Work in Progress: Using Symbolic Planning with Deep RL to Improve Learning

Tianpei Yang[1], **Srijita Das**[1], **Christabel Wayllace**[2] **and Matthew E. Taylor**[1,3]
[1]University of Alberta, Canada
[2]New Mexico State University, USA
[3]Alberta Machine Intelligence Institute (Amii), Canada
{tianpei.yang,srijita1,matthew.e.taylor}@ualberta.ca
cwayllac@nmsu.edu

Abstract

Deep Reinforcement Learning (DRL) has achieved expressive success across a wide range of domains. However, it is still faced with the sample-inefficiency problem that requires massive training samples to learn the optimal policy. Furthermore, the trained policy is highly dependent on the training environment which limits the generalization. In this paper, we propose the Planner-guided RL (PRL) approach to explore how symbolic planning can help DRL in terms of efficiency and generalization. Our PRL is a two-level structure that incorporates any symbolic planner as the meta-controller to derive the subgoals. The low-level controller learns how to achieve the subgoals. We evaluate PRL on Montezuma's Revenge and results show that PRL outperforms previous hierarchical methods. The evaluation of generalization is a work-in-progress.

1 Introduction

Deep Reinforcement Learning (DRL) has achieved notable success across a diverse range of fields, from mastering complex games such as Go [22] and Dota [4] to interesting applications like navigating super-pressure balloons in the stratosphere [3]. Many of these domains share common traits, characterized by high-dimensional state spaces and continuous action spaces. Nevertheless, the application of these algorithms to complex real-world domains such as robotics faces a significant challenge because of sample inefficiency [13] as DRL agents require millions of interactions with the environment to learn an optimal policy. Another challenge that DRL agents face is their poor generalization capability [25] when executing the learned policy on a similar environment with slight variation in logic.

Symbolic models have been used in the literature to increase the generalization capability of RL algorithms. One of the earlier directions was Relational Reinforcement Learning [8], where the agent learned a non-parametric tree-based value function representation that generalized well to tasks with a similar structure. This solution, however, was restricted to tabular RL algorithms on smaller domains. Jiang et al. [15] proposed a method to integrate differentiable Inductive logic Programming with policy gradient algorithms using minimum background knowledge. High-level programs [23, 28] have also been used to propose sub-tasks in order to make the policies learned by a low-level agent generalizable. Cao et al. [6] integrate program synthesis with inductive logic programming to construct policies that are more robust to slight logical changes in the training environment.

Symbolic Planning has long been used at the highest level to guide DRL agents with exploration [19, 16]. However, most prior work uses some combination of meta-controller and symbolic

NeurIPS 2023 Workshop on Generalization in Planning (GenPlan 2023)

planner at the higher level to choose the appropriate subgoals for the lower-level DRL agent. In this work, we propose a two-level structure that incorporates any symbolic planner as the meta-controller to get the subgoals. Since the subgoals are directly derived from the symbolic plan; they are already ordered and there is no need to use a meta-controller separately to choose the subgoals. The low-level RL agent learns to achieve the subgoals sequentially as proposed by the planner. As a result, our proposed method is more efficient than other baselines in terms of training time. The proposed algorithm should also generalize well to similar environments with slight logical differences by virtue of the cause-effect relationship which is inherently incorporated in the domain representation at the symbolic level. This is a work-in-progress, and we leave the evaluations related to the generalization of our algorithm as immediate future work.

