Guided Skill Learning and Abstraction for Long-Horizon Manipulation

Shuo Cheng and Danfei Xu
Georgia Institute of Technology
United States
{shuocheng, danfei}@gatech.edu

Abstract: To assist with everyday human activities, robots must solve complex long-horizon tasks and generalize to new settings. Recent deep reinforcement learning (RL) methods show promises in fully autonomous learning, but they struggle to reach long-term goals in large environments. On the other hand, Task and Motion Planning (TAMP) approaches excel at solving and generalizing across long-horizon tasks, thanks to their powerful state and action abstractions. But they assume predefined skill sets, which limits their real-world applications. In this work, we combine the benefits of these two paradigms and propose an integrated task planning and skill learning framework named LEAGUE (Learning and Abstraction with Guidance). LEAGUE leverages symbolic interface of a task planner to guide RL-based skill learning and creates abstract state space to enable skill reuse. More importantly, LEAGUE learns manipulation skills *in-situ* of the task planning system, continuously growing its capability and the set of tasks that it can solve. We demonstrate LEAGUE on three challenging simulated task domains and show that LEAGUE outperforms baselines by a large margin, and that the learned skills can be reused to accelerate learning in new tasks and domains. Additional resource is available at https://bit.ly/3eUOx4N.

Keywords: Skill Learning and Abstraction, Long-horizon Manipulation

1 Introduction

A longstanding challenge in robotics is to develop robots that can *autonomously learn* to work in everyday human environments such as households. Recently, Deep Reinforcement Learning (DRL) has emerged as a promising paradigm to allow robots to acquire skills with minimal supervision [1, 2, 3, 4, 5]. However, DRL methods are still far from enabling home robots on their own. Among the multitudes of challenges, two have stood out. First, complex real-world tasks are often *long-horizon*. This requires a learning agent to explore a prohibitively large space of possible action sequences that scales exponentially with the task horizon. Second, effective home robots must carry out diverse tasks in varying environments. As a result, a learner must *generalize* or *quickly adapt* its knowledge to new settings.

To better learn long-horizon tasks, many DRL methods propose to leverage domain knowledge and structural prior [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]. Automatic goal generation in curriculum learning guides a learning process using intermediate subgoals, enabling an agent to effectively explore and make incremental progress toward a long-horizon task goal [10, 11, 21, 22]. Other methods use predefined behavior primitives or learn hierarchical policies to enable temporally-extended decision-making [6, 7, 8, 13, 15, 17, 18, 24, 23, 25, 28, 29]. Although these approaches can outperform vanilla DRL, they still suffer from low sample efficiency, lack of interpretability, and fragile generalization. Most importantly, the learned policies are often task-specific and fall short in cross-task and cross-domain generalization.

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

In the meantime, more established paradigms in robotics have long sought to address these challenges. In particular, Task and Motion Planning (TAMP) [30, 31, 32, 33, 34] leverages symbolic action abstractions to enable tractable planning and strong generalization. Specifically, the symbolic action operators divide a large planning problem into pieces that are each easier to solve. And the "lifted" action abstraction allows skill reuse across tasks and even domains. For example, a `grasp` skill operator and its underlying implementation can be easily adapted to solve a new task in a new domain. At the same time, most TAMP-style approaches assume access to a complete set of skills before deployment. This is impractical for two reasons. First, it is hard to prepare skills for all possible tasks. A robot must be able to grow its skill set on demand. Second, it is hard to hand-engineer manipulation skills for complex or contact-rich tasks (i.e., pouring or insertion). The challenges make TAMP methods difficult to deploy in real-world settings.

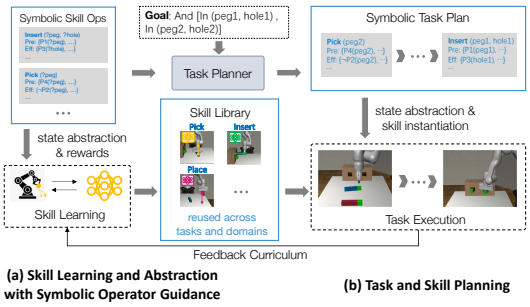

Figure 1: **Overview of the LEAGUE framework.** We present an integrated task planning and skill learning framework. (a) The system uses the symbolic operator interface of a TAMP-like system as guidance to learn reusable manipulation skills. (b) A task planner composes the learned skills to solve long-horizon tasks. As an integrated system, the task planner acts as a feedback curriculum to guide the skill learning, and the RL-based skill learner continuously grows the set of tasks that the system can solve.

In this work, we introduce LEAGUE (**LE**arning and **A**bstraction with **GU**idanc**E**), an *integrated task planning and skill learning* framework that learns to solve and generalize across long-horizon tasks. LEAGUE harnesses the merits of the two research paradigms discussed above. Starting with a task planner that is equipped with skills that are easy to implement (e.g., `reaching`), LEAGUE continuously grows the skill set *in-situ* using a DRL-based learner. The intermediate goals in a task plan are prescribed as reward for the learner to acquire and refine skills, and the mastered skills are used to reach the initial states of the new skills. Moreover, LEAGUE leverages the action operator definition, i.e., the preconditions and the effects, to determine a reduced state space for each learned skill, akin to the concept of information hiding in feudal learning [35, 36]. The key idea is to *abstract away* task-irrelevant features to make the learned skills modular and reusable. Together, the result is a virtuous cycle where the task planner guides skill learning and abstraction, and the learner continuously expands the set of tasks that the overall system can perform.

