SDRL: Interpretable and Data-Efficient Deep Reinforcement Learning Leveraging Symbolic Planning

Daoming Lyu,[1] Fangkai Yang,[2] Bo Liu,[1] Steven Gustafson[2]
[1]Auburn University, Auburn, AL, USA
[2]Maana Inc., Bellevue, WA, USA
daoming.lyu@auburn.edu, fyang@maana.io, boliu@auburn.edu, sgustafson@maana.io

Abstract

Deep reinforcement learning (DRL) has gained great success by learning directly from high-dimensional sensory inputs, yet is notorious for the lack of interpretability. Interpretability of the subtasks is critical in hierarchical decision-making as it increases the transparency of black-box-style DRL approach and helps the RL practitioners to understand the high-level behavior of the system better. In this paper, we introduce symbolic planning into DRL and propose a framework of Symbolic Deep Reinforcement Learning (SDRL) that can handle both high-dimensional sensory inputs and symbolic planning. The task-level interpretability is enabled by relating symbolic actions to options.This framework features a *planner – controller – meta-controller* architecture, which takes charge of subtask scheduling, data-driven subtask learning, and subtask evaluation, respectively. The three components cross-fertilize each other and eventually converge to an optimal symbolic plan along with the learned subtasks, bringing together the advantages of long-term planning capability with symbolic knowledge and end-to-end reinforcement learning directly from a high-dimensional sensory input. Experimental results validate the interpretability of subtasks, along with improved data efficiency compared with state-of-the-art approaches.

Introduction

Deep reinforcement learning (DRL) algorithms have achieved tremendous success on sequential decision-making problems involving high-dimensional sensory inputs such as Atari games (Mnih et al. 2015). The input states of Atari games are usually raw pixel images, and the deep neural network is used to approximate Q-value, i.e., "Deep Q-Network" (DQN). This approach can learn fine granular policies that surpass human experts but is criticized for the lack of data efficiency and interpretability. DRL algorithms usually require several millions of samples but still cannot learn long-horizon sequential actions for problems with sparse and delayed rewards, such as Montezuma's Revenge (Mnih et al. 2015). The learning behavior based on the black-box neural network is nontransparent and hard to explain and understand. The goal of interpretability is to describe the internals of a system in a way that is understandable to humans. In real applications of decision-making, it is instrumental to make the system behavior interpretable to gain the trust from the user and provide insights for their decision-making process (Gilpin et al. 2018). Studying on these two topics has received increasing attention from the machine learning community. Recent study in Neuroscience suggests human plays video games by learning and planning from a high-level object-based representation of a deterministic transition model of the underlying problem (Tsividis et al. 2017). This observation sheds lights on improving data efficiency by introducing a hierarchy over flat MDP problem using subtasks, i.e., a temporal abstraction over a course of primitive actions, and utilizing object representation for learning (Kulkarni et al. 2016) or strategic planning (Keramati et al. 2018). Although these approaches achieved some success on data efficiency, none of them reaches the same level of interpretability as much as object-based subtask planning and learning that human performs.

In this paper, we argue that performing reasoning and planning on explicitly represented knowledge is an effective way to achieve the task-level interpretability and propose to use *Symbolic Planning* (SP) (Cimatti, Pistore, and Traverso 2008) as a descriptive and intuitive high-level technique to improve the data-efficiency and interpretability of DRL. Symbolic planning has been used to build mobile robots that co-inhabit with human, perform tasks for human and communicate with human for task-relevant information (Hanheide et al. 2015; Chen, Yang, and Chen 2016; Khandelwal et al. 2017), all requiring high-level interpretability of their behavior. Different from RL approaches, a planning agent carries prior symbolic knowledge of objects, properties and how they are changed by executing actions in the dynamic system, represented in a formal, logic-based language such as PDDL (McDermott et al. 1998) or an action language (Gelfond and Lifschitz 1998) that relates to logic programming under answer set semantics (answer set programming) (Lifschitz 2008). The agent utilizes a symbolic planner, such as a PDDL planner FastDownward (Helmert 2006) or an answer set solver Clingo (Gebser, Kaufmann, and Schaub 2012) to generate a sequence of actions based on its symbolic knowledge, executes the actions to achieve its goal, perform execution monitor and replan to handle execution failure and domain uncertainty. Compared with RL approaches, an SP agent does not require a large number of trial-and-error to behave reasonably well, and the interpretability of

the agent's behavior is well guaranteed by white-box algorithms of planning and reasoning with predefined and human-readable symbolic knowledge. Finally, recent work on integrating symbolic planning with RL (Yang et al. 2018; Lu et al. 2018) provides SP+RL frameworks where symbolic plans based on prior knowledge can guide RL for meaningful exploration, leading to improvement on decision-making.

This paper advances the state of the art SP+RL approach by integrating symbolic planning with a hierarchical approach of DRL. We made two assumptions. First, it is relatively easy for human expert to represent knowledge about high-level abstract, albeit inaccurate, dynamics of the problem domain. Second, due to the recent progress on computer vision, it is relatively easy to build perception modules to recognize objects and their properties. Based on the assumptions, we propose the *intrinsic goal*, a measurement of plan quality based on an internal utility function, to enable reward-driven planning. Afterwards, we propose a *Symbolic Deep Reinforcement Learning* (SDRL) framework that features a *planner–controller–meta-controller* architecture, i.e.,

1. A planner uses prior symbolic knowledge to perform long-term planning by a sequence of symbolic actions (subtasks) that achieve its intrinsic goal;
2. A controller uses DRL algorithms to learn the sub-policy for each subtask based on *intrinsic rewards*;
3. A meta-controller learns on *extrinsic rewards* by measuring the training performance of controllers and propose new intrinsic goals to the planner.

