DEEP REINFORCEMENT LEARNING WITH INCOMPLETE SYMBOLIC KNOWLEDGE IN PARAMETRIZED ACTION SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Specifying a complete symbolic model of a decision-making domain is difficult, tedious, and error-prone. However, even incomplete domain knowledge can constrain decision spaces without providing a complete domain model. In this paper, we study how Deep Reinforcement Learning can use such incomplete symbolic domain knowledge in parametrized action spaces, where each decision combines a discrete action with continuous parameters. We introduce our novel Knowledge- and Gradient-guided Reinforcement Learning (KGRL) algorithm that leverages on incomplete symbolic domain knowledge to reduce training time and improve safety during exploration. KGRL uses a Datalog knowledge base that encodes action restrictions and parameter bounds, which can be instantiated e.g. from domain expert knowledge. At each decision, the knowledge base is evaluated with a symbolic state abstraction, deriving state-specific action constraints and parameter bounds, while a gradient-based refinement searches the permitted parameter space for optimal parameter values. The symbolic derivations of the knowledge base additionally provide local traces that explain the imposed restrictions. Across six evaluation tasks in four different environments, KGRL achieves higher returns faster than five RL baselines for parametrized action spaces. Ablations show that including the incomplete domain knowledge accounts for substantial performance gains, whereas the benefit of the gradient guidance remains task-dependent.

1 INTRODUCTION

Reinforcement Learning (RL) is an alternative to symbolic decision-making algorithms for planning and control, when complete symbolic domain models are unavailable or too costly to craft (Sutton & Barto, 2018; Moerland et al., 2023). However, training RL agents requires an immense amount of experiences in the form of recorded training data or direct domain interactions (Yu, 2018; Duan & Wainwright, 2024), while collecting such experience through exploration typically is costly or unsafe (Levine et al., 2020; Dulac-Arnold et al., 2021). Yet, even in complex real-world domains, there is often incomplete symbolic domain knowledge available (e.g., rules, safety constraints, or expert heuristics), which rules out unsafe or infeasible decisions without fully determining the optimal behavior (Achiam et al., 2017; Ghanadbashi & Golpayegani, 2021). However, knowing which decisions exclude is different from knowing which decision to take. Hence, the central question we want to answer in this paper is: *How can incomplete symbolic domain knowledge constrain a decision space while leaving RL to optimize what cannot be symbolically specified?*

We study this question in parametrized action spaces, where a decision involves not only choosing a discrete action but also choosing numerical parameters that influence the action's effect (Masson et al., 2016). The hybrid structure is interesting, as it is common in many real-world decision-making problems, e.g., industrial control and planning (Zhu et al., 2021; Heesch et al., 2024), where high-level actions are naturally discrete (e.g., `open_valve`, `start_pump`, `fill_tank`), but can accommodate numeric parameters (e.g., `valve_opening`, `pump_speed`, `fill_height`). The underlying problem of such scenarios is that the decision space is a union of disjoint continuous spaces, often of different dimensionality. This calls for algorithms that are specifically tailored for such problems.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Related work on RL in parametrized action spaces relies on one-shot estimators that estimate the parameters for actions at decision time (Masson et al., 2016; Hausknecht & Stone, 2016; Xiong et al., 2018; Bester et al., 2019). However, these approaches require numerous training episodes when deploying the RL agent. Related work on integrating knowledge into RL agents typically focusses on crafting domain models (Silver et al., 2017; 2018; Moerland et al., 2023), specifying goals in out-of-distribution scenarios (Ghanadbashi & Golpayegani, 2021), or constraining the agent for safe exploration and deployment (Achiam et al., 2017; Alshiekh et al., 2018; Könighofer et al., 2020; 2025). Few approaches started considering a Neuro-Symbolic perspective on RL that integrates symbolic decision-making as guardrail or augmentation of an RL policy (Mazzi et al., 2023; Meli et al., 2024; Shindo et al., 2025; Veronese et al., 2026). Yet, these approaches are typically only evaluated on discrete toy domains. To our knowledge, there is no work that integrates a structured, queryable knowledge base directly into the training and parameter estimation of an RL agent in a parametrized action space in order to guide exploration and deployment toward feasible decisions and to improve training data efficiency.

To address this gap, we propose our novel, Neuro-Symbolic Knowledge- and Gradient-Guided Reinforcement Learning (KGRL) algorithm (cf. Figure 1). KGRL introduces two coupled extensions to the DQN algorithm (Mnih et al., 2015): (i) A *Datalog knowledge base* that contains incomplete domain knowledge in the form of state-dependent rules. The knowledge base is evaluated at every decision together with a symbolic state representation to derive the set of applicable actions and parameter constraints for the given state. (ii) A *gradient-guided parameter refinement* that uses the gradient of the action-value function in a nested optimization loop. This allows KGRL to iteratively adapt the parameter values until they converge to an optimum. By using projected gradient ascent (Levitin & Polyak, 1966), we can integrate known parameter constraints from the knowledge base to exclude unrealistic parameter regions (e.g., temperatures below 0K) or parameters that violate known domain constraints. When evaluating the knowledge base at each step, KGRL can log the grounded rule instances and symbolic state facts that support the derivation of excluded actions and active parameter constraints from the decision space, as an explanation. Such logs provide local and procedural explanations (Krarup et al., 2019) along the trajectory of the agent. This connects KGRL to work on explainable and safe RL (Alshiekh et al., 2018; Milani et al., 2024; Könighofer et al., 2025).

We evaluate our KGRL algorithm against five baselines from RL in parametrized action spaces from literature (Masson et al., 2016; Hausknecht & Stone, 2016; Xiong et al., 2018; Li et al., 2022; Bester et al., 2019). KGRL outperforms all baselines, starting with higher returns and converging quicker to optimal results. Our main contributions are:

- A formalization of knowledge-constrained Parametrized Action Markov Decision Processes that allows RL agents to include symbolic domain knowledge into their decision-making.
- Our novel KGRL algorithm, that integrates reasoning over a symbolic knowledge base and a gradient-guided parameter refinement into its decision-making, leading to higher sample efficiency and performance.
- An empirical evaluation of KGRL against state-of-the-art baselines.

2 Related Work

RL in Parameterized Action Spaces. Parameterized action spaces combine a discrete action choice with action-specific continuous parameters (Masson et al., 2016). They are formalized in Parameterized Action Markov Decision Processes (PAMDPs) (Masson et al., 2016). Multiple approaches for solving PAMDPs with RL have been proposed: QPAMDP uses two policies to tackle this problem, one policy for selecting discrete actions and a policy for selecting parameters (Masson et al., 2016). The policies are alternatingly updated during training (Masson et al., 2016). Fan et al. (2019) picked up on this idea and suggested an actor-critic approach, where separate policy heads for discrete actions and numerical parameters operate on a joint state encoding in parallel. Other approaches model the decision problem as a joint policy. For example, Hausknecht & Stone (2016) proposed an actor-critic approach (PA-DDPG), where one policy network outputs a discrete action and corresponding parameters. P-DQN instead decouples the parameter estimation in a separate parameter estimator network that suggests parameter values, which are then jointly evaluated with

the state by a Deep Q-Network (Mnih et al., 2015; Xiong et al., 2018). Bester et al. (2019) improved this approach by using batch-processing to compute separate passes through the Deep Q-Network for preventing unrelated action parameters from affecting an action's Q-value. Lastly, Li et al. (2022) propose to leverage on a joint latent decision-space in which the decision-problem is solved, and from which discrete actions and numerical parameters are decoded. Across these approaches, action applicability and parameter feasibility are learnt implicitly, which yields sample-intensive exploration and can lead to infeasible parameters if not prevented by the environment.

Injecting Domain Knowledge into RL. Domain knowledge can guide RL through predictive models, constraints, and heuristics. Model-based RL uses known or learned transition models for planning and policy improvement (Moerland et al., 2023). Constrained Policy Optimization regularizes policy optimization via cost functions (Achiam et al., 2017). Constrained Hybrid Policy Optimization extends this idea to hybrid action spaces (Zhou et al., 2025). Safety layers and OptLayer correct continuous action proposals to satisfy constraints (Dalal et al., 2018; Pham et al., 2018). Mazumder et al. (2022) restrict exploration through state-action permissibility knowledge. Continuous action masking maps actions into state-dependent relevant sets (Stolz et al., 2024), and Action Mapping separates learning feasible action representations from optimizing return (Theile et al., 2025).

