HRLMS: A Data-driven Hierarchical Reinforcement Learning System for Interactive Rule Intervention and Visualization

Haodi Zhang
Shenzhen University
Shenzhen, China
hdzhang@szu.edu.cn

Xiangyu Zeng
Shenzhen University
Shenzhen, China
zxc962790623@qq.com

Chen Zhang
PolyU
Hong Kong SAR, China
jason-c.zhang@polyu.edu.hk

Yuanfeng Song
WeBank Co., Ltd
Shenzhen, China
songyf@outlook.com

Kaishun Wu
HKUST (Guangzhou)
Guangzhou, China
wuks@hkust-gz.edu.cn

ABSTRACT

In recent years, an increasing number of deep reinforcement learning methods have achieved success in domains such as gaming, yet their inherent black-box nature poses significant challenges to the interpretability of the training process. Moreover, there is an urgent need for the ability to intervene directly and simply during training. To address these issues, we present the Interactive Hierarchical Reinforcement Learning Monitoring System (HRLMS). This framework integrates a set of rules derived from both autonomously generated rules and those input through user interaction, showcasing these rules in real-time during the training process. Throughout the system's operation, the input, integration, display, and reuse of rules form a comprehensive chain, enhancing the entirety of the workflow by seamlessly blending the training and display processes.

PVLDB Reference Format:
Haodi Zhang, Xiangyu Zeng, Chen Zhang, Yuanfeng Song, and Kaishun Wu. HRLMS: A Data-driven Hierarchical Reinforcement Learning System for Interactive Rule Intervention and Visualization. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

1 INTRODUCTION

In the current technological landscape, Reinforcement Learning (RL) has emerged as a significant breakthrough in the realm of artificial intelligence, demonstrating its potential across a wide array of dynamic decision-making scenarios. From gaming to autonomous driving, complex optimization to intelligent robotics, the applications of RL span a diverse set of fields, showcasing its efficacy and versatility [8, 14, 16, 17]. However, despite its success, a primary challenge associated with RL methods is their "black box" nature - a lack of transparency in the decision-making process that constrains our understanding and control over model behavior [3, 7]. In terms of data processing, there is a lot of work on interpretable processing [2, 12, 15], but once combined with a model that does not have interpretable capabilities, it becomes extra difficult to obtain and intervene in its internal processes.

With the advent of interactive reinforcement learning systems, a glimpse into a more controllable and interpretable future of AI has been unveiled. Some of these interactive systems might present training conditions through parameters or visuals. Systems like EasyRL [6] and PantheonRL [13] have made strides in lowering the barriers to entry and enhancing transparency to some extent. EasyRL, for instance, facilitates the training and evaluation of RL agents through an interactive graphical user interface, whereas PantheonRL enhances multi-agent reinforcement learning capabilities with support for dynamic training interactions, including loops, adaptivity, and episodic training. But this still fails to understand the decision-making logic behind reinforcement learning in time. Some interactions are conducted through language dialogue, such as in Interactive Task Learning (ITL) [4], which aims to develop AI agents capable of learning new tasks through online interactions. Rosie [9] engages in bidirectional interactive learning for specific tasks within simulated environments, mainly through natural language instructions, building a significant task hierarchy that includes various task types. This interaction modality is procedural and makes it hard to intervene and adjust the training process on the fly. Other methods engage in interactive reinforcement through direct learning of human preferences for expressions to guide symbolic regression (SR) [1], allowing users to steer exploration towards areas more relevant to them within the symbolic space. Hi-Viscont [5] enables robots to learn new visual concepts and tasks through live language interaction with human users. Plansformer [11] utilizes a fine-tuned language model based on the Transformer architecture to generate symbolic plans. However, the form of rules in these interpretable reinforcement learning systems is complex, and their reusability for explanations is weak.

To address these challenges and limitations, we developed the HRLMS system, which aims to push the boundaries of interactive reinforcement learning through the fusion of hierarchical reinforcement learning models and interactive monitoring systems. A core innovation of the HRLMS system is that it allows users to directly input rules in the form of first-order logic and integrates user-defined rules with rules generated by the model itself. These rules not only reflect the agent's understanding of the environment, but also enable users to guide and adjust the training process based on their own intuition and knowledge. In addition, the HRLMS system