2 Related work

Integration of Symbolic Planning and RL: A variety of work exists along guiding Reinforcement Learning agents to useful states using the integration of symbolic planners at the higher level. Yang et al. [26] integrated a planner with hierarchical R-learning for small domains with a direct mapping between symbolic transitions and options. In this work, the planner updates its plan after every step of R-learning. Lyu et al. [19] further generalized [26] for Deep RL by using a symbolic planner that suggests sub-tasks; a controller that learns sub-policy for the suggested sub-task based on intrinsic reward and a meta-controller that proposes intrinsic goals back to the planner. Jin et al. [16] proposed a hierarchical 2-level framework where the top level includes interaction between the meta-controller and symbolic planner; the former generates the action model automatically and the goal for the planner to plan and also selects the best option. The lower-level agent learns the policy for the option chosen by the higher level. Chester et al. [7] propose a three-level hierarchy for incomplete symbolic domain models consisting of a meta-controller that chooses symbolic goals; a planner that plans for these goals and a controller that uses the generated plan to guide its policy. Symbolic Planners have also been used in Taskable RL and Relational MDP representation [14, 17] for better generalization and transfer; however, these were restricted to tabular RL algorithms for learning the tasks. Guan et al. [11] combine symbolic planning and hierarchical RL in a setting where symbolic domain knowledge by humans is incomplete by using skill diversity to account for the incompleteness at the symbolic level. Sarathy et al. [21] consider a slightly different integration task where they use RL for discovering new operators and adding them to the planning representation to construct a valid plan. Contrary to prior work [19, 16, 7], we remove the meta-controller from our framework and let the planner suggest sequential subgoals to the low-level learner with the future objective to be able to generalize to logically similar tasks.

3 Background

3.1 Markov Decision Process (MDP)

A Markov Decision Process (MDP) is generally defined as $\langle S, A, T, R, \rho_0, \gamma \rangle$, with a set of states S, a set of actions A, a stochastic transition function $T : S \times A \rightarrow P(S)$, which represents the probability distribution over possible next states, given the current state and action, a reward function $R : S \times A \rightarrow \mathbb{R}$, an initial state distribution $\rho_0 : S \rightarrow \mathbb{R}_{\in[0,1]}$, and a discounted factor $\gamma \in [0, 1)$. An agent interacts with the environment (i.e., MDP) at discrete time steps by performing its policy $\pi : S \rightarrow P(A)$, receiving a reward r. The agent's objective is to learn a policy to maximize the expected cumulative discounted reward: $J(\pi) = \mathbb{E}_{\rho_0,\pi,T}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$.

3.2 Reinforcement Learning

In this paper, we focus on value-based RL algorithms and introduce one representative DRL method, DQN [20]. DQN [20] is a popular value-based DRL method, which replaces the Q-table of Q-learning [24] as a deep neural network. DQN learns an action-value function as $Q(s, a|\theta)$, parameterized by θ, by minimizing the loss:

$$L(\theta) = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} Q'(s', a'|\theta') - Q(s, a|\theta)\right)^2\right] \tag{1}$$

where Q' is the target Q-network parameterized by θ' and periodically updated from θ.

3.3 Classical Planning

In classical planning, the objective is to select actions in deterministic environments whose initial state is fully known. The classical planning model is represented as a tuple $\langle \mathcal{S}, s_0, S_G, \mathcal{A}, \mathcal{T}, C \rangle$, where $\mathcal{S}$ is the set of states and $\mathcal{A}$ is the set of actions, $s_0 \in \mathcal{S}$ is the known initial state, $S_G \in \mathcal{S}$ is the non-empty set of goal states, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is a deterministic transition function, and $C : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a positive cost function. A solution or *plan*, is a sequence of applicable actions $a_0 \ldots a_n$ that generates a state sequence $s_0 \ldots s_{n+1}$ where s_{n+1} is a goal state.

3.3.1 Languages for Classical Planning

The simplest classical planning language in use is STRIPS [9], a language based on boolean variables. A STRIPS planning problem is a tuple $P = \langle F, I, O, G \rangle$, where F is the set of propositions of interest, $I \subseteq F$ represents the initial situation, O denotes the set of actions, and $G \subseteq F$ represents the goal. The actions $a_s \in O$ in STRIPS are represented by three sets of propositions over F: $Add(a_s)$, $Del(a_s)$, and $Pre(a_s)$. $Add(a_s)$ describes the propositions that a_s makes true, $Del(a_s)$, the propositions that a_s makes false, and $Pre(a_s)$, the propositions that must be true in order for a_s to be applicable. In STRIPS, a state f_s is a possible *collection* of propositions over F, where an atom $p \in F$ is $true$ in $f_s \iff p \in f_s$, $s_0 = I$, $f_s \in S_G \wedge G \subseteq f_s$, $\mathcal{A}(f_s) = O$ with $Prec(a_s) \subseteq f_s$, $\mathcal{T}(f_s, a_s) = \big(f_s \setminus Del(a_s)\big) \cup Add(a_s)$, and cost $c(f_s, a_s) = 1$ by default. The Planning Domain Definition Language (PDDL) [1] accommodates the STRIPS language and has been used in planning competitions. There are many free classical planners and hundreds of planning problems expressed in PDDL for use with such planners [10].

