Symbolic Plans as High-Level Instructions for Reinforcement Learning (Abridged)

León Illanes, Xi Yan, Rodrigo Toro Icarte, Sheila A. McIlraith†
University of Toronto & Vector Institute, Toronto, Canada
† Schwartz Reisman Institute for Technology and Society, Toronto, Canada
Toronto, Ontario, Canada
{lillanes, rntoro, sheila}@cs.toronto.edu, xi.yan@mail.utoronto.ca

Abstract

Reinforcement learning (RL) agents seek to maximize the cumulative reward obtained when interacting with their environment. Users define tasks or goals for RL agents by designing specialized reward functions such that maximization aligns with task satisfaction. This work explores the use of techniques from knowledge representation and reasoning, and in particular high-level symbolic action models, as a framework for defining final-state goal tasks and automatically producing their corresponding reward functions. We also show how automated planning can be used to synthesize high-level plans that can guide hierarchical RL (HRL) techniques towards efficiently learning adequate policies. We provide a formal characterization of taskable RL environments and describe sufficient conditions that guarantee we can satisfy various notions of optimality (e.g., minimize total cost, maximize probability of reaching the goal). In addition, we do an empirical evaluation that shows that our approach converges to near-optimal solutions faster than standard RL and HRL methods and that it provides an effective framework for transferring learned skills across multiple tasks in a given environment.

The full version of this paper appeared at the ICAPS 2020 conference [9].

1 Introduction

Reinforcement learning (RL) methods represent the state of the art for solving complex continuous control problems in robotics and other domains that evade human modeling [19, 12, 11, 6, 3, 1]. For instance, the OpenAI lab recently showed that model-free RL can be used to learn to control a human-like robot hand to purposefully manipulate complex objects, such as a Rubik's Cube [1]. The strength of model-free RL comes from being able to learn policies that maximize an external reward signal by directly interacting with the environment—without requiring a predefined model of the complex physics equations that control it (nor trying to learn them).

This generality comes with a cost, though. As environment dynamics and reward structures are initially unknown, RL mostly relies on random exploration to collect rewards and then improve the current policy. As such, RL can be *sample inefficient* (i.e., requires billions of interactions with the environment before learning better-than-random policies). Further, RL systems are typically not *taskable*. If you would like an RL agent to solve task A, then you would have to program a reward function such that its optimal policy would solve A. If, later on, you would like the agent to perform task B, then you would have to program a new reward function for B and the RL agent would have to learn a brand new policy for B from scratch—which is a problem known as *transfer learning* [16]. A number of approaches have been proposed to address these shortcomings including efforts to learn hierarchical representations [5], to define *options* or macro-actions that can be used by the RL system [15], or to learn skills that are independent of the state space where they were learned [10].

4th Knowledge Representation and Reasoning Meets Machine Learning Workshop (KR2ML 2020), at NeurIPS.

Our interest in this paper is in leveraging high-level symbolic planning models and automated plan synthesis techniques, in concert with state-of-the-art RL techniques, with the objective of improving sample efficiency and creating systems that are human taskable. Our efforts are based on the observation that some approximated understanding of the environment can be characterized as a symbolic planning model—a set of properties of the world and a formal description of actions that cause those properties to change in predictable ways—while leaving (possibly complex) low-level aspects of the environment (e.g., the frame by frame outcome of dropping a pen) unspecified.

As a result, our AI agent gets the best of both worlds: (1) it is taskable as the user can define tasks as goal conditions in the symbolic domain (and a reward function is automatically computed for such a task), (2) it improves sample efficiency as the high-level plans can be used for transferring learning from previously learned policies, and (3) it can learn complex low-level control policies as it relies on model-free RL to accommodate for all the information missing in the high-level model. To achieve this, we build on ideas for *learning by instructions* in RL [2, 17, 18]. That work shows that sample efficiency can be improved if a *manually* generated description of the task is given to the agent. In this work, we propose to *automatically* generate useful instructions for RL agents using the high-level model of the environment and describe an approach—based on hierarchical reinforcement learning (HRL)—that exploits such instructions. We compare our approach to standard forms of HRL and show that the combination of high-level symbolic planning and low-level reinforcement learning is an effective method for specifying tasks to RL agents and, more importantly, for learning high-quality policies—for previously unseen tasks—up to an order of magnitude faster than using standard RL.

