Neuro-Symbolic Artificial Intelligence: The State of the Art

Pascal Hitzler [a] and Md Kamruzzaman Sarker [b]
[a] *Kansas State University*
[b] *University of Hartford*

Contents

Chapter 1

Explainable Neuro-Symbolic Hierarchical Reinforcement Learning

Daoming Lyu, Auburn University

Fangkai Yang, NVIDIA

Hugh Kwon, Auburn University

Bo Liu[1], Auburn University

Wen Dong, SUNY Buffalo

Levent Yilmaz, Auburn University

Abstract

Human-robot interactive decision-making is increasingly becoming ubiquitous, and explainability is an influential factor in determining the reliance on autonomy. However, it is not reasonable to trust systems beyond our comprehension, and typical machine learning and data-driven decision-making are black-box paradigms that impede explainability. Therefore, it is critical to establish computational efficient decision-making mechanisms enhanced by explainability-aware strategies. To this end, we propose the Trustworthy Decision-Making (TDM[2]), which is an explainable neuro-symbolic approach by integrating symbolic planning into hierarchical reinforcement learning. The framework of TDM enables the subtask-level explainability from the causal relational and understandable subtasks. Besides, TDM also demonstrates the advantage of the integration between symbolic planning and reinforcement learning, reaping the benefits of both worlds. Experimental results validate the effectiveness of proposed method while improving the explainability in the process of decision-making.

Key Words: Symbolic Planning, Sequential Decision-making, Interactive Machine Learning, Explainable Machine Learning.

[1]Corresponding author: Bo Liu <boliu@auburn.edu>.

[2]This chapter is primarily adapted from [1].

1.1. Introduction

From self-driving cars to voice assistants on the phone, artificial intelligence (AI) is progressing rapidly. Though AI systems are undeniably powerful and continue to expand their role in our daily lives, emerging issues, such as non-transparency and potential risk in decision-making, give rise to a vital concern: ***why should the end-users trust the decision support system?*** Reliance on technology and autonomy is strongly influenced by trust [2]. However, a recent study reveals that a mere belief in interacting with autonomous teammates led to diminished performance and passive behavior among human participants even though they were collaborating with a remote human partner [3], indicating that we do not trust the capabilities of AI systems as much as we trust humans.

Building explainable AI systems [4, 5, 6, 7] , therefore, becomes an important issue for humans to reap the full spectrum of societal and industrial benefits from AI, as well as for minimizing the undesired consequences in the use of the system [8]. Indeed, in modern AI systems, there are many interactive relations, primarily in the form of *interactions* between humans, between humans and the smart agents (human-computer interaction systems), between agents and the environment (interactive machine learning), and among smart agents (multi-agent systems). Besides, AI systems, in general, need to comply with the norms of the social and cultural context if they are expected to operate side by side with humans. Therefore, trust is particularly important for normative agents that are expected to exhibit behavior consistent with the norms of a society or group. From the humans' perspective, we trust things that behave as we expect them to, indicating that trust derives from the process of minimizing the perceived risk [9]. Perceived risk is often defined in terms of uncertainty about the possibility of failure or the likelihood of exhibiting improper behavior.

To open up the AI black-box and facilitate trust, it is critical to enable the AI system to be reliable, transparent, and explainable so that the system can achieve reproducible results, justify the decisions it makes, and understand what is important in the decision-making process. This combination of features can be called "explainability". Deep learning algorithms have been successfully applied to sequential decision-making problems involving high-dimensional sensory inputs such as Atari games [10]. However, deep learning algorithms have limited explainability and transparency, which gives rise to doubt by the human end-users due to a lack of trust and confidence in the AI systems. Prior research efforts [11] on explainability often focus on explaining the results of learning when the underlying problem and architecture are complex. Symbolic planning [12, 13, 14, 15, 16] has been used to break down the complex task into simpler explainable subtasks. However, existing studies [17, 18] focus on the performance gain of symbolic planning when it is applied to general sequential decision-making rather than explainability.

Motivated by understanding the task-level behavior of the algorithm and elevating the trust between human end-users and the smart agents, we propose an explainable neuro-symbolic hierarchical reinforcement learning approach, which we call it Trustworthy Decision-Making (TDM). Symbolic planning is therefore utilized to perform causal reasoning and planning on explicitly represented knowledge, allowing humans to predict the future and to understand the causes of events. This ultimately results in task-level explainability. The key insight is that the success is better measured when the tasks are explainable. Achieving a reliable success rate of the tasks is equivalent to minimizing perceived risks of the agent's behavior, thereby enhancing its trustworthiness.

The rest of the paper is structured as follows. Section 1.2 introduces the background on symbolic planning along with an overview of the interactive sequential decision problem. Section 1.3 provides a detailed description of the TDM framework. We introduce a learning algorithm facilitated by our formal framework in Section 1.4. The optimality analysis for the converged plan generated by the algorithm is presented to substantiate the utility of our strategy. The experimental results of Section 1.5 validate the explainability of the high-level behavior of the system while maintaining improved data efficiency. Section 1.6 concludes by summarizing our results and delineating potential avenues of future research.

1.2. Preliminaries

In this section, we introduce the sequential decision making, hierarchical reinforcement learning, and symbolic planning.

Interactive Sequential Decision-Making. In sequential decision-making problems, an agent takes a sequence of actions to determine its utility. However, the utility is often difficult to model in practical settings when the environment is complex or changes dynamically. In such cases, the utility is determined through feedback from interaction. This can be modeled as a Markov Decision Process (MDP) [3], which is defined by a tuple $(\mathscr{S}, \mathscr{A}, P^a_{ss'}, r, \gamma)$. Specifically, $\mathscr{S}$ is the set of states and $\mathscr{A}$ is the set of actions. Then $P^a_{ss'}$ is the transition kernel that, given a state $s \in \mathscr{S}$ and an action $a \in \mathscr{A}$, defines the probability the next state will be $s' \in \mathscr{S}$. Moreover, $r(s,a) : \mathscr{S} \times \mathscr{A} \mapsto \mathbb{R}$ 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 $\pi : \mathscr{S} \mapsto \mathscr{A}$ that maps a state to an action.