Integrating Symbolic Reasoning into RL. There are different approaches for connecting symbolic reasoning with RL. Alshiekh et al. (2018) create shields from temporal-logic specifications to exclude unsafe actions. Anderson et al. (2020) constrain exploration by projecting learned policies into a verifiable symbolic policy class. Beyond safe RL, Kimura et al. (2021) use logical neural networks to infer action recommendations and exclusions. Guan et al. (2022) extract landmarks from approximate symbolic models to guide subgoal and skill learning, while Meli et al. (2024) learn logical rules from traces as heuristics for online POMDP planning. Ling et al. (2023) compile Boolean constraints into distributions over valid combinatorial actions, and Han et al. (2026) learn symbolic state representations and action masks consistent with domain constraints. Closest to our work is (Veronese et al., 2026) who use an Answer Set Programming knowledge base to augment a neural policy with a symbolic policy that is generated from evaluating the knowledge base. However, to our knowledge there is no work focussing on the combination of incomplete symbolic knowledge and RL in parametrized action spaces.

3 PRELIMINARIES

Parametrized Action Markov Decision Processes. Our work in this paper builds upon PAMDPs as introduced by Masson et al. (2016). A PAMDP is a tuple $\langle S, A, \{\Psi_a\}_{a\in A}, T, r, \gamma\rangle$, where $S \subseteq \mathbb{R}^n$ is the continuous state space, A is a finite set of symbolic actions. Each action $a \in A$ is associated with a continuous parameter space $\Psi_a \subseteq \mathbb{R}^{m_a}$. Hence, the hybrid parametrized action space is the set $\mathcal{A} = \{(a, \psi) | a \in A, \psi \in \Psi_a\}$. T is the transition function $T = P(s'|s, a, \psi)$ that describes the probability of transitioning into state $s' \in S$ given state $s \in S$, action $a \in A$ and a parameter $\psi \in \Psi_a$. r is the reward function $r : S \times \mathcal{A} \to \mathbb{R}$ that returns the scalar reward when transitioning from s to s'. $\gamma \in [0, 1)$ is a discount factor.

A policy is a function $\pi : S \to \mathcal{A}$. Executing π on a PAMDP leads to a trajectory $(s, (a, \psi), r, d, s', (a', \psi'), r', d'...)$, where $\pi(s) = (a, \psi)$, $s' \sim T(\cdot|s, a, \psi)$, $r = r(s, a, \psi)$, and $d \in \{0, 1\}$ indicates whether an episode is done, $d = 1$, or not $d = 0$. The performance of a policy is measured by its expected discounted return

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, \psi_t) \right].$$

Symbolic Abstraction. In many domains there is incomplete symbolic domain knowledge available, e.g. in the form of rules, safety constraints, or expert heuristics. This knowledge typically excludes unsafe or infeasible decisions, but does not specify optimal behavior. A simple and expressive way of representing such domain knowledge is in the form of a symbolic knowledge base $\mathcal{K}$. However, $\mathcal{K}$ operates in a symbolic state space, while π operates in a numerical state space. Hence, both state space representations must be connected. One way to achieve this is by using an abstraction function α (Veronese et al., 2026). Therefore, we assume a fixed finite set $\mathcal{F}$ of propositional

atoms that describe features of S. α maps the numeric PAMDP states into a symbolic representation $\alpha : S \to 2^{\mathcal{F}}$, either via numeric threshold tests over s, or via a learned function (Ciatto et al., 2024). This allows to obtain the symbolic state representation $\alpha(s)$ as the set of propositional feature atoms that hold true in s.

As an example consider an industrial tank system. The state s contains a feature tank level $\texttt{level}(s) \in [0, 42]$. α may be defined by threshold tests, like $\texttt{tank_near_full} \in \alpha(s) \iff \texttt{level}(s) \geq 41$ or $\texttt{tank_near_empty} \in \alpha(s) \iff \texttt{level}(s) \leq 1$, where $\texttt{tank_near_full}, \texttt{tank_near_empty} \in \mathcal{F}$. Given a state s, we could obtain the symbolic state representation by applying α to s. E.g., $\texttt{level}(s) = 41.25$ would yield $\alpha(s) = \{\texttt{tank_near_full}\}$, while $\texttt{level}(s) = 0.25$ yields $\alpha(s) = \{\texttt{tank_near_empty}\}$.

Knowledge Base and Evaluation. A lightweight, yet powerful way of expressing symbolic domain knowledge is stratified Datalog. A Datalog knowledge base consists of a finite set of rules with negation-as-failure in the form of $h \leftarrow b_1, \ldots, b_x, \texttt{not } b_{x+1}, \ldots, \texttt{not } b_z$, where h and all b_i are atoms over a fixed predicate vocabulary, and $\texttt{not}$ denotes negation-as-failure. To allow for deriving applicable actions and active constraints, we introduce static declarations of actions and constraints into $\mathcal{K}$. When A is the finite set of symbolic actions from the PAMDP and C is the finite set of parameter constraint identifiers, we can denote the static declarations $Act := \{\texttt{action}(a) | a \in A\}$ and $Con := \{\texttt{constr}(c) | c \in C\}$. As $\mathcal{K}$ is stratified, $\mathcal{K} \cup \alpha(s) \cup Act \cup Con$ has a unique intended model for every state s. Hence, we can write $\mathcal{K} \cup \alpha(s) \cup Act \cup Con \models q$, to denote that the atom q is entailed in state s.

Applicable Actions. Evaluating $\mathcal{K}$ with a symbolic state abstraction $\alpha(s)$ allows us to obtain the set of actions that are applicable in state s. Therefore, we use the predicates $\texttt{app}(a)$ to mark applicability of actions and $\texttt{napp}(a)$ to mark non-applicability of actions. The set of applicable actions can be defined as $\bar{A}(s) = \{a \in A | \mathcal{K} \cup \alpha(s) \cup Act \cup Con \models \texttt{app}(a)\}$. As the knowledge in $\mathcal{K}$ is assumed to be incomplete, we adopt a default applicability for actions, resembling a closed-world assumption regarding the discrete action space. This means, an action a is considered applicable unless the predicate $\texttt{napp}(a)$ (non-applicable) can be derived, enforced by an additional rule $\texttt{app}(a) \leftarrow \texttt{action}(a), \texttt{not napp}(a)$.

Feasible Parameters. Evaluating $\mathcal{K}$ with $\alpha(s)$ can additionally determine which parameter constraints are active for a discrete action a in state s. Therefore, we introduce the predicates $\texttt{active}(a, c)$ to denote that the constraint identifier $c \in C$ is active for $a \in A$, and define $C_{\mathcal{K}}(s, a) = \{c \in C \mid \mathcal{K} \cup \alpha(s) \cup Act \cup Con \models \texttt{active}(a, c)\}$. Each constraint identifier c is grounded by a numeric feasibility condition for a in s, inducing the feasible subset $\Psi_{a,c}(s) \subseteq \Psi_a$. The feasible parameter region for a in s is then $\bar{\Psi}_a(s) = \bigcap_{c \in C_{\mathcal{K}}(s,a)} \Psi_{a,c}(s)$, where $\bigcap_{c \in \emptyset} \Psi_{a,c}(s) = \Psi_a$.

Together, $\bar{A}(s)$ and $\{\bar{\Psi}_a(s)\}_{a \in A}$ characterize the decision space that remains feasible for the RL agent after incorporating the symbolic domain knowledge.

