Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning
KR meets Machine Learning and Explanation

Symbolic Knowledge Transfer for Sample-Efficient Deep Reinforcement Learning

Celeste Veronese , Alessandro Farinelli , Daniele Meli
University of Verona, Italy
{celeste.veronese, alessandro.farinelli, daniele.meli}@univr.it

Abstract

Reinforcement Learning (RL) provides a principled framework for sequential decision-making in complex environments. However, state-of-the-art Deep Reinforcement Learning (DRL) algorithms typically require large amounts of training data and often fail to generalize beyond small-scale training scenarios, even on standard benchmarks. We propose a neuro-symbolic DRL approach that incorporates background symbolic knowledge to improve both sample efficiency and generalization to more challenging, unseen tasks. Specifically, partial policies learned in simple domain instances, where high performance can be achieved reliably, are transferred as structured priors to accelerate learning in more complex environments, eliminating the need to tune DRL parameters from scratch. Our method represents partial policies as axioms in the Answer Set Programming (ASP) formalism and performs online reasoning to guide training through two complementary mechanisms: (i) biasing the action distribution during exploration, and (ii) rescaling Q-values during exploitation. The integration of ASP reasoning with DRL enhances interpretability and trustworthiness while accelerating convergence, particularly in sparse-reward settings and tasks with long planning horizons, without introducing significant computational overhead. We empirically evaluate our approach on challenging variants of gridworld environments under both fully and partially observable settings. Results demonstrate consistent performance improvements over a state-of-the-art reward machine baseline.

1 Introduction

Deep Reinforcement Learning (DRL) can be successfully applied to solve sequential decision-making problems, offering invaluable benefits for many real-world domains of application, e.g., robotic tasks (Ibarz et al. 2021) and sustainability (Zuccotto et al. 2024), involving complex dynamics, multiple performance objectives, and large observation spaces. However, DRL algorithms still present drawbacks that limit their wide adoption to real systems. Firstly, DRL policies are black-box, which makes them hardly interpretable to humans, affecting trustworthiness and social acceptance (Vouros 2022). Moreover, one big challenge of DRL lies in *sample inefficiency* (Dulac-Arnold et al. 2021), as the agent needs to collect numerous experiences from the environment to build an accurate model and achieve optimality. Sample inefficiency is particularly problematic when scaling and generalizing DRL to environments with longer planning horizons, more sub-goals, and sparse rewards (Dulac-Arnold et al. 2021) (e.g., larger grids with more objects in gridworlds). Although several approaches have been proposed to mitigate this issue, they either require access to a large variety of historical data (Bertran et al. 2020) or rely on specific assumptions about the policy's parametrization (Wang et al. 2019). Heuristic-guided DRL can potentially mitigate the aforementioned issues, but the currently established approaches based on reward shaping or augmentation (Toro Icarte et al. 2018; De Giacomo et al. 2019) are, in practice, sample-inefficient and sensitive to the accuracy of heuristics (Cheng, Kolobov, and Swaminathan 2021).

In this paper, we propose a novel neuro-symbolic approach to improve the sample efficiency of DRL when generalizing to domains with longer planning horizons, more sub-goals, and sparse rewards. Specifically, we adapt interpretable symbolic knowledge from small-scale, easy-to-solve scenarios to guide the training of DRL agents in more challenging settings, in which the neural algorithm obtains poor performance. We include symbolic knowledge as logical specifications (rules) representing an approximation of the policy learned by the agent in simpler domain instances (e.g., small grids with few objects in gridworlds), which require limited exploration. When facing more complex or diverse scenarios, the training of the DRL agent is then enhanced by performing autonomous reasoning on the knowledge transferred from the easier setting, in order to deduce the set of most promising actions given the observation of the agent. We then leverage this knowledge as a planning heuristic to be used directly at the algorithmic level of DRL. In contrast to existing works requiring an exact definition of sub-goal specifications or sub-plans (Kokel et al. 2023), our methodology works even with imperfect symbolic knowledge (e.g., learned from data (Furelos-Blanco et al. 2021)), and does not require DRL parameter re-tuning when generalizing large domain instances. In more detail, the contributions of this paper are the following:

- we propose a novel neuro-symbolic approach for DRL, which exploits autonomous reasoning on partial logical policy specifications (either handcrafted or acquired in easier settings) to deduce the most promising actions to be taken. We present two different integrations of the

symbolic knowledge into the DRL training process, both in exploration and exploitation. Our solution is implemented for the popular class of ϵ-greedy DRL algorithms, which are relevant for the DRL community (Liu, Viano, and Cevher 2022) but are still sample-inefficient (Dann et al. 2022).

- we exploit an ϵ-decay strategy to balance between the neural and symbolic components without compromising the exploration-exploitation tradeoff of the original DRL algorithm;
- we assess the performance of our methodology in two relevant gridworlds, *OfficeWorld* (Toro Icarte et al. 2018) and *DoorKey* (Chevalier-Boisvert et al. 2023), characterized by sparse rewards, multiple sub-goals, and long planning horizons. In particular, we leverage logical heuristics learnt from DRL executions in small grids with few objects, and then assess sampling efficiency and training performance on larger grids with more items (hence, longer planning horizon and more sub-goals) and partially observable scenarios. We show that our methodology is more robust to imperfect partial policies, against an established neuro-symbolic DRL approach based on reward machines (Toro Icarte et al. 2018). We finally perform an ablation study to evidence the necessary balance between symbolic reasoning and DRL via both the ϵ-decay mechanism and a *confidence* parameter ρ that assesses how much we trust the symbolic knowledge.

2 Related Works

The problem of sampling efficiency prevents DRL scalability and generalization in the presence of long planning horizons, sparse rewards, and many sub-goals. Initially, ad-hoc algorithms have been developed to face this problem, such as Deep Q-Network (DQN), Rainbow DQN, SAC, etc. (Sutton and Barto 2018). Some approaches then tackled this problem by increasing the variability of the environment at the training stage, possibly with past training information (Bertran et al. 2020) or human intervention (Strouse et al. 2021). In (Igl et al. 2019; Lee et al. 2020) regularization techniques are employed to prevent overfitting in DRL, which is a major cause for the lack of generalization. Nevertheless, these approaches still require numerous training iterations in large domains; furthermore, the interpretability of the learned policies is still hindered, especially when they are represented by deep neural networks.