Note that the idea of combining high-level symbolic planning with low-level RL has a long history. Well-known examples include work done by Ryan (2002), Grounds and Kudenko (2008), Grześ and Kudenko (2008), Yang et al. (2018), and Lyu et al. (2019). Informed by this work, we contribute a formal characterization of a relevant problem, which we call *taskable RL*, and a novel approach to transfer learned policies and guide exploration in RL based on high-level plans. Building on this, we provide theoretical analysis regarding sufficient conditions for ensuring our approach satisfies various notions of optimality and show empirical results that validate the efficiency of our approach.

2 Taskable Reinforcement Learning

We assume reader familiarity with RL, HRL, the *options* framework used in HRL, and with the basics of symbolic planning (see [9] for more details). One of the great advantages of symbolic planning is that specifying new simple tasks for a given domain model is very easy. It is with this in mind that we define the problem of taskable RL, where final-state goal tasks can be specified trivially for a given RL environment. To define goals for tasks in such an environment, we assume the existence of a set of high-level propositions, P, and a labeling function $L\colon S \to 2^P$ that establishes a mapping between low-level states and high-level propositions. These propositions are supposed to represent important state properties that may affect the outcomes of actions or their costs, or that may be of significance to an end user of the system. Finally, we also assume the existence of a constant R that establishes a reward bonus received by the agent when it accomplishes a task. With this, we define taskable RL environments and their associated final-state goal tasks.

Definition 1 (Taskable RL Environment)**.** A *taskable reinforcement learning environment* is defined by a tuple $E = \langle S, A, r, p, \gamma, P, L, R\rangle$, where $\langle S, A, r, p, \gamma\rangle$ is an MDP, P is a set of propositional variables, $L\colon S \to 2^P$ is a labeling function, and $R \in \mathbb{R}$ is a parameter called the *goal reward*.

Definition 2 (Final-state Goal Task)**.** A *final-state goal task* for taskable RL environment $E = \langle S, A, r, p, \gamma, P, L, R\rangle$ is defined as a tuple $G = \langle G^+, G^-\rangle$ where its elements are disjoint subsets of P. We say a state $g \in S$ is a goal state when $G^+ \subseteq L(g)$ and $G^- \cap L(g) = \varnothing$. We denote the set of all goal states as $\mathbb{G}$. The objective for this task is to find the optimal policy for the MDP $M_G = \langle S, A, r_G, p_G, \gamma\rangle$, where r_G and p_G are defined as follows. $r_G(s, a, s) = R + r(s, a, s')$ if $s' \in \mathbb{G}$ and $s \neq s'$; and $r_G(s, a, s') = 0$ if $s' \in \mathbb{G}$ and $s = s'$; and $r_G(s, a, s) = r(s, a, s')$ otherwise. Similarly, $p_G(s'|s, a) = 0$ if $s \in \mathbb{G}$ and $s \neq s'$; and $p_G(s'|s, a) = 1$ if $s \in \mathbb{G}$ and $s = s'$, and $p_G(s'|s, a) = p(s'|s, a)$ otherwise.

Intuitively, the goal conditions are used for defining *fictional* terminal states in the environment. The modified transition probabilities ensure that exiting a goal state is impossible. In turn, the modified reward function ensures that a reward bonus is given when a goal state is first reached, and that no further reward can be accrued after that.

The main motivation behind Definitions 1 and 2 is to allow end-users to define tasks for RL agents with minimal effort. In the same spirit, the main guarantee that we should provide to that end-user is that the RL agent will optimize its behaviour towards actually accomplishing their tasks. Interestingly, whether a taskable RL environment provides such a guarantee depends on how it defines r, γ, and R. For instance, if r is of the restricted form $r\colon S \times A \times S \to \mathbb{R}^-$, $\gamma = 1$, and $R = 0$, then the optimal policy is guaranteed to reach any final-state goal in a taskable environment (as long as such a goal is reachable). Another combination that holds this property is $\gamma \leq 1$, $R = 1$, and $r(s, a, s') = 0$.