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

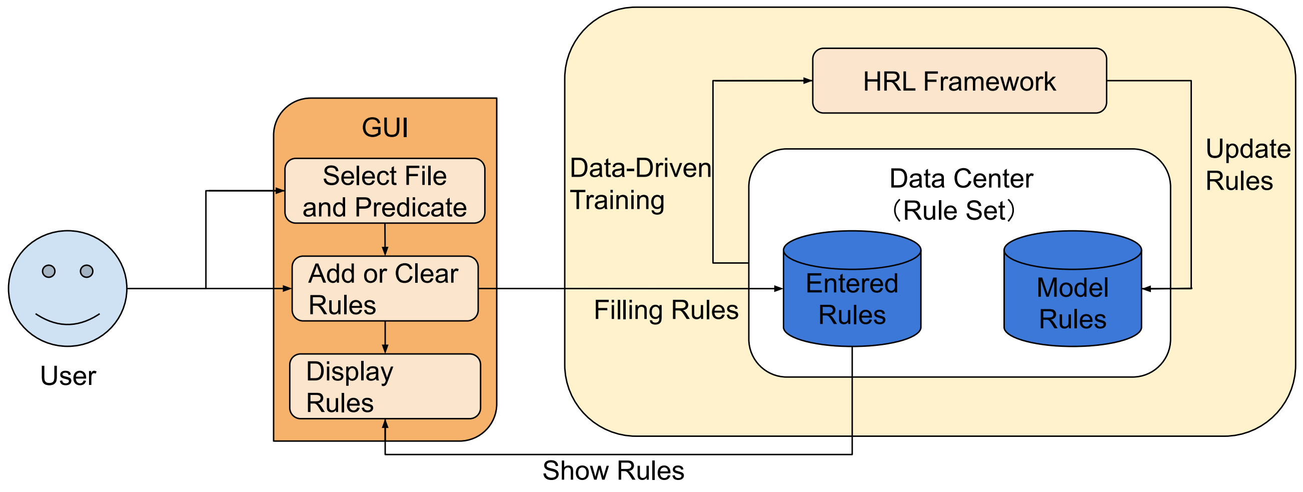


Figure 1: The overall architecture of HRLMS. HRLMS first requires the user to select a configuration file, and then the system loads specific predicates and environment objects. The user needs to select predicates and object construction rules in order to input them into the data center where the rules are stored in HRL. The user can also observe the rules generated by the HRL training process. The rules are input by the user or generated by HRL.

can display these rules in real time. The HRLMS system provides a more transparent window, allowing users to witness and influence the learning process of the model, allowing users to dynamically intervene in the model training process in real time and receive feedback quickly.

2 PRELIMINARIES

We introduce the fundamental concepts necessary for understanding the proposed framework. Our proposal is based on knowledge representation and reasoning (KRR) in form of first-order logic (FOL), and hierarchical reinforcement learning (HRL).

2.1 Knowledge Representation and Reasoning in FOL

The language of FOL consists of three key components: **Entities**, **Predicates**, and **Formulas**. Entities represent constants (e.g., objects), while predicates define relations between entities. An **atom** $\alpha = P(t_1, t_2, \ldots, t_n)$ comprises an n-ary predicate P and n terms $\{t_1, t_2, \ldots, t_n\}$, where each term can be either a constant or a variable. An atom becomes **grounded** when all its terms are constants. A **formula** is a construct formed from atoms, logical connectives, and potential existential or universal quantifiers. A **rule**, also referred to as a **clause**, is expressed as follows:

$$\alpha \leftarrow \alpha_1 \wedge \alpha_2 \cdots \wedge \alpha_n$$

Here, α is the *head* atom, and $\alpha_1, \alpha_2, \ldots, \alpha_n$ are *body* atoms. A clause becomes grounded when all associated atoms are grounded. The head atom is considered true only if all body atoms are true. Leveraging prior knowledge, FOL-based clauses offer exceptional interpretability.

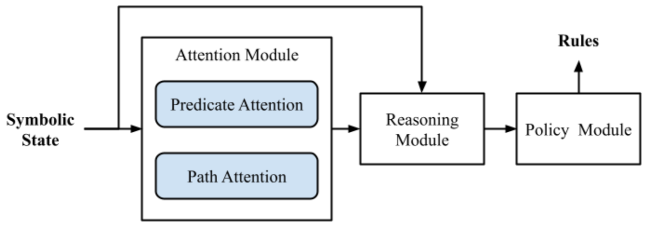


Figure 2: Automated rule generation for HRL

2.2 Hierarchical Reinforcement Learning (HRL)

Hierarchical Reinforcement Learning extends conventional reinforcement learning, proving particularly advantageous in scenarios with sparse rewards. This approach decomposes the primary objective into manageable subtasks. The high-level agent handles subtask allocation, while low-level sub-strategies target the corresponding sub-goals. Upon completing the exploration of each sub-strategy, the overarching task's strategy is achieved. HRL often incorporates abstraction, including state, task, and action abstraction, to distill crucial information from diverse subtasks and enhance differentiation between sub-goals.