The methodology proposed in this paper lies in the neuro-symbolic DRL research area, which combines the abstraction and generalization capabilities of interpretable symbolic (logical) representation and reasoning tools with the inherent advantages of neural approaches when dealing with uncertain data from environmental interaction. The most prominent approach to neuro-symbolic DRL exploits the definition of logical specifications to derive automata, driving the agent towards sub-goals in a hierarchical planning framework (Kokel et al. 2023) or by shaping the reward (De Giacomo et al. 2019; Furelos-Blanco et al. 2021). However, reward shaping approaches still require many environmental interactions and do not solve the sample-inefficiency, especially with a long planning horizon or with imperfect heuristics (Cheng, Kolobov, and Swaminathan 2021). Recent works (Meli, Castellini, and Farinelli 2024; Sreedharan and Katz 2023; Veronese, Meli, and Farinelli 2025b; Veronese, Meli, and Farinelli 2025a) propose to learn symbolic knowledge from past traces (state-action pairs) of an agent's executions, instead of handcrafting symbolic knowledge. However, they focused either on the exploration phase only (Meli, Castellini, and Farinelli 2024; Veronese, Meli, and Farinelli 2025b; Veronese, Meli, and Farinelli 2025a), or on a soft initialization of Q-values in tabular RL (Sreedharan and Katz 2023), without comprehensively realizing an efficient and adaptive neuro-symbolic integration for DRL. On the contrary, we propose a novel neuro-symbolic methodology for DRL, which jointly leverages the symbolic knowledge at the algorithmic level, both in exploitation and exploration. Specifically, we consider the popular class of ϵ-greedy DRL algorithms, where exploitation and exploration are neatly separated. We modulate the exploration factor ϵ to probabilistically favour the selection of symbolically entailed actions. This is achieved by biasing the action distribution in exploration and adaptively re-scaling the Q-values during exploitation, according to ϵ-parameter. Thanks to an ϵ-decay strategy, the early exploration of the agent is more sample-efficient, a fundamental requirement for scalable DRL (Jiang, Kolter, and Raileanu 2024). At the same time, we progressively modulate the impact of symbolic reasoning over DRL, mimicking the human-like synergy between fast and slow thinking (Kautz 2022).

Finally, our methodology presents some similarities with Statistical Relational Learning (SRL) (Marra et al. 2024), as proposed in (Hazra and De Raedt 2023). However, SRL hardly scales to complex domains, requiring the accurate definition of the policy search space (Hazra and De Raedt 2023). Furthermore, by exploiting the advantages of both symbolic reasoning and DRL, with respect to pure logical learning proposed in (Hazra and De Raedt 2023), our algorithm efficiently generalizes to more challenging domains with longer planning horizons and more sub-goals, where standard DRL algorithms fail.

3 Background

We here introduce the relevant background to our methodology, i.e., solving Markov Decision Processes (MDPs) with ϵ-greedy DRL, our testing domains, and the logical framework of Answer Set Programming (ASP) (Lifschitz 1999), which is the state of the art for logical representation and reasoning on planning problems (Meli, Nakawala, and Fiorini 2023).

3.1 MDPs and Reinforcement Learning

Markov Decision Processes (MDPs) provide a formal framework for planning and decision-making in deterministic and stochastic environments(Sutton and Barto 2018). A MDP is defined by the tuple (S, A, T, R, γ), where S is the set of states describing the environment; A is the set of possible actions; $T : S \times A \rightarrow \Pi(S)$ is the state transition function, representing the probability of transitioning to

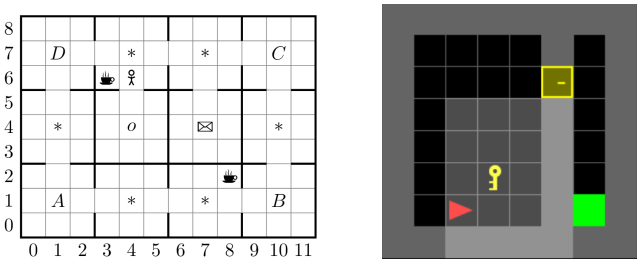

Figure 1: Our testing domains: *OfficeWorld* (left) and *DoorKey* (right).

state s' from state s when action a is taken; $R : S \times A \to \mathbb{R}$ is the reward function, providing the immediate reward received after taking action a in state s; $\gamma \in [0, 1]$ is the discount factor, which determines the present value of future rewards. Planning with an MDP aims at finding an optimal policy $\pi^* : S \to A$ that maximizes the expected cumulative reward (return) over time. The return, starting from state s and following a policy π, is captured by the value function $V^\pi(s)$:

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,|\, s_0 = s, \pi\right]$$

In model-free DRL, the agent learns to solve a MDP by interacting with the environment directly, learning an optimal policy and the value function as it performs actions in the environment and collects rewards. Typically, DRL is based on the interleave between exploitation and exploration. During *exploitation*, the agent selects the action according to the currently learned policy and value models; in the *exploration* phase, the agent randomizes its decision in order to explore new regions of the policy space and ultimately improve its learned model. In this paper, we focus on a specific class of DRL algorithms based on ϵ-greedy exploration strategy (Liu, Viano, and Cevher 2022), where typically a random action is chosen with ϵ probability. DQN is the most popular representative of this class of algorithms (Sutton and Barto 2018). It uses a deep neural network (Q-network) to approximate an action-value function $Q(s, a; \theta)$, where s represents the state, a the action, and θ the parameters of the neural network. DQN employs an ϵ-greedy policy with an ϵ-decay strategy to balance exploration and exploitation, by gradually reducing ϵ to favor exploitation as the training progresses. Initially, ϵ is set to a high value, encouraging exploration by selecting random actions. Over time, ϵ is gradually reduced according to a decay schedule, allowing the agent to exploit the learned Q-values more frequently as training progresses. This decay is crucial for the agent to sufficiently explore the environment in the early stages and focus on exploiting its knowledge later.

3.2 Domains

In the **OfficeWorld** domain (Figure 1) (Toro Icarte et al. 2018), the agent is moving on a 9×12 grid, representing various rooms. The state space is given by the set of all locations in the grid (i.e., the agent knows the coordinate of the location it is stepping on), and the observable elements represented by coffee, mail, decorations (marked as $*$ in Figure 1), rooms A, B, C, D and office o. The fully deterministic action space is defined as $A = \{\texttt{left}, \texttt{right}, \texttt{up}, \texttt{down}\}$. We tested our methodology on the *DeliverCoffeeAndMail* task, which requires bringing both coffee and mail to the office, and on the *PatrolABC* task, in which the agent must visit rooms A, B, and C to complete the task. In both cases, stepping on a decoration ends the episode with a failure.

In the **DoorKey** domain (Figure 1), an agent (red) has to pick up a key to unlock a door (the key and the door must be of the same color) and then get to the green goal square. The environment is discretized in uniform cells, and the state space the agent can observe is a 7×7 portion of the full grid in front of it (unless walls are present), as shown in light gray in Figure 1. Each tile is encoded as a 3-dimensional tuple `(object, color, state)`, indicating, respectively, the object that occupies that cell (e.g. a key, a door or a wall), its color and its state (open or closed, in case the object is a door). The action space for the domain is $A = \{\texttt{left}, \texttt{right}, \texttt{forward}, \texttt{pickup}, \texttt{open}\}$. Performing actions `left` and `right`, the agent turns in the corresponding direction; the `forward` action makes the agent move one cell ahead in the current direction; `pickup` is only effective in front of a key and it results in picking the key up; and `open` can be used by the agent to open a door when in front of it, by using a previously collected key. Both environments are fully deterministic, and a reward of $1 - 0.9 * (\texttt{step\_count}/\texttt{max\_steps})$ is given for completing the task, 0 otherwise.

Both domains are particularly challenging because of the highly sparse reward and the need to coordinate multiple macroactions towards sub-goals (e.g., picking the key before opening the door and reaching the goal) for the winning strategy.