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 un-discounted sum of rewards for finite horizon problems. In this paper we adopt the latter metric defined as $J^\pi_{\text{avg}}(s) = \mathbb{E}[\sum_{t=0}^{T} r_t | s_0 = s]$. We define the *gain reward* $\rho^\pi(s)$ reaped by policy π from s as $\rho^\pi(s) = \lim_{T\to\infty} \frac{J^\pi_{\text{avg}}(s)}{T} = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}[\sum_{t=0}^{T} r_t]$. Gain reward $\rho^\pi(s)$ is often used to measure the lower bound of the expected cumulative rewards of a task w.r.t a given policy. For example, $\rho^\pi(s)$ is instrumental in deciding whether a specific task is rewarding enough to make the "stay-or-leave" decision [20]. More mathematical details of gain reward can be referred to [21, 22, 23, 24].

Hierarchical Reinforcement Learning and Options. Hierarchical reinforcement learning (HRL) [25] overcomes the "curse of dimensionality" via hierarchical decomposition within the policy space of the problem. Options [26] are a well-known HRL framework which is the temporally extended course of actions. An *option* consists of three components: a policy $\pi : \mathscr{S} \times \mathscr{A} \mapsto [0,1]$, a termination condition $\beta : \mathscr{S} \mapsto [0,1]$, and an initiation set $\mathscr{I} \subseteq \mathscr{S}$. 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

[3] We follow the style of notation in the 1st edition of reinforcement learning book by Sutton and Barto [19].

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.

Action Language $\mathscr{BC}$. An *action description D* in the language $\mathscr{BC}$ [27] includes two kinds of symbols on signature σ, *fluent constants* that represent the properties of the world, and *action constants* , representing actions that influences the world. A *fluent atom* associates a fluent constant f to a value v and is expressed as $f = v$. In particular, we consider a boolean domain for f. *Causal laws* can now be defined on the fluent atoms and action constants, describing the relationship among fluent atoms and the effects of actions on the value of fluent atoms. A *static law* may state that a fluent atom A is true at a given time step when $A_1,\ldots,A_m$ are true (A **if** $A_1,\ldots,A_m$) or the value of f equals v by default (**default** $f = v$). On the other hand, a *dynamic law* describes an action a (**nonexecutable** a **if** $A_1,\ldots,A_m$) or the effect of a on the fluent atom A (a **causes** A **if** $A_1,\ldots,A_m$). An inertia is a special case of dynamic law that states the value of fluent constant f does not change with time (**inertial** f). An action description is a finite set of causal laws, capturing the domain dynamics as transitions.

Symbolic Planning. Given the action description D, symbolic planning (SP) is performed with action language $\mathscr{BC}$. With a slight abuse of notations, let $\langle s,a,s'\rangle$ denote a transition from a symbolic state s to a symbolic state s' by a set of action a. Then a planning problem can be formulated as a tuple (I,G,D). The planning problem has a plan of length $l-1$ if and only if there exists a transition path of length l such that $I = s_1$ and $G = s_l$. In the rest of the paper, Π denotes both the plan and the transition path by following that plan. Given the above semantics represented in $\mathscr{BC}$, automated planning can be achieved by an answer set solver such as CLINGO [28], and provide the solution to the planning problem.

SP has been used in tasks that must be highly explainable to human users, such as interacting and cooperating with mobile robots [12, 13, 14, 15, 16]. By abstracting the planning problem into a symbolic representation, including the knowledge of objects, properties, and the effects of executing actions in a dynamic system, SP can solve the problem based on logical reasoning. Such symbolic representation is typically implemented via a formal, logic-based language as the Planning Domain Definition Language (PDDL) [29] or an action language [30] that relates to logic programming under answer set semantics (answer set programming) [31].

SP algorithms are white-box algorithms. With the symbolic knowledge designed to be human-readable, the behavior of an SP agent that plans and reasons based on causal relations entailed by actions and events, which is naturally predictable and explainable. Prior works combine symbolic planning with other sequential-decision making algorithms to bring stability to plan-based learning [17, 18]. However, these works only highlight the performance improvement from the prior domain knowledge represented by SP.

1.3. The TDM Framework

This section will introduce the main framework of Trustworthy Decision-Making (TDM), which is a Neuro-Symbolic HRL approach. The underlying sequential decision-making problem is first defined as an MDP tuple $(\widetilde{\mathscr{S}},\widetilde{\mathscr{A}},\widetilde{P^a_{ss'}},r,\tilde{\gamma})$, where $\widetilde{\mathscr{S}}$ refers to a set of states in the problem (e.g., pixel images in Atari games), $\widetilde{\mathscr{A}}$ is the set of primitive actions, $\widetilde{P^a_{ss'}}$ is the transition kernel, r is the reward function, and $\tilde{\gamma}$ is a discounting factor. To make the notations clear, in the following paper, $\widetilde{\mathscr{S}},\widetilde{\mathscr{A}}$ are used to denote the MDP

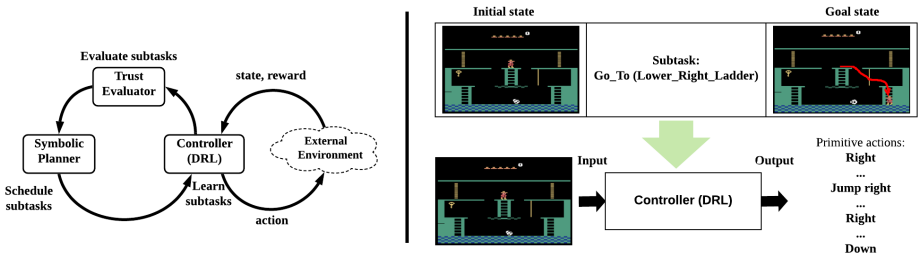


Figure 1. The left subfigure (A) shows the architecture overview of TDM, while the right subfigure (B) gives an example of subtask learning on Montezuma's Revenge. The subtask of "Go_To (Lower_Right_Ladder)" means the avatar in the middle platform needs to arrive at the position of lower right ladder. In this case, the controller accepts the pixel images as an input, and learns a sub-policy for this subtask.

state space and action space, while $\mathscr{S}, \mathscr{A}$ represent the symbolic state space and action space. The goal of TDM is to seek a reasonable sequence of subtasks and learn the corresponding sub-policies such that performing the learned subtasks in order can lead to the maximal cumulative reward. It should be noted that our framework is not restricted to high-dimensional sensory inputs. However, we choose to use them as an example since learning from such inputs is more challenging and relevant to real-world applications.