3 HRLMS OVERVIEW

Figure 1 illustrates the framework of the HRLMS and the process of interaction and visualization available to users. From the user's perspective, the system's input consists of predicates and objects necessary for generating model rules, which can be entered through a file in JSON format. After loading the predicates and objects from the input file, users can select predicates and objects to form the premises of rules. Upon completing their selection, they can synthesize the rule; the system then combines one or more selected predicates into a complete first-order logic rule. The rules added by

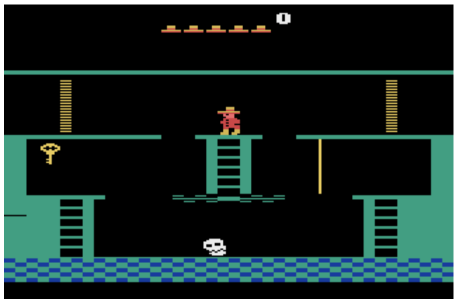
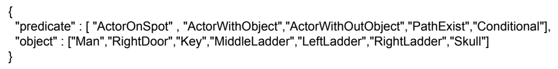


Figure 3: Montezuma's Revenge



```
{
  "predicate" : [ "ActorOnSpot" , "ActorWithObject","ActorWithOutObject","PathExist","Conditional"],
  "object" : ["Man","RightDoor","Key","MiddleLadder","LeftLadder","RightLadder","Skull"]
}
```

Figure 4: Content format of user input file

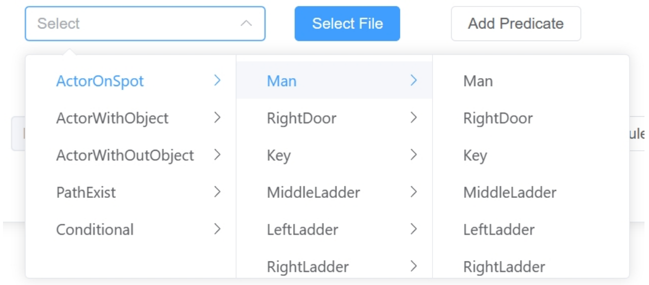
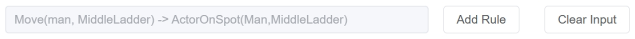


Figure 5: Add file and select predicates



Move(man, MiddleLadder) -> ActorOnSpot(Man,MiddleLadder)

Figure 6: Add rules and clear rules

users are eventually pushed to the model's rule set, where they are combined with the model's own generated rules for inclusion in the next training process.

We have integrated a rule generator in the HRL framework. The rules are generated by the rule generator, and its structure is shown in Figure 2, which mainly consists of attention module, reasoning module and policy module. The attention module employs a hierarchical stack of transformers to generate dynamic attention weights. By utilizing a multi-head dot-product attention module (MHDPA), dynamic weights are computed, enabling the spotlighting of specific elements. The symbolic state is structured as a tensor, undergoing transformation into a matrix to facilitate attention weight generation. This module enhances the model's sensitivity to varying symbolic information. The reasoning process of the reasoning module is similar to multi-hop reasoning, visualized as nodes (objects) and edges (relationships). For the relationship path from x to x':

$$query(x, x') \leftarrow R_1(x, z_1) \wedge R_2(z_1, z_2) \cdots \wedge R_n(z_{n-1}, x') \quad (1)$$

where $R_1 ... R_n$ denotes the predicate $x ... x$ denotes the object. Predicate reasoning is similar to matrix multiplication, where the predicate or relationship P_k is expressed as a binary matrix M_k of $\{0, 1\}^{|\chi| \times |\chi|}$, where χ represents the number of objects. And if there is a certain relationship P_k between x_i and x_j, then the corresponding (i, j) in M_k is 1. These matrices (e.g. M_k) encode graph connectivity information.