3.3 Answer Set Programming and Reasoning

Answer Set Programming (ASP) (Lifschitz 1999) represents a planning domain by a sorted signature $\mathcal{D}$, with a hierarchy of symbols defining the alphabet of the domain (variables, constants, and predicates). Logical axioms or rules are then built on $\mathcal{D}$. In this paper, we leverage the ASP formalism to represent the background knowledge about the MDP policy, either defined by human experts or learned via DRL in small scenarios (Furelos-Blanco et al. 2021; Meli, Castellini, and Farinelli 2024; Veronese, Meli, and Farinelli 2025b). Hence, we consider normal rules, i.e., axioms in the form $\texttt{h} \texttt{ :- } \texttt{b}_1, \ldots, \texttt{b}_\texttt{n}$, where the body of the rule $\mathcal{B} = \texttt{b}_1 \wedge \cdots \wedge \texttt{b}_\texttt{n}$ (i.e. the logical conjunction of the literals) serves as the precondition for the head $\texttt{h}$. In our setting, body literals represent features generated from the state of the environment (e.g., $\texttt{samecolor}(\texttt{X}, \texttt{Y})$ denoting that two elements are of the same color in *DoorKey*), while the head literal represents an action (e.g., `pickup` in *DoorKey*).

Given an ASP problem formulation P, an ASP solver computes the *answer sets*, i.e., the minimal models satisfying ASP axioms, after all variables have been *grounded* (i.e., assigned with constant values). Answer sets lie in Herbrand base $\mathcal{H}(P)$, defining the set of all possible ground terms that

can be formed. In our setting, answer sets contain the feasible actions available to the DRL agent.

4 Methodology

Our neuro-symbolic DRL strategy (SR-DQN) combines ϵ-greedy DRL exploration and exploitation with Symbolic Reasoning (SR) over logical knowledge, approximating a good partial policy for small and simple domain instances. For simplicity, we frame our methodology in the context of the well-established DQN algorithm; however, it can be easily extended to any other ϵ-greedy DRL approach. We now detail the main phases of SR-DQN.

4.1 Logical Representation of the MDP

We begin by representing the MDP domain using the logical formalism of ASP. This involves encoding the state and action spaces as ASP terms. In the ASP signature, we define terms corresponding to environmental features $\mathcal{F}$, which capture essential aspects of the state (e.g., `locked(X)` in the *DoorKey* domain, indicating that door `X` is locked), as well as terms for actions $\mathcal{A}$ (e.g., `left`, a constant term representing the corresponding action).

To bridge the MDP and the corresponding logical representations, we introduce a *feature map* $F_{\mathcal{F}} : S \to \mathcal{H}(\mathcal{F})$ and an *action map* $F_{\mathcal{A}} : A \to \mathcal{H}(\mathcal{A})$, where $\mathcal{H}(\mathcal{F})$ and $\mathcal{H}(\mathcal{A})$ denote the Herbrandt bases of $\mathcal{F}$ and $\mathcal{A}$, respectively. More specifically, the feature map $F_{\mathcal{F}}$ translates an MDP state s into a logical description composed of ground terms from $\mathcal{F}$. For example, in the example environment depicted in Figure 1, a specific state s might be represented as $F_{\mathcal{F}}(s) = \texttt{key(X)}, \texttt{notcarrying}$. Likewise, the action map $F_{\mathcal{A}}$ provides a logical representation of MDP actions using ground terms from $\mathcal{A}$. This part of our methodology relies on the following assumptions:

Assumption 1. *The sets $\mathcal{A}$, $\mathcal{F}$ and the mappings $F_{\mathcal{F}}$ and $F_{\mathcal{A}}$ are known a priori.*

This is a standard assumption in the literature (Meli, Castellini, and Farinelli 2024; Furelos-Blanco et al. 2021), and it is considerably weaker than requiring full symbolic task specifications or the existence of a symbolic planner (Kokel et al. 2023). In particular, $F_{\mathcal{A}}$ can be seen as a straightforward symbolic encoding of the MDP's action space. Furthermore, we do not assume that $\mathcal{F}$ is *complete*, i.e., it does not need to contain all task-relevant predicates. This relaxation is justified because our neuro-symbolic integration is robust to imperfect partial policies (see Section 4.3). Thus, we assume that a minimal set of domain predicates, along with a grounding mechanism $F_{\mathcal{F}}$, is either known or can be obtained via automated symbol grounding techniques (Umili, Capobianco, and De Giacomo 2023).

Assumption 2. *The action map $F_{\mathcal{A}}$ is surjective.*

In other words, to each ground logical predicate corresponds at least one MDP action. This assumption is typically satisfied in practice. For discrete action spaces, distinct predicates can be assigned to each action (i.e., $F_{\mathcal{A}}$ is bijective). In continuous settings, it is trivial to define a discretization of the action space, where each grounded action predicate maps to a set of continuous actions. It is also possible to learn this mapping (Umili, Capobianco, and De Giacomo 2023).

4.2 Logical Representation of Policy Knowledge

Once the MDP is expressed in ASP formalism, we can represent the background information about the policy in terms of the maps $F_{\mathcal{F},A}$. To this aim, we define a *partial logical policy* $\pi_{ASP} : \mathcal{F} \to \mathcal{A}$, which maps environmental features to action terms. The logical policy encodes normal rules in the form $\texttt{a :-f}_1, \ldots, \texttt{f}_n$, with $f_i \in \mathcal{F}, a \in \mathcal{A}$. For instance, in *DoorKey* domain, π_{ASP} may correspond to the rule:

$$\texttt{pickup(X) :-key(X), door(Y), samecolor(X, Y).}$$

meaning that the agent should pick up a key if it is of the same colour as the door.

We remark that this knowledge can be inaccurate, e.g., learned from previous example executions (Furelos-Blanco et al. 2021). In order to account for the possible inaccuracy of partial policies, we also define a confidence level $\rho \in [0, 1)$ about π_{ASP}.

4.3 Neuro-symbolic Training

Algorithm 1 SR-DQN training loop

Require: Env, maps $F_{\mathcal{F}}$, $F_{\mathcal{A}}$, partial policy π_{ASP}, max episodes E, max steps T, exploration parameters ϵ_i, ϵ_f, ϵ_r, partial policy confidence $\rho \in [0, 1)$

Initialize replay memory M, $Q(s, a; \theta)$, $\epsilon \leftarrow \epsilon_i$
for episode $e = 1$ to E **do**
  Initialize state s_0
  for step $t = 1$ to T **do**
    Uniform sample $x \sim [0, 1]$
    if $x \geq \epsilon$ **then**
      $a \leftarrow$ SR-Exploitation(s_t, ϵ, ρ)
    else
      $a \leftarrow$ SR-Exploration(s_t, ϵ, ρ)
    end if
    $s_{t+1}, r \leftarrow$ Env.step(a)
    Store (s_t, a, r, s_{t+1}) in M
    $\theta \leftarrow$ **DQNUpdate**(θ)
    $s_t \leftarrow s_{t+1}$
    if s_t is terminal **then**
      break
    end if
  end for
  if $\epsilon > \epsilon_f$ **then**
    $\epsilon \leftarrow$ LinearDecrease($\epsilon, \epsilon_f, \epsilon_r$)
  end if
end for

Algorithm 1 gives a general view of how we integrate symbolic reasoning into DQN. The main components of our methodology are highlighted in red. Our goal is to improve the outcome of ϵ-greedy DRL, by reasoning over the logical policy π_{ASP} in order to efficiently bias the agent towards the most promising actions from background policy knowledge. As in standard DQN, after initializing the Q-network