A symbolic representation of the problem provided by human experts is assumed to be available, including a set of causal rules that captures objects, fluents and how their values are changed by executing subtasks. Although this may need some effort, it is possible to develop general-purpose action modules [32, 33, 34] as it has been shown that dynamic domains share many actions in common. Therefore, the symbolic formulation for one problem can be adapted to another with a little changes by instantiating a different set of objects or adding a few more rules. To make TDM general, the domain formulation in our framework is always in a coarse granular and high-level abstraction. This also enables the decision-making process to be robust and flexible when facing uncertainty and domain changes.

The overview of TDM framework is shown in Fig. 1(A). With a symbolic representation, the symbolic planner performs planning by a sequence of symbolic actions based on the evaluation feedback from the trust evaluator. These symbolic actions are actually the subtasks that can be sequenced as a high-level plan. In addition, a mapping function is implemented to perform symbol grounding in our experiments, associating each MDP state (e.g., pixel images in Atari games) with a symbolic state. In different domains, such function is assumed to be available. As a result, the subtasks defined on the MDP space can be induced based on symbolic states. After symbol grounding, the subtasks in the plan will be sent to the controller to be executed so that each sub-policy for the corresponding subtask is learned by the controller. Since it is possible that the controller cannot successfully learn the sub-policies, a metric is introduced to evaluate the competence of learned sub-policies, such as the success ratio over several episodes. After all subtasks in the plan have been executed, the trust evaluator computes the plan quality score by utilizing the *"degree of trustworthiness"* for subtasks. The evaluation for subtasks is returned to the symbolic planner and is used to generate new plans by either exploring new subtasks or sequencing learned subtasks that supposedly can achieve a higher plan quality score in the next iteration.

1.3.1. A Computational Framework for Plan Quality Evaluation

We formulate the plan evaluation with a success ratio of task execution as the plan quality score. This is used to quantitatively measure both the performance level of a plan, and the reliability of learning for each subtask in this plan. Therefore, the plan with a low plan quality score will have less ability to be considered for the final solution. Besides, the insight here is not only just explain how and why the agent's behavior can produce outcomes but also maintain high performance level on achieving the task. So the plan evaluation can be tied to the trustworthiness, including both the explainability for the plan and the reliability for each subtask learning. In essence, we try to solve an optimization problem to maximize the explicit plan quality score subject to the constraint of implicit explainability:

$$\max_{\Pi} \frac{1}{|\Pi|} \sum_{o \in \Pi} \varepsilon(o),$$

where ε is the success ratio of a subtask o, Π is a symbolic plan.

The success ratio with respect to explainability is the key to our plan evaluation mechanism. But explainability is a qualitative measure, and it is difficult to formulate a traditional optimization approach to satisfy this constraint. Instead, we use symbolic representation to naturally induce explainability in our problem. We discuss the details of our approach in the subsequent sections.

1.3.2. Explainability Enhancement via Symbolic Representation

Let us consider a planning problem (I,G,D). Although a symbolic formulation is possible with various planning or action languages, we will use $\mathscr{BC}$ to represent D as a demonstration. Specifically, we add the following causal laws to D to formulate gain rewards of executing actions and their effects on cumulative plan quality:

- For any symbolic state that contains atoms $\{A_1,\ldots,A_n\}$, D contains static laws of the form:

$$s \textbf{ if } A_1,\ldots,A_n, \text{ for state } s \in \mathscr{S}.$$

- 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 that the gain reward is initialized optimistically by default to promote exploration and is denoted as *INF*:

$$\textbf{default } \rho(s,a) = INF, \text{ for } s \in \mathscr{S}, a \in \sigma_A(D).$$

- We use fluent symbol *quality* to denote the total *"degree of trustworthiness"* of a plan, termed as *plan quality* or *plan quality score*. D contains dynamic laws of the form

$$a \textbf{ causes } quality = C + Z \textbf{ if } s, \rho(s,a) = Z, quality = C.$$

- D contains a set P of facts of the form $\rho(s,a) = z$.

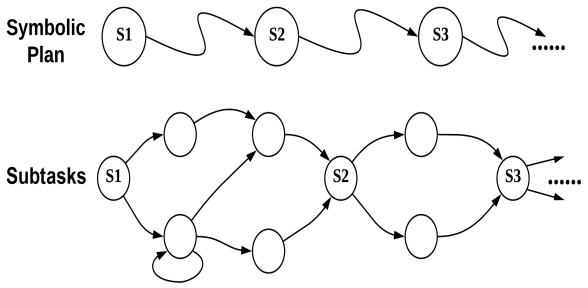


Figure 2. The mapping from a symbolic transition path to subtasks

In our case, I is still the initial symbolic planning state, but the goal state G is also a linear constraint of the form

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

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

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

It should be noted that (1) in TDM doesn't have the logical constraint part and enables "model-based exploration by planning", which is more suitable for the problems where the plan quality score drives the agent's behavior.

From the causal point of view, symbolic planning centers on reasoning about changes and causal relations entailed by actions and events. This enables to predict the outcomes of the agent's behavior with causality, and further help humans to understand and justify the outcomes.

1.3.3. Implicit Explainability Constraint Satisfaction: From Symbolic Transitions to Subtasks

Our discussion of plan evaluation and explainability enhancement has been restricted to general machine learning. Symbolic planning can be used in many problem domains, e.g., image classification, and success ratio may be replaced with a suitable metric, such as accuracy, in the respective domain. Here, we narrow our focus to a sequential decision-making problem, which will allow us to devise a specific algorithm based on the TDM framework.

We assume an oracle exists to determine whether the symbolic properties specified as fluent atoms of the form $f = v$ in s are true in $\tilde{s}$. This oracle is actually a mapping function as $\mathbb{F} : \mathscr{S} \times \widetilde{\mathscr{S}} \mapsto \{\texttt{True}, \texttt{False}\}$. If a symbolic state $s \in \mathscr{S}$ corresponds to an MDP state $\tilde{s} \in \tilde{\mathscr{S}}$, then $\mathbb{F} = \texttt{True}$. $\mathbb{F} = \texttt{False}$ if otherwise. For example, this oracle can be a pre-trained perception module for object recognition in images when it is used for Atari games.