We conduct empirical studies on three challenging long-horizon manipulation tasks built on the Robosuite simulation framework [37]. We show that LEAGUE is able to outperform state-of-the-art hierarchical reinforcement learning methods [6] by a large margin. We also highlight that our method can achieve strong generalization to new task goals and even task domains by reusing and adapting learned skills. As a result, LEAGUE can solve a challenging simulated coffee making task where competitive baselines fall flat.

In summary, our primary contributions are: 1) we leverage the state and action abstractions readily available in a TAMP system to learn reusable skills, 2) we instantiate the strong synergy between the task planner and the skill learner as an integrated task planning and skill learning framework, and 3) we show that the framework can progressively learn skills to solve complex long-horizon tasks and generalize the learned skills to new task goals and domains.

2 Related Work

TAMP and Learning for TAMP. Task and Motion Planning (TAMP) [38, 39, 40, 41, 42, 43] is a powerful paradigm to solve long-horizon manipulation tasks. The key idea is to break a challenging planning problem into a set of symbolic-continuous search problems that are individually easier to solve. However, TAMP methods require high-level skills and their kinematics or dynamics models

a priori. The assumptions preclude domains for which hand-engineering manipulation skills is difficult, such as contact-rich tasks. Recent works proposed to learn dynamics models for TAMP by characterizing skill preconditions and effects [44, 45, 46, 47, 48, 49, 26]. For example, Konidaris *et al.* [48] learns compact symbolic models of an environment through trial-and-error. Liang *et al.* [26] uses graph neural networks to model skill effects. However, these works still require hand-engineering complete skill sets that can solve the target task, which may not be feasible in real-world applications. Our work instead aims to progressively learn new skills to extend the capability of a TAMP-like system.

Curriculum for RL. Our idea to guide skill learning with a symbolic task planner is connected to curriculum for RL. The key idea is to expose a learning agent to incrementally more difficult intermediate tasks before mastering a target task [50]. The intermediate tasks can take the form of state initialization [51, 52, 53], environments [54, 55, 56], and subgoals [57, 58, 59, 60]. For example, Florensa *et al.* [51] starts with near-success initialization and gradually moves the initial states further away. Campero *et al.* [58] trains a goal-generator teacher and a goal-seeking student. While effective at accelerating task learning, existing curricula focus on teaching task or domain-specific policies. In contrast, our method leverages the symbolic abstraction of a task planner to learn a repertoire of modular and composable skills. We show that we can compose learned skills to achieve new goals and even transfer skills to new task domains.

State and Action Abstractions. State and action abstractions are crucial for learning complex tasks in a large environment [61]. State abstraction allows agents to focus on task-relevant features of the environment. Action abstraction enables temporally-extended decision-making for long-horizon tasks. There exists a large body of work on learning either or both types of abstractions [62, 63, 64, 48, 65, 66, 67]. For example, Jonschkowski *et al.* [64] explores different representation learning objectives for effective state abstraction. Abel *et al.* [66] introduces a theory for value-preserving state-action abstraction. However, autonomously discovering suitable abstractions remains an open challenge. Our key insight is that a TAMP framework provides powerful state and action abstractions that can readily guide skill learning. Specifically, the symbolic interface of an action operator defines both the precondition and the effect (action abstraction) and the state subspace that is relevant to the action (state abstraction). The abstractions allow us to train skills that are compatible with the task planner and prevent the learned skills from being distracted by irrelevant objects, making skill reuse across tasks and domains possible.

Hierarchical Modeling in Robot Learning. Our method inherits the bi-level hierarchy of a TAMP framework. Hierarchical modeling has a rich history in robotics and robot learning. In addition to TAMP, various general frameworks including hierarchical task networks [68, 69], logical-geometric programming [41, 42], and hierarchical reinforcement learning (HRL) [23, 24, 36] have been proposed to exploit the hierarchical nature of common robotics tasks. In the context of HRL, a small number of works have explored symbolic planner-guided HRL [70, 71]. However, these methods require tabular state representations and are thus limited to simple grid-world domains. In robotics domains, a closely related research thread is to use behavior primitives in RL [6, 72]. For example, MAPLE [6] trains a high-level policy that chooses among hand-engineered behavior primitives and atomic actions. Our method instead leverages a symbolic planner to serve as the high-level controller to compose learned skills, allowing us to continuously extend the skill set while also leading to better generalization.

3 Method

We seek to enable robots to solve and generalize across long-horizon tasks. Our primary contribution is a novel integrated task planning and skill learning framework named LEAGUE. Here, we first provide necessary background in Sec. 3.1, and describe how LEAGUE (1) learns reusable skills guided by the symbolic operators of a task planner in Sec. 3.2 and (2) uses planner-generated task plans as an autonomous curriculum to continuously learn skills and expand the capability of the overall system in Sec. 3.3.

3.1 Background

MDP. A Markov decision process (MDP) is a tuple $< \mathcal{X}, \mathcal{A}, \mathcal{R}(x,a), \mathcal{T}(x'|x,a), p(x^0), \gamma >$, where $\mathcal{X}$ is the set of environment state, $\mathcal{A}$ is the set of actions the agent can taken, $\mathcal{R}$ is the reward function, $\mathcal{T}$ is the transition model of the environment. $p(x^0)$ denotes the distribution of the initial states, and γ is the discount factor. The objective for RL training is to maximize the expected total reward with related to the policy $\pi(a|x)$ that the agent uses to interact with the environment: $J = \mathbb{E}_{x^0,a^0,\ldots a^{t-1},s^T \sim \pi, p(x^0)} [\sum_t \gamma^t \mathcal{R}(x^t, a^t)]$.