Planning Models in RL. The general idea is to use planning models and solutions computed for them as guidance for solving RL tasks in the low-level environment. To do so, we associate symbolic models to taskable RL environments. Given a taskable environment $E = \langle S, A, r, p, \gamma, P, L, R\rangle$, a symbolic model for E is specified as $\mathcal{M} = \langle \mathcal{D}, \alpha\rangle$, where $\mathcal{D} = \langle \mathcal{F}, \mathcal{A}\rangle$ is a planning domain with $\mathcal{F} = P$ and $\alpha\colon \mathcal{A} \to 2^P \times 2^P$ is a function that associates planning actions with conditions over P.

We will use α to associate finite-state goal tasks for the taskable RL environment E with the planning actions. Each such task defines an option. Whenever the task is accomplished, the option terminates. Formally, given a condition $C = \langle C^+, C^-\rangle$, we can define the set of all low-level states that satisfy it as $T(C) = \{s \in S \mid C^+ \subseteq L(s), C^- \cap L(s) = \varnothing\}$. Then, for each planning action $a \in \mathcal{A}$, we can define an associated option with termination set $T_a = T(\alpha(a))$ and reward function $r_a = r_{\alpha(a)}$ (see Definition 2). Finally, we will define an option set O consisting of one option for each distinct $\alpha(a)$ generated this way. Note that two or more planning actions may be associated with the same option.

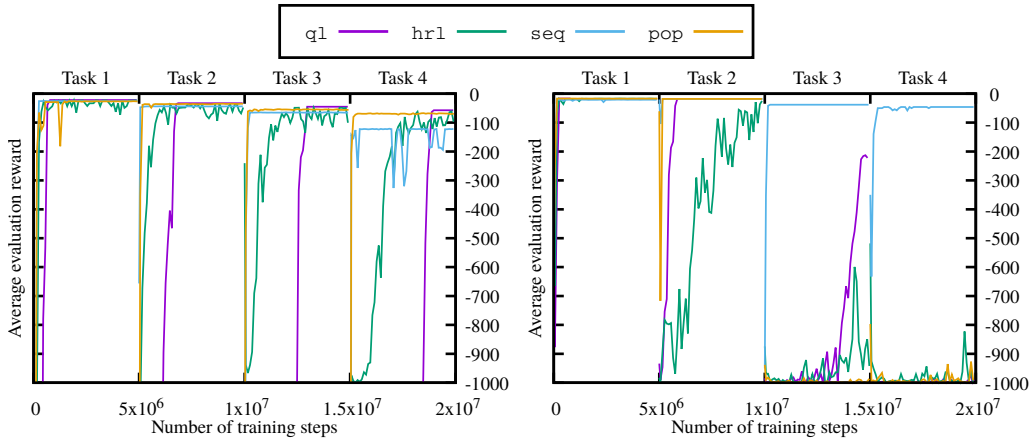
For a given symbolic model and planning task, we can easily compute a sequential plan by using an off-the-shelf planner. Such a plan can subsequently be used as a naive meta-controller for an HRL system by directly executing—in the provided order—each of the options associated with the actions in the plan. This basic approach is denoted as `seq`. A variant in which we relax this sequential plan into a partial-order plan [4] is denoted `pop`. For both cases, we also consider the use of regression-based plan execution monitoring ($\texttt{seq}_\texttt{m}$ and $\texttt{pop}_\texttt{m}$) described in the full paper [9]. Finally, we compare and contrast against two basic benchmark approaches: direct use of q-learning over the low-level environment (`ql`), and use of the options framework over the set of options associated with the model (`hrl`).