Knowledge-Constrained PAMDPs. Building on the above, we can define the underlying formalism of this paper as Knowledge-Constrained PAMDP. As we can obtain the set of feasible parametrized actions in a state s as $\bar{\mathcal{A}}_{\mathcal{K}}(s) = \{(a, \psi) | a \in \bar{A}(s), \psi \in \bar{\Psi}_a(s)\}$, we define a knowledge-constrained PAMDP as a tuple $\langle S, A, \{\Psi_a\}_{a \in A}, T, r, \gamma, \alpha, \mathcal{K} \rangle$.

A policy π is $\mathcal{K}$-feasible if it only selects decisions that are admissible by the knowledge base $\forall s \in S : \pi(s) \in \bar{\mathcal{A}}_{\mathcal{K}}(s)$. Let $\Pi_{\mathcal{K}}$ denote the set of all $\mathcal{K}$-feasible policies, then an RL problem in a knowledge-constrained PAMDP lies in finding a policy $\pi^\star \in \Pi_{\mathcal{K}}$ that maximizes the return $\pi^\star \in \arg\max_{\pi \in \Pi_{\mathcal{K}}} J(\pi)$.

4 SOLUTION

To solve decision making in knowledge-constrained PAMDPs, we present our novel Neuro-Symbolic Knowledge- and Gradient-Guided Reinforcement Learning (KGRL) algorithm (cf. Figure 1). KGRL is based a Deep Q-Network (DQN) (Mnih et al., 2015), which approximates the action-value function of a domain in a Neural Network Q_θ. The key idea of KGRL is to use symbolic domain knowledge from a Datalog knowledge base to prune non-applicable actions from the decision space and restrict parameters to feasible regions. Therefore, we extend the DQN with two additional mechanisms: (i) A Datalog knowledge base $\mathcal{K}$ that can derive the set of applicable actions and feasible parameter regions for every state, (ii) PARAMOPT, a gradient-guided parameter refinement loop that uses the gradient of Q_θ to estimate optimal parameters within the feasible parameter regions.

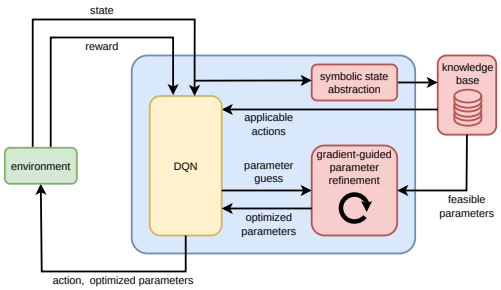


Figure 1: We propose the novel Neuro-Symbolic Knowledge- and Gradient-Guided Reinforcement Learning (KGRL) algorithm for PAMDPs (novel additions marked in red). KGRL evaluates a knowledge base and a symbolic state abstraction at every decision to exclude inapplicable discrete actions, and uses a gradient-based parameter optimization to refine an initial suboptimal parameter guess into an optimal parameter estimate.

At each step, KGRL evaluates $\mathcal{K}$ and a symbolic state abstraction $\alpha(s)$ to derive the set of applicable actions $\bar{A}(s)$ and feasible parameters $\{\Psi_a\}_{a\in\bar{A}(s)}(s)$. This allows KGRL to prune non-applicable actions. Subsequently, PARAMOPT uses the gradient information of Q_θ to perform a gradient-guided parameter refinement as an optimization over the feasible parameter spaces of the available actions. At the end of each step, KGRL greedily selects the action-parameter tuple with the highest action-value.

The step-wise evaluation of $\mathcal{K}$ has the additional side-effect that a trace of local, procedural explanations on which actions and parameters were excluded from the decision space is logged along each rollout. Depending on the coverage of $\mathcal{K}$ this allows to explain the behavior of the agent.

4.1 KNOWLEDGE-GUIDANCE

KGRL uses $\mathcal{K}$ to restrict the decision space of the DQN agent. At each state s, $\mathcal{K}$ is evaluated together with $\alpha(s)$ to obtain the set of applicable actions $\bar{A}(s)$ and feasible parameter regions $\bar{\Psi}_a(s)$ as action specific parameter boxes (cf. Algorithm 1, line 3). By encoding a default action applicabilty in $\mathcal{K}$, KGRL can operate operate on incomplete knowledge, as actions remain available unless excluded by a rule from $\mathcal{K}$, and parameters retain their original bounds unless an active constraint restricts them.

During exploration, KGRL samples an applicable action and corresponding parameters uniformly from $\bar{A}(s)$ and $\bar{\Psi}_a(s)$ (cf. Algorithm 1, line 5). During exploitation, KGRL uses PARAMOPT (cf. Algorithm 2) to estimate the parameters for all actions (cf. Algorithm 1, line 7-8), and greedily picks the action parameter tuple with the highest Q value.

Knowledge also guides the learning by restricting the successor decisions used to compute the DQN targets. The executed transitions $(s_i, a_i, \psi_i, r_i, d_i', s_i')$ are stored in the replay buffer $\mathcal{D}$. For each sampled transition, KGRL evaluates $\mathcal{K}$ at s_i' and refines candidate parameters $\tilde{\psi}_{a'}$ using PARAMOPT. The target for updating Q_θ is computed as

$$y_i \leftarrow r_i + \gamma(1 - d_i') \max_{a' \in \bar{A}(s_i')} Q_{\theta^-}(s_i', a', \tilde{\psi}_{a'}'), \tag{1}$$

where $d_i' = 1$ denotes termination or truncation. If all actions are excluded from the set of applicable actions, the implementation resets without storing a transition.

Algorithm 1: KGRL training

Input: $\mathcal{E}, \mathcal{K}, \alpha$;
Initialize $Q_\theta, Q_{\theta^-}, \mathcal{D}$; reset s
while *in training budget* **do**
  Derive $\bar{A}(s), \bar{\Psi}(s)$ from $\mathcal{K}, \alpha(s)$
  if $\text{Uniform}(0,1) < \epsilon$ **then**
    $a \sim \bar{A}(s)$; $\psi \sim \bar{\Psi}_a(s)$ (uniform)
  else
    for $a \in \bar{A}(s)$ **do**
      $\tilde{\psi}_a \leftarrow \textsc{ParamOpt}(s, a)$
    $a \leftarrow \arg\max_{a \in \bar{A}(s)} Q_\theta(s, a, \tilde{\psi}_a)$
    $\psi \leftarrow \tilde{\psi}_a$
  $(s', r, d') \leftarrow \mathcal{E}.\text{step}(a, \psi)$; store in $\mathcal{D}$
  if *update due* **then**
    Sample minibatch B from $\mathcal{D}$
    foreach $(s_i, a_i, \psi_i, r_i, d'_i, s'_i) \in B$ **do**
      Derive $\bar{A}(s'_i), \bar{\Psi}(s'_i)$ from $\mathcal{K}, \alpha(s'_i)$
      foreach $a' \in \bar{A}(s'_i)$ **do**
        $\tilde{\psi}'_{a'} \leftarrow \textsc{ParamOpt}(s'_i, a')$
      $y_i \leftarrow r_i + \gamma(1 - d'_i) \max_{a' \in \bar{A}(s'_i)} Q_{\theta^-}(s'_i, a', \tilde{\psi}'_{a'})$
    Update θ by gradient descent on $\frac{1}{|B|} \sum_{i \in B} \big(Q_\theta(s_i, a_i, \psi_i) - y_i\big)^2$
  Synchronize $\theta^- \leftarrow \theta$ when due
  $s \leftarrow s'$; reset if $d' = 1$

Algorithm 2: ParamOpt

Input: Q_θ, s, a; bounds ℓ, u, mask M; step ζ, threshold ξ, budget U
1 $\hat{\psi} \leftarrow \text{init}$
2 Hold network weights θ fixed
3 **for** $j \leftarrow 1$ **to** U **do**
4 $g \leftarrow \nabla_{\hat{\psi}} Q_\theta(s, a, \hat{\psi})$
5 Scale g to $\tilde{g}$ using Eq. 2
6 $\hat{\psi}^+ \leftarrow M \odot \text{clip}_{[\ell,u]}(\hat{\psi} + \zeta\tilde{g})$
7 $\delta \leftarrow \|\hat{\psi}^+ - \hat{\psi}\|_2$
8 $\hat{\psi} \leftarrow \hat{\psi}^+$
9 **if** $\delta < \xi$ **then break**
10 **end**
11 **return** $\tilde{\psi}^\star \leftarrow \hat{\psi}$

