitioning from state s to state s' upon taking action a), and initial states $\eta : S \to \mathbb{R}_{\geq 0}$ (i.e., $\eta(s)$ is the probability density of the initial state being s). A

SYNT 21, ,
2021.

trajectory $\zeta \in \mathcal{Z}$ is either an infinite sequence $\zeta = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$ or a finite sequence $\zeta = s_0 \xrightarrow{a_0} \cdots \xrightarrow{a_{t-1}} s_t$ where $s_i \in S$ and $a_i \in A$. We let $\mathcal{Z}_f$ denote the set of finite trajectories. A (deterministic) *policy* $\pi : \mathcal{Z}_f \to A$ maps a finite trajectory to a fixed action.

Specification language. We consider the specification language Spectrl for specifying reinforcement learning tasks [7]. A specification ϕ in this language is a logical formula over trajectories that indicates whether a given trajectory ζ successfully accomplishes the desired task. Formally, a specification is defined over a set of *predicates* $\mathcal{P}$, where every $p \in \mathcal{P}$ is associated with a function $p : S \to \mathbb{B}$; we say a state s *satisfies* p (denoted $s \models p$) if and only if $p(s) = \texttt{true}$. Finally, the syntax of Spectrl specifications is given by

$$\phi \;::=\; \texttt{achieve}\ b \mid \phi_1\ \texttt{ensuring}\ b \mid \phi_1; \phi_2 \mid \phi_1\ \texttt{or}\ \phi_2,$$

where $b \in \mathcal{P}$. In this case, each specification ϕ corresponds to a function $\phi : \mathcal{Z} \to \mathbb{B}$, and we say $\zeta \in \mathcal{Z}$ satisfies ϕ (denoted $\zeta \models \phi$) if and only if $\phi(\zeta) = \texttt{true}$. Letting ζ be a finite trajectory of length t, this function is defined by

$\zeta \models \texttt{achieve}\ b$	if $\exists\ i \le t,\ s_i \models b$
$\zeta \models \phi\ \texttt{ensuring}\ b$	if $\zeta \models \phi$ and $\forall\ i \le t,\ s_i \models b$
$\zeta \models \phi_1; \phi_2$	if $\exists\ i < t,\ \zeta_{0:i} \models \phi_1$ and $\zeta_{i+1:t} \models \phi_2$
$\zeta \models \phi_1\ \texttt{or}\ \phi_2$	if $\zeta \models \phi_1$ or $\zeta \models \phi_2$.

Learning from Specifications. Given an MDP $\mathcal{M}$ with unknown transitions and a specification ϕ, our goal is to compute a policy $\pi^* : \mathcal{Z}_f \to \mathcal{A}$ such that

$$\pi^* \in \arg\max_{\pi} \Pr_{\zeta \sim \mathcal{D}_\pi}[\zeta \models \phi],$$

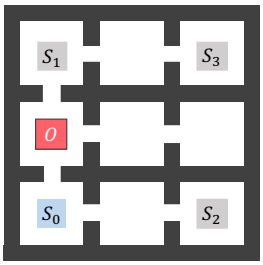
where $\mathcal{D}_\pi$ is the distribution over infinite trajectories generated by π. In other words, we want to learn a policy π^* that maximizes the probability that a generated trajectory ζ satisfies the specification ϕ.

3 DiRL: Algorithm Sketch

Consider an RL-agent in the environment of interconnected rooms in Figure 1 (top). The agent is initially in the blue box, and their goal is to navigate to either the top-left room S_1 or the bottom-right room S_2, followed by the top-right room S_3, all the while avoiding the red block O. Its goal is formally captured by the specification (middle). This specification is comprised of four simpler RL subtasks—namely, navigating between corners of the room while avoiding the obstacle. First, based on the specification alone, our approach DiRL constructs the abstract graph $\mathcal{G}_{\text{ex}}$ (see bottom) whose vertices represent the initial region and the three subgoal regions, and the edges correspond to subtasks (labeled with a safety constraint that must be satisfied).

However, $\mathcal{G}_{\text{ex}}$ by itself is insufficient to determine the optimal path—e.g., it does not know that there is no path leading directly from S_2 to S_3, which is a property of the environment. These differences can be represented as edge costs in $\mathcal{G}_{\text{ex}}$. In particular, DiRL trains a policy π_e for each edge e in , and sets the cost of e to be $c(e; \pi_e) = -\log P(e; \pi_e)$, where $P(e; \pi_e)$ is the probability that π_e succeeds in achieving e. For instance, for the edge $S_0 \to S_1$, π_e is trained to reach S_1 from a random state in S_0 while avoiding O.

Thus, to identify the optimal path, we could (i) train a policy π_e for each edge e, (ii) use it to estimate the edge cost $c(e; \pi_e)$, and (iii) run Dijkstra's algorithm with these costs. One challenge is that π_e depends on the initial states used in its training. For instance, training π_e for $e = S_1 \to S_3$ requires a distribution over S_1. Using the wrong distribution can lead to poor performance due to distribution shift. Furthermore, training a policy for all edges may unnecessarily waste effort training policies for unimportant edges. For these reasons, DiRL interweaves training policies with the execution of Dijkstra's algorithm, only training π_e once Dijkstra's algorithm requires the cost of edge e. As a result, DiRL scales to complex tasks and succeeds in generating a policy that satisfies the specification with high probability.