3 Empirical Evaluation

We evaluated our approach by considering two low-level environments and respective high-level models. To best evaluate the effectiveness at leveraging previous experience, we also defined a sequence of 4 tasks for each environment, ordered roughly by level of complexity. For each tested algorithm, the evaluation proceeded as follows. Each task was trained on for a fixed number of training steps. Whenever a goal state was reached, the task was restarted. If a task ran for more than $1,000$ steps without reaching the goal, it was also restarted. When the limit of total training steps for a task was reached, the meta-controller policy was reset and training began on the next task. When using `seq`, `pop`, $\texttt{seq}_\texttt{m}$, $\texttt{pop}_\texttt{m}$, or `hrl`, the trained policies for the options were transferred between tasks. In all cases, option policies and meta-controllers were trained by q-learning. To actually evaluate the quality of the learned policies, we paused the training every $10,000$ training steps and ran a number of independent trials using the policy as learned at that point.

3.1 Benchmark Environments.

The first test environment is the OFFICEWORLD running example. Each training episode was initialized with a random initial state, and the evaluation trials were done from 10 different predefined initial states. To account for different outcomes when tie-breaking, each such trial was run 10 times. Our second environment is the Minecraft-inspired gridworld [2]. The grid contains raw materials (e.g., wood, iron) and locations where the agent can combine materials to produce refined materials (e.g., wooden planks), tools (e.g., hammer), and goods (e.g., goldware). The high-level actions allow for collecting each of the raw materials, and for achieving the combinations. The types of tasks that we evaluated on include examples such as *"make a pickaxe,"* which requires getting wood and iron and taking them to various locations, or *"get a gem,"* which requires first making a pickaxe and then going to the location with the gem. We ran experiments using random initial states for training and evaluating on 5 predefined initial states. Each experiment was run 5 times.



(a) OFFICEWORLD (b) MINECRAFTWORLD

Figure 1: Experimental performance obtained in two separate environments. We report the mean reward obtained by two baseline algorithms, q-learning (`ql`) and standard HRL (`hrl`), by our basic approach in which a sequential plan is executed directly (`seq`), and by our main approach in which HRL is restricted to execute a partial-order plan (`pop`).

3.2 Results and Discussion.

To adequately display how our approach is capable of converging quickly to high-quality solutions, Figure 1 displays a comparison between our main approaches—`seq` and `pop`—and the two basic baseline algorithms in both benchmark domains. Each graph displays the reward obtained after training with the labeled algorithm for the specified number of steps.

The experimental results show that—once the option policies are sufficiently well trained—our approach can significantly outperform `ql` and `hrl` when the number of training steps is limited. For instance, in the last task of the OFFICEWORLD, `pop` needed only $70,000$ training steps to converge to a policy that resulted in traces that were typically 10 steps away from optimal. In contrast, `hrl` needed at least $1,800,000$ steps before finding a policy of comparable quality and did not appear to converge to stability in less than the $5,000,000$ training steps we allowed. That said, `hrl` did reach policies that resulted in slightly better solutions—only 5 steps worse than optimal. Q-learning converged to optimality after $3,850,000$ training steps.

The tasks in the MINECRAFTWORLD domain are significantly harder than those of the OFFICEWORLD domain and serve as a stress test for our approaches. In particular, the planning model is not strictly consistent with the low-level environment so the tasks exhibit a variety of pitfalls that make accidentally undoing previous work very easy. For example, if the agent is carrying a piece of wood and walks through a cell marked as a tool bench, it will automatically convert the wood into a wooden plank, even if it actually needs the wood for some other reason. Despite this, our results show that `seq` leads to reasonable results after very little training. In contrast, `pop` does not perform any better than `hrl`. In Figure 2 (see the full paper [9]), we show what happens when we address the unexpected outcomes with execution monitoring. In the OFFICEWORLD, the policies obtained by $\texttt{seq}_\texttt{m}$ seem to result in slightly more stable performance. For the MINECRAFTWORLD, $\texttt{pop}_\texttt{m}$ significantly outperforms `pop`, even if it still does not converge to high-quality policies.

4 Conclusions and Future Work

To conclude, we believe that the automatic generation of goal-relevant instructions is one of the key aspects that will enable RL systems to be both taskable and scalable. The combination of symbolic action models with model-free RL allows for solving problems that require both intricate control and long-term combinatorial planning. Taskable RL represents a valuable formalism for describing problems of this kind, and planning has shown to be a useful technique to aid in improving sample efficiency by enabling structured methods of exploration and transfer learning.