4.2 Gradient-Guidance

To estimate optimal parameters for an action, we propose the ParamOpt algorithm, a nested refinement loop that leverages on autodifferentiation and the gradient of Q_θ (cf. Algorithm 2). ParamOpt starts with an initial parameter guess $\hat{\psi}$, which can be zero, the midpoint of the parameter box, or a heuristic derived from $\mathcal{K}$ (cf. Algorithm 2, line 1). Using autodifferentiation, ParamOpt calculates the gradient $g = \nabla_{\hat{\psi}} Q_\theta(s, a, \hat{\psi})$ with respect to $\hat{\psi}$ (cf. Algorithm 2, line 4). To account for the parameter bounds ℓ, u derived from $\mathcal{K}$, ParamOpt scales each gradient component by the remaining distance to the bound in its ascent direction (Hausknecht & Stone, 2016):

$$\tilde{g}_j = M_j g_j \begin{cases} \text{clip}_{[0,1]}\left(\dfrac{u_j - \hat{\psi}_j}{\max(u_j - \ell_j, 10^{-6})}\right), & g_j > 0, \\ \text{clip}_{[0,1]}\left(\dfrac{\hat{\psi}_j - \ell_j}{\max(u_j - \ell_j, 10^{-6})}\right), & g_j \le 0, \end{cases} \tag{2}$$

where M_j is a mask that indicates if parameter dimension j is being refined (cf. Algorithm 2, line 5). Using projected gradient-ascent with step size ζ updates $\hat{\psi}$ along the gradient of Q_θ

$$\hat{\psi}^+ = M \odot \text{clip}_{[\ell,u]}\big(\hat{\psi} + \zeta\tilde{g}\big).$$

Clipping keeps the active parameters within their feasible bounds, while the mask M sets inactive dimensions to zero (cf. Algorithm 2, line 6).

The gradient-guided refinement of ParamOpt stops when either the maximum refinement budget U is used or the parameter displacement $\|\hat{\psi}^+ - \hat{\psi}\|_2$ falls below the threshold ξ (cf. Algorithm 2, line 7,9). The resulting estimate is a $\mathcal{K}$-feasible parameter candidate. However, since Q_θ typically is non-convex, gradient ascent does not guarantee a global optimum. We therefore denote the returned estimate by $\tilde{\psi}^\star$.

4.3 Explanation Traces

Since KGRL evaluates $\mathcal{K}$ together with $\alpha(s)$ at every decision step, it can record the supporting derivations as an explanation trace along the trajectory. At each time step, the trace contains the symbolic facts from $\alpha(s)$ and the rules in $\mathcal{K}$ that derive $\texttt{app}(a)$, $\texttt{napp}(a)$, and $\texttt{active}(a, c)$. This makes explicit why actions were pruned or admitted and why the parameter search was restricted to a particular feasible region. The resulting trace constitutes a local procedural explanation (Krarup et al., 2019), as it explains parts of the concrete decision process at each state rather than the policy as a whole.

5 Evaluation

In this section, we present the empirical evaluation of our KGRL algorithm. Our experiments address three research questions: (RQ1) Does integrating domain knowledge from a symbolic knowledge base into the training and deployment of an RL agent in a parametrized action space increase sample efficiency and performance? (RQ2) Can a gradient-guided parameter refinement increase the sample efficiency and performance of an RL agent in a parametrized action space, compared to one-shot estimators? (RQ3) Which explanations can be derived from deploying KGRL in a parametrized action space?

We evaluate our KGRL algorithm against five baseline RL algorithms for PAMDPs from literature: QPAMDP (Masson et al., 2016), PA-DDPG (Hausknecht & Stone, 2016), P-DQN (Xiong et al., 2018), MP-DQN (Bester et al., 2019), and HyAR (Li et al., 2022), following the experimental design guidelines for empirical Machine Learning research by (Vranješ et al., 2024). A detailed description of the experimental setup can be found in Appendix A. As evaluation domains we use the *HardMoveX*, *CatchPoint*, *HardGoal*, and *Platform* domains (Masson et al., 2016; Li et al., 2022). All domains have parametrized action spaces that consist of a finite set of symbolic actions and associated numerical parameters. For each domain we crafted a set of simple rules to utilize in KGRL. A detailed description of the domains and their knowledge bases can be found in Appendix B.

5.1 KGRL vs. Baselines

In the first experiment, we evaluated the normalized Area Under the Learning Curve (AULC) of our KGRL algorithm against the baseline algorithms. We chose the normalized AULC as it combines algorithmic performance and convergence speed in a single metric. The results are reported in the upper part of Table 1 and Figure 2. We could show that across all evaluation domains, KGRL consistently achieves the highest average AULC across the eight different seeds. Additionally, we could show that KGRL already starts with higher episodic returns and converges quicker to higher return values, with lower variance across the seeds than the baseline algorithms (cf. Figure 2). Both findings support RQ1, that integrating explicit but incomplete knowledge into the training and application of an RL agent improves learning efficiency and performance.

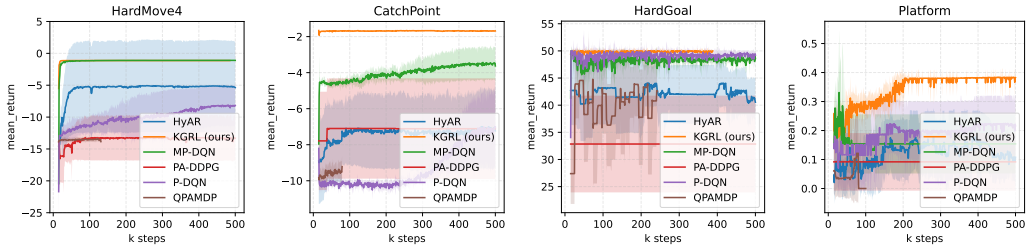


Figure 2: The Figure shows the learning curves of KGRL and the baselines on the *HardMove4, CatchPoint, HardGoal*, and *Platform* evaluation domains. Environment steps are on the x-axis, episodic return on the y-axis. Shaded areas are one standard deviation across eight different seeds on the evaluation tasks. Steeper slopes indicate faster learning and higher values indicate better learned policies.

Additionally, we evaluated ablations of KGRL, to estimate the influence of the knowledge and gradient guidance. Therefore, we removed the knowledge base (nokb) or replaced PARAMOPT algorithm with a mean estimate of the parameter range (nogg). We ablated both components either during training (train), or during evaluation (eval), or during both (train eval). The results are reported in the lower part of Table 1. Ablating knowledge- and gradient-guidance showed mixed influence on the AULC and a strong dependence on the evaluation domain. While ablating the knowledge base during evaluation showed a consistently worse AULC (supporting RQ1). Ablating the gradient-guidance during training led to better results in the *HardMove* domain and comparable results to the full version of KGRL in the other domains pointing toward a mixed support of RQ2. However, removing knowledge- and gradient-guidance during training and evaluation from KGRL, rendered consistently worse results than keeping them, except for the *Platform* domain, which generally showed little variance across the ablations, which is majorly connected to its limited decision space (cf. Appendix B). Additional results and ablations can be found in Appendix C.

Table 1: Normalized AULC (the higher, the better) of our KGRL algorithm and its ablations against the baseline algorithms. We compare ablations of KGRL using no gradient-based parameter refinement (nogg), no knowledge-guidance (nokb), or neither (nogg nokb) during training (train) or deployment (eval).