Task planning space. To support task planning, we assume the environment is augment with a symbolic interface $< \mathcal{O}, \Lambda, \bar{\Psi}, \bar{\Omega}, \mathcal{G} >$, where $\mathcal{O}$ denotes the object set and Λ denotes a finite set of object types. Each object entity $o \in \mathcal{O}$ (e.g., `peg1`) has a specific type $\lambda \in \Lambda$ (e.g., `peg`) and a tuple of $\texttt{dim}(\lambda)$ dimential feature (i.e., (x, y, z, quaternion, ...)), and the environment state $x \in \mathcal{X}$ is a mapping from object entities to features: $x(o) \in \mathcal{R}^{\texttt{dim}(\texttt{type}(o))}$. Predicates Ψ describe the relationships among multiple objects. Each predicate ψ (i.e., `Holding`) is characterized by a tuple of object types $(\lambda_1, ..., \lambda_m)$ and a binary classifier that determines whether the relationship holds: $c_\psi : \mathcal{X} \times \mathcal{O}^m \rightarrow \{True, False\}$, where each substitute entity $o_i \in \mathcal{O}$ should have type $\lambda_i \in \Lambda$. Evaluating a predicate on the state by substituting corresponding object entities will result in a ground atom (e.g., `Holding(peg1)`), where a lifted atom is a predicate that maps to typed object variables, which can be viewed as placeholders (e.g., `Holding(?object)`). A task goal $g \in \mathcal{G}$ is represented as a set of ground atoms, where a symbolic state x_Ψ can be obtained by evaluating a set of predicates $\bar{\Psi}$ and keeping all positive ground atoms: $x_\Psi = \textsc{parse}(x, \Psi) \triangleq \{\underline{\psi} : c_{\underline{\psi}}(x) = True, \forall \psi \in \Psi\}$.

Symbolic skill operators. Following prior works [43, 44], we characterize lifted skill operator $\bar{\omega} \in \bar{\Omega}$ by a tuple $< \text{PAR}, \text{PRE}, \text{EFF}^+, \text{EFF}^- >$, where PRE denotes the precondition of the operator, which is a set of lifted atoms defining the condition that the operator is executable. EFF^+ and EFF^- are lifted atoms that describe the expected effects (changes in conditions) upon successful skill execution. PAR is an ordered parameter list that defines all object types used in PRE, EFF^+, and EFF^-. A ground skill operator $\underline{\omega}$ substitutes lifted atoms with object instances: $\underline{\omega} = <\bar{\omega}, \delta> \triangleq < \underline{\text{PRE}}, \underline{\text{EFF}^+}, \underline{\text{EFF}^-} >$, where $\delta : \text{PAR} \rightarrow \mathcal{O}$. Given a task goal, a symbolic task plan is a list of ground operators that, when the instantiated skills executed successfully, leads to a environment state that satisfies the goal condition.

We are interested in learning primitive manipulation skills for accomplishing individual subgoal induced by the expected effects of the corresponding operators – the building blocks that constitute a symbolic task plan. In our setting, each lifted operator $\bar{\omega}$ will have a corresponding skill policy π to be learned, while during execution the ground operators belong to the same lifted operator $\bar{\omega}$ share the same skill policy. We assume access to the predicates $\bar{\Psi}$ and the lifted operators $\bar{\Omega}$ of the environments and focus on efficiently learning the skills for achieving the effects. Note that it is possible to invent and learn predicates and operators [49, 46, 73, 74, 75, 76, 77, 33, 78], but the topics are beyond the scope of this work.

3.2 Skill Learning and Abstraction with Operator Guidance

Action and state abstractions [61] are fundamental to TAMP systems' abilities to solve and generalize across long-horizon tasks [43]. Our key insight is that these abstractions, in the form of symbolic action operators (see Fig. 2 for example), can readily guide RL-trained policies to gain similar abilities. Specifically, for action abstraction, we train temporally-extended skills to reach desired effects of a skill operator by prescribing the effect condition as shaped reward. For state abstraction, we take inspiration from the idea of *information hiding* in feudal learning [35, 36] and use the precondition and effect signature of an operator to determine a *skill-relevant* state space for its corresponding learned policy. This allows the policy to be robust against domains shift and achieve generalization, especially in large environments where most elements are impertinent to a given skill. To further accelerate skill learning, we leverage the existing motion planning capability of a TAMP system to augment the learned skill with a transition primitive. Below we describe each component in details.

Symbolic operators as reward guidance. Our skill learning scheme can build on top of any RL method that supports continuous action space. In this work, we use Soft Actor-Critic (SAC) [79] as the basis for skill learning. SAC leverages entropy regularization to enhance exploration. Given the ground operator $\underline{\omega}$ of a skill, we can define an operator-guided reward $\mathcal{R}_\Psi$ for each individual skill based on continuous environment state x and the action a produced by the corresponding policy π that takes in skill-related state $\dot{x}$ (which will be described later), the objective for our skill learning is therefore rewritten as: $J = \mathbb{E}_{x^0,a^0,\ldots a^{t-1},s^T\sim\pi,p(x^0)}\left[\sum_t \gamma^t(\mathcal{R}_\Psi(x^t,a^t,\underline{\omega}) + \alpha\mathcal{H}(\pi(\cdot|\dot{x}^t))\right]$, where $R_\Psi(\cdot) \mapsto [0,1]$, and $\mathcal{H}(\cdot)$ is the entropy term introduced by SAC.