Acknowledgments and Disclosure of Funding

We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chairs Program, and Microsoft Research. The third author also acknowledges funding from ANID (Becas Chile). Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute for Artificial Intelligence (`www.vectorinstitute.ai/partners`). Finally, we thank the Schwartz Reisman Institute for Technology and Society for providing a rich multi-disciplinary research environment.

References

[1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. "Solving Rubik's Cube with a Robot Hand". In: *arXiv preprint arXiv:1910.07113* (2019).

[2] Jacob Andreas, Dan Klein, and Sergey Levine. "Modular Multitask Reinforcement Learning with Policy Sketches". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. Vol. 70. PMLR. 2017, pp. 166–175.

[3] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. "Learning dexterous in-hand manipulation". In: *arXiv preprint arXiv:1808.00177* (2018).

[4] Anthony Barrett and Daniel S. Weld. "Partial-Order Planning: Evaluating Possible Efficiency Gains". In: *Artificial Intelligence* 67.1 (1994), pp. 71–112.

[5] Thomas G. Dietterich. "Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition". In: *Journal of Artificial Intelligence Research* 13 (2000), pp. 227–303.

[6] Pietro Falco, Abdallah Attawia, Matteo Saveriano, and Dongheui Lee. "On policy learning robust to irreversible events: An application to robotic in-hand manipulation". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1482–1489.

[7] Matthew Jon Grounds and Daniel Kudenko. "Combining Reinforcement Learning with Symbolic Planning". In: *Adaptive Agents and Multi-Agents Systems III (AAMAS III)*. Vol. 4865. LNCS. 2008, pp. 75–86.

[8] Marek Grześ and Daniel Kudenko. "Plan-based reward shaping for reinforcement learning". In: *Proceedings of the 4th IEEE International Conference on Intelligent Systems (IS)*. Vol. 2. 2008, 10-22–10-29.

[9] León Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A. McIlraith. "Symbolic Plans as High-Level Instructions for Reinforcement Learning". In: *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS)*. 2020, pp. 540–550.

[10] George Konidaris and Andrew G. Barto. "Building Portable Options: Skill Transfer in Reinforcement Learning". In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*. 2007, pp. 895–900.

[11] Vikash Kumar, Abhishek Gupta, Emanuel Todorov, and Sergey Levine. "Learning dexterous manipulation policies from experience and imitation". In: *arXiv preprint arXiv:1611.05095* (2016).

[12] Vikash Kumar, Emanuel Todorov, and Sergey Levine. "Optimal control with learned local models: Application to dexterous manipulation". In: *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 378–383.

[13] Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. "SDRL: Interpretable and Data-efficient Deep Reinforcement Learning Leveraging Symbolic Planning". In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*. 2019.

[14] Malcolm R. K. Ryan. "Using Abstract Models of Behaviours to Automatically Generate Reinforcement Learning Hierarchies". In: *Proceedings of the 19th International Conference on Machine Learning (ICML)*. 2002, pp. 522–529.

[15] Richard S. Sutton, Doina Precup, and Satinder P. Singh. "Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning". In: *Artificial Intelligence* 112.1-2 (1999), pp. 181–211.

[16] Matthew E Taylor and Peter Stone. "Transfer learning for reinforcement learning domains: A survey". In: *Journal of Machine Learning Research* 10.Jul (2009), pp. 1633–1685.

[17] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. "Advice-Based Exploration in Model-Based Reinforcement Learning". In: *Canadian Conference on Artificial Intelligence*. 2018, pp. 72–83.

[18] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. Vol. 80. PMLR. 2018, pp. 2112–2121.

[19] Herke Van Hoof, Tucker Hermans, Gerhard Neumann, and Jan Peters. "Learning robot in-hand manipulation with tactile features". In: *Proceedings of the IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. 2015, pp. 121–127.

[20] Fangkai Yang, Daoming Lyu, Bo Liu, and Steven Gustafson. "PEORL: Integrating Symbolic Planning and Hierarchical Reinforcement Learning for Robust Decision-Making". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*. 2018, pp. 4860–4866. DOI: 10.24963/ijcai.2018/675. URL: https://doi.org/10.24963/ijcai.2018/675.