4 Planner-guided RL

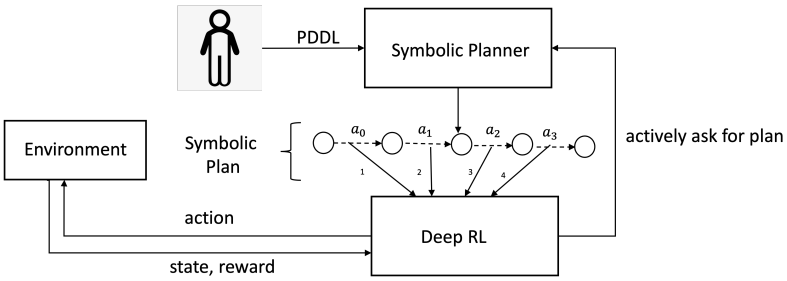

Figure 1: Planner-guided Reinforcement Learning

4.1 Problem Formulation

We represent the problem with a tuple $\langle F, I, O, G, S, A, T, R, \rho_0, \gamma, \mathcal{G} \rangle$, where F, I, O, G are as defined in a STRIPS planning problem, S, A, T, ρ_0, γ are MDP components, R is the intrinsic reward function, and $\mathcal{G} \subseteq S$ represents the set of subgoals that a RL agent should achieve. The final goal of the RL agent is to learn a policy π which learns faster (due to guidance from the planner) and generalizes well (due to symbolic representation of the domain)

Our proposed framework for Planner-guided RL is shown in Figure 1. At the top level, a domain expert specifies the environment dynamics and constraints in the form of PDDL. This domain definition is taken as input by the Symbolic Planner to generate a symbolic plan which is an ordered set of symbolic actions (a_0, a_1, a_2, a_3) as per Figure 1. These symbolic actions are mapped into subgoals through a mapping function and are achieved sequentially by the low-level Deep RL agent. In order to learn the subgoals, the low-level RL agent receives an intrinsic reward based on R when it achieves a subgoal.

4.1.1 Interaction Between Planner and RL Agent

An illustration of the interaction between Planner and RL agent is shown in Figure 2. The representation of an environment differs for the planner and the reinforcement learning algorithm to suit

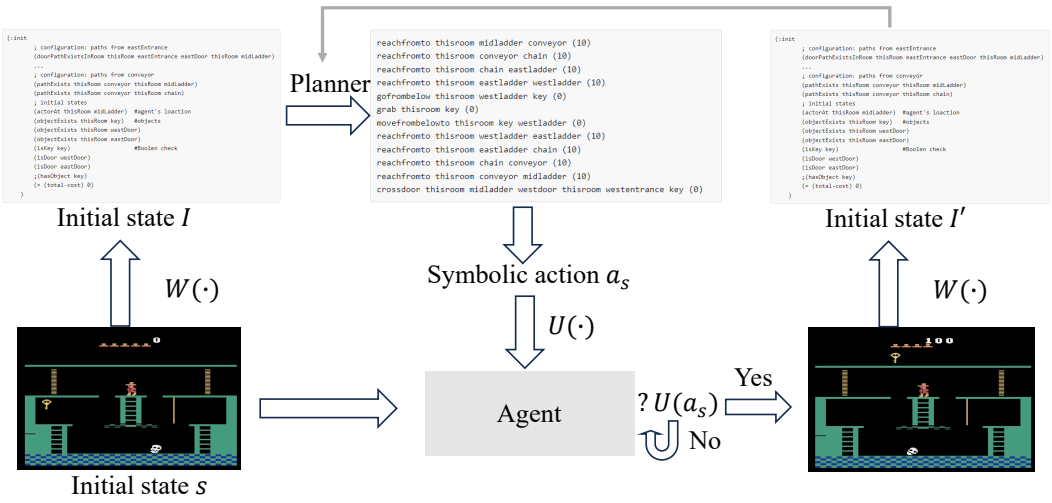

Figure 2: The execution of PRL. First, given the initial state s from the environment, we map s to the symbolic state I and generate the *plan*. Then, the symbolic action is derived following the *plan* and mapped to the low-level subgoal state. Last, the low-level agent learns the policy to reach the subgoal, which becomes the new symbolic state I'. The above process repeats until the agent learns to achieve the final goal.