Algorithm 2 SR-Exploration

Require: $F_{\mathcal{F}}$, $F_{\mathcal{A}}$, π_{ASP},current state s_t, ϵ, ρ
1: $\mathcal{A}_{\pi_{ASP}} \leftarrow$ **ComputeAnswerSet(**$\pi_{ASP}, F_{\mathcal{F}}(s_t)$**)**
2: $A_{\pi_{ASP}} \leftarrow F_{\mathcal{A}}^{-1}(\mathcal{A}_{\pi_{ASP}})$
3: **if** $A_{\pi_{ASP}} \neq \emptyset$ **then**
4: sample $a \sim$ **WeightProb(**$A, A_{\pi_{ASP}}, \rho, \epsilon$**)**
5: **else**
6: Uniform sample $a \sim A$
7: **end if**
8: **return** a

Algorithm 3 SR-Exploitation

Require: $F_{\mathcal{F}}$, $F_{\mathcal{A}}$, π_{ASP}, access to current network Q, current state s_t, ϵ, ρ
1: Qvals $\leftarrow Q(s_t, a; \theta)$
2: $\mathcal{A}_{\pi_{ASP}} \leftarrow$ **ComputeAnswerSet(**$\pi_{ASP}, F_{\mathcal{F}}(s_t)$**)**
3: **if** $A_{\pi_{ASP}} \neq \emptyset$ **then**
4: Qvals$_{\pi_{ASP}} \leftarrow$ **RescaleQvals(**Qvals, $\mathcal{A}_{\pi_{ASP}}, \rho, \epsilon$**)**
5: $a \leftarrow \arg\max_a$ Qvals$_{\pi_{ASP}}$
6: **else**
7: $a \leftarrow \arg\max_a$ Qvals
8: **end if**
9: **return** a

Q and the replay buffer M, we set the exploration rate ϵ to a suitable high initial value ϵ_i (Line 1), in order to favour exploration at the early stages of training, where the agent has not collected enough experience to build an accurate Q-network. During each episode of training, starting at state s_0 (Line 3), at each time step t, the algorithm samples a random number $x \in [0, 1]$ uniformly (Line 5), to decide whether to perform exploitation (Line 6-7) or exploration (Line 8-9), both of which are improved via symbolic reasoning and will be explained in detail in the next subsections. After an action is taken, the agent observes the next state and reward, storing this experience in the replay memory for later training. The Q-network is then updated according to the original DRL algorithm. Finally, once the episode reaches termination (Line 15), the exploration rate ϵ is decreased until the lower bound ϵ_f is reached (Line 17)[1].

The exploration fraction parameter ϵ_r controls how quickly, during the training, ϵ_f must be reached, in terms of the number of episodes. For example, $\epsilon_r = 0.5$ means that $\epsilon = \epsilon_f$ is reached at episode $e = E/2$. In this way, the agent favours exploitation in place of exploration as the training progresses, while also relying more on the knowledge gained by the network and less on the background logical knowledge derived from smaller domains.

We now explain in detail how we employ symbolic reasoning in the different phases of training and then analyze how these changes affect the optimality guarantees of the original DRL algorithm.

Neuro-symbolic exploration The exploration phase in standard ϵ-greedy approaches consists of picking a uniformly random action from the set A. On the contrary, in SR-Exploration (as shown in Algorithm 2) automated reasoning is performed over π_{ASP} to identify the set $\mathcal{A}_{\pi_{ASP}}$ of actions (ground terms in ASP formalism) entailed by the background policy knowledge, given the current set of ground environmental features $F_{\mathcal{F}}(s_t)$ (Line 1). Suggested ASP ground actions are then translated to the MDP action space $A_{\pi_{ASP}} \subseteq A$, by considering the pre-image $F_{\mathcal{A}}^{-1}$ of the action map (Line 2, see Assumption 1). The agent then selects an action from A according to a weighted probability distribution (Line 4), where the weights are defined for each action as follows:

$$w_a = \begin{cases} \rho & \text{if } a \in A_{\pi_{ASP}} \\ 1 - \rho & \text{otherwise} \end{cases} \quad (1)$$

and then normalized, such that $\sum_{a \in A} w_a = 1$. As explained in Section 4.2, ρ represents the level of confidence about the knowledge encoded in π_{ASP}, which may be inaccurate, especially when it is learned (Meli, Castellini, and Farinelli 2024). The higher ρ value, the higher the probability that the agent selects an action suggested by background policy knowledge. If $A_{\pi_{ASP}} = \emptyset$ (i.e., π_{ASP} cannot suggest any valuable action at the given state s_t), then a uniformly random action is selected from A as in standard DQN (Line 6).

Neuro-symbolic exploitation Our approach, shown in Algorithm 3, aims at enhancing the standard DRL exploitation phase by biasing the choice towards the most promising action according to the symbolic knowledge. Given the current state s_t, the agent first queries the Q-network Q to obtain estimated Q-values for all possible actions (Line 1). As in DQN, these Q-values represent the expected return for each action under the current policy. We then employ the ASP policy π_{ASP} in the same way as explained in Section 4.3 to compute the set of preferred actions $\mathcal{A}_{\pi_{\text{ASP}}}$ (Line 2). Subsequently, the Q-values are rescaled (Line 3) by a factor $k_a = 1 + (\epsilon * w_a)$ for each action, with w_a determined following Equation 1. This is aimed at adjusting the action values within the context of the most promising action set $\mathcal{A}_{\pi_{\text{ASP}}}$, according to the confidence parameter ρ. Adding ϵ as an additional rescaling parameter allows the agent to increasingly trust the estimations produced by the neural network as the training proceeds. Finally, the action with the highest rescaled Q-value is selected for execution (Line 4).

4.4 Impact of Symbolic Knowledge on Training Convergence

The integration of symbolic knowledge we propose is designed to improve the efficiency of DRL exploration while maintaining the stability and empirical convergence behavior of the base algorithms. This balance is achieved by modulating the influence of symbolic guidance through the exploration factor (ϵ_r, ϵ_f), which gradually decreases as training progresses, and the confidence parameter ρ. At the beginning of training, when the agent has limited experience

[1]The linear decrease rule is chosen empirically in our methodology, but more sophisticated strategies can be adopted to reduce ϵ, depending on the specific task (Gimelfarb, Sanner, and Lee 2020).

and the Q-value estimates are still inaccurate, the symbolic component plays a stronger role in shaping the agent's behavior. By biasing the action selection toward more promising actions, the agent can more efficiently explore relevant regions of the state space, thereby accelerating early learning. As ϵ_t decays, the contribution of symbolic guidance becomes progressively weaker, allowing the learned value function to increasingly dominate the action selection process. This gradual reduction prevents the logical policy from over-constraining the agent's behavior or trapping it in suboptimal local minima that may arise from incomplete or imperfect symbolic knowledge. The symbolic component thus acts as a form of guided exploration in the early stages, while the long-term learning dynamics of the underlying DRL algorithm remain unaffected.