Algorithm	HardMove4	CatchPoint	HardGoal	Platform
QPAMDP	-13.63 $\pm$ 0.30	-9.67 $\pm$ 0.29	39.19 $\pm$ 2.34	0.03 $\pm$ 0.05
PA-DDPG	-14.12 $\pm$ 2.49	-7.40 $\pm$ 2.48	32.86 $\pm$ 8.86	0.09 $\pm$ 0.09
P-DQN	-12.84 $\pm$ 1.24	-9.79 $\pm$ 0.34	48.17 $\pm$ 0.48	0.13 $\pm$ 0.07
HyAR	-7.49 $\pm$ 5.66	-8.19 $\pm$ 1.77	42.04 $\pm$ 5.30	0.08 $\pm$ 0.04
MP-DQN	-3.48 $\pm$ 0.48	-5.23 $\pm$ 0.31	47.62 $\pm$ 0.64	0.19 $\pm$ 0.06
KGRL	**-1.64 $\pm$ 0.21**	**-1.69 $\pm$ 0.01**	**49.76 $\pm$ 0.07**	**0.22 $\pm$ 0.02**
KGRL (train nogg)	**-1.43 $\pm$ 0.13**	-1.69 $\pm$ 0.00	49.74 $\pm$ 0.07	0.21 $\pm$ 0.02
KGRL (train nokb)	-1.84 $\pm$ 0.17	-1.75 $\pm$ 0.02	**49.89 $\pm$ 0.06**	0.21 $\pm$ 0.02
KGRL (train nogg nokb)	-1.67 $\pm$ 0.19	-1.74 $\pm$ 0.02	49.86 $\pm$ 0.05	**0.23 $\pm$ 0.02**
KGRL (eval nogg)	-1.68 $\pm$ 0.17	**-1.66 $\pm$ 0.00**	49.72 $\pm$ 0.06	0.21 $\pm$ 0.03
KGRL (eval nokb)	-4.33 $\pm$ 0.43	-6.28 $\pm$ 0.42	34.00 $\pm$ 1.22	0.19 $\pm$ 0.02
KGRL (eval nogg nokb)	-4.81 $\pm$ 0.36	-6.34 $\pm$ 0.44	34.67 $\pm$ 1.85	0.20 $\pm$ 0.03
KGRL (train eval nogg)	-1.74 $\pm$ 0.22	**-1.66 $\pm$ 0.00**	49.66 $\pm$ 0.05	0.22 $\pm$ 0.03
KGRL (train eval nokb)	-4.19 $\pm$ 0.40	-5.49 $\pm$ 0.35	43.23 $\pm$ 0.86	0.22 $\pm$ 0.03
KGRL (train eval nogg nokb)	-5.21 $\pm$ 0.43	-6.22 $\pm$ 0.40	42.55 $\pm$ 1.50	**0.23 $\pm$ 0.02**

5.2 KNOWLEDGE BASE UTILIZATION AND EXPLANATION TRACES

In addition to the algorithm performance, we also quantified how strongly KGRL utilizes the knowledge base during application, to evaluate whether its use is substantial enough to support rule-grounded explanations. Therefore, we evaluated the knowledge-base utilization $\mathcal{K}$-utilization describing the ratio of $\mathcal{K}$ evaluations that resulted in pruning rules or parameter constraints, and the ratio of discrete actions that were pruned from A through $\mathcal{K}$ (cf. Equation 4).

Table 2 shows $\mathcal{K}$-utilization and A-pruning for our KGRL algorithm. Across all domains, we could show that for the given knowledge-bases the $\mathcal{K}$-utilization is high, indicating that $\mathcal{K}$ contributes frequently to the decision-making of KGRL. Especially in the *HardMove* domain, $\mathcal{K}$ caused pruning almost half of the actions, which leads to a large simplification of the decision space (an ablation of the *HardMove* domain with increasingly larger action spaces can be found in the Appendix C.1). While A-pruning at *CatchPoint* and *HardGoal* ranged around a

Table 2: Mean $\mathcal{K}$-utilization $\pm$ one standard deviation and the mean A-pruning $\pm$ one standard deviation across eight seeds.

domain	rules	$\mathcal{K}$-utilization	$\lvert A \rvert$	A-pruning
HardMove4	48	1.00 $\pm$ 0.00	8	0.49 $\pm$ 0.00
CatchPoint	7	0.64 $\pm$ 0.08	2	0.32 $\pm$ 0.04
HardGoal	70	1.00 $\pm$ 0.00	10	0.34 $\pm$ 0.00
Platform	15	0.49 $\pm$ 0.01	3	0.00 $\pm$ 0.00

third, we could show that in the *Platform* domain mainly parameter constraints were deduced from $\mathcal{K}$.

As we can record individual rule activations in $\mathcal{K}$, e.g., $\texttt{napp}(a)$ or $\texttt{active}(a, c)$, we can construct step-wise explanation traces which explain why certain actions were excluded and how feasible parameter regions were shaped along the agent's trajectory. This directly supports our RQ3 by showing that the information which is required for rule-grounded explanations is constructed at decision-time. Figure 3 shows an exemplary excerpt from the explanation trace of the *HardGoal* domain.

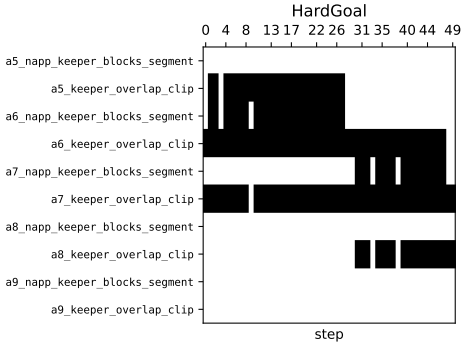


Figure 3: The figure shows an exemplary excerpt from the *HardGoal* domain's explanation trace. During different phases of the rollout, different $\texttt{napp}(a)$ rules fired due to the goalkeeper blocking them for the agent.

6 Discussion

Knowledge Base. KGRL demonstrates that the knowledge-guidance improves sample efficiency and performance. However, it depends on *when* the knowledge is applied. Removing knowledge during evaluation reduces the AULC across all evaluation domains, whereas removing it during training has smaller effects and even improves the HardGoal result. Additionally, the hard integration of knowledge in the form of action pruning makes KGRL sensitive to incorrect rules in $\mathcal{K}$. Adaptive rule repair (Feng et al., 2025) or a soft integration (Veronese et al., 2026) could mitigate this risk.

ParamOpt. We could show that the gradient-guidance has a smaller, task-dependent effect on the agent. This could stem from unreliable gradients in early training, when the action-value function is not fitted. Refinement also adds computation, and projected ascent on a non-convex network cannot guarantee a global optimum. Scheduling the integration of gradient-guidance in the training and deployment, as well as multiple initial guesses could improve the gradient-guidedance.

Explanation Traces. Rule activations explain which actions were excluded and how parameter regions were constrained. Knowledge-base utilization shows that these restrictions contribute to decisions, but does not measure explanation quality. The traces provide local procedural explanations (Krarup et al., 2019). However, they do not explain the DQN's ranking of the remaining choices. Aggregating recurring rule activations into broader explanations, e.g. by using LLMs is part of future work.

Learning Knowledge Bases and Abstraction Functions. Our experiments use handcrafted rules and handcrafted abstraction functions. Learning these components can reduce the effort of instantiating a KGRL agent. Examples from ILP (Mazzi et al., 2023; Meli et al., 2024) show that learning rule bases is an active complementary research field. However in this work, we were especially focussed on integrating incomplete domain knowledge from domain experts. Integrating the approaches above remains part of future work.

7 Conclusion

In this paper, we introduced KGRL for reinforcement learning in PAMDPs with incomplete symbolic domain knowledge. KGRL adds a knowledge- and gradient-guidance to DQN in order to apply it in parametrized action spaces. The knowledge guidance operates on a Datalog knowledge base of incomplete domain knowledge and the gradient-guidance leverages on the differentiability of the action value function in order to find optimal parameters. KGRL outperforms all baseline algorithms on the employed PAMDP benchmarks. However, we also found that is important *when* to employ the knowledge- and gradient guidance. Future work will investigate adaptive refinement schedules, learned rules and abstraction functions, and the aggregation and evaluation of explanation traces.