Given $\mathbb{F}$ and a pair of symbolic states $s, s' \in \mathscr{S}$, we can induce a subtask as a triple (I, π, β) where the initiation set $I = \{\tilde{s} \in \widetilde{\mathscr{S}} : \mathbb{F}(s, \tilde{s}) = \texttt{True}\}$, $\pi : \widetilde{\mathscr{S}} \mapsto \widetilde{\mathscr{A}}$ is the intra-subtask policy, and β is the termination condition such that

$$\beta(\tilde{s'}) = \begin{cases} 1, \ \mathbb{F}(s', \tilde{s'}) = \texttt{True}, \text{ for } \tilde{s'} \in \widetilde{\mathscr{S}} \\ 0, \ \text{otherwise} \end{cases}$$

This definition maps symbolic transition to a subtask (as illustrated in Fig. 2), which is similar to an option and incorporates the hierarchical structure into the decision-making. At the high-level, the symbolic planner generates the symbolic plan that includes a sequence of subtasks, while each subtask will be a accomplish with the primitive actions at the low-level. It also implies that the execution of the subtask will be feasible if the termination condition is 1, while it will be infeasible if the termination condition is 0. Since the symbolic knowledge is provided by the human experts, subtasks are pre-defined by the human experts as well.

1.3.4. Explicit Success Ratio Maximization

To motivate the subtask learning from the low-level control, $t_e(\tilde{s'})$ can be assigned in a binary form. It is defined as:

$$t_e(\tilde{s'}) = \begin{cases} +1, \beta(\tilde{s'}) = 1 \\ -1, \text{otherwise} \end{cases} \tag{3}$$

which means that the $t_e(\tilde{s'})$ will be $+1$ if the subtask terminated at $\tilde{s'}$ can be achieved, while the $t_e(\tilde{s'})$ will be -1 if subtask fails to be accomplished. Other more sophisticated distribution (e.g., Bernoulli distribution) can also be modeled other than this 0-1 distribution model in practice.

To evaluate the high-level planning, we further define $r_e(s,o)$ as $r_e(s,o) = f(\varepsilon)$, where f is a function of ε, and ε is the success ratio that denotes the average rate of completing the subtask successfully over the previous 100 episodes. This metric is used to measure whether the subtask o at symbolic state s is learnable or not. This is different from the definition of $t_e(\tilde{s'})$ since $r_e(s,o)$ requires reliably achieving the subtask after the training by episodes and keeping the success ratio above a certain threshold. Therefore, f is defined as

$$f(\varepsilon) = \begin{cases} -\psi, & \varepsilon < \mathrm{E} \\ r(s,o), & \varepsilon \geq \mathrm{E} \end{cases} \tag{4}$$

ψ is a large positive numerical value, $r(s,o)$ is the reward obtained from the MDP environment by following the subtask o hyper-parameter E is a predefined threshold of success ratio. In our experiments, we set $\mathrm{E} = 0.9$ according to empirical performance observations. Intuitively, it means if the sub-policy can reliably achieve the subtask after the training by episodes. The *"degree of trustworthiness"* of $r_e(s,o)$ at s' reflects true cumulative reward from the MDP environment by following the subtask; otherwise, the *"degree of trustworthiness"* will be very low (as a large negative number), 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 $\sum_{\langle s,a,s' \rangle} r_e(s,o)$ is maximal among all plans.

1.4. Algorithm Design

We present a learning algorithm for the TDM framework in Algorithm 1 and show TDM's planning and learning process. Each episode starts with the symbolic planner generating a symbolic plan Π_t for the problem (I,G,D) (Line 4). The symbolic transitions of Π_t are mapped to subtasks (Line 10) to be learned by a controller (Line 11). The controller performs deep Q-learning with t_e using experience replay (Lines 12–15) to estimate the Q value $Q(\tilde{s},\tilde{a};o) \approx Q(\tilde{s},\tilde{a};\theta,o)$, where θ is the parameter of the non-linear function approximator. The observed transition $(\tilde{s}_t,\tilde{a}_t,r_e(\tilde{s}_{t+1},g),\tilde{s}_{t+1})$ is stored as an experience in $\mathscr{D}_o$. The loss at i^{th} iteration is calculated as an expectation over the collected experience and is given by

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

When the inner loop terminates, a symbolic transition $\langle s_t,a_t,s_{t+1}\rangle$ for the subtask o_t is completed, and we are ready to update the plan quality score of the subtask (Line 18). Although various learning methods, such as Q-learning, may be used here, we use R-learning [35] to evaluate the plan quality score as an average reward case rather than a discounted reward case. With R-learning, the update rule becomes

$$\begin{aligned} R_{t+1}(s_t,o_t) &\overset{\alpha}{\leftarrow} r_e - \rho_t^{o_t}(s_t) + \max_o R(s_t,o) \\ \rho_{t+1}^{o_t}(s_t) &\overset{\beta}{\leftarrow} r_e + \max_o R_t(s_{t+1},o) - \max_o R_t(s_t,o) \end{aligned} \tag{6}$$

The *quality* of Π_t is measured by (2) (Line 20) and the plan quality score of the plan is updated according to *quality*(Π_t) (Line 21). The symbolic formulation is then updated with the learned ρ values (Line 22). With the symbolic formulation changed, an improved plan may be generated in the next episode. The loop terminates when no such improvement can be made. The algorithm guarantees symbolic level optimality conditioned on R-learning convergence.

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

Proof. When R-learning converges, for any transition $\langle s,a,t\rangle$, the increment terms in (6) diminish to 0, which implies

$$\begin{aligned} R(s,o) &= \max_{o'} R(s,o'), \\ \rho^a(s) &= r_e(s,o) - \max_{o'} R(s,o') + \max_{a'} R(t,o') \end{aligned} \tag{7}$$

Algorithm 1 terminates iff there exists an upper bound of plan quality iff there does not exist a plan with a loop L such that $\sum_{\langle s,a,t\rangle\in L}\rho^a(s) > 0$. By (7), it is equivalent to $\sum_{\langle s,o,t\rangle\in L}(r_e(s,o) - R(s,o) + R(t,o)) \le 0$ iff $\sum_{\langle s,o,t\rangle\in L} r_e(s,o) - R(s_{|L|},o) + R(s_0,o) \le 0$. Since L is a loop, $s_{|L|} = s_0$, so $\sum_{\langle s,a,t\rangle\in L} r_e(s,o) \le 0$ iff any plan Π does not have a positive loop of cumulative reward iff optimal plan exists, which completes the proof.