4.5 Complexity of Symbolic Inference in Training

At each training step (Lines 7 an 9 of Algorithm 1), the RL agent needs to evaluate whether $F_{\mathcal{F}}(s)$ (i.e. the grounding of the current MDP state) satisfies π_{ASP}, which is a fixed set of non-disjunctive ASP rules. It is known that the grounding part of ASP reasoning is the most computationally intensive (Gebser et al. 2019). Specifically, let each normal rule $r \in \pi_{ASP}$ include v variables that range over a finite domain of N constants. Denote the finite set of all possible ground instances as $g(r, F_{\mathcal{F}}(s))$. The number of such ground instances is $O(N^v)$. Verifying whether these instances are satisfied by $F_{\mathcal{F}}(s)$ then requires checking, for each ground instance, the truth of every literal in its body. For propositional (ground) non-disjunctive programs, this process is linear in the number of literal occurrences (Eiter, Gottlob, and Leone 1997). Therefore, the total computational cost of verifying all grounded instances of rules in π_{ASP} can be expressed as:

$$T_{\text{ASP}}(F_{\mathcal{F}}(s), \pi_{ASP}) = O\left(\sum_{r \in \pi_{ASP}} |g(r, F_{\mathcal{F}}(s))| \cdot |\mathcal{B}_r| \right),$$

where $|\mathcal{B}_r|$ denotes the number of literals composing the body of rule r. In our domain representations, each rule references only a small subset of state predicates (e.g., objects within the agent's view), thus the combinatorial term $|g(r, F_{\mathcal{F}}(s))|$ remains small in practice. In more general and complex settings, we remark that π_{ASP} doesn't evolve during training and $\mathcal{F}$ is known a priori. Hence, it is possible to compute the full grounding before training starts, thereby eliminating the exponential factor of the complexity. Consequently, the symbolic reasoning over π_{ASP} introduces an overhead which is proportional to the number of variable instantiations actually realized in the current state. For compact rule sets and moderate grounding sizes, this practically results in a negligible increment of the original DQN step time.

5 Empirical Evaluation

We evaluate our SR-DQN methodology on the *DoorKey* and *OfficeWorld* domains presented in Section 3.2. These domains are widely used in related literature (Hazra and De Raedt 2023; Toro Icarte et al. 2018; Furelos-Blanco et al. 2021), since they present unique challenges, namely i) sparse reward definition; ii) long planning horizon; iii) the need for optimal strategic coordination towards the achievement of sub-goals. Hence, they represent the ideal benchmark to validate our methodology.

In the following, we primarily evaluate the scalability and sampling efficiency of our method in both domains, increasing the planning horizon and number of sub-goals. To this aim, we compare with a standard DQN baseline and a state-of-the-art reward machine methodology as proposed by (Toro Icarte et al. 2018). We then perform a thorough ablation study in the *DoorKey* domain (which was the most challenging one in our experiments), to separately investigate the performance of symbolic exploration vs. exploitation, and to assess the impact of relevant parameters of our methodology. Importantly, we tuned the DQN baseline on the easier settings (e.g., maps with one key only in Doorkey) and applied it to more challenging scenarios using the same set of hyperparameters. This allowed us to test the capabilities of our approach in achieving generalization without requiring the DRL algorithm to be tuned from scratch[2].

5.1 Logical Domain Representations and Policies

We now introduce the symbolic knowledge π_{ASP} for our testing domains. Crucially, symbolic knowledge represents partial policies which are learned by the authors of (Furelos-Blanco et al. 2021; Hazra and De Raedt 2023) in small-scale domains. Hence, they may fail to generalize to more complex settings, e.g., with more items and sub-goals and a longer planning horizon.

OfficeWorld We formalize the *OfficeWorld* domain by introducing environmental features $\mathcal{F}$ that represent the known positions of the observables in the map: `coffee(X)`, `mail(Y)`, `office(Z)`. Moreover, we introduce the `hasCoffee` and `hasMail` predicates to state that the agent has already picked up that item, and `hittingDecors`, which represents the presence of a decoration (that must not be broken) in the agent's moving direction. Finally, predicate `visited(X)` states that the agent has already visited room `X`. As logical policies, we take the ones learned by (Furelos-Blanco et al. 2021). Namely, the policy for the *DeliverCoffee* task:

```
goto(X) :- coffee(X), not hasCoffee,          (2)
           not hittingDecors.
goto(X) :- office(X), hasCoffee,              (3)
           not hittingDecors.
```

and the one for the *VisitAB* task:

```
goto(A) :- not visited(A), not hittingDecors.  (4)
goto(B) :- visited(A), not hittingDecors.      (5)
```

[2]Code to replicate results in both Section 5.2 and Section 5.3 is available in the supplementary material.

where $\mathcal{A} = \{\texttt{goto(X)}\}$ denotes the action of moving to an item. In order to respect Assumption 2, we map this to

$$\texttt{left :- goto(X), on\_left(X).}$$
$$\texttt{right :- goto(X), on\_right(X).}$$
$$\texttt{forward :- goto(X), straight(X).}$$

adding the necessary body predicates to $\mathcal{F}$. The above specifications suggest that the agent should first pick up the coffee (Rule (2)) and then reach the office (Rule (3)), both without breaking any decoration.

Doorkey For simplicity, we replicate the ASP formulation of the Doorkey domain proposed in (Hazra and De Raedt 2023). As environmental features $\mathcal{F}$, we define $\texttt{door(X)}, \texttt{key(Y)}, \texttt{goal(Z)}$ to denote doors, keys and the goal. We then introduce predicate $\texttt{locked(X)}$ to state that a door is locked; `unlocked` to state the intermediate door to the goal is open; $\texttt{samecolor(X, Y)}$ to denote items (either doors or keys) of the same colour; $\texttt{carrying(Y)}$ to specify that the agent is carrying an object (key); and `notcarrying` to specify that the agent is not carrying anything.

We also consider the logical policy learned in (Hazra and De Raedt 2023), in a small 5×5 grid with only one door and key:

$$\texttt{pickup(X) :- key(X), samecolor(X, Y), door(Y), notCarryingKey.} \quad (6)$$
$$\texttt{open(X) :-key(Z), samecolor(X, Z), door(X), locked(X), carrying(Z).} \quad (7)$$
$$\texttt{goto(X) :- goal(X), unlocked.} \quad (8)$$

Rule (6) suggests that the agent should pick up a key `X` if it matches the colour of the door `Y` and the agent is not currently carrying another key. Then, from Rule (7), the agent can unlock a door `X` if it is holding a matching-colored key `Z`, the door `X` is locked and `Z` is the correct key for that door. Finally, Rule (8) prescribes that the agent moves towards the goal `X` if all doors along the path are unlocked.

5.2 Scalability Study

We compare SR-DQN against the performance of a standard DQN algorithm (DQN in the figures) and DQN with reward machines (RM-DQN in the figures) as designed in (Toro Icarte et al. 2018). For reward machines, we test different rewards for state transitions in both Doorkey and OfficeWorld and keep the best-performing ones in the tuning scenarios. For SR-DQN, we empirically choose $\epsilon_f = \epsilon_r = 0.3$ in Doorkey tasks, and $\epsilon_f = 0.05$, $\epsilon_r = 0.1$ in OfficeWorld tasks. Since we do not have information about the confidence level of π_{ASP} from (Hazra and De Raedt 2023) and (Furelos-Blanco et al. 2021), we empirically set $\rho = 0.8$ for both domains. For each method, we evaluate the discounted return[3] achieved over 5 random seeds.

Finally, in Table 1, we report the execution times of DQN and SR-DQN across all tested tasks. It is evident that the additional overhead introduced by the symbolic inference, indicated in the last row, has a negligible impact on the overall training duration. This demonstrates that integrating symbolic guidance does not compromise the efficiency of the learning process while still providing the benefits evidenced throughout this section.