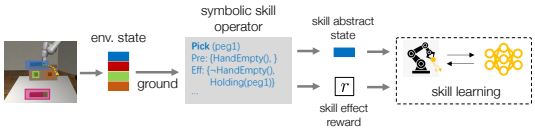

Figure 2: **Operator-guided skill learning and abstraction**. We leverage TAMP-style skill operators as a guidance for skill learning (use desired effect as reward) and state abstraction (enforce skill-relevant state space).

Enhance skill reuse with feudal state abstraction. With the precondition and effect signature of a ground operator $\underline{\omega}$, we can determine a skill-relevant state space to further prevent the learned policy from being distracted by task-irrelevant objects: $\dot{x} = \textsc{extract}(x, \underline{\omega}, \mathcal{O}) \triangleq \{x(o) : o \in \underline{\text{PAR}}, \forall o \in \mathcal{O}\}$, where $\underline{\text{PAR}}$ is the parameter list of the ground operator. For example, for skill `Pick(peg1)`, the skill-related state $\dot{x}$ includes the 6D pose of `peg1` and the end-effector, and the joint parameters of the robot. This design echos previous work that learn to impose constraints on states [67, 80, 34], except that constraints are directly informed by the task planner.

Accelerate learning with transition motion primitives. A key to our method is learning modular manipulation skills that can be composed to solve long tasks. However, for complex manipulation problems, even learning such short skills can be challenging. On the other hand, although TAMP systems fall short when facing contact-rich manipulation, they excel at finding collision-free paths. To this end, we propose to augment our policy with motion planner-based transition primitives. The key idea is to first approach the skill-relevant object (per the skill operator) using off-the-shelf motion planner, before convening RL-based skill learning. The component significantly speeds up the exploration while still allows the system to learn closed-loop contact-rich manipulation skills.

3.3 Integrated Task Planning and Skill Learning

So far, we have described a recipe for learning reusable skills using symbolic skill operators as guidance. But these skills are not learned in silos. A key to LEAGUE's success is to learn skills *in-situ* of a task planning system. The integrated planning and learning scheme ensures that the learned skills are compatible with the planner, and the skill learner can continuously extend the capability of the overall system to solve more tasks. Here we first describe how LEAGUE performs task planning and execution at inference time, and then we introduce an algorithm that uses task plans as an autonomous curriculum to schedule skill learning.

Task planning and skill execution. To plan for task goal g, we first PARSE (See Eq. 3.1) the continuous environment state x for obtaining symbolic state x_Ψ, which affords symbolic search with ground operators. We then ground each lifted operator $\bar{\omega} \in \bar{\Omega}$ on the object set $\mathcal{O}$ by substituting object entities in preconditions and effects, leading to ground operators $\underline{\omega} =< \underline{\text{PRE}}, \underline{\text{EFF}^+}, \underline{\text{EFF}^-} >$ that support operating with symbolic states. A ground operator is considered executable only when its preconditions are satisfied: $\underline{\text{PRE}} \subseteq x_\Psi$. The operators induce an abstract transition model $F(x_\Psi, \underline{\omega})$ that allows planning in symbolic space: $x'_\Psi = F(x_\Psi, \underline{\omega}) \triangleq (x_\Psi \setminus \underline{\text{EFF}^-}) \cup \underline{\text{EFF}^+}$. We use PDDL [81, 82] to build the symbolic planner and use A$*$ search for generating high-level plans.

With the generated task plan, we sequentially invoke the corresponding skill π_l to reach the subgoal that complies with the effects of each ground operator $\underline{\omega}_l$ in the plan. We rollout each skill controller until it fulfills the effects of the operator or a maximum skill horizon H is reached. To verify whether the skill is executed successfully, we first obtain the corresponding symbolic state x^l_Ψ by parsing the ending environment state x^l. The execution is considered successful only when the environment

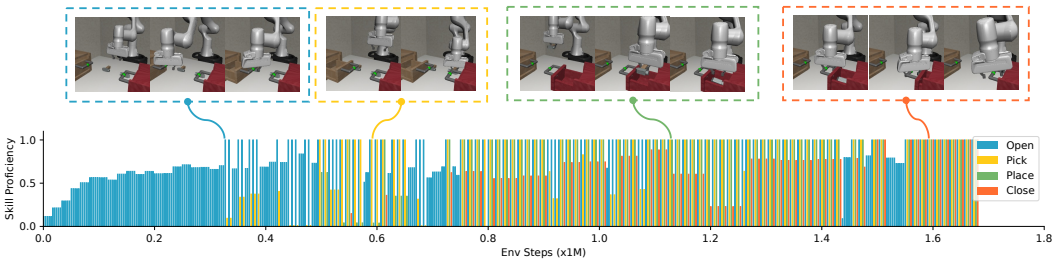

Figure 3: **Visualizing skill learning progress.** The plot shows the proficiency level of each skills throughout the process of learning the task **HammerPlace**. The proficiency is the average normalized reward a skill receive at an iteration.

state x^l conforms to the expected effects: $F(x_\Psi^{l-1}, \underline{\omega}) \subseteq x_\Psi^l$. We keep track of the failed skills to inform the learning curriculum, as described next.

Task planner as autonomous curriculum.