AI USE STATEMENT

In this work, we used generative AI tools for refining hypotheses, providing feedback on research methodology and experiments, and assisting in implementing methods. We have not used generative AI tools helping in developing the conceptual framework, formulating mathematical claims, interpreting results, and for providing critical ingredients for mathematical claims, while tasks such as generating synthetic datasets, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, and assisting in writing proofs are not relevant tasks for this work. Additionally, we used generative AI tools for editing and debugging code, brainstorming about ideas, editing the research paper to improve readability, and checking if we did not miss out relevant literature. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work complies with the ICLR Code of Ethics. All experiments were conducted in simulated environments. Hence, no human participants, sensitive data, or real-world deployments were involved, eliminating ethical risks associated with such settings

REPRODUCIBILITY STATEMENT

The algorithm, experimental setup and domains for reproducing the results are described in the paper in Sections 3, 4, 5, as well as in Appendices A, and B. The source code for replicating the results is included in the supplementary material. Upon acceptance, we will make the complete source code publicly available on GitHub.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 22–31. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/achiam17a.html.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018. ISSN 2159-5399. doi: 10.1609/aaai.v32i1.11797. URL http://dx.doi.org/10.1609/aaai.v32i1.11797.

Greg Anderson, Abhinav Verma, Isil Dillig, and Swarat Chaudhuri. Neurosymbolic reinforcement learning with formally verified exploration. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6172–6183. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/448d5eda79895153938a8431919f4c9f-Paper.pdf.

Craig J. Bester, Steven D. James, and George D. Konidaris. Multi-pass q-networks for deep reinforcement learning with parameterised action spaces. *arXiv preprint arXiv:1905.04388*, 2019. doi: 10.48550/ARXIV.1905.04388. URL https://arxiv.org/abs/1905.04388.

Giovanni Ciatto, Federico Sabbatini, Andrea Agiollo, Matteo Magnini, and Andrea Omicini. Symbolic knowledge extraction and injection with sub-symbolic predictors: A systematic literature review. *ACM Computing Surveys*, 56(6):1–35, March 2024. ISSN 1557-7341. doi: 10.1145/3645103. URL http://dx.doi.org/10.1145/3645103.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018. URL https://arxiv.org/abs/1801.08757.

Yaqi Duan and Martin J Wainwright. Taming "data-hungry" reinforcement learning? stability in continuous state-action spaces. In A. Globerson, L. Mackey, D. Belgrave, A. Fan,

U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 71352–71380. Curran Associates, Inc., 2024. doi: 10.52202/079017-2279. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/8364ad56291e22ec6041794355a46802-Paper-Conference.pdf.

Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, April 2021. ISSN 1573-0565. doi: 10.1007/s10994-021-05961-4. URL http://dx.doi.org/10.1007/s10994-021-05961-4.

Zhou Fan, Rui Su, Weinan Zhang, and Yong Yu. Hybrid actor-critic reinforcement learning in parameterized action space. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 2279–2285. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/316. URL https://doi.org/10.24963/ijcai.2019/316.

Yao Feng, Jun Zhu, André Platzer, and Jonathan Laurent. Adaptive shielding via parametric safety proofs. *Proceedings of the ACM on Programming Languages*, 9:816–843, April 2025. ISSN 2475-1421. doi: 10.1145/3720450. URL http://dx.doi.org/10.1145/3720450.

Saeedeh Ghanadbashi and Fatemeh Golpayegani. Using ontology to guide reinforcement learning agents in unseen situations: A traffic signal control system case study. *Applied Intelligence*, 52 (2):1808–1824, May 2021. ISSN 1573-7497. doi: 10.1007/s10489-021-02449-5. URL http://dx.doi.org/10.1007/s10489-021-02449-5.

Lin Guan, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging approximate symbolic models for reinforcement learning via skill diversity. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7949–7967. PMLR, 2022. URL https://proceedings.mlr.press/v162/guan22c.html.

Shuai Han, Mehdi Dastani, and Shihan Wang. Neuro-symbolic action masking for deep reinforcement learning. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS. International Foundation for Autonomous Agents and Multiagent Systems, 2026. doi: 10.65109/jwph6906. URL http://dx.doi.org/10.65109/JWPH6906.

Matthew J. Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.04143.

René Heesch, Alessandro Cimatti, Jonas Ehrhardt, Alexander Diedrich, and Oliver Niggemann. A lazy approach to neural numerical planning with control parameters. In *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI 2024)*. IOS Press, October 2024. ISBN 9781643685489. doi: 10.3233/faia241000. URL http://dx.doi.org/10.3233/FAIA241000.

Daiki Kimura, Subhajit Chaudhury, Akifumi Wachi, Ryosuke Kohita, Asim Munawar, Michiaki Tatsubori, and Alexander Gray. Reinforcement learning with external knowledge by using logical neural networks. *arXiv preprint arXiv:2103.02363*, 2021. URL https://arxiv.org/abs/2103.02363.

Bettina Könighofer, Florian Lorber, Nils Jansen, and Roderick Bloem. Shield synthesis for reinforcement learning. In *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*, pp. 290–306. Springer International Publishing, 2020. ISBN 9783030613624. doi: 10.1007/978-3-030-61362-4_16. URL https://dx.doi.org/10.1007/978-3-030-61362-4_16.

Bettina Könighofer, Roderick Bloem, Nils Jansen, Sebastian Junges, and Stefan Pranger. Shields for safe reinforcement learning. *Communications of the ACM*, 68(11):80–90, October 2025. ISSN 1557-7317. doi: 10.1145/3715958. URL https://dx.doi.org/10.1145/3715958.

Benjamin Krarup, Michael Cashmore, Daniele Magazzeni, and Tim Miller. Towards model-based contrastive explanations for explainable planning. In *ICAPS Workshop on Explainable AI Planning (XAIP)*, 2019.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. doi: 10.48550/ARXIV.2005.01643. URL https://dx.doi.org/10.48550/ARXIV.2005.01643.

E.S. Levitin and B.T. Polyak. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50, January 1966. ISSN 0041-5553. doi: 10.1016/0041-5553(66)90114-5. URL http://dx.doi.org/10.1016/0041-5553(66)90114-5.

Boyan Li, Hongyao Tang, Yan Zheng, Jianye Hao, Pengyi Li, Zhen Wang, Zhaopeng Meng, and Li Wang. Hyar: Addressing discrete-continuous action reinforcement learning via hybrid action representation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=64trBbOhdGU.

Jiajing Ling, Moritz Lukas Schuler, Akshat Kumar, and Pradeep Varakantham. Knowledge compilation for constrained combinatorial action spaces in reinforcement learning. In *Proceedings of the 22th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2023.

Warwick Masson, Pravesh Ranchod, and George Konidaris. Reinforcement learning with parameterized actions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), February 2016. ISSN 2159-5399. doi: 10.1609/aaai.v30i1.10226.

Sahisnu Mazumder, Bing Liu, Shuai Wang, Yingxuan Zhu, Xiaotian Yin, Lifeng Liu, and Jian Li. Knowledge-guided exploration in deep reinforcement learning. *arXiv preprint arXiv:2210.15670*, 2022. doi: 10.48550/ARXIV.2210.15670.

Giulio Mazzi, Daniele Meli, Alberto Castellini, and Alessandro Farinelli. Learning logic specifications for soft policy guidance in POMCP. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 373–381, 2023.

Daniele Meli, Alberto Castellini, and Alessandro Farinelli. Learning logic specifications for policy guidance in pomdps: an inductive logic programming approach. *Journal of Artificial Intelligence Research*, 79:725–776, February 2024. ISSN 1076-9757. doi: 10.1613/jair.1.15826.

Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. Explainable reinforcement learning: A survey and comparative review. *ACM Computing Surveys*, 56(7):1–36, April 2024. ISSN 1557-7341. doi: 10.1145/3616864. URL http://dx.doi.org/10.1145/3616864.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023. ISSN 1935-8245. doi: 10.1561/2200000086.

Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana. OptLayer: Practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018. URL https://arxiv.org/abs/1709.07643.

Hikaru Shindo, Quentin Delfosse, Devendra Singh Dhami, and Kristian Kersting. Blendrl: A framework for merging symbolic and neural policy learning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 3615–3646, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/0a443a000e1cb2281480b3bac395b3b8-Paper-Conference.pdf.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, October 2017. ISSN 1476-4687. doi: 10.1038/nature24270. URL http://dx.doi.org/10.1038/nature24270.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, December 2018. ISSN 1095-9203. doi: 10.1126/science.aar6404. URL http://dx.doi.org/10.1126/science.aar6404.