their respective decision-making approaches. We use a symbolic representation, PDDL, to allow the planner to reason about the environment at a high level regarding symbolic predicates, preconditions, and effects. On the other hand, the reinforcement learning algorithm uses a state-action representation at a low level. The planner generates an optimal sequence $p = \langle a_0, a_1, \ldots, a_n \rangle$ of symbolic actions $a_i \in O$, which are sent to the RL agent as subgoals. We consider the mapping function $U : O \rightarrow \mathcal{G}$[1], which takes as input a symbolic action $a \in O$ and maps it to a subgoal $g \in \mathcal{G}$. Therefore, the RL agent receives a sequence:

$$u = \langle g_0 = U(a_0), g_1 = U(a_1), \ldots, g_n = U(a_n) \rangle \text{ where } i = 0, \ldots, n \wedge g_i \in \mathcal{G} \tag{2}$$

We consider another mapping function $W : S \rightarrow F$ which maps the low-level states to symbolic states for the planner to generate a plan.

In a static environment, the planner only needs to run once. However, the environment may change, especially during training, causing unreachable or suboptimal sequences of actions and, hence, subgoals. For instance, consider the initial room (Figure 3) in the Montezuma's Revenge environment. If the agent successfully follows all subgoals, after getting the key, it would need to go to the ladder on the right and climb it, climb the middle ladder, and open one door. In contrast, if the agent dies after getting the key, its next initial position will be at the top of the first ladder, and since it already has a key, the next optimal subgoal should be to open a door (Figure 4). To handle such a dynamic environment, the RL agent sends its current state as the new initial state to the planner after reaching every subgoal or at the beginning of every episode. Our system accounts for the agent's progress and current state to generate a new PDDL file (Figure 1). While requesting a new plan after reaching every subgoal is not optimal, the running time is dominated by the learning process.

4.2 Algorithm

We present the algorithm for Planner-guided RL (PRL) in Algorithm 1. The Planner generates a plan starting from a symbolic representation of the current state s_s (line 5). The symbolic action a_s following the plan is converted to subgoal g_s in the low-level learner's state and action-space representation using the mapping function U (lines 6 and 8). The low-level Deep RL agent collects experience by taking action a, getting a reward r, and transitioning to the next state s' and stores this to replay buffer D (lines 10-12). A mini-batch of experience is sampled from D to update the

[1]In this work, this function is predefined manually; however, we plan to investigate learning it automatically as future work

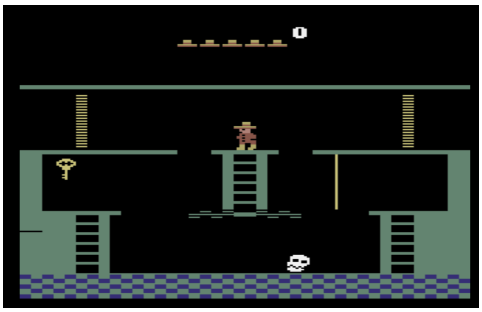

Figure 3: Initial room (Montezuma's Revenge.)

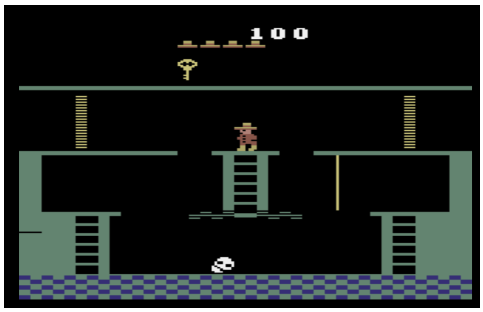

Figure 4: A different initial state.