To efficiently acquire all necessary skills for solving a given multi-step task, we leverage task plans as an autonomous curriculum to schedule skill learning in a progressive manner. The key idea is to use more proficient skills to reach the preconditions of skills that require additional learning. The learning algorithm is sketched in Alg. 2 and Alg. 1. On a high level, we repeat task planning and skill learning until convergence. We keep track of failed skills during N task executions and adopt a strict scheduling criteria, where a skill is scheduled for learning (Sec. 3.2) if it ever fails during the N episodes. Notably, for different skill instances (e.g., `Pick(peg1)` and `Pick(peg2)`) that belong to the same lifted operator, we share the replay buffers so that the relevant experience can be reused to further improve the learning efficiency and generalization.

4 Experiments

Our experiments aim to show that 1) our integrated task planning and skill learning framework can progressively learn and refine skills to solve long-horizon tasks and 2) our novel operator-guided skill learning and abstraction algorithm method produces composable and reusable skills, enabling quick adaptation to new tasks and domains.

4.1 Experimental Setup

We conduct evaluations on three simulated manipulation domains: **HammerPlace**, **PegInHole**, and **MakeCoffee**, in which we devise tasks that require multi-step reasoning and long-horizon interactions. The environments are built on Robosuite [37] with Mujoco [83] as the physics engine. We use a Franka Emika Panda robot arm that controlled at frequency 20Hz with an operational space controller (OSC) [84], which has 5 degrees of freedom: the position of the end-effector, the yaw angle, and the position of the gripper. See Figure 4 for an illustration. **HammerPlace** requires the robot to place two hammers into different closed cabinets, where four skill operators are applicable in the environment: `Pick(?object)`, `Place(?object)`, `Pull(?handle)`, `Push(?handle)`. Since the workspace is tight, the robot needs to close an opened cabinet before being able to open the other one, which requires complex reasoning over the task plan. **PegInHole** is to pick up and insert two pegs into two target holes. The applicable operators are `Pick(?object)` and `Insert(?object, ?hole)`. This task challenges the robot with contact-rich manipulations and multi-step planning; **MakeCoffee** is the most challenging task that requires the robot to pick up a coffee pod from a closed cabinet and insert it into the holder of the coffee machine. Finally, the robot also needs to close the lid and the cabinet before finishing the task. The applicable operators are `Pick(?object)`, `Pull(?handle)`, `Push(?handle)`, `CloseLid(?machine)`, and `InsertHolder(?object, ?machine)`. This task is difficult due to the fine-grained manipulation (e.g., take out a round pod from a small drawer and insert pod to a tight hole) and reasoning over multiple steps (i.e., first insert the pod then close the lid and cabinet).

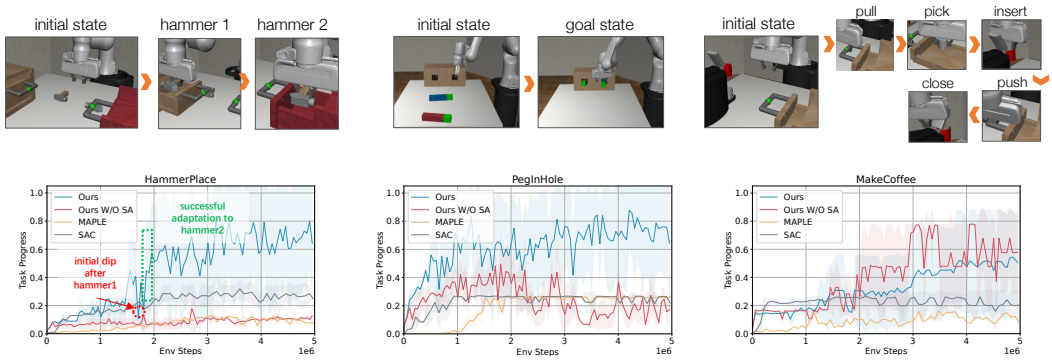

Figure 4: **Main results.** (Top) We visualize the key stages of the three evaluation tasks. (Bottom) We compare relevant methods on the three task domains. The plot shows the corresponding task completion progress (0 for initial state, 1 for task completion) throughout training. The results are reported using four random seeds, with standard deviation shown as the shaded area.

4.2 Visualize Progressive Skill Learning

Before discussing quantitative comparisons, we seek to gain intuitive understanding of our progressive skill learning scheme (Sec. 3.3), where the learning curriculum adjust based on the proficiencies of the skills. In Fig. 3, we visualize the proficiency level of each skill throughout the process of learning the task **HammerPlace**. The y axis shows the average normalized reward a skill receives at an iteration. The corresponding task progress of each skill is visualized in the snapshots on top of the plot. At the beginning of the training, the `Pull(?cabinet)` skill is repeatedly selected for training, until the agent is able to open one of the cabinets. The `Pick(?object)` is then instantiated for learning and execution. Finally at the end of the training, all skills become proficient to be used to execute the entire task. The result qualitatively shows that LEAGUE's automated curriculum is effective at progressively learning skills to achieve long-horizon task goals.

4.3 Quantitative Evaluation

We compare **LEAGUE** with three baseline methods. The first baseline **MAPLE** [6] is a recent state-of-the-art hierarchical RL method that learns a task controller to invoke parametric action primitives or atomic actions. MAPLE is shown to outperform competitive hierarchical RL methods such as DAC [85] and HIRO [23], and learned task controller with open-loop policies [29]. Different than MAPLE that assumes access to a variety of open-loop hand-engineered skills, our method aims to learn closed-loop manipulation skills augmented with a transition motion primitive (Sec. 3.2). To facilitate a fair comparison, we provide MAPLE with staged dense reward based on task plans generated by our task planner, in addition to the affordance-based reward used in their original implementation. The second baseline is a variant of our approach without the proposed state abstraction. We also report the performance of **SAC** [79] that trained with the staged dense reward. We report the normalized task progress score (0 for initial state, 1 for task completion) over the training stages. For example, for a task composed of 8 skills, the successful execution of first 4 skills achieves a progress score of 0.5.