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

Algorithm 1 TDM Planning and Learning Loop

Input: $(I, G, D, \mathbb{F})$ where $G = (quality > 0)$, and an exploration probability ε
Output: An optimal symbolic plan Π^*

```
P_0 ⇐ ∅, Π ⇐ ∅
while True do
    Π* ⇐ Π
    Take ε probability to solve planning problem and obtain a plan Π ⇐ CLINGO.solve(I, G, D∪P_t)

    if Π = ∅ then
        return Π*
    end if
    for symbolic transition ⟨s, a, s'⟩ ∈ Π do
        Obtain current state s̃
        Correspond to subtask o by using 𝔽 to obtain initiation set and terminate condition
        while β(s̃) ≠ 1 and maximal step is not reached do
            Pick up an action ã and obtain transition (s̃, ã, s̃', t_e(s̃'))
            Store transition in experience replay buffer 𝒟_o
            Estimate Q(s̃, ã; θ, o) by minimizing loss function (5) when there are sufficient samples
            in 𝒟_o
            Update current state s̃ ⇐ s̃'
        end while
        Calculate r_e(s, o) with "degree of trustworthiness"s
        Update R(s, o) and ρ^o(s) using (6).
    end for
    Calculate quality of Π by (2).
    Update planning goal G ⇐ (quality > quality_t(Π)).
    Update facts P_t ⇐ {ρ(s, a) = z : ⟨s, a, s'⟩ ∈ Π, ρ^a_t(s) = z}
end while
```

Proof. By Lee et al. [27, Theorem 2], Π^* is a plan for planning problem (I, G, D). For Π_o returned by Algorithm 1 when it terminates, $quality(\Pi) \leq quality(\Pi^*)$ for any Π iff

$$\sum_{\langle s,a,t\rangle \in \Pi} \rho^a(s) \leq \sum_{\langle s,a,t\rangle \in \Pi_o} \rho^a(s).$$

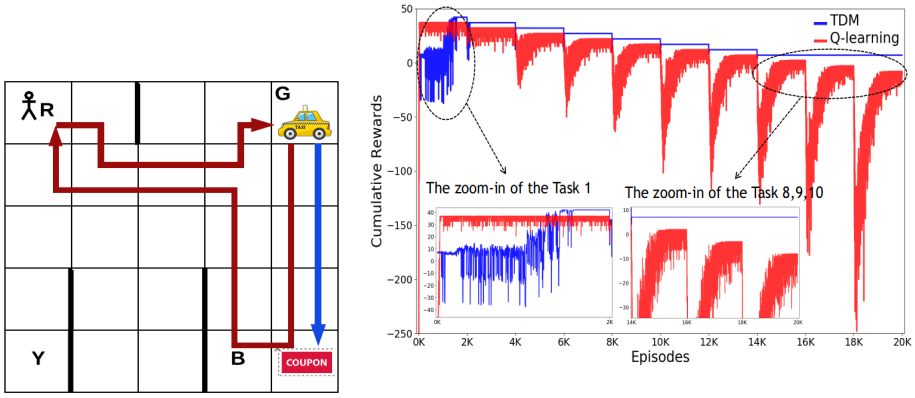
By (7), the inequality is equivalent to

$$\sum_{\langle s,a,t\rangle \in \Pi} r_e(s,o) + R(s_{|\Pi|}, o) \leq \sum_{\langle s,a,t\rangle \in \Pi_o} r_e(s,o) + R(s_{|\Pi^*|}, o).$$

Since $s_{|\Pi|}$ and $s_{|\Pi^*|}$ are terminal states of each symbolic plan with no subtask policies available, we have $\sum_{\langle s,a,t\rangle \in \Pi} r_e(s,o) \leq \sum_{\langle s,a,t\rangle \in \Pi_o} r_e(s,o)$. This completes the proof.

1.5. Experiment

We use the Taxi domain [25] to demonstrate the behavior of learning and planning, Grid World [17] to show the ability to learn unmodeled domain knowledge, and on



(a) Taxi Domain **(b)** Learning Curve

Figure 3. Experimental Results on Taxi Domain

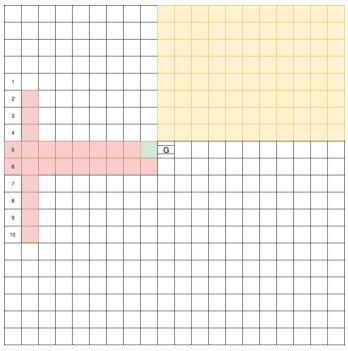
Montezuma's Revenge [10] for explainability and data efficiency. We use 1M to denote 1 million and 1K to denote 1000.

1.5.1. Taxi Domain

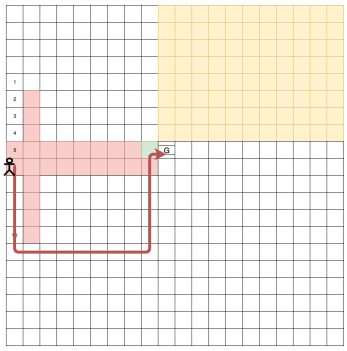
The objective of the Taxi domain is to maximize reward by successfully picking-up and dropping-off a passenger at a specified destination on a grid map (Fig. 3a). A taxi agent moves from a cell to an adjacent cell with -1 reward collected at each time step. The agent receives a reward of 50 for dropping the passenger off at the destination while receiving a reward of -10 when attempting to pick-up or drop-off at an incorrect location. Additionally, there is a one-time reward coupon the taxi can collect at $(4,4)$. With a reward of 10, we will observe the behavior of learning agents when the environment changes.

Setup. For each episode, the taxi starts at $(0,4)$, and we will call every 2000 episodes a task. The experiment consists of 10 tasks where the reward for the successful drop-off decreases in each task by 5, i.e., 50 in Task 1, 45 in Task 2, and so forth. Other reward settings stay constant throughout the tasks. We compare TDM with Q-learning.