Algorithm 1: **P**lanner-guided **RL**

Initialize: Q-network parameters θ, target Q-network parameters θ^-, replay buffer $\mathcal{D}$, target Q-network update interval τ, G
repeat
  Start from state s
  // Planning stage
  Get the symbolic state $I = W(s)$
  Generate a plan to reach goal G
  Get the symbolic action a_s following the plan ▷ see Eq. (2)
  //RL stage
  Receive the subgoal $g_s = U(a_s)$
  repeat
    Select an action a following a ϵ-greedy strategy over Q-values
    Obtain a reward r and next state s'
    Store $\{s, g_s, a, r, s'\}$ to replay buffer $\mathcal{D}$
    // Update Q-network
    Sample a batch of trajectories from $\mathcal{D}$
    Update Q-network w.r.t. θ ▷ see Eq. (1)
    Update target Q-network w.r.t. θ^- every τ episodes
  until *reaching maximum steps or achieving* g_s;
  $s \leftarrow$ Reset to initial state
until *reaching maximum training steps*;

parameter θ and θ^- of the Q-network and target Q-network respectively as per the loss function in Equation 1. The agent's next initial state is used by the symbolic planner to generate a new plan for the defined goal G.

5 Experimental Evaluations

Experimental settings: We present our results on Montezuma's revenge. Montezuma's Revenge [5] is a notorious example with sparse, delayed rewards. Fig. 3 shows the first room of Montezuma's Revenge. An ideal agent needs to climb down the ladder, move left, and collect the key where obtains a reward ($+100$); then the agent backtracks and navigates to the door and opens it with the key, resulting in a final reward ($+300$). A long sequence of specific actions is required to solve this task. Without effective exploration, an RL agent would not be able to finish the task in such an environment [27]. We use this environment to test the effectiveness of our method. Note that this game requires some primary skills that are common throughout a panorama of 24 different rooms [2]. For example, the agent needs to collect different objects (e.g., keys and treasures) and solve relations among objects (e.g., the correspondence between keys and doors) to trigger a series of critical events (e.g., discovering the sword). Meanwhile, collecting the items requires avoiding monsters and traps. Thus, some tasks learned in one room will be helpful in other rooms. Therefore,

it is an adequate domain to test planning, learning, and generalization skills[2]. We used the Fast Downward planning system [12] to generate the subgoals.

Baselines: We compare PRL to 2 baselines-(1) H-DQN which is a hierarchical DRL method using human-designed subgoals [18]; (2) SORL, which is a start-of-the-art algorithm that uses a three-level hierarchy consisting of planner, meta-controller and controller [16].

Result: Our preliminary results are shown in Figure 5. We compare the average training reward and see that our proposed approach PRL gets an average reward of 400 in around 2000 episodes as compared to the other baseline approaches, presenting the effectiveness of using our method to facilitate more efficient training. H-DQN converges in around 5000 episodes but gets a much lower average reward (around 100) as compared to our approach. SORL is not able to learn within 10000 episodes which shows that SORL might need a lot of training episodes to learn the optimal policy. This validates our hypothesis that removing the meta-controller improves the training efficiency, as PRL takes into account the ordered subgoals output by the planner.

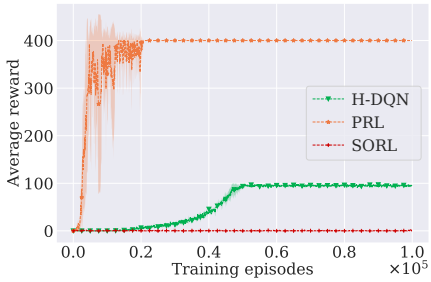

Figure 5: Training results on Montezuma's Revenge.

6 Conclusion and Future Work

This paper presents a first step to show the advantages of combining symbolic planning with deep RL to improve learning and generalization. The experiments illustrate how our approach requires less training than prior work, proving our first point.