In this process, planner, controllers, and meta-controller cross-fertilize each other and eventually converge to an optimal symbolic plan along with the learned subtasks. To the best of our knowledge, this is the first work that integrates symbolic planning with DRL that gains interpretability and data-efficiency for decision-making in complex domains. While our framework is generic enough so that various planning and DRL techniques can be used, we instantiate our framework using action language $\mathcal{BC}$ (Lee, Lifschitz, and Yang 2013) for planning and R-learning (Schwartz 1993; Mahadevan 1996) for meta-controller learning. We prove that the symbolic plan converges to optimal conditioned on the convergence of meta-controller. The framework is evaluated on Taxi domain (Barto and Mahadevan 2003) and Montezuma's Revenge, demonstrating improved interpretability through explicitly encoding planning knowledge and learning into human-readable subtasks, and also improved data-efficiency through automatic selecting and learning control policies of modular subtasks.

Preliminaries

Planning with Action Language $\mathcal{BC}$. An *action description* D in the language $\mathcal{BC}$ includes two kinds of symbols on signature σ, *fluent constants* that represent the properties of the world, and *action constants*. A *fluent atom* is an expression of the form $f = v$, where f is a fluent constant and v is an element of its domain. For boolean domain, denote $f = \mathbf{t}$ as f and $f = \mathbf{f}$ as $\sim f$. An action description is a finite set of *causal laws* that describe how fluent atoms are related with each other in a single time step, or how their values are changed from one step to another, possibly by executing actions. For instance, $(A$ **if** $A_1, \ldots, A_m)$ is a *static law* that states at a time step, if $A_1, \ldots, A_m$ holds then A is true. Another static law (**default** $f = v$) states that by default, the value of f equals v at any time step. $(a$ **causes** A_0 **if** $A_1, \ldots, A_m)$ is a *dynamic law*, stating that at any time step, if $A_1, \ldots, A_m$ holds, by executing action a, A_0 holds in the next step. (**nonexecutable** a **if** $A_1, \ldots, A_m)$ states that at any step, if $A_1, \ldots, A_m$ holds, action a is not executable. Finally, the dynamic law (**inertial** f) states that by default, the value of fluent f does not change with time.

An action description captures a dynamic transition system. A *symbolic state* s is a complete set of fluent atoms, and a transition is a tuple $\langle s, a, s' \rangle$ where s, s' are states and a is an action. Let I and G be states. The triple (I, G, D) is called a planning problem. (I, G, D) has a plan of length $l - 1$ iff there exists a transition path of length l such that $I = s_1$ and $G = s_l$. Throughout the paper, we use Π to denote both the plan and the transition path by following the plan. Due to the semantic definition above, automated planning with an action description in $\mathcal{BC}$ can be achieved by an answer set solver such as CLINGO, and the output answer sets encode the transition paths that solve the planning problem.

Reinforcement Learning. A Markov Decision Process (MDP) is defined as the tuple $(\mathcal{S}, \mathcal{A}, P^a_{ss'}, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of symbols denoting states and actions, the transition kernel $P^a_{ss'}$ specifies the probability of transition from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$, $r(s, a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is a reward function bounded by $r_{\max}$, and $0 \leq \gamma < 1$ is a discount factor. A solution to an MDP is a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ that maps a state to an action. RL learns a near-optimal policy by executing actions and observing the state transitions and rewards, and can be applied when the underlying MDP is not known.

To evaluate a policy π, there are two types of performance measures: the expected discounted sum of reward for infinite-horizon problems and the expected undiscounted sum of rewards for finite horizon problems. In this paper we adopt the latter metric defined as $J^\pi_{\text{avg}}(s) = \mathbb{E}[\sum_{t=0}^{T} r_t | s_0 = s]$, where T denotes the horizon length. We define the *gain reward* $\rho^\pi(s)$ reaped by policy π from s as $\rho^\pi(s) = \lim_{T\to\infty} \frac{J^\pi_{\text{avg}}(s)}{T} = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}[\sum_{t=0}^{T} r_t]$. R-learning is a model-free value iteration algorithm that can be used to find the optimal policy for average reward criteria. At the t-th iteration (s_t, a_t, r_t, s_{t+1}), update:

$$R_{t+1}(s_t, a_t) \xleftarrow{\alpha_t} r_t - \rho_t(s_t) + \max_a R_t(s_{t+1}, a),$$

$$\rho_{t+1}(s_t) \xleftarrow{\beta_t} r_t + \max_a R_t(s_{t+1}, a) - \max_a R_t(s_t, a)$$

where α_t, β_t are the learning rates, and $a_{t+1} \xleftarrow{\alpha} b$ denotes the update law as $a_{t+1} = (1 - \alpha)a_t + \alpha b$.

Options. Compared with regular RL, hierarchical reinforcement learning (HRL) (Barto and Mahadevan 2003) specifies on real-time-efficient decision-making problems over a se-

ries of tasks. An MDP can be considered as a flat decision-making system where the decision is made at each time step. On the contrary, humans make decisions by incorporating temporal abstractions. An *option* is temporally extended course of action consisting of three components: a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, a termination condition $\beta : \mathcal{S} \mapsto [0, 1]$, and an initiation set $\mathcal{I} \subseteq \mathcal{S}$. An option (I, π, β) is available in state s_t iff $s_t \in I$. After the option is taken, a course of actions is selected according to π until the option is terminated stochastically according to the termination condition β. With the introduction of options, the decision-making has a hierarchical structure with two levels, where the upper level is the option level (also termed as task level) and the lower level is the (primitive) action level. Markovian property exists among different options at the option level.