OfficeWorld To test the scalability performance of SR-DQN in the *OfficeWorld* domain, we employ the policy learned by (Furelos-Blanco et al. 2021) in the *DeliverCoffee* and *PatrolAB* tasks as partial policies in the more complex *DeliverCoffeeAndMail* and *PatrolABC* tasks, respectively. In this way, we assess the sampling efficiency of our methodology when generalizing to longer planning horizons and more sub-goals. Figure 2 shows the performance of SR-DQN and the baselines. For both tasks, we tuned the base DQN algorithm to solve the easier setting (i.e. *DeliverCoffee* and *PatrolAB*) and then used the same set of hyperparameters to train all the agents in the more challenging tasks. On average, SR-DQN achieves the highest return by the end of training, also proving to be more stable with a lower standard deviation with respect to DQN in particular. On the other hand, RM-DQN converges more slowly to a lower average return with larger variance, proving the inefficiency of reward augmentation, as theoretically suggested by (Cheng, Kolobov, and Swaminathan 2021).

DoorKey For *DoorKey*, we evaluate performance across a range of scenarios with increasing complexity. We begin by scaling up the environment from a 5×5 to an 8×8 grid, and simultaneously increase the number of keys (with distinct colors), either 2 or 4. We further extend the evaluation to a larger 16×16 grid, considering both one-key and two-key configurations[4]. In these settings, the agent must also correctly decide which key to pick, depending on the door's color. These scenarios provide a compelling demonstration of the benefits of our neuro-symbolic approach, which effectively leverages prior knowledge to generalize to more complex domains. Importantly, none of these larger map configurations were considered in (Hazra and De Raedt 2023), the source of the π_{ASP} policy.

Figure 3 shows the training performances of our algorithm and the baselines in random 8×8 maps with a single key. Since all methods achieve optimal performance in this setting, we freeze the fine-tuned parameters and reuse them in more challenging configurations. This allows us to evaluate the ability of both the baselines and our approach to generalize effectively without requiring re-tuning parameters from scratch. Figure 4 shows the performance of our algorithm and the baselines in 8×8 and 16×16 random maps with different amounts of keys. The proposed SR-DQN algorithm clearly outperforms both the baselines in the more challenging tasks, both enlarging the size of the grid and increasing the number of keys. showing a significantly higher average return. Overall, the proposed approach demonstrates clear improvements over both base DQN and RM-DQN in tasks requiring longer planning horizons and in case of imperfect symbolic policies.

[3] For RM-DQN, we exclude the additional reward from the plots for a fair comparison.

[4] We omit the 4-keys configuration in the 16x16 map, as all tested algorithms, including ours, exhibit similarly low performance in this setting.

Domain	Task	Steps	DQN Exec Time	SR-DQN Exec Time	Exec Time Increment
DoorKey	8x8, 1 Key	3M	1h 20min	1h 24min	4 min **(5%)**
DoorKey	8x8, 2 Keys	5M	2h 10min	2h 15min	5 min **(3.85%)**
DoorKey	8x8, 4 Keys	5M	2h 10min	2h 15min	5 min **(3.85%)**
DoorKey	16x16, 1 Key	10M	2h 46min	2h 54min	8 min **(4.82%)**
DoorKey	16x16, 2 Keys	10M	2h 46min	2h 54min	8 min **(4.82%)**
OfficeWorld	CoffeeAndMail	250k	38min 30s	39 min	30s **(1.3%)**
OfficeWorld	PatrolABC	250k	40min	41min	1min **(2.5%)**

Table 1: Execution times of DQN and SR-DQN algorithms. The last column shows the time increment introduced by symbolic reasoning.

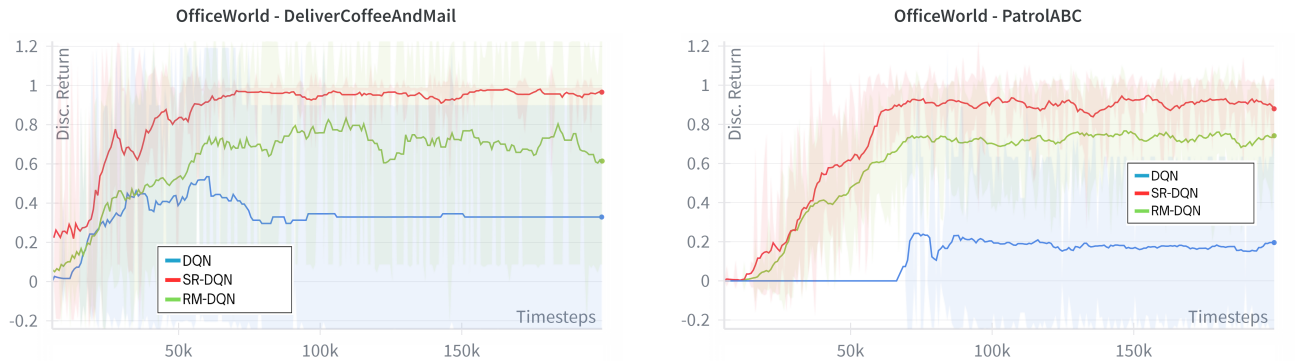

Figure 2: OfficeWorld results on the *DeliverCoffeeAndMail* task (left) and on the *PatrolABC* task (right).

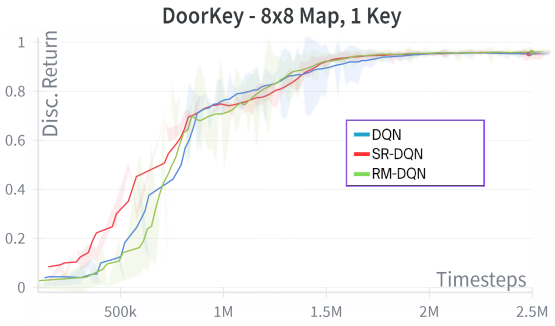

Figure 3: Training results on 8×8 single-key DoorKey.

5.3 Ablation Study

Our SR-DQN (Algorithm 1) combines symbolic reasoning both in the exploration (Algorithm 2) and the exploitation (Algorithm 3) phases of DRL, modulated by the decay law of ϵ (Line 20). Together with ρ, the values of ϵ_r and ϵ_f determine the impact of π_{ASP} on the training loop. We now want to investigate the role of these components independently.

Figure 5 shows the performance of both SR-Exploration (Algorithm 2) and SR-Exploitation (Algorithm 3) when employed as the only symbolic component within the full Algorithm 1. Specifically, as in standard DQN, when disabling SR-Exploration, we sample a uniformly from A during exploration; for SR-Exploitation, we do not rescale Q-values. We perform this ablation study on the same 8×8 Doorkey environment with 4 keys, which provides a challenging yet tractable scenario where good performance can still be achieved. Both SR-Exploration and SR-Exploitation alone outperform standard DQN, but SR-Exploitation alone achieves performance very close to that of the full SR-DQN, underscoring the importance of value shaping as a key contributor to overall performance. In contrast, SR-Exploration alone yields a smaller improvement but faster return growth at the very beginning of training.

In Figure 6 (top), we instead keep the full SR-DQN algorithm, but vary the values of ϵ_f and ϵ_r. Starting from an equal initial value $\epsilon_i = 1$, in this way we modify the decremental behaviour of ϵ, thus varying the impact of the symbolic component of our algorithm. We perform this test in the *DoorKey* environment with an 8×8 grid and 4 keys, which poses a significant challenge for all tested baselines but still allows them to learn good policies. The optimal curve corresponding to Figure 4 (top-right) is reported in red ($\epsilon_f = 0.3, \epsilon_r = 0.3$). Figure 6 (top) shows that, when decreasing ϵ_f and ϵ_r, thus reducing the impact of the symbolic policy, the training performance decreases significantly. Moreover, a too high ϵ_f value results in unstable policies, thus gaining worse returns.