The results are shown in Fig. 4. For easier task **PegInHole** with fewer object states and shorter horizon, **MAPLE** is able to learn to pick up a peg (progress≈ 0.25), but struggles to learn insertion. Our baseline variant that takes in full environment state learns to insert the first peg, but plateaus before picking up the second peg. We hypothesize that learning to pick the second peg causes the policy to forget the policy for the first one. And **LEAGUE** with state abstraction can effectively reuse the knowledge. For harder task like **HammerPlace**, which requires 8 skills to finish, **MAPLE** is stuck at the initial stage after opening the first cabinet, while our method is able to learn all skills efficiently and solve the task. We also found that **SAC** is able to reach the second stage by exploiting the reward function with unexpected behavior (i.e., grasp the head part of the hammer instead of the handle). In addition, we note that our method experiences a performance dip when

switching to the second hammer (as illustrated in the plot). This is because there is a kinematic structure difference between pulling the left drawer and the right one. This phenomenon is also observed when fine-tuning RL policy for a new goal and has been reported in the literature [86, 87]. But the state abstraction allows the skill to quickly adapt the skill to the new task, thus the sharp improvement highlighted in the green region. In the most challenging **MakeCoffee**, **LEAGUE** is able to make reasonable progress but plateaus at inserting the pod and closing the cabinet and lid. Note that because this task does not facilitate in-domain skill reuse, **LEAGUE** performs on par with its full-state baseline.

4.4 Generalization to New Tasks and Domains

To validate that our method can effectively generalize to new task goals and even new task domains by reusing learned skills, we present the following experiments.

Table 1: We report the performance of applying our method to new task goals in the **HammerPlace** (H.P.) and the **PegInHole** (P.I.H.) domains without additional learning.

	Train goal	Test goal1	Test goal2	Mean
H.P.	0.94 ± 0.21	0.90 ± 0.12	0.73 ± 0.31	0.81 ± 0.25
P.I.H.	0.87 ± 0.23	0.53 ± 0.05	1.00 ± 0.00	0.76 ± 0.24

Generalize to new task goals. Here we devise new task goals for the **HammerPlace** and the **PegInHole** domains. For **HammerPlace** domain, the first test goal is to swap the hammer-cabinet mapping. The second test goal is to place `hammer1` into `cabinet2` and keep `cabinet1` open. For **PegInHole**, the first test goal is to swap the peg-hole mapping. The second goals to only insert `peg1` into `hole2`. The results are in Table 1. We observe that LEAGUE experiences little performance drop when generalizing to new task goals without additional training, demonstrating strong compositional generalization capability and skill modularity.

Quick adaptation to new domains.

Another exciting possibility of LEAGUE is to transfer skills learned from one domain to another. We design an experiment to verify this feature. The target domain is **MakeCoffee**, which is the hardest task of the three. We adapt skills `Pick(?object)`, `Pull(?cabinet)`, and `Push(?cabinet)` learned in the **HammerPlace** domain by slightly modifying the preconditions and effects and integrate the skills into learning the **MakeCoffee** task. As shown in Fig. 5, compared to learning from scratch, transferring learned skills can significantly accelerate learning (the x-axis is shorter than in Fig. 4) and enables the robot to solve the entire task. This highlights LEAGUE's strong potential for continual learning.

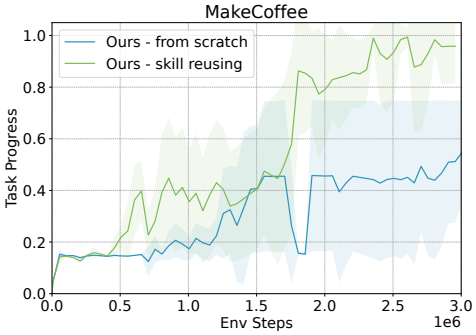

Figure 5: **Generalization to new domain.** For **MakeCoffee** task, we compare (a) learning the task from scratch and (b) learning by adapting the skills (`Pick(?object)`, `Pull(?cabinet)`, and `Push(?cabinet)`) learned from the **HammerPlace** domain.

5 Conclusion

We present LEAGUE, an integrated task planning and skill learning framework. Through challenging manipulation tasks, we demonstrated that LEAGUE is effective at solving long-horizon tasks and generalizing the learned skills to new tasks and domains. Our idea of leveraging TAMP-style skill abstractions for RL-based skill learning allude to a number of open challenges. As we discussed in Sec. 3.1, we assume access to a library of skill operators that serve as the basis for skill learning. Relatedly, our assumptions pertaining to skill-relevant state abstraction, although empirically effective, may not hold in certain cases (e.g. unintended consequences during exploration). A possible path to address both challenges is to learn skill operators with sparse transition models from experience [47, 46, 49].

6 Appendix

We provide the pseudo code of our proposed algorithm for skill learning and planning.