Related Work

Interpretability. Studying on interpretability concerns on describing the internals of a system in a way that is understandable to humans (Doshi-Velez and Kim 2017; Gilpin et al. 2018). Interpretability of deep learning involves studying on explaining deep network processing (Ribeiro, Singh, and Guestrin 2016). For interpretive DRL, program induction approach is used (Verma et al. 2018) to enable policy interpretability. Our work is the first to use symbolic knowledge to enable task-level interpretability for DRL and it is interpretable by construction.

Hierarchical Deep Reinforcement Learning. Hierarchical RL approach such as the options framework (Sutton, Precup, and Singh 1999) formulates the problem using a two-level hierarchy as aforementioned and is one way to solve the challenge of learning long horizon action sequences with sparse rewards. It often assumes that a set of useful options are predefined. (Machado, Bellemare, and Bowling 2017; Machado et al. 2017) focus on discovering eigen-based options and also attempt to solve the problem of learning policies over long time horizons. However, it is difficult to interpret the options in their approaches. The closest hierarchical RL work to ours are (Kulkarni et al. 2016; Le et al. 2018), utilizing a meta-controller to learn to sequence subtasks defined on objects. In addition, (Keramati et al. 2018) focuses on performing strategic planning and learning in Object-oriented MDP, a deterministic abstraction over the original problem. By contrast, we leverage symbolic action languages to explicitly represent objects, properties, and the high-level transition dynamics. We use an out-of-box symbolic planner to generate and improve plans, with symbolic transitions automatically mapped to subtasks, leading to a more interpretable and expressive representation.

Integrating Symbolic Planning with Reinforcement Learning. SP-RL integration has been studied for a long time (Parr and Russell 1998; Ryan 2002; Hogg, Kuter, and Munoz-Avila 2010; Leonetti, Iocchi, and Stone 2016; Yang et al. 2018; Lu et al. 2018), most of which are based on tabular representation of the domain. Our work inherits the interpretability of symbolic planning with symbolic knowledge and further generalizes previous PEORL framework (Yang et al. 2018) with intrinsic goals and the integration with DRL. In particular, the meta-controller is introduced to bridge the gap of *planning over symbolic states and DRL over pixel images*, by learning at the task level using extrinsic reward derived from training performance of DRL. Meta-controller learning enables the planner to automatically select subtasks and improve the plan by sequencing learnable subtasks.

Computational Models of Intrinsic Motivation. Recent studies (Kulkarni et al. 2016) showed that characterizing *intrinsic motivation* is important to address learning goal-directed behavior facing sparse and delayed rewards. In psychology, intrinsic motivation is defined as accomplishing an activity for its inherent satisfaction rather than for some separable consequences, driven by an internal utility function (Oudeyer and Kaplan 2009). Intrinsically-motivated RL (Chentanez, Barto, and Singh 2005) uses the framework of options. In comparison, our work provides a computational model where symbolic planning uses its internal utility function to measure its plan quality, in order to motivate the agent to improve the plan by accumulating larger rewards.

SDRL Framework

We model the underlying sequential decision-making problem as an MDP tuple $(\widetilde{\mathcal{S}}, \widetilde{\mathcal{A}}, \widetilde{P^a_{ss'}}, r, \gamma)$ where $\widetilde{\mathcal{S}}$ consists of states of high-dimensional sensory inputs such as pixel images, $\widetilde{\mathcal{A}}$ is the set of primitive actions, $\widetilde{P^a_{ss'}}$ is the transition matrix, r is the reward function, and γ is a discounting factor. In the following paper, $\widetilde{\mathcal{S}}, \widetilde{\mathcal{A}}$ are used to denote the MDP state space and action space, while $\mathcal{S}, \mathcal{A}$ represent the symbolic state space and action space. Our goal is to learn both a sequence of subtasks and the corresponding sub-policies, so that executing the sub-policy for each subtask one by one can achieve maximal cumulative reward.

To solve this problem, we assume a symbolic structure, i.e., a set of causal rules that captures objects, fluents and how their values are changed by executing subtasks, is given by human experts. While a pre-defined symbolic representation requires some work from human experts, it has been observed that majority of discrete dynamic domains share surprising similarities and can be formulated based on a set of general-purpose action modules, and symbolic representation is elaboration-tolerant: adding new information usually doesn't require significant change of existing knowledge. Consequently, the symbolic formulation for one problem can be easily applied to another, by instantiating a different set of objects or adding a few more rules. For instance, Taxi domain, a benchmark problem of HRL, concerns the movement of a taxi in a gridworld, carrying passengers and dropping off passenger at destination, while Montezuma's Revenge, a seemingly drastically different Atari game, concerns the movement of an Avatar among a set of locations (ladders, platforms, doors, ropes, etc), picking up a key and using the key to open a door. Both domains involve the formulation of spatial movement, co-location of objects and utility of objects. Furthermore, our framework is intended to start with a coarse granular, high-level abstract domain formulation, so that decision-making can be robust and flexible facing domain change and uncertainty. Consequently, the laborious effort of crafting an accurate symbolic model is neither necessary nor useful.