Finally, Figure 6 (bottom) shows the performance of SR-DQN under different values of the confidence parameter ρ. Both excessively low ($\rho \leq 0.5$) and excessively high (ρ close to 1) confidence values lead to suboptimal performance. In the former case, the symbolic component exerts too little influence, preventing the agent from effectively exploiting the structured prior knowledge. In the latter, too much reliance is placed on the symbolic rules, whose accuracy is limited since they are learned from simplified ver-

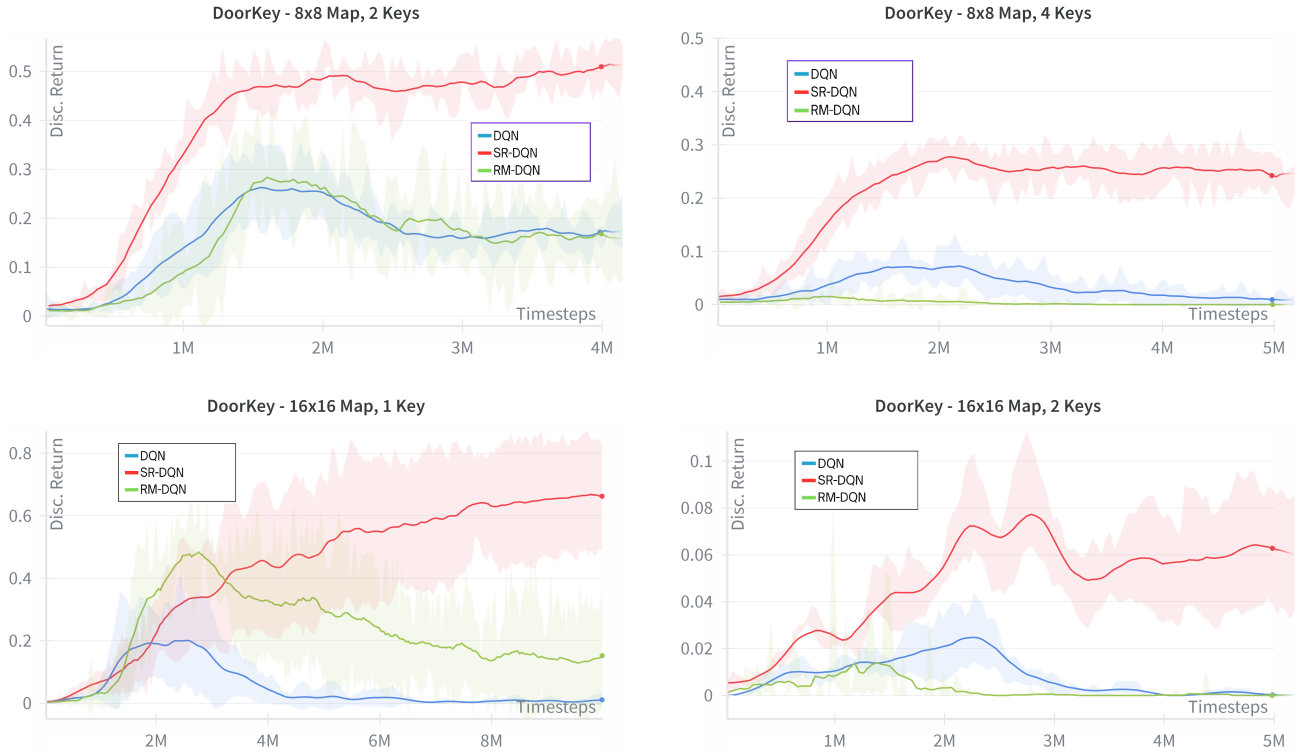

Figure 4: Training results on the DoorKey environment in random maps, varying map size and number of keys.

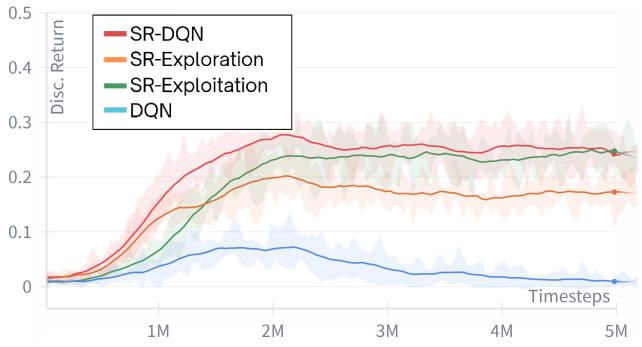

Figure 5: Ablation study over the SR-DQN components on 8×8, 4-keys DoorKey.

sions of the tasks. This analysis highlights the need for a balanced integration between neural and symbolic components, where ρ, ϵ_r, and ϵ_f regulate the trust in imperfect symbolic knowledge.

6 Conclusion and Future Work

We presented SR-DQN, a novel neuro-symbolic DRL approach to tackle the problems of scalability and sampling inefficiency in DRL, in environments with long planning horizons, sparse rewards, and multiple sub-goals. Our methodology exploits partial logical policy specifications representing the optimal strategy in easy-to-solve domain instances with a limited planning horizon. Then, we perform automated reasoning to entail suggested actions from the logical specifications, biasing both the exploration phase of ϵ-greedy DRL agents and the Q-values produced by the neural component during training, to encourage the choice of promising symbolic actions. We exploit an ϵ-decay schedule to balance symbolic reasoning and neural learning over time. Importantly, the added symbolic component doesn't represent a significant computational overhead for the original DRL algorithm. We empirically demonstrated the benefits of SR-DQN in two benchmarks, *OfficeWorld* and *DoorKey*, both of which present the challenges mentioned above, as well as partial observability in larger maps. SR-DQN consistently outperformed all the selected baselines (namely, standard DQN, DQN with reward machines, which represent a state-of-the-art technique in neuro-symbolic DRL), also being the only method capable of achieving significant returns in challenging, partially observable *DoorKey* tasks with more items (e.g., multiple keys) and sub-goals, where all tested baselines performed much worse.

In future work, we plan to generalize our methodology to a broader class of DRL algorithms beyond ϵ-greedy strategies. This includes integrating our framework with policy gradient and actor-critic methods. Additionally, we aim to extend our approach to more expressive logical representations, such as temporal or probabilistic logic.