Roland Stolz, Hanna Krasowski, Jakob Thumm, Michael Eichelbeck, Philipp Gassert, and Matthias Althoff. Excluding the irrelevant: Focusing reinforcement learning through continuous action masking. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-3013. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/acf4a08f67724e9d2de34099f57a9c25-Abstract-Conference.html.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, Cambridge, Massachusetts, 2018.

Mirco Theile, Lukas Dirnberger, Raphael Trumpp, Marco Caccamo, and Alberto Sangiovanni-Vincentelli. Action mapping for reinforcement learning in continuous environments with constraints. *Reinforcement Learning Journal*, 6:343–363, 2025. URL https://rlj.cs.umass.edu/2025/papers/Paper27.html.

Celeste Veronese, Alessandro Farinelli, and Daniele Meli. Sample-efficient neurosymbolic deep reinforcement learning. *arXiv preprint arXiv:2601.02850*, 2026. doi: 10.48550/ARXIV.2601.02850.

Daniel Vranješ, Jonas Ehrhardt, René Heesch, Lukas Moddemann, Henrik Sebastian Steude, and Oliver Niggemann. Design Principles for Falsifiable, Replicable and Reproducible Empirical Machine Learning Research. In Ingo Pill, Avraham Natan, and Franz Wotawa (eds.), *35th International Conference on Principles of Diagnosis and Resilient Systems (DX 2024)*, volume 125 of *Open Access Series in Informatics (OASIcs)*, pp. 7:1–7:13, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-356-0. doi: 10.4230/OASIcs.DX.2024.7. URL https://drops.dagstuhl.de/entities/document/10.4230/OASIcs.DX.2024.7.

Jiechao Xiong, Qing Wang, Zhuoran Yang, Peng Sun, Lei Han, Yang Zheng, Haobo Fu, Tong Zhang, Ji Liu, and Han Liu. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:1810.06394*, 2018. doi: 10.48550/ARXIV.1810.06394.

Yang Yu. Towards sample efficient reinforcement learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, IJCAI-2018, pp. 5739–5743. International Joint Conferences on Artificial Intelligence Organization, July 2018. doi: 10.24963/ijcai.2018/820. URL http://dx.doi.org/10.24963/ijcai.2018/820.

Ao Zhou, Jiayi Guan, Li Shen, Fan Lu, Sanqing Qu, Junqiao Zhao, Ziqiao Wang, Ya Wu, and Guang Chen. CHPO: Constrained hybrid-action policy optimization for reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=KMaBmPHkVj.

Jie Zhu, Fengge Wu, and Junsuo Zhao. An overview of the action space for deep reinforcement learning. In *2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*, ACAI'21. ACM, December 2021. doi: 10.1145/3508546.3508598.

APPENDIX

A EXPERIMENTAL SETUP

For creating our experimental setup, we followed the experimental design guidelines for empirical Machine Learning research by Vranješ et al. (2024). Therefore, we compared the performance of our KGRL algorithm against the baseline algorithms on the evaluation domains. We trained all algorithms with a fixed budget of 500,000 training steps and a maximum computation wall time of 12 hours. To rule out lucky initialization, we repeated the trainings on eight different seeds. On checkpoints at every 1,000 steps, we performed an evaluation run on 16 episodes with environment different environemnt seeds as the training seeds. The code for reproducing the results, including the knowledge bases, can be found under `in-supplementary-materials-now--released-upon-acceptance`.

As main performance metric, we used the area under the learning curve (AULC), as it captures the performance but also the pace on how fast an agent adapts to a domain. For checkpoints $t_0 < \cdots < t_m$ of a run i, we computed the AULC as

$$\text{AULC}_i = \frac{1}{t_m - t_0} \sum_{j=1}^{m} (t_j - t_{j-1}) \frac{R_i(t_{j-1}) + R_i(t_j)}{2}, \tag{3}$$

where R_i is the mean return over the 16 evaluation runs. Although the nominal training budget is 500,000 steps, not all baselines could reach this budget within the wallclock time. Especially the Q-PAMDP algorithm ran into timeout for every domain. The reported AULC values hence are computed on a common interval within each domain, ending at the earliest final checkpoint among the compared runs.

We calculated the $\mathcal{K}$-utilization and A-pruning for N decisions and $|A|$ discrete actions as

$$\mathcal{K}\text{−utilization} = \frac{1}{N} \sum_{j=1}^{N} z_j, \qquad A\text{−pruning} = \frac{1}{N|A|} \sum_{j=1}^{N} p_j, \tag{4}$$

where $z_j \in \{0, 1\}$ indicates whether $\mathcal{K}$ derived active rules, and p_j is the number of excluded actions.

B EVALUATION DOMAINS AND KNOWLEDGE-BASES

Below follows a short description of our evaluation domains (cf. Figure 4) and the knowledge bases that we instanciated for every domain. The full domains and knowledge-bases can be found in the supplementary repository. We evaluated our algorithm on the *HardMoveX*, *CatchPoint*, *Platform* , and *HardGoal* domains.

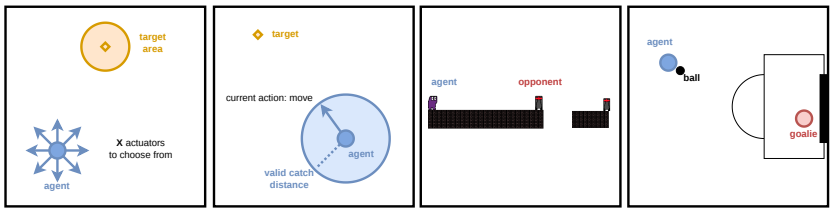


Figure 4: The Figure shows our evaluation domains. We evaluated our algorithm on the (from left to right) *HardMoveX*, *CatchPoint*, *Platform* , and *HardGoal* domains.

HardMoveX. The *HardMoveX* domains describe a 2D navigational task in which an agent must reach a target. The agent has $n = 2X$ discrete actions that point into a specific direction $\texttt{moveX}(\psi)$ and which are parametrized by a magnitude $\psi \in [0, 1]$. We used *HardMoveX* to evaluate the performance on the same domain but with increasingly large decision spaces using *HardMove4, HardMove6, HardMove8* and *HardMove10*, yielding 8, 12, 16, and 20 discrete actions.

The abstraction function uses the following symbolic features:

`dist_small`,
`dist_tiny`,
`in_goal`,
`late_episode`
`away_from_goal_a{i}`.

Action pruning rules are for each action index i:

`a{i}_napp_in_goal`,
`a{i}_napp_away_from_goal`

Parameter constraints are activated for each action index i via

`a{i}_small_step`,
`a{i}_tiny_step`,
`a{i}_late_episode_step`

CatchPoint. The *CatchPoint* describes a 2D reach-and-catch task. The agent can $\texttt{move}(\psi), \psi \in [-1, 1]$ and $\texttt{catch}(\psi), \psi \in [0, 1]$, both parameters are mapped to an angle.

The abstraction function uses the following symbolic features:

`catch_impossible`,
`catch_possible`,
`in_target`.

Action pruning rules are:

`a0_napp_in_target`,
`a1_napp_catch_impossible`

There are no rules for parameter constraints.

Platform. The *Platform* domain is a side-scrolling locomotion domain with three parameterized actions and one continuous parameter per action: $\texttt{run}(\psi), \psi \in [0, 30]$, $\texttt{hop}(\psi), \psi \in [0, 720]$, and $\texttt{leap}(\psi), \psi \in [0, 430]$. The objective is to maximize progress while avoiding enemies and gaps.

The abstraction function uses the following symbolic features:

`enemy_close`,
`enemy_very_close`,
`high_speed`,
`approaching_gap`,
`gap_imminent`,
`hazard_combo`.

Action pruning rules are:

`platform_run_napp_near_gap`,
`platform_run_napp_enemy_very_close`.