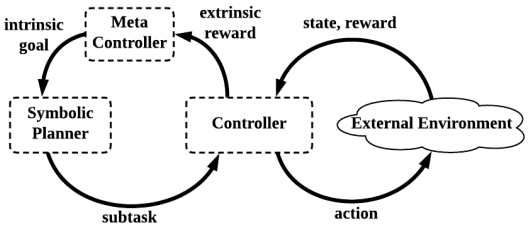


Figure 1: Architecture illustration

With a symbolic representation given by the human expert, the SDRL architecture is shown in Fig. 1. A symbolic planner generates high-level plans, i.e., a sequence of subtasks, to meet its *intrinsic goal*. An intrinsic goal is a measurement on plan quality, which approximates how much cumulative reward the plan may achieve. We assume a pre-trained function can associate each sensory input with a symbolic state, i.e., performing symbol grounding, so that subtasks on the MDP space can be induced based on symbolic states and the pre-trained function. We extend the reward structure of core MDP by introducing *intrinsic reward* and *extrinsic reward* to facilitate two levels of learning tasks. The sub-policies for the action level are learned using DRL algorithms based on intrinsic reward, with pseudo-rewards to encourage the agent to learn skills to achieve each subtask. As DRL continues, a metric is used to evaluate the competence of learned sub-policies, such as the success ratio over a number of episodes, from which extrinsic rewards is derived. When the sub-policy is learned and reliably achieves the subtask, the extrinsic reward is equivalent to the environmental reward. Using extrinsic rewards, meta-controller performs R-learning that reflects the long-term average reward and gains the reward of selecting each subtask. The learned values are returned to the symbolic planner, and used to measure plan quality and propose new intrinsic goals for the planner to improve the plan, by either exploring new subtasks or by sequencing learned subtasks that supposedly can achieve higher rewards in the next iteration. We define the process formally as follows.

Symbolic Representation

A tuple (I, G, D) is a symbolic planning problem, where D is an *action description* defined on signature σ, I is initial state and G is the *intrinsic goal*.

Similar to PEORL, we use action language $\mathcal{BC}$ to demonstrate the essence of the action description, but similar formulation can be represented easily using other planning languages such as PDDL. In addition to usual causal laws that describe the preconditions and effects of actions (dynamic laws) and static relationships between fluents (static law), D consists of causal laws that formulate gain rewards of executing actions and its effect on cumulative plan quality:

- For any symbolic state that contains atoms $\{A_1, \ldots, A_n\}$, D contains static laws of the form: s **if** $A_1, \ldots, A_n,$ for state $s \in \mathcal{S}$.
- We introduce new fluent symbols of the form $\rho(s, a)$ to denote the gain reward at state s following action a. D contains a static law stating by default, the gain reward is initialized optimistically, denoted as *INF*, to promote exploration: **default** $\rho(s, a) = \mathit{INF},$ for $s \in \mathcal{S}, a \in \sigma_A(D)$.
- We use fluent symbol *quality* to denote the cumulative average-adjusted reward of a plan, termed as *plan quality*. D contains dynamic laws of the form: a **causes** $\mathit{quality} = C + Z$ **if** $s, \rho(s, a) = Z, \mathit{quality} = C$.
- D contains a set P of facts of the form $\rho(s, a) = z$.

I is the initial symbolic planning state, and G is an *intrinsic goal* which is a linear constraint of the form

$$\mathit{quality} > \mathit{quality}(\Pi), \tag{1}$$

for a symbolic plan Π measured by the internal utility function *quality* defined as

$$\mathit{quality}(\Pi) = \sum_{\langle s_i, a_i, s_{i+1}\rangle \in \Pi} \rho(s_i, a_i). \tag{2}$$

The definition of intrinsically motivated goal (1) is different from standard PEORL, in which a goal consists of the linear constraint of form (1) plus a set of logical constraints specifying the goal condition, given by the human designer towards a particular task. Intrinsic goal in SDRL drops the logical constraint part and enables "model-based exploration by planning", which is more suitable for RL problems where the agent's behavior is driven by reward.

From Symbolic Transitions to Options

Given the set $\mathcal{S}$ of symbolic states, i.e., a complete set of fluent atoms, we assume there is a pre-trained oracle capable of answering whether the symbolic properties specified as fluent atoms of the form $f = v$ in s are true in the high-dimensional sensory input $\tilde{s}$, and define the mapping $\mathbb{F}$ as $\mathbb{F} : \mathcal{S} \times \widetilde{\mathcal{S}} \mapsto \{\mathbf{t}, \mathbf{f}\}$. In the case of Atari games, such pre-trained function can be a perception module that performs object recognition and performs symbol grounding based on the predefined semantics of symbols. For instance, the perception module can answer if the avatar picked the key by checking if the bounding box of the avatar overlaps with the bounding box of the key. Due to the recent progress of computer vision, we assume such a perception module is generally available.

Given $\mathbb{F}$ and a pair of symbolic states $s, s' \in \mathcal{S}$, we can induce a semi-Markov option, as a triple (I, π, β) where the initiation set $I = \{\tilde{s} \in \widetilde{\mathcal{S}} : \mathbb{F}(s, \tilde{s}) = \mathbf{t}\}$, $\pi : \widetilde{\mathcal{S}} \mapsto \widetilde{\mathcal{A}}$ is the intra-option policy, and β is the termination condition such that $\beta(\tilde{s'}) = \begin{cases} 1 & \mathbb{F}(s', \tilde{s'}) = \mathbf{t}, \text{for } \tilde{s'} \in \widetilde{\mathcal{S}} \\ 0 & \text{otherwise} \end{cases}$. The formulation above maps symbolic transition to a similar structure of options.

Rewards

To facilitate learning at the action level and the task level, we define intrinsic reward at the action level as

$$r_i(\tilde{s'}) = \begin{cases} \phi & \beta(\tilde{s'}) = 1 \\ r & \text{otherwise} \end{cases} \tag{3}$$

where ϕ is a positive constant encouraging achieving subtasks and r is the reward from the environment at state $\tilde{s'}$. If reward

is sparse, (3) is usually a simple binary form. We further define extrinsic reward for selecting subtask g at symbolic state s as $r_e(s,g) = f(\epsilon)$ where f is a function about ϵ, a criterion that measures the competence of the learned sub-policy for each subtask. We define ϵ as the success ratio, which is the average rate of successfully completing the subtask over the previous 100 episodes. f can be defined as

$$f(\epsilon) = \begin{cases} -\psi & \epsilon < 0.9 \\ r(s,g) & \epsilon \geq 0.9 \end{cases} \quad (4)$$

where $-\psi$ is a negative constant to punish selecting unlearnable subtasks, $r(s,g)$ is the discounted cumulative reward obtained from the environment by following the subtask g, and 0.9 is the threshold. Unlearnable sub-tasks here refers to the sub-tasks that are too difficult to learn by the controller on the condition that the success ratio of achieving a sub-task cannot keep above the threshold value of 0.9 after the training by episodes. Intuitively, the definition of extrinsic rewards means if the sub-policy can reliably achieve the subgoal, then the extrinsic reward at s' reflects true cumulative environmental reward of following the subtask; otherwise, the extrinsic reward at s' is negative, indicating that the sub-policy performs badly and is probably not learnable.