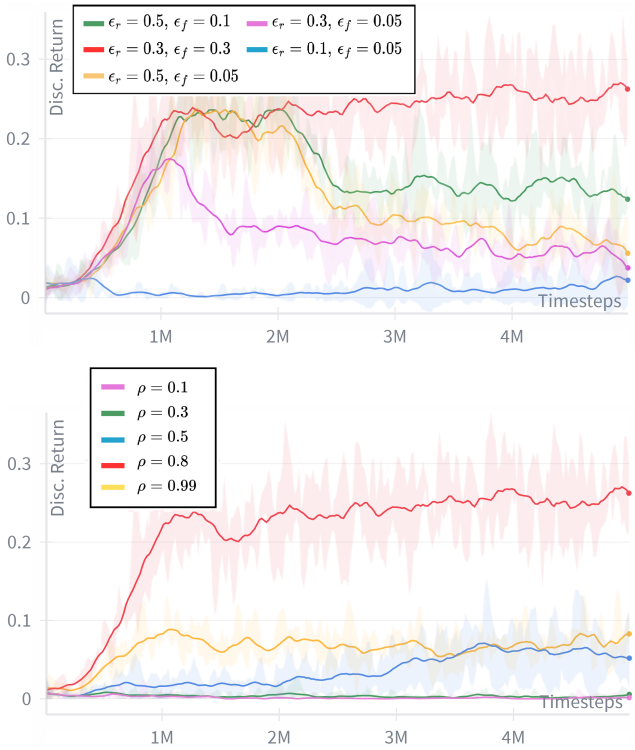

Figure 6: Ablation study over the SR-DQN main parameters, ϵ (top) and ρ (bottom), both performed on 8×8, 4-keys DoorKey.

AI Declaration

No AI-generated content was used for scientific conclusions, data analysis, or interpretation within this manuscript. All results, experiments, and claims were independently developed and verified by the authors, who assume full responsibility for the content of this work.

References

Bertran, M.; Martinez, N.; Phielipp, M.; and Sapiro, G. 2020. Instance-based generalization in reinforcement learning. *Advances in Neural Information Processing Systems* 33:11333–11344.

Cheng, C.-A.; Kolobov, A.; and Swaminathan, A. 2021. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems* 34:13550–13563.

Chevalier-Boisvert, M.; Dai, B.; Towers, M.; de Lazcano, R.; Willems, L.; Lahlou, S.; Pal, S.; Castro, P. S.; and Terry, J. 2023. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR* abs/2306.13831.

Dann, C.; Mansour, Y.; Mohri, M.; Sekhari, A.; and Sridharan, K. 2022. Guarantees for epsilon-greedy reinforcement learning with function approximation. In *International conference on machine learning*, 4666–4689. PMLR.

De Giacomo, G.; Iocchi, L.; Favorito, M.; and Patrizi, F. 2019. Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications. In *Proceedings of the international conference on automated planning and scheduling*, volume 29, 128–136.

Dulac-Arnold, G.; Levine, N.; Mankowitz, D. J.; Li, J.; Paduraru, C.; Gowal, S.; and Hester, T. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning* 110(9):2419–2468.

Eiter, T.; Gottlob, G.; and Leone, N. 1997. Abduction from logic programs: Semantics and complexity. *Theoretical Computer Science* 189(1):129–177.

Furelos-Blanco, D.; Law, M.; Jonsson, A.; Broda, K.; and Russo, A. 2021. Induction and exploitation of subgoal automata for reinforcement learning. *Journal of Artificial Intelligence Research* 70:1031–1116.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2019. Multi-shot asp solving with clingo. *Theory and Practice of Logic Programming (TPLP)* 19(1):27–82.

Gimelfarb, M.; Sanner, S.; and Lee, C.-G. 2020. Epsilon-bmc: A bayesian ensemble approach to epsilon-greedy exploration in model-free reinforcement learning. In *Uncertainty in Artificial Intelligence*, 476–485. PMLR.

Hazra, R., and De Raedt, L. 2023. Deep explainable relational reinforcement learning: a neuro-symbolic approach. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 213–229. Springer.

Ibarz, J.; Tan, J.; Finn, C.; Kalakrishnan, M.; Pastor, P.; and Levine, S. 2021. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research* 40(4-5):698–721.

Igl, M.; Ciosek, K.; Li, Y.; Tschiatschek, S.; Zhang, C.; Devlin, S.; and Hofmann, K. 2019. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems* 32.

Jiang, Y.; Kolter, J. Z.; and Raileanu, R. 2024. On the importance of exploration for generalization in reinforcement learning. *Advances in Neural Information Processing Systems* 36.

Kautz, H. 2022. The third ai summer: Aaai robert s. engelmore memorial lecture. *AI Magazine* 43(1):105–125.

Kokel, H.; Natarajan, S.; Ravindran, B.; and Tadepalli, P. 2023. Reprel: a unified framework for integrating relational planning and reinforcement learning for effective abstraction in discrete and continuous domains. *Neural Computing and Applications* 35(23):16877–16892.

Lee, K.; Lee, K.; Shin, J.; and Lee, H. 2020. Network randomization: A simple technique for generalization in deep reinforcement learning. In *Eighth International Conference on Learning Representations, ICLR 2020*. International Conference on Learning Representations.

Lifschitz, V. 1999. Answer set planning. In *Logic Programming and Nonmonotonic Reasoning: 5th International Conference, LPNMR'99 El Paso, Texas, USA, December 2–4, 1999 Proceedings 5*, 373–374. Springer.

Liu, F.; Viano, L.; and Cevher, V. 2022. Understanding deep neural function approximation in reinforcement learning via

ϵ-greedy exploration. *Advances in Neural Information Processing Systems* 35:5093–5108.

Marra, G.; Dumančić, S.; Manhaeve, R.; and De Raedt, L. 2024. From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence* 104062.

Meli, D.; Castellini, A.; and Farinelli, A. 2024. Learning logic specifications for policy guidance in pomdps: an inductive logic programming approach. *Journal of Artificial Intelligence Research* 79:725–776.

Meli, D.; Nakawala, H.; and Fiorini, P. 2023. Logic programming for deliberative robotic task planning. *Artificial Intelligence Review* 56(9):9011–9049.

Sreedharan, S., and Katz, M. 2023. Optimistic exploration in reinforcement learning using symbolic model estimates. *Advances in Neural Information Processing Systems* 36:34519–34535.

Strouse, D.; McKee, K.; Botvinick, M.; Hughes, E.; and Everett, R. 2021. Collaborating with humans without human data. *Advances in Neural Information Processing Systems* 34:14502–14515.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2018. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2112–2121.

Umili, E.; Capobianco, R.; and De Giacomo, G. 2023. Grounding ltlf specifications in image sequences. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, 668–678.

Veronese, C.; Meli, D.; and Farinelli, A. 2025a. Learning symbolic persistent macro-actions for pomdp solving over time. In H. Gilpin, L.; Giunchiglia, E.; Hitzler, P.; and van Krieken, E., eds., *Proceedings of The 19th International Conference on Neurosymbolic Learning and Reasoning*, volume 284 of *Proceedings of Machine Learning Research*, 1026–1040. PMLR.

Veronese, C.; Meli, D.; and Farinelli, A. 2025b. Online inductive learning from answer sets for efficient reinforcement learning exploration. In Bruno, P.; Calimeri, F.; Cauteruccio, F.; and Terracina, G., eds., *Hybrid Models for Coupling Deductive and Inductive Reasoning*, 93–106. Cham: Springer Nature Switzerland.

Vouros, G. A. 2022. Explainable deep reinforcement learning: state of the art and challenges. *ACM Computing Surveys* 55(5):1–39.

Wang, H.; Zheng, S.; Xiong, C.; and Socher, R. 2019. On the generalization gap in reparameterizable reinforcement learning. In *International Conference on Machine Learning*, 6648–6658. PMLR.

Zuccotto, M.; Castellini, A.; Torre, D. L.; Mola, L.; and Farinelli, A. 2024. Reinforcement learning applications in environmental sustainability: a review. *Artificial Intelligence Review* 57(4):88.