Parameter constraints are activated via :

`platform_enemy_close_run`,
`platform_speed_limited_run`,
`platform_emergency_run_cap`,
`platform_enemy_close_hop`,
`platform_gap_hop`,
`platform_gap_imminent_hop`,
`platform_hazard_combo_hop`,
`platform_enemy_close_leap`,
`platform_gap_imminent_leap`,
`platform_hazard_combo_leap`.

HardGoal. The *HardGoal* domain is a simplified soccer benchmark, where the agent aims to shoot a goal against a goalkeeper. Therefore, it can choose from ten shooting actions $\texttt{a0}(\psi)$ to $\texttt{a9}(\psi), \psi \in [-1, 1]$. The goalkeeper is continuously moving. Rules in $\mathcal{K}$ prune the lower half actions, when the goalkeeper is in the left goal corner and upper-half actions, when the goalkeeper is in the right corner, as well as any action that is blocked by the keeper.

The abstraction function uses the following symbolic features:

`keeper_left`,
`keeper_right`,
`keeper_center`,
`ball_fast`,
`keeper_overlap_a{i}`, `keeper_blocks_a{i}`.

Action pruning rules are:

`a{i}_napp_keeper_left` for lower-half shots,
`a{i}_napp_keeper_right` for upper-half shots,
`a{i}_napp_keeper_blocks_segment` for all shots.

Parameter constraints are activated via :

`a{i}_keeper_left_bias`,
`a{i}_keeper_right_bias`,
`a{i}_fast_ball_narrow_offset`,
`a{i}_keeper_center_narrow_offset`,
`a{i}_keeper_overlap_clip`.

Across all domains, we include default applicability rules `default_app_a{i}`, which keep action a_i applicable unless a corresponding `napp` rule is derived.

C Additional Results

C.1 Increasing the HardMove Action Space

To examine the performance with larger decision spaces, we evaluated the algorithms on the HardMove4, HardMove8, and HardMove10 domains, which contain 8, 16, and 20 discrete actions, respectively. The evaluation setup remains equal to the main evaluation. The results are reported in Table 3. Figure 5 shows the learning curves of KGRL and the baselines on the increasing decision spaces.

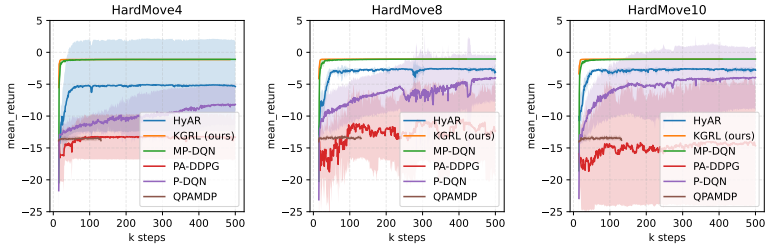


Figure 5: Learning curves of KGRL and the baselines on the *HardMove4, HardMove8*, and *HardMove10* evaluation domains over the full recorded training range. Environment steps are on the x-axis, episodic return on the y-axis. Shaded areas are one standard deviation across the available seeds on the evaluation tasks.

Across evaluation domains, full KGRL maintains an advantage in AULC over the five baselines. Interestingly, it became easier for the algorithms to solve the task in domains with larger decision spaces, also showing a decrease in variance. This can be explained by the redundancy of actions in the growing action spaces. Removing $\mathcal{K}$ during evaluation reduces performance for each domain, whereas removing gradient refinement during training has a task-dependent effect. Hence, the results support the usefulness of knowledge guidance across these action sets.

Table 3: HardMove action-space comparison: AULC reported as mean ± one standard deviation across eight seeds.

Algorithm	HardMove4	HardMove8	HardMove10
QPAMDP	-13.626 ± 0.304	-13.493 ± 0.124	-13.559 ± 0.224
PA-DDPG	-14.119 ± 2.489	-14.651 ± 3.139	-16.093 ± 6.963
P-DQN	-12.844 ± 1.237	-10.400 ± 2.789	-9.774 ± 2.941
HyAR	-7.494 ± 5.662	-5.172 ± 0.406	-5.100 ± 0.407
MP-DQN	-3.478 ± 0.476	-3.492 ± 0.297	-3.294 ± 0.359
KGRL	-1.643 ± 0.212	-1.442 ± 0.133	-1.361 ± 0.061
KGRL (train nogg)	-1.432 ± 0.127	-1.340 ± 0.105	-1.488 ± 0.185
KGRL (train nokb)	-1.842 ± 0.173	-1.563 ± 0.094	-1.621 ± 0.138
KGRL (train nogg nokb)	-1.675 ± 0.190	-1.812 ± 0.208	-1.745 ± 0.143
KGRL (eval nogg)	-1.675 ± 0.171	-1.529 ± 0.149	-1.505 ± 0.127
KGRL (eval nokb)	-4.327 ± 0.433	-4.184 ± 0.349	-4.426 ± 0.557
KGRL (eval nogg nokb)	-4.813 ± 0.364	-5.007 ± 0.459	-5.039 ± 0.278
KGRL (train eval nogg)	-1.738 ± 0.220	-1.534 ± 0.122	-1.599 ± 0.147
KGRL (train eval nokb)	-4.186 ± 0.398	-4.524 ± 0.311	-4.661 ± 0.448
KGRL (train eval nogg nokb)	-5.212 ± 0.428	-5.351 ± 0.411	-4.703 ± 0.297

C.2 Sensitivity to Incomplete and Incorrect Rules

Additionally, we examined the sensitivity of KGRL to pruning rules from $\mathcal{K}$ or adding incorrect, malicious rules, on the HardMove4 domain. The full HardMove4 knowledge base contains 48 rules, including eight default-applicability rules. We modified it to create five variants:

1. The first variant removes the eight `late_episode_step` rules.
2. The second variant additionally removes the eight `tiny_step` rules.
3. The third variant additionally removes the eight `small_step` rules.
4. The fourth variant retains only the eight default-applicability rules.
5. The fifth variant adds an incorrect exclusion rule `hardmove_incorrect_global_napp_a0` to the complete knowledge base, prohibiting action a_0.

Table 4 and Figure 6 show that removing rules does not cause a monotonic decrease in performance. The default-only variant has substantially lower AULC than the full set, but a higher final return.

Table 4: Sensitivity to incomplete and incorrect rule sets on HardMove4. AULC and final return are reported as mean ± one sample standard deviation across eight seeds.

$\mathcal{K}$ variants	rules	AULC	final return
Full rule set	48	-2.07 ± 0.06	-1.79 ± 0.02
Without late-episode rules	40	-2.11 ± 0.11	-1.78 ± 0.01
Also without tiny-step rules	32	-2.14 ± 0.12	-1.77 ± 0.02
Also without small-step rules	24	-2.00 ± 0.12	-1.64 ± 0.01
Default applicability only	8	-3.51 ± 0.21	-1.65 ± 0.02
Full set + incorrect exclusion	49	-2.17 ± 0.15	-1.83 ± 0.01

Retaining the exclusion rules while removing the three parameter-constraint families also yields a higher final return. Thus, guidance that helps during learning can also restrict the performance eventually reached by the policy. The added incorrect exclusion lowers the final return and AULC. This demonstrates sensitivity to this particular incorrect rule. However, the experiment is only performed over a single domain. This explains the weak effect of gradient-guidance, as it showed already in the main evaluation that for particularly this domain gradient-guidance does not have a positive contribution.

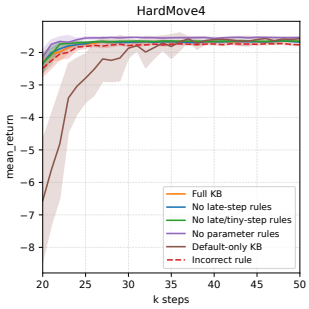


Figure 6: Learning curves of KGRL with incomplete and incorrect knowledge bases on *HardMove4*. Environment steps are noted on the x-axis, episodic return on the y-axis. Shaded areas are one standard deviation across eight seeds on the evaluation tasks.