A plan Π of (I, G, D) is considered to be *optimal* iff $\sum_{\langle s,a,s' \rangle} r_e(s,g)$ is maximal among all plans.

Planning and Learning

The planning and learning process is shown in Algorithm 1. At any episode t, symbolic planner uses a logical representation D, an initial state I, and an intrinsic goal G to generate a symbolic plan Π_t (Line 4).The symbolic transitions of Π_t correspond to subtasks (Line 10) and sent to a controller to learn the sub-policy for each subtask over a predefined number of steps (Line 11). The controller performs deep Q-learning with intrinsic rewards r_i using experience replay (Lines 12–15). The controller estimates the Q value $Q(\tilde{s}, \tilde{a}; g) \approx Q(\tilde{s}, \tilde{a}; \theta, g)$, where θ is the parameter of the non-linear function approximator. The experience of executing actions in the environment $(\tilde{s}_t, \tilde{a}_t, r_e(\tilde{s}_{t+1}, g), \tilde{s}_{t+1})$ is stored in memory $\mathcal{D}_g$, and the loss function is defined as

$$L(\theta; g) = \mathrm{E}_{(\tilde{s},\tilde{a},g,r_i,\tilde{s}') \sim D_g} \\ [r_e + \gamma \max_{\tilde{a}'} Q(\tilde{s}, \tilde{a}'; \theta_{i-1}, g) - Q(\tilde{s}, \tilde{a}; \theta_i, g)]^2 \quad (5)$$

where i denotes the iteration number. After maximal steps are reached, the success ratio of controller's sub-policy or the true environmental rewards are used to derive extrinsic rewards (Line 17). Meta-controller performs R-learning (Line 18) based on extrinsic rewards for the symbolic transitions $\langle s_t, a_t, s_{t+1} \rangle$ that corresponds to the subtask g_t:

$$R_{t+1}(s_t, g_t) \xleftarrow{\alpha} r_e - \rho_t^{g_t}(s_t) + \max_g R(s_t, g) \\ \rho_{t+1}^{g_t}(s_t) \xleftarrow{\beta} r_e + \max_g R_t(s_{t+1}, g) - \max_g R_t(s_t, g) \quad (6)$$

After R-learning is performed, the *quality* of the symbolic plan Π_t is measured by (2) (Line 20). The plan quality *quality*(Π_t) is used to update intrinsic goal (Line 21) and learned ρ values are passed back into symbolic formulation for a new plan to be generated (Line 22). The loop continues until the symbolic plan Π^* cannot be further improved.

The algorithm guarantees symbolic level optimality conditioned on R-learning convergence. See Appendix for proof.

Algorithm 1 SDRL Planning and Learning Loop

Require: $(I, G, D, \mathbb{F})$ where $G = (\textit{quality} > 0)$, and an exploration probability ϵ

1: Initialization: $P_0 \Leftarrow \emptyset$, $\Pi_0 \Leftarrow \emptyset$
2: **for** t = 1 ... end of episodes **do**
3: $\quad \Pi^* \Leftarrow \Pi_{t-1}$
4: $\quad$ take ϵ probability to solve planning problem and obtain a plan $\Pi_t \Leftarrow$ CLINGO.*solve*$(I, G, D \cup P_{t-1})$
5: $\quad$ **if** $\Pi_t = \emptyset$ **then**
6: $\quad\quad$ **return** Π^*
7: $\quad$ **end if**
8: $\quad$ **for** symbolic transition $\langle s, a, s' \rangle \in \Pi_t$ **do**
9: $\quad\quad$ obtain current state $\tilde{s}$
10: $\quad\quad$ correspond to subtask g by using $\mathbb{F}$ to obtain initiation set and terminate condition
11: $\quad\quad$ **while** $\beta(\tilde{s}) \neq 1$ and maximal step is not reached **do**
12: $\quad\quad\quad$ pick up an action $\tilde{a}$ and obtain transition $(\tilde{s}, \tilde{a}, \tilde{s'}, r_i(\tilde{s'}))$
13: $\quad\quad\quad$ store transition in experience replay buffer $\mathcal{D}_g$
14: $\quad\quad\quad$ estimate $Q(\tilde{s}, \tilde{a}; \theta, g)$ by minimizing loss function (5) when there are sufficient samples in $\mathcal{D}_g$
15: $\quad\quad\quad$ update current state $\tilde{s} \Leftarrow \tilde{s'}$
16: $\quad\quad$ **end while**
17: $\quad\quad$ calculate extrinsic reward $r_e(s, g)$
18: $\quad\quad$ update $R(s, g)$ and $\rho^g(s)$ using (6).
19: $\quad$ **end for**
20: $\quad$ calculate quality of Π_t by (2).
21: $\quad$ update planning goal $G \Leftarrow (\textit{quality} > \textit{quality}(\Pi_t))$.
22: $\quad$ update facts $P_t \Leftarrow \{\rho(s, a) = z : \langle s, a, s' \rangle \in \Pi_t, \rho_t^a(s) = z\}$
23: **end for**