Experimental Results. First, let us consider the optimal policies as shown in Fig. 3a. The dark red colored route that collects the coupon, picking up, and then dropping off the passenger is the optimal policy from Task 1 to Task 7. From Task 8 and onward, the reward for the passenger drop-off (≤ 15) is no longer worth the effort, and hence, the optimal policy changes to simply collecting the coupon as indicated by the blue line. As shown in the learning curve in Fig. 3b, averaged over 10 runs, TDM successfully learns the optimal policy for all Tasks. As noted, the optimal policy changes at Task 8, and TDM quickly abandons the sub-tasks that lead to a sub-optimal policy and learns the new optimal policy. All these are in contrast to Q-learning. As shown in the zoom-in of Task 1 in Fig. 3b), Q-learning converges faster than TDM since the model-free exploration of Q-learning is more aggressive than the TDM's exploration guided by the planning. However, the policy learned by Q-learning is worse than that of TDM over all 10 Tasks, which results in a gap of cumulative rewards between them. This gap becomes larger when the optimal policy has changed, especially from Task 8 to Task 10. On the contrary,

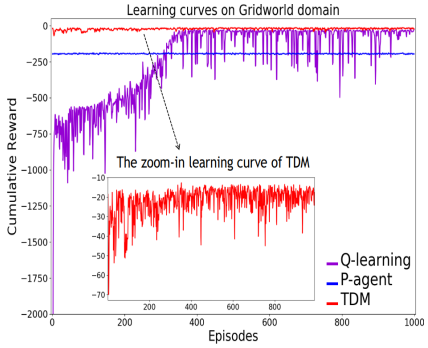


(a) Grid World



(b) A solution

(c) Execution failure of TDM and Planning



(d) Learning Curve

Figure 4. Grid World

the plan evaluation mechanism in TDM enables to adapt to the domain changes very well and select the plan with higher plan quality score. In practice, the reward is often collected from the interaction and may change over time in a dynamic environment. This experiment shows the advantage of TDM in such situations.

1.5.2. Grid World

TDM can learn domain details that are not modeled into symbolic knowledge. We demonstrate this behavior with the Grid World adapted from Leonetti et al. [17], shown in Fig. 4a. An agent must navigate to (9,10), which can only be entered through a door at (9, 9). The agent must grab a doorknob at the door, turn it, and then push the door to reach the goal, all of which may fail and incur a -10 penalty. As in the Taxi domain, every movement has a reward of -1.

Setup. Horizontal and vertical bumpers represent the domain details that the agent needs to learn. The agent receives an additional penalty for a movement into a bumper. Penalties are -30 and -15 for red and yellow bumpers, respectively. The agent starts at random on one of the numbered grids in the first column. We use Q-learning and a standard planning agent (P-agent) as baselines for this domain. P-agent generates plans using CLINGO and executes the plans without any learning capability.

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

Table 1. Neural Network Architecture for Montezuma's Revenge

Experimental Results. Learning curves of cumulative reward are shown in Fig. 4d where the performance of TDM surpasses both that of Q-learning and P-agent. Specifically, TDM can learn domain details that are not included in the symbolic knowledge, such as bumpers in this case. With plan quality scores, TDM can abandon the unsuccessful plans, avoid the bumpers and learn to reliably open and push through the door (e.g., Fig. 4b). This also achieves the optimal behavior. As a comparison, P-agent prefers shortest plans and goes over the bumpers directly, which are costly in this case. Besides, P-agent relies on re-planning upon encountering failures without learning. As such, the number of execution failures to enter the door remains high throughout episodes, as shown in Fig. 4c.

1.5.3. Montezuma's Revenge

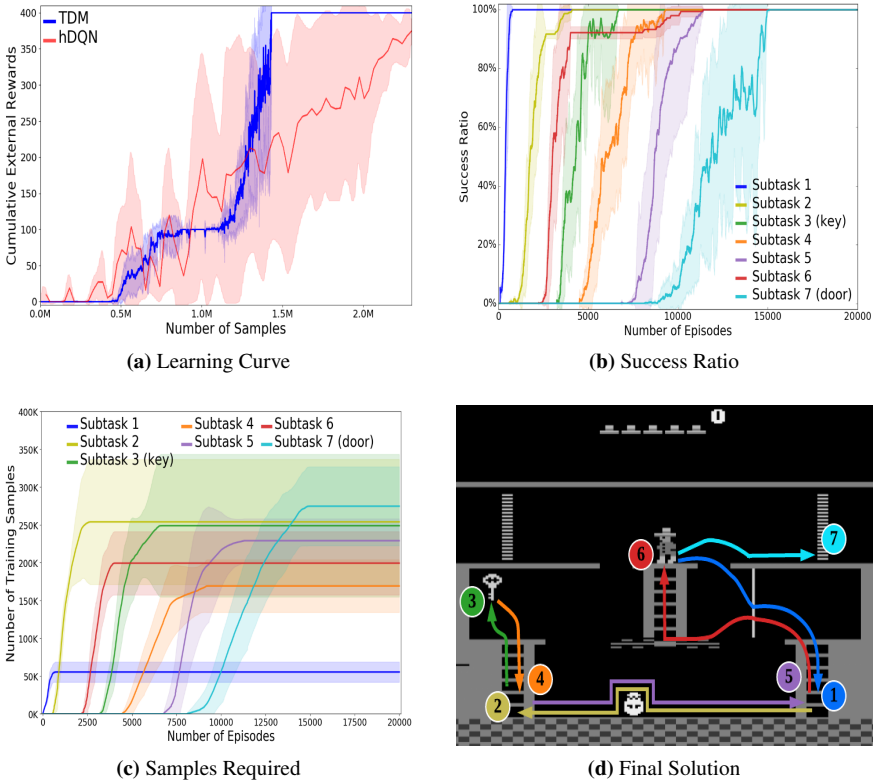
In "Montezuma's Revenge," the player controls a game character through a labyrinth avoiding dangerous enemies and collecting items helpful to the player. We focus on the first room of the labyrinth. Here, the player has to collect a key to unlock the door to the next room. In a typical setting for DRL, an agent receives +100 reward for collecting the key and +300 for opening the door with the key. This is a challenging domain for DRL as it involves a long sequence of high-level tasks and many primitive actions to achieve those tasks. This is a challenging domain where the vanilla DQN often fails to learn [10].