Algorithm 1 PlanningWithSkills

```
hyperparameters:
Number of maximal trials M
Maximal skill horizon H
input:
env                                   ▷ the environment simulator
g                                     ▷ the symbolic task goal
Ψ̄                                     ▷ the state predicates
Ω̄                                     ▷ the lifted operators
Π                                     ▷ the skill controllers
start
O, x^0 ← env.get_state()
x^0_Ψ ← PARSE(x^0, Ψ̄)
Ω ← GROUND(O, Ω̄)
[ω_1, ..., ω_L] ← SEARCH(x^0_Ψ, g, Ω)
T ← 0⃗ ∈ R^L
D ← []
S ← []
l ← 0
while l < L do
    T[l] ← T[l] + 1                   ▷ update number of trials
    i ← map_to_skill(ω_l)
    π_l ← Π[i]
    x^l ← ROLLOUT(env, π_l)
    s^i ← env.get_sim()               ▷ get simulator state
    S ← S ∪ s^i
    if VERIFY(x^l, ω_l) then
        l ← l + 1                     ▷ advance to next skill
        continue
    else if N is odd then
        D ← D ∪ (i, s_i, ω_l)         ▷ collect failed skill and state
    end if
    // backtracking
    while T[l] == M do
        T[l] ← 0
        l ← l − 1                     ▷ reached upper limit of trials
        if l < 0 then
            return D
        end if
    end while
    sim.set_sim(S[l])
end while
return D
```

Algorithm 2 Skill Learning

```
hyperparameters:
Number of training iterations K
input:
env                                   ▷ task environment
g                                     ▷ symbolic task goal
Ψ̄                                     ▷ state predicates
Ω̄                                     ▷ lifted operators
start
Π ← [π^0_1, ..., π^0_N]
// initialize all skill controllers
t ← 0
while Not Converged do
    D ← ∅
    for i ← [1, ..., N] do
        D ← D ∪ PLANNINGWITHSKILLS(env, g, Ψ̄, Ω̄, Π)
    end for
    // evaluate planner and collect failed skills
    for i, s_i, ω ← D do
        π^t_i ← Π[i]
        for k ← [1, ..., K] do
            π^{t+k}_i ← OPTIMIZE(env, π^{t+k−1}_i, ω)
            // RL training
        end for
        Π[i] ← π^{t+K}_i
    end for
    t ← t + K
end while
return Π
```

References

[1] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

[2] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[3] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[4] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[6] S. Nasiriany, H. Liu, and Y. Zhu. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[7] K. Pertsch, Y. Lee, and J. J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020.

[8] M. Dalal, D. Pathak, and R. R. Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21847–21859. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/b6846b0186a035fcc76b1b1d26fd42fa-Paper.pdf.

[9] D. Xu, R. Martín-Martín, D.-A. Huang, Y. Zhu, S. Savarese, and L. F. Fei-Fei. Regression planning networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[10] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1515–1528. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/florensa18a.html.

[11] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=SkT5Yg-RZ.

[12] S. Nair and C. Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1gzR2VKDH.

[13] Y. Zhu, P. Stone, and Y. Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *arXiv preprint arXiv:2109.13841*, 2021.

[14] C. Wang, D. Xu, and L. Fei-Fei. Generalizable task planning through representation pretraining. *arXiv preprint arXiv:2205.07993*, 2022.

[15] V. Veeriah, T. Zahavy, M. Hessel, Z. Xu, J. Oh, I. Kemaev, H. van Hasselt, D. Silver, and S. Singh. Discovery of options via meta-learned subgoals. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 29861–29873, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/fa246d0262c3925617b0c72bb20eeb1d-Abstract.html.

[16] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, S. Savarese, and L. Fei-Fei. Deep affordance foresight: Planning through what can be done in the future. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6206–6213. IEEE, 2021.

[17] T. Nam, S.-H. Sun, K. Pertsch, S. J. Hwang, and J. J. Lim. Skill-based meta-reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022.

[18] K. Lu, A. Grover, P. Abbeel, and I. Mordatch. Reset-free lifelong learning with skill-space planning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[19] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3795–3802. IEEE, 2018.

[20] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8565–8574, 2019.

[21] A. Sharma, A. Gupta, S. Levine, K. Hausman, and C. Finn. Autonomous reinforcement learning via subgoal curricula. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18474–18486. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/99c83c904d0d64fbef50d919a5c66a80-Paper.pdf.

[22] S. Forestier, R. Portelas, Y. Mollard, and P.-Y. Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

[23] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[24] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[25] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

[26] J. Liang, M. Sharma, A. LaGrassa, S. Vats, S. Saxena, and O. Kroemer. Search-based task planning with learned skill effect models for lifelong robotic manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6351–6357. IEEE, 2022.

[27] L. Guan, S. Sreedharan, and S. Kambhampati. Leveraging approximate symbolic models for reinforcement learning via skill diversity. *arXiv preprint arXiv:2202.02886*, 2022.

[28] Y. Lee, J. Yang, and J. J. Lim. Learning to coordinate manipulation skills via skill behavior diversification. In *International conference on learning representations*, 2019.

[29] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta. Efficient bimanual manipulation using learned task schemas. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1149–1155. IEEE, 2020.

[30] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 639–646. IEEE, 2014.

[31] R. Chitnis, T. Silver, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. *arXiv preprint arXiv:2105.14074*, 2021.

[32] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

[33] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez. Learning symbolic operators for task and motion planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3182–3189. IEEE, 2021.

[34] D. Driess, J.-S. Ha, and M. Toussaint. Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. *arXiv preprint arXiv:2006.05398*, 2020.