Theorem 1 (Termination)**.** *If the meta-controller's R-learning converges, Algorithm 1 terminates iff an optimal symbolic plan exists.*

Theorem 2 (Optimality)**.** *If meta-controller's R-learning converges, when Algorithm 1 terminates,* Π^* *is an optimal symbolic plan.*

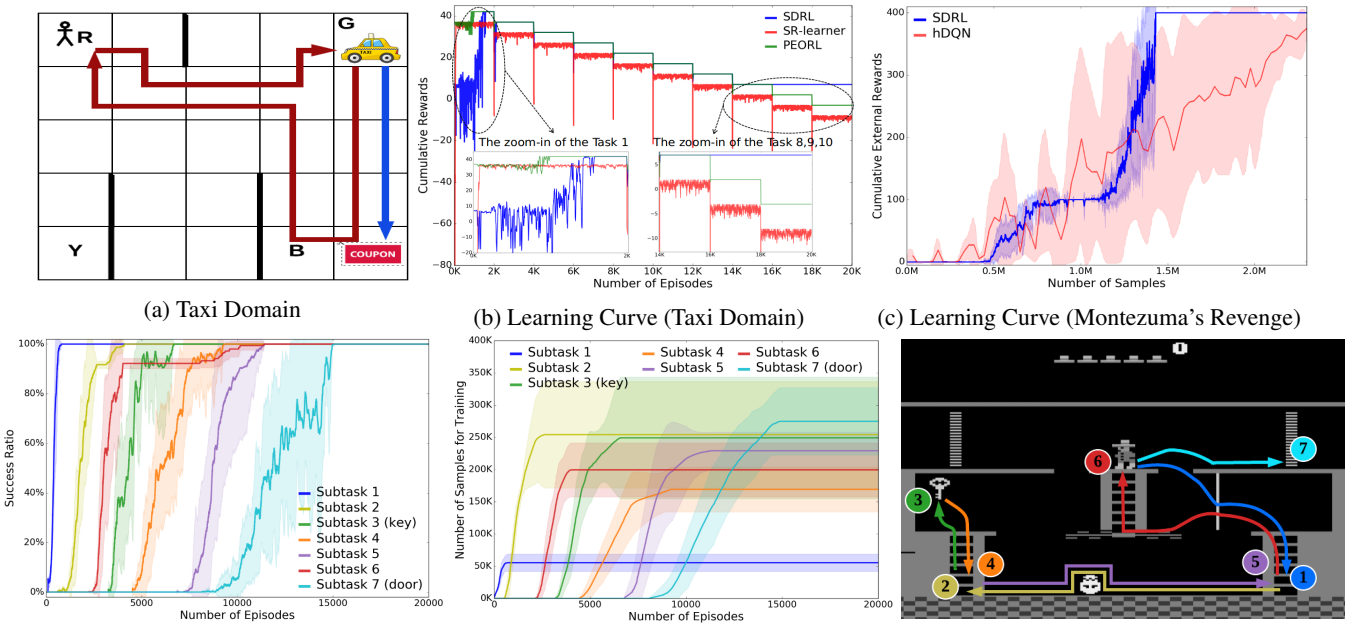
Experiment

We use Taxi domain to demonstrate the behavior of intrinsically motivated planning, and on Montezuma's Revenge for interpretability and data-efficiency. We use 1M to denote 1 million and 1k to denote 1000.

Taxi Domain

A Taxi starts at any location in a 5×5 grid map (Fig. 2a) with a passenger and a destination. Every movement has a reward -1. Successful drop-off receives reward 50. Improper pick-up or drop-off receive a reward -10. We introduce an extra coupon at $(4, 4)$ where the taxi can only collect once, with gaining a reward of 10. In tabular representation, the intrinsic goal is the only difference between SDRL and standard PEORL, and we will demonstrate how intrinsically motivated goal affects exploration.

Setup. We consider a sequence of 10 tasks. For each task, the reward of successfully dropping off the passenger declines by 5. For example, the reward of successful dropping off the passenger in Task 1 is 50, while that of Task 2 would



(a) Taxi Domain

(b) Learning Curve (Taxi Domain)

(c) Learning Curve (Montezuma's Revenge)

(d) Success Ratio (Montezuma's Revenge)

(e) Samples Required (Montezuma's Revenge)

(f) Final Solution (Montezuma's Revenge)

Figure 2: Experimental Results

be 45. Reward change happens after every 2000 episodes, and the taxi's location is always reset to $(0, 4)$. Standard PEORL has a fixed final goal, i.e., drop off the passenger at the destination. We compare SDRL with standard PEORL and a linear successor representation (SR) learner (Lehnert, Tellex, and Littman 2017), a common approach to implement transferable RL for tasks with reward change.

Experimental Results. Result is collected and averaged over 10 runs, and learning curves of cumulative reward are shown in Fig. 2b. From Task 1 to Task 7, the optimal policy is to pick up the coupon and then drop off the passenger (the route with dark red color in Fig. 2a). Both Standard PEORL and SDRL can learn the optimal policy. As the zoom-in of the Task 1 (first 2k episodes) shown in Fig. 2a, SDRL does not converge as fast as SR-learner initially, due to the fact that the exploration of SDRL based on planning with intrinsic goals is not as aggressive as model-free exploration. SDRL does not converge as fast as standard PEORL either because SDRL is not given the explicit planning goal of dropping off the passenger and it needs to explore the states to discover the reward, unlike standard PEORL. SR-learner performs the worst by only dropping off the guest without picking up the coupon. After exploring the state space in Task 1, from Task 2 onwards, SDRL converges as fast as SR-learner and standard PEORL. However, as the reward of dropping off the passenger keeps declining, from Task 8 onwards, the optimal policy changes to just pick up the coupon. The new optimal policy is successfully learned by SDRL due to intrinsic goal (the route with blue color in Fig. 2a), while both standard PEORL and SR learner does not change its policy from previously learned ones. The inadequacy of SR in transferring to an optimal policy with a different goal was also pointed out in (Lehnert, Tellex, and Littman 2017). Standard PEORL cannot change its plan due to the fixed planning goal, showing that an explicitly given planning goal may unnecessarily restrict exploration of RL, while intrinsic goals in SDRL allows the agent to flexibility changes its goal based on reward structure of the tasks, which is more suitable to solve RL problems.