(achieve (reach S_1) or achieve (reach S_2);
achieve (reach S_3)) ensuring avoid O

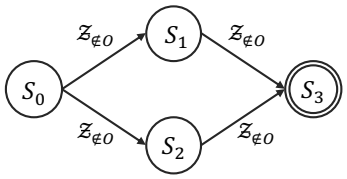
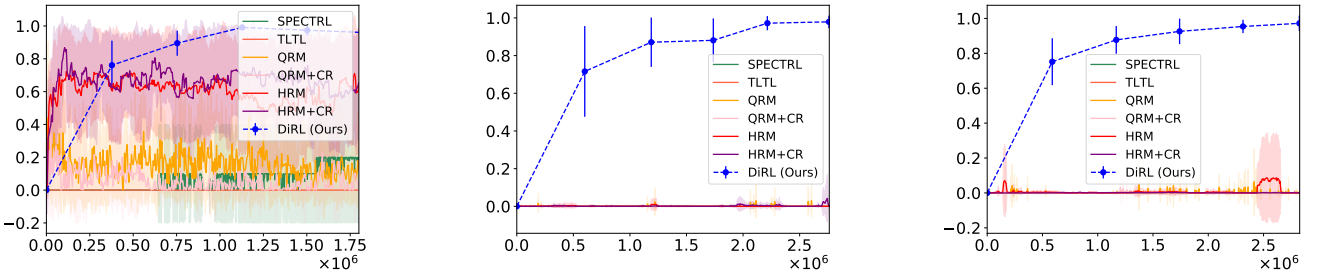


Figure 1. Top: The 9-rooms environment, with initial region S_0 in the bottom-left, an obstacle O in the middle-left, and three subgoal regions S_1, S_2, S_3 in the remaining corners. Middle: A user-provided specification that requires the agent to reach S_1 or S_2, followed by S_3, while avoiding O for the duration of the rollout. Bottom: The abstract graph $\mathcal{G}_{\text{ex}}$ that DiRL constructs for the specification.

4 Empirical Evaluation

Set-up. We compare our approach to four state-of-the-art algorithms for learning from specifications, Spectrl [7], Qrm [5], Hrm [13] and a Tltl [9] based approach. The variants Qrm+cr and Hrm+cr use counterfactual reasoning to reuse samples during training.



(a) Specification ϕ_1 **(b)** Motivating Example **(c)** Motivating Example without Obstacles

Figure 2. Learning curves for 9-Rooms environment with different specifications. x-axis denotes the number of samples (steps) and y-axis denotes the estimated probability of success. Results are averaged over 10 runs with error bars indicating ± standard deviation.

We consider the 9-Rooms environment shown in Figure 1. It has states $(x, y) \in \mathbb{R}^2$ encoding 2D position, actions $(v, \theta) \in \mathbb{R}^2$ encoding speed and direction, and transitions $s' = s + (v\cos(\theta), v\sin(\theta))$. We learn policies using ARS [11] (model-free RL) with shaped rewards; each policy is a fully connected NN with 2 hidden layers of 30 neurons each.

Results. Figure 2 shows learning curves on the specifications encoding sequential tasks in the 9-Rooms environment.

The task in Figure 2a is formally given by $\phi_1 =$ `achieve (reach` S_1`); achieve (reach` S_0`)` which is for the agent to go to S_1 and then return to S_0. This is a challenging task for standard RL algorithms since they do not store whether the first task has been accomplished. In these cases, a stateless policy will be unable to determine whether to move upwards or downwards from a state along the right hallway. Among all the tools examined here, all but TLTL support stateful policies. We observe that several tools (including DIRL) learn high-quality policies for task as it does not require any high-level planning.

The task in Figure 2b corresponds to the motivating example from Figure 1. The tasks in Figure 2c is identical to the motivating example except that it does not impose the safety condition to avoid the red obstacle. Both of these specifications require high-level planning to inform the agent of the absence of a direct path from S_2 to S_3. We observe that while DIRL quickly converges to high-quality policies for both of these specifications, none of the baselines comes close in performance. Even though, SPECTRL, QRM, and HRM can learn stateful policies, they scale poorly for two reasons (i) they do not decompose the learning problem into simpler ones and (ii) they are completely based on model-free RL and do not employ model-based planning at the high-level.

References

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[2] Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Modular Multitask Reinforcement Learning with Policy Sketches. In *International Conference on Machine Learning*. 166–175.

[3] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. 2020. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research* 39, 1 (2020), 3–20.

[4] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. 2005. Efficient bipedal robots based on passive-dynamic walkers. *Science* 307, 5712 (2005), 1082–1085.

[5] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. 2018. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2107–2116.

[6] Jeevana Priya Inala, Yichen Yang, James Paulos, Yewen Pu, Osbert Bastani, Vijay Kumar, Martin Rinard, and Armando Solar-Lezama. 2021. Neurosymbolic Transformers for Multi-Agent Communication. *arXiv preprint arXiv:2101.03238* (2021).

[7] Kishor Jothimurugan, Rajeev Alur, and Osbert Bastani. 2019. A Composable Specification Language for Reinforcement Learning Tasks. In *Advances in Neural Information Processing Systems*, Vol. 32. 13041–13051.

[8] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.

[9] Xiao Li, Cristian-Ioan Vasile, and Calin Belta. 2017. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3834–3839.

[10] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275* (2017).

[11] Horia Mania, Aurelia Guy, and Benjamin Recht. 2018. Simple random search of static linear policies is competitive for reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 1805–1814.

[12] Shao-Hua Sun, Te-Lin Wu, and Joseph J. Lim. 2020. Program Guided Agent. In *International Conference on Learning Representations*.

[13] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. 2020. Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning. *arXiv preprint arXiv:2010.03950* (2020).