[35] P. Dayan and G. E. Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

[36] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

[37] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

[38] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *ICRA*, 2011.

[39] L. P. Kaelbling and T. Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

[40] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 440–448, 2020.

[41] M. Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning.

[42] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. 2018.

[43] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *arXiv preprint arXiv:2010.01083*, 2020.

[44] L. P. Kaelbling and T. Lozano-Pérez. Learning composable models of parameterized skills. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 886–893. IEEE, 2017.

[45] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Active model learning and diverse action sampling for task and motion planning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4107–4114. IEEE, 2018.

[46] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, 29:309–352, 2007.

[47] V. Xia, Z. Wang, and L. P. Kaelbling. Learning sparse relational transition models. *International Conference on Learning Representations*, 2018.

[48] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61: 215–289, 2018.

[49] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Perez, L. P. Kaelbling, and J. Tenenbaum. Inventing relational state and action abstractions for effective and efficient bilevel planning. *arXiv preprint arXiv:2203.09634*, 2022.

[50] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[51] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

[52] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[53] M. Fang, T. Zhou, Y. Du, L. Han, and Z. Zhang. Curriculum-guided hindsight experience replay. *Advances in neural information processing systems*, 32, 2019.

[54] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

[55] K. Fang, Y. Zhu, S. Savarese, and L. Fei-Fei. Adaptive procedural task generation for hard-exploration problems. In *International Conference on Learning Representations*, 2020.

[56] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

[57] V. Sarathy, D. Kasenberg, S. Goel, J. Sinapov, and M. Scheutz. Spotter: Extending symbolic planning operators through targeted reinforcement learning. In *AAMAS Conference proceedings*, 2021.

[58] A. Campero, R. Raileanu, H. Küttler, J. Tenenbaum, T. Rocktäschel, and E. Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. In *ICLR*. OpenReview. net, 2021.

[59] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

[60] S. Narvekar, J. Sinapov, M. Leonetti, and P. Stone. Source task creation for curriculum learning. In *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*, pages 566–574, 2016.

[61] D. Abel. A theory of abstraction in reinforcement learning. *arXiv preprint arXiv:2203.00397*, 2022.

[62] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[63] D. Koller and R. Parr. Computing factored value functions for policies in structured mdps. In *IJCAI*, volume 99, pages 1332–1339, 1999.

[64] R. Jonschkowski and O. Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.

[65] A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2020.

[66] D. Abel, N. Umbanhowar, K. Khetarpal, D. Arumugam, D. Precup, and M. Littman. Value preserving state-action abstractions. In *International Conference on Artificial Intelligence and Statistics*, pages 1639–1650. PMLR, 2020.

[67] R. Chitnis, T. Silver, B. Kim, L. P. Kaelbling, and T. Lozano-Perez. Camps: Learning context-specific abstractions for efficient planning in factored mdps. *arXiv preprint arXiv:2007.13202*, 2020.

[68] N. Nejati, P. Langley, and T. Konik. Learning hierarchical task networks by observation. In *Proceedings of the 23rd international conference on Machine learning*, pages 665–672, 2006.

[69] S. Sohrabi, J. A. Baier, and S. A. McIlraith. Htn planning with preferences. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[70] F. Yang, D. Lyu, B. Liu, and S. Gustafson. Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. *arXiv preprint arXiv:1804.07779*, 2018.

[71] L. Illanes, X. Yan, R. T. Icarte, and S. 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.

[72] M. Dalal, D. Pathak, and R. R. Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. *Advances in Neural Information Processing Systems*, 34: 21847–21859, 2021.

[73] H. H. Zhuo, Q. Yang, D. H. Hu, and L. Li. Learning complex action models with quantifiers and logical implications. *Artificial Intelligence*, 174(18):1540–1569, 2010.

[74] E. Ugur and J. Piater. Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2627–2633. IEEE, 2015.

[75] B. Ames, A. Thackston, and G. Konidaris. Learning symbolic representations for planning with parameterized skills. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 526–533. IEEE, 2018.

[76] J. Loula, T. Silver, K. R. Allen, and J. Tenenbaum. Discovering a symbolic planning language from continuous experience. In *CogSci*, page 2193, 2019.

[77] H. H. Zhuo, T. Nguyen, and S. Kambhampati. Refining incomplete planning domain models through plan traces. In *Twenty-third international joint conference on artificial intelligence*. Citeseer, 2013.

[78] A. Arora, H. Fiorino, D. Pellier, M. Métivier, and S. Pesty. A review of learning planning action models. *The Knowledge Engineering Review*, 33, 2018.

[79] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1856–1865, 2018.

[80] A. M. Wells, N. T. Dantam, A. Shrivastava, and L. E. Kavraki. Learning feasibility for task and motion planning in tabletop environments. *IEEE robotics and automation letters*, 4(2): 1255–1262, 2019.

[81] M. Fox and D. Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

[82] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl-the planning domain definition language. 1998.

[83] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[84] O. Khatib. Inertial properties in robotic manipulation: An object-level framework. *The international journal of robotics research*, 14(1):19–36, 1995.

[85] S. Zhang and S. Whiteson. Dac: The double actor-critic architecture for learning options. *Advances in Neural Information Processing Systems*, 32, 2019.

[86] A. Nair, A. Gupta, M. Dalal, and S. Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[87] J. Li, Q. Vuong, S. Liu, M. Liu, K. Ciosek, H. Christensen, and H. Su. Multi-task batch reinforcement learning with metric learning. *Advances in Neural Information Processing Systems*, 33:6197–6210, 2020.