As for future work, we plan to test our approach on more domains and show that our method is also effective in generalization in domains that contain the same domain definitions as the training environment with slight variations in the environment. The symbolic actions (mapped to subgoals) carry the cause and effect information, which allows planners to use the same domain information in different instances and which will support the generalization for the RL agent. Since the agent receives the same subgoals for similar tasks (i.e., the planner does not consider low-level details such as background color and types of enemies in Montezuma's revenge domain), it should be able to use a learned task in different contexts.

Acknowledgments and Disclosure of Funding

This work has taken place in the Intelligent Robot Learning (IRL) Lab at the University of Alberta, which is supported by research grants from the Alberta Machine Intelligence Institute (Amii); a Canada CIFAR AI Chair, Amii; Compute Canada; Huawei; Mitacs; and NSERC.

References

[1] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, et al. PDDL| the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[2] Atariage. Atari 2600 archives: Montezuma's revenge. https://atariage.com/2600/archives/strategy_MontezumasRevenge_Level1.html.

[3] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

[4] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[2]In this paper, we show the results for the first part. The evaluation of generalization is a work in progress.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[6] Yushi Cao, Zhiming Li, Tianpei Yang, Hao Zhang, Yan Zheng, Yi Li, Jianye Hao, and Yang Liu. GALOIS: boosting deep reinforcement learning via generalizable logic synthesis. In *NeurIPS*, pages 19930–19943, 2022.

[7] Andrew Chester, Michael Dann, Fabio Zambetta, and John Thangarajah. SAGE: generating symbolic goals for myopic models in deep reinforcement learning. *CoRR*, abs/2203.05079, 2022.

[8] Sašo Džeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. *MLJ*, 2001.

[9] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[10] Hector Geffner and Blai Bonet. A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(1):1–141, 2013.

[11] Lin Guan, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging approximate symbolic models for reinforcement learning via skill diversity. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *Proceedings of the Thirty-ninth International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7949–7967. PMLR, 2022.

[12] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[13] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 2021.

[14] León Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A McIlraith. Symbolic plans as high-level instructions for reinforcement learning. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, pages 540–550, 2020.

[15] Zhengyao Jiang and Shan Luo. Neural logic reinforcement learning. In *International conference on machine learning*, pages 3110–3119. PMLR, 2019.

[16] Mu Jin, Zhihao Ma, Kebing Jin, Hankz Hankui Zhuo, Chen Chen, and Chao Yu. Creativity of AI: automatic symbolic option discovery for facilitating deep reinforcement learning. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*, pages 7042–7050. AAAI Press, 2022.

[17] Harsha Kokel, Arjun Manoharan, Sriraam Natarajan, Balaraman Ravindran, and Prasad Tadepalli. Reprel: Integrating relational planning and reinforcement learning for effective abstraction. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pages 533–541, 2021.

[18] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[19] Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. SDRL: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 2970–2977. AAAI Press, 2019.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[21] Vasanth Sarathy, Daniel Kasenberg, Shivam Goel, Jivko Sinapov, and Matthias Scheutz. SPOTTER: extending symbolic planning operators through targeted reinforcement learning. In Frank Dignum, Alessio Lomuscio, Ulle Endriss, and Ann Nowé, editors, *Proceedings of the Twentieth International Conference on Autonomous Agents and Multiagent Systems*, pages 1118–1126. ACM, 2021.

[22] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

[23] Shao-Hua Sun, Te-Lin Wu, and Joseph J Lim. Program guided agent. In *International Conference on Learning Representations*, 2019.

[24] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[25] Markus Wulfmeier, Ingmar Posner, and Pieter Abbeel. Mutual alignment transfer learning. In *Conference on Robot Learning*, pages 281–290. PMLR, 2017.

[26] Fangkai Yang, Daoming Lyu, Bo Liu, and Steven Gustafson. PEORL: integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4860–4866. ijcai.org, 2018.

[27] Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Jianye Hao, Zhaopeng Meng, and Peng Liu. Exploration in deep reinforcement learning: A comprehensive survey. *CoRR*, abs/2109.06668, 2021.

[28] Yichen Yang, Jeevana Priya Inala, Osbert Bastani, Yewen Pu, Armando Solar-Lezama, and Martin Rinard. Program synthesis guided reinforcement learning for partially observed environments. *Advances in neural information processing systems*, 34:29669–29683, 2021.