Montezuma's Revenge.

"Montezuma's Revenge" requires the player to navigate the explorer through several rooms while collecting treasures. For the first room, the player has to first pick up the key by climbing down the ladders and moving towards the key in order to pass through doors, resulting in a long sequence of actions before receiving a reward for collecting the key $(+100)$. After that, the player has to move towards the door and open it, which results in another reward $(+300)$. Optimal execution requires more than 200 primitive actions. Vanilla DQN frequently achieves a score of 0 on this domain (Mnih et al. 2015).

Setup. Our experiment setup follows the DQN controller architecture (Kulkarni et al. 2016) with double-Q learning (Van Hasselt, Guez, and Silver 2016) and prioritized experience replay (Schaul et al. 2015). The architecture of the deep neural networks is shown in Table 1. The experiment is conducted using Arcade Learning Environment (ALE) (Bellemare et al. 2013). We build customized algorithms based on ALE API to recognize the locations of the agent, the skull, ladders and platforms from pixels and the mapping function $\mathbb{F}$. The intrinsic reward follows (3) with $\phi = 1$ and $r = -1$ when the agent loses its life. Extrinsic reward follows (4) where $\psi = 100$ and define $r(s, g) = -10$ for $\epsilon > 0.9$ to

encourage shorter plan. We use hierarchical DQN (hDQN) (Kulkarni et al. 2016) as the baseline.

No.	Layer	Details
1	Convolutional Layer	32 filters, kernel size=8, stride=4, activation='relu'
2	Convolutional Layer	64 filters, kernel size=4, stride=2, activation='relu'
3	Convolutional Layer	64 filters, kernel size=3, stride=1, activation='relu'
4	Fully Connected Layer	512 nodes, activation='relu'
5	Output Layer	activation='linear'

Table 1: Neural Network Architecture for Montezuma's Revenge

```
% object declaration
location(mp;rd;ls;lll;lrl;key).
% dynamic causal law declaration
move(L) causes loc=L if location(L).
move(L) causes cost=L+Z if rho((at(L1)),move(L))=Z,
           loc=L1,picked(key)=false.
move(L) causes cost=L+Z if rho((at(L1),picked(key)),
           move(L))=Z,loc=L1,picked(key)=true.
inertial loc. inertial quality.
% static causal law declaration
picked(key)=true if loc=key.
nonexecutable move(key) if picked(key).
default rho((at(L1)),move(L))=10.
default rho((at(L1),picked(key)),move(L))=10.
```

Figure 3: Montezuma's Revenge in $\mathcal{BC}$

No.	subtask	policy learned	in optimal plan
1	MP to LRL, no key	✓	✓
2	LRL to LLL, no key	✓	✓
3	LLL to key, no key	✓	✓
4	key to LLL, with key	✓	✓
5	LLL to LRL, with or without key	✓	✓
6	LRL to MP, with or without key	✓	✓
7	MP to RD, with key	✓	✓
8	LRL to LS, with or without key	✓	
9	LS to key, with or without key	✓	
10	MP to RD, no key	✓	
11	LRL to key, with or without key		
12	key to LRL, with key		
13	LRL to RD, with key		

Table 2: Subtasks for Montezuma's Revenge

Symbolic Representation. We formulated domain knowledge in action language $\mathcal{BC}$ (Fig. 3) based on 6 pre-defined locations: middle platform (`mp`), right door (`rd`), left of rotating skull (`ls`), lower left ladder (`lll`), lower right ladder (`lrl`), and key (`key`). The input language can be processed by software CPLUS2ASP and translated into the input language of CLINGO for symbolic planning. The function $\mathbb{F}$ maps the symbolic transition to 13 subtasks (Table 2). It is worth noting that our subtask definition is different from hDQN. In hDQN, subtask is associated with an object, but in our work, a subtask is defined as a symbolic transition with initiation set and termination condition mapped from a pair of states whose properties are satisfied by a set of logical literals. Our approach is more descriptive and interpretable, and also makes sub-policy for each subtask to be more easily learned and subtasks more easily sequenced.

Experimental Result. While data-efficiency is easy to demonstrate quantitatively, interpretability is a qualitative metric. We show that the planning and learning process on subtasks and their sequencing can be easily understood from the figures, providing insights and transparency on how learning optimal behavior progresses under the hood.

We collect data from 10 runs and the shadow in the Fig. 2c, 2d, 2e is the variance among these runs. The description of subtasks can refer to both Fig.2f and Table2. Only achieving Subtask 3 (picking up the key) and Subtask 7 (opening the right door) can receive the external reward of $+100$ and $+300$ respectively, while other subtasks will receive the reward of 0 from the environment. The learning curve of SDRL (Fig. 2c) shows that the agent first discovered the plan of collecting key after 0.5M samples by sequencing subtasks 1–3. Intrinsically motivated planning encourages exploring untried subtasks, and by learning more subtasks to move to other locations, the agent finally converges to the maximal cumulative external reward of 400 around 1.5M samples by sequencing subtasks 1–7 (Fig. 2f). By comparison, hDQN cannot reliably achieve the score of 400 around 2.5M samples. The variance of SDRL is smaller than that of hDQN, partially due to the fact that our definition of subtask is easier to learn than the one defined in hDQN, leading to more robust and stable learning.