Setup. Our experiment setup follows the DQN controller architecture [36] with double-Q learning [37] and prioritized experience replay [38]. The architecture of the deep neural networks is shown in Table 1. The experiment is conducted using Arcade Learning Environment (ALE) [39]. We have implemented the symbolic mapping function $\mathbb{F}$ based on ALE API. The binary *"degree of trustworthiness"* follows (3) for when a subtask successfully completes or the agent loses its life. The subtask plan quality score follows (4) where $\psi = 100$ and define $r(s,o) = -10$ for $\varepsilon > 0.9$ to encourage shorter plan. We use hierarchical DQN (hDQN) [36] as the baseline.

Symbolic Representation. We represent the transition dynamics and domain knowledge in action language $\mathscr{BC}$. There are 6 pre-defined locations or objects: middle platform (`mp`), right door (`rd`), left of rotating skull (`ls`), lower left ladder (`lll`), lower right ladder (`lrl`), and key (`key`). Note that the number of predefined locations or objects depends on the users and their domain knowledge. We then formulate 13 subtasks based on the symbolic locations. The corresponding symbolic transitions from the mapping function $\mathbb{F}$ are shown in Table 2. The difference between our and hDQN's subtask definitions should be noted. A subtask is only associated with an object in hDQN; However, in our work, it is defined as a symbolic transition with an initiation set and termination condition based on the states satisfying symbolic properties. With this informative symbolic representation, our approach is more general, descriptive, and explainable. It also makes sub-policy for each subtask to be more easily learned and subtasks more easily sequenced.

No.	Subtask	Policy learned	In the 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



(a) Learning Curve **(b)** Success Ratio

(c) Samples Required **(d)** Final Solution

Figure 5. Experimental Results on Montezuma's Revenge

Experimental Results. It is easy to see that TDM is more data-efficient than the baseline hDQN from the learning curve (Fig. 5a). Explainability requires a qualitative analysis. We

will emphasize how plan quality score guides TDM on planning, learning, and sequencing of the subtasks, and therefore learns the optimal behavior, all of which are visualized in our figures.

The results have been averaged over 10 runs and we shown mean-variance plots in the Fig. 5a, 5b, 5c. The description of subtasks can refer to both Fig. 5d and Table 2. The environment rewards are only given for completing Subtask 3 (picking up the key, $+100$) and Subtask 7 (opening the right door, $+300$). Since other subtasks do not receive any reward from the environment, we can infer from Fig. 5a that TDM first learns the plan to sequence Subtasks 1–3. The agent learns other subtasks through exploration as the plan quality score-based planning encourages executing untried subtasks and try different locations until it reaches the door. At that point, we can see in the learning curve that the agent quickly converges to the optimal plan that sequences through Subtasks 1–7 (Fig. 5d), resulting in the maximal reward of 400. In contrast, hDQN does not reliably achieve the maximal reward even after TDM stably converges. Also, it's difficult to see how hDQN sequenced through its subtasks due to high variance. TDM achieves a smaller variance than hDQN partially because our definition of the subtask is easier to learn than the one defined in hDQN, leading to more robust and stable learning.

The symbolic planning of TDM allows flexibility in reusing the learned subtasks in various plans. Fig. 5b shows that TDM learns Subtask 6 before Subtask 3, but in the optimal plan, Subtask 3 must be sequenced before Subtask 6. This means Subtask 6 has been learned as a part of a different plan but re-selected in the optimal plan, which is possible because subgoals of the subtasks are associated by their starting states and only activated by the planner when the agent satisfies the starting states.

During the experiment, Subtasks 1–10 are successfully learned by controllers (or DQNs), with 7 of them being selected in the final solution with achieving a success ratio of 100%, shown in Fig. 5b. TDM prunes other Subtasks 8–13 based on the low plan quality score achieved during training. Subtask 8, from the lower right ladder to the left of the rotating skull, reaches a success ratio of 0.9 but later quickly drops back to 0, due to the instability of DQN. Subtasks 9 and 10 reach the required success ratio but are discarded as they do not contribute to the optimal plan, whereas the success ratios on subtasks 11–13 are poor, suggesting they are difficult to learn.

Explainability and Trust The evaluation criteria are designed through human involvement in our proposed framework, based on the human expert's interpretation of the agent's execution. The symbolic representation can then be refined to improve the agent's performance. Therefore, the agent's decision-making trustworthiness is rooted in the trust we place in the underlying symbolic representation. Although many existing explainable frameworks focus on simplifying the underlying model to generate human-understandable explanations [6, 40, 41], we show that the trade-off between the explainability and performance of the model is not always necessary. In our case, a performance-based plan performance metric is used to guide the learning agent to maximize its performance and justify the subtasks it learns. This is, in essence, a separation of descriptive and persuasive explanation tasks [42], where we handle the descriptive task through symbolic representation and perform persuasive explanation through the use of the plan quality score.

We now relate these concepts to the comparison of our framework to the baseline. In both hDQN and TDM, the bounding box defined by a human expert is utilized, considering the locations of objects only. As a result, the bounding box can be meaningful and

		TDM	hDQN
Explainability	Explainable	Yes	Yes
	Descriptive	Strong	Weak
	Persuasive	Strong	Weak

Table 3. Comparison of TDM and hDQN

explainable with the human expert's domain knowledge. The subtasks of hDQN are only associated with the bounding box. However, TDM utilizes symbolic representation to derive subtasks except for the bounding box and can provide a relational perspective, such as objects/entities and their relations. This makes the subtasks of TDM more descriptive since a symbolic state may contain more rich information than hDQN. According to the plan evaluation on subtasks, plan quality scores would motivate the TDM agent to maximize its performance by executing the learnable subtasks while discarding the unlearnable ones. The experiment results show TDM's persuasive explanation is more convincing than hDQN. Also, the variance about the curves (Fig. 5a) demonstrates the degree of robustness when the agent faces uncertainties. The summarization is shown in Table 3.