During the experiment, Subtasks 1–10 are successfully learned by DQNs, with 7 of them being selected in the final solution with achieving success ratio of 100%, shown in Fig. 2d. It should be noted that the order of learning subtasks does not depend on the order they appear in the final optimal plan. For instance, Subtask 6 was learned early but appears later in the final optimal plan. This suggests that the subgoals proposed for learning by symbolic planner is activated only when the starting state is satisfied, and once learned, can be easily sequenced and reused in other plans. Subtasks 8–13 are pruned during learning process due to bad extrinsic rewards derived from training performance. Subtask 8, from the lower right ladder to the left of the rotating skull reaches success ratio of 0.9 but later quickly drops back to 0, due to the instability of DQN. Subtasks 9 and 10 do not contribute to the optimal plan and are therefore dropped by the planner as well. Subtasks 11 – 13 are shown to be too difficult to learn in our experiments and discarded by the planner due to poor extrinsic rewards.

Conclusions

In this paper we propose SDRL framework that uses explicitly represented symbolic knowledge to perform high-level symbolic planning based on intrinsic goal and utilizes DRL to learn low-level control policy, leading to improved task-level interpretability for DRL and data-efficiency, which are validated by evaluation on benchmark problems.

Acknowledgments

We thank the donation of GPU card from NVIDIA Corporation.

References

Barto, A., and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Systems Journal* 13:41–77.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

Chen, K.; Yang, F.; and Chen, X. 2016. Planning with task-oriented knowledge acquisition for a service robot. In *IJCAI*, 812–818.

Chentanez, N.; Barto, A. G.; and Singh, S. P. 2005. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, 1281–1288.

Cimatti, A.; Pistore, M.; and Traverso, P. 2008. Automated planning. In van Harmelen, F.; Lifschitz, V.; and Porter, B., eds., *Handbook of Knowledge Representation*. Elsevier.

Doshi-Velez, F., and Kim, B. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2012. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence* 187-188:52–89.

Gelfond, M., and Lifschitz, V. 1998. Action languages. *Electronic Transactions on Artificial Intelligence (ETAI)*.

Gilpin, L. H.; Bau, D.; Yuan, B. Z.; Bajwa, A.; Specter, M.; and Kagal, L. 2018. Explaining explanations: An approach to evaluating interpretability of machine learning. *arXiv preprint arXiv:1806.00069*.

Hanheide, M.; Göbelbecker, M.; Horn, G. S.; et al. 2015. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*.

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

Hogg, C.; Kuter, U.; and Munoz-Avila, H. 2010. Learning methods to generate good plans: Integrating htn learning and reinforcement learning. In *AAAI*.

Keramati, R.; Whang, J.; Cho, P.; and Brunskill, E. 2018. Strategic object oriented reinforcement learning. *arXiv preprint arXiv:1806.00175*.

Khandelwal, P.; Zhang, S.; Sinapov, J.; Leonetti, M.; Thomason, J.; Yang, F.; Gori, I.; Svetlik, M.; Khante, P.; Lifschitz, V.; and Stone, P. 2017. Bwibots: A platform for bridging the gap between ai and human–robot interaction research. *The International Journal of Robotics Research* 36(5-7):635–659.

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 3675–3683.

Le, H. M.; Jiang, N.; Agarwal, A.; Dudík, M.; Yue, Y.; and Daumé III, H. 2018. Hierarchical imitation and reinforcement learning. *arXiv preprint arXiv:1803.00590*.

Lee, J.; Lifschitz, V.; and Yang, F. 2013. Action Language $\mathcal{BC}$: A Preliminary Report. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Lehnert, L.; Tellex, S.; and Littman, M. L. 2017. Advantages and limitations of using successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1708.00102*.

Leonetti, M.; Iocchi, L.; and Stone, P. 2016. A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artificial Intelligence* 241:103–130.

Lifschitz, V. 2008. What is answer set programming? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1594–1597. MIT Press.

Lu, K.; Zhang, S.; Stone, P.; and Chen, X. 2018. Robot represention and reasoning with knowledge from reinforcement learning. *arXiv preprint arXiv:1809.11074*.

Machado, M. C.; Bellemare, M. G.; and Bowling, M. 2017. A laplacian framework for option discovery in reinforcement learning. *arXiv preprint arXiv:1703.00956*.

Machado, M. C.; Rosenbaum, C.; Guo, X.; Liu, M.; Tesauro, G.; and Campbell, M. 2017. Eigenoption discovery through the deep successor representation. *arXiv preprint arXiv:1710.11089*.

Mahadevan, S. 1996. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning* 22:159–195.

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

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Oudeyer, P.-Y., and Kaplan, F. 2009. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics* 1:6.

Parr, R., and Russell, S. J. 1998. Reinforcement learning with hierarchies of machines. In *Advances in neural information processing systems*, 1043–1049.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144. ACM.

Ryan, M. R. K. 2002. Using abstract models of behaviours to automatically generate reinforcement learning hierarchies. In *In Proceedings of The 19th International Conference on Machine Learning*, 522–529. Morgan Kaufmann.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schwartz, A. 1993. A reinforcement learning method for maximizing undiscounted rewards. In *Proc. 10th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA.

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

Tsividis, P. A.; Pouncy, T.; Xu, J. L.; Tenenbaum, J. B.; and Gershman, S. J. 2017. Human learning in atari.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, 2094–2100.

Verma, A.; Murali, V.; Singh, R.; Kohli, P.; and Chaudhuri, S. 2018. Programmatically interpretable reinforcement learning. *arXiv preprint arXiv:1804.02477*.

Yang, F.; Lyu, D.; Liu, B.; and Gustafson, S. 2018. Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. In *International Joint Conference of Artificial Intelligence (IJCAI)*.