1.6. Conclusions

Since explainability of the high-level behavior is critical to enhancing applicability in a complex sequential decision-making problem, we proposed the TDM framework in this paper by integrating symbolic planning with hierarchical reinforcement learning. Specifically, deep RL is used to learn low-level control policies for subtasks managed by high-level symbolic planning based on explicit symbolic knowledge. Both theoretical analysis and empirical studies on benchmark problems validate that our plan quality score-based algorithmic framework brings *task-level explainability* to deep reinforcement learning and *improved data efficiency* induced by the symbolic-planning-learning framework of the agent.

There are several promising future work potentials in this research direction. For example, it would be intriguing to apply TDM in the human-computer interaction systems, allowing layperson to understand the system behavior and provide meaningful evaluation feedback to the machine. The second interesting direction is to apply the framework of TDM in multi-agent systems. Trust is important in interactions among different agents where the integrity of some agents is not always guaranteed. Explainability may be used to help improve the behavior of the agents and lead to a better equilibrium.

References

[1] Lyu D, Yang F, Kwon H, et al. TDM: Trustworthy decision-making via interpretability enhancement. IEEE Transactions on Emerging Topics in Computational Intelligence. 2021;.

[2] Lee JD, See KA. Trust in automation: Designing for appropriate reliance. Human factors. 2004;46(1):50–80.

[3] Demir M, McNeese NJ, Cooke NJ. The impact of perceived autonomous agents on dynamic team behaviors. IEEE Transactions on Emerging Topics in Computational Intelligence. 2018;2(4):258–267.

[4] Marsh S. Trust in distributed artificial intelligence. In: European Workshop on Modelling Autonomous Agents in a Multi-Agent World; Springer; 1992. p. 94–112.

[5] O'Donovan J, Smyth B. Trust in recommender systems. In: Proceedings of the 10th international conference on Intelligent user interfaces; ACM; 2005. p. 167–174.

[6] Ribeiro MT, Singh S, Guestrin C. 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; ACM; 2016. p. 1135–1144.

[7] Ong YS, Gupta A. Air 5: Five pillars of artificial intelligence research. IEEE Transactions on Emerging Topics in Computational Intelligence. 2019;3(5):411–415.

[8] Hoff KA, Bashir M. Trust in automation: Integrating empirical evidence on factors that influence trust. Human Factors. 2015;57(3):407–434.

[9] Rousseau DM, Sitkin SB, Burt RS, et al. Not so different after all: A cross-discipline view of trust. Academy of management review. 1998;23(3):393–404.

[10] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature. 2015;518(7540):529–533.

[11] Biran O, McKeown KR. Human-centric justification of machine learning predictions. In: IJCAI; 2017. p. 1461–1467.

[12] Hanheide M, Göbelbecker M, Horn GS, et al. Robot task planning and explanation in open and uncertain worlds. Artificial Intelligence. 2015;.

[13] Chen K, Yang F, Chen X. Planning with task-oriented knowledge acquisition for a service robot. In: International Joint Conference on Artificial Intelligence; 2016. p. 812–818.

[14] Khandelwal P, Zhang S, Sinapov J, et al. Bwibots: A platform for bridging the gap between ai and human–robot interaction research. The International Journal of Robotics Research. 2017;36(5-7):635–659.

[15] Jeong IB, Ko WR, Park GM, et al. Task intelligence of robots: Neural model-based mechanism of thought and online motion planning. IEEE Transactions on Emerging Topics in Computational Intelligence. 2016; 1(1):41–50.

[16] Lyu D, Yang F, Liu B, et al. SDRL: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In: Proceedings of the AAAI Conference on Artificial Intelligence; Vol. 33; 2019. p. 2970–2977.

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

[18] Yang F, Lyu D, Liu B, et al. PEORL: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. In: International Joint Conference on Artificial Intelligence; 2018.

[19] Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT press Cambridge; 1998.

[20] Shuvaev S, Starosta S, Kvitsiani D, et al. R-learning in actor-critic model offers a biologically relevant mechanism for sequential decision-making. NeurIPS. 2020;33.

[21] Puterman ML. Markov Decision Processes. New York, USA: Wiley Interscience; 1994.

[22] Mahadevan S. Average reward reinforcement learning: Foundations, algorithms, and empirical results. Machine learning. 1996;22(1):159–195.

[23] Tadepalli P, Ok D, et al. H-learning: A reinforcement learning method to optimize undiscounted average reward. Oregon State University; 1994. 94–30–1.

[24] Mahadevan S, Liu B. Basis construction from power series expansions of value functions. Advances in Neural Information Processing Systems (NIPS). 2010;23:1540–1548.

[25] Barto A, Mahadevan S. Recent advances in hierarchical reinforcement learning. Discrete Event Systems Journal. 2003;13:41–77.

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

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

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

[29] McDermott D, Ghallab M, Howe A, et al. PDDL-the planning domain definition language; 1998.

[30] Gelfond M, Lifschitz V. Action languages. ETAI. 1998;.

[31] Lifschitz V. What is answer set programming? In: AAAI. MIT Press; 2008. p. 1594–1597.

[32] Erdoğan ST, Lifschitz V. Actions as special cases. In: KR; 2006. p. 377–387.

[33] Erdoğan ST. A library of general-purpose action descriptions [dissertation]. University of Texas at Austin;

2008.
[34] Inclezan D, Gelfond M. Modular action language ALM. Theory and Practice of Logic Programming. 2016;16(2):189–235.
[35] Mahadevan S. Average reward reinforcement learning: Foundations, algorithms, and empirical results. Machine Learning. 1996;22:159–195.
[36] Kulkarni TD, Narasimhan K, Saeedi A, et al. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In: Advancement on Neural Information Processing Systems; 2016. p. 3675–3683.
[37] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning. In: Association for the Advancement of Artificial Intelligence; Vol. 16; 2016. p. 2094–2100.
[38] Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay. arXiv preprint arXiv:151105952. 2015;.
[39] Bellemare MG, Naddaf Y, Veness J, et al. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research. 2013;47:253–279.
[40] Lipton ZC. The mythos of model interpretability. arXiv preprint arXiv:160603490. 2016;.
[41] Wood DA. A transparent open-box learning network provides insight to complex systems and a performance benchmark for more-opaque machine learning algorithms. Advances in Geo-Energy Research. 2018;2(2):148–162.
[42] Herman B. The promise and peril of human evaluation for model interpretability. arXiv preprint arXiv:171107414. 2017;.