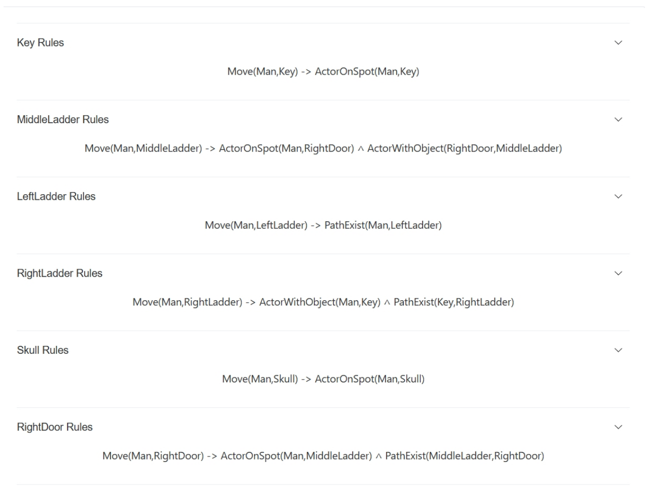
The Policy module drives the introduction of Deep Reinforcement Learning (DRL) policies. The predicates of the last hop are customized as action predicates, and a multi-perceptron layer is applied to the inference output, effectively deriving the desired policy. The rule updating process aims to continually refine and enhance the rule set during the model training process.

The process of integrating and utilizing rules within the hierarchical reinforcement learning (HRL) model during training comprises several steps: First, rules input by users are processed and merged with rules generated by the model itself. This process also filters out any potentially duplicate rules. During the HRL training process, the HRL model initially generates a sub-goal and then performs rule matching to determine if the sub-goal needs to be changed. If a match with the rules is found, the sub-goal is modified accordingly. Finally, interaction training is conducted based on the selected sub-goal and the training environment. The success rate of the sub-goal is then updated based on the completion status of the sub-goal. Users can introduce new rules at any moment during training, but it's essential to understand that the rules generated by the model itself during the HRL (Hierarchical Reinforcement Learning) training process will automatically update. The frequency at which the model-generated rules are overwritten is determined by the completion rates of sub-goals. Each sub-goal has a completion rate, which is the ratio of the number of times that sub-goal was achieved in its last hundred selections. If the success rate of a sub-goal changes by ten percent (or more) compared to its completion rate at the time of the last rule update, it suggests a significant shift in the model's understanding of the environment, prompting an update and overwrite of the rule set. At this juncture, users can view the latest developments through the interactive interface, keeping them informed and engaged with the training process's current state.

In addition to integrating rules into the training process, these rules can also be displayed within an interactive GUI. This display includes not just user inputs but also the content of rules generated by the model itself. The presentation is organized by sub-goals, showcasing different rules for different sub-goals, allowing users to inspect them easily. It also indicates that even if users do not input any rules, they can still observe the rules generated during the model's training process.

4 DEMONSTRATION

We will conduct a demonstration of HRLMS on Montezuma's Revenge [10] (shown in 3), which is a typical Atari game characterized by intricate levels and multi-faceted challenges. In this game, the agent takes a series of actions to obtain a reward, but the reward is very sparse. We want the agent to successfully get the key, but on the way to the key's location, the agent may have to pass multiple targets (such as ladders and skulls).


Key Rules

Move(Man,Key) -> ActorOnSpot(Man,Key)

MiddleLadder Rules

Move(Man,MiddleLadder) -> ActorOnSpot(Man,RightDoor) ∧ ActorWithObject(RightDoor,MiddleLadder)

LeftLadder Rules

Move(Man,LeftLadder) -> PathExist(Man,LeftLadder)

RightLadder Rules

Move(Man,RightLadder) -> ActorWithObject(Man,Key) ∧ PathExist(Key,RightLadder)

Skull Rules

Move(Man,Skull) -> ActorOnSpot(Man,Skull)

RightDoor Rules

Move(Man,RightDoor) -> ActorOnSpot(Man,MiddleLadder) ∧ PathExist(MiddleLadder,RightDoor)

Figure 7: Rule display. The classification in each box shows the rules related to a certain object, where Key, MiddleLadder, etc. are objects in the environment.

We construct interactive components using Spring Boot and Vue3. The backend is tasked with monitoring rules generated by the model; new rule changes are then sent to the frontend for display. The frontend is responsible for building rules and displaying both constructed and model-generated rules. The upload of configuration files must be relevant to the training environment, as predicates and objects that are completely unrelated hold no value. Furthermore, the configuration file format is required to be in JSON. During the demonstration, we will guide participants through all steps. Figure 4 shows a snapshot of an example file.

Step ① (Uploading Configuration Files): Initially, users can click the "Select File" button to choose the configuration file relevant to their environment, allowing the system to load the content of predicates and objects. The content loaded can be seen in the cascading menu on the left side of Figure 4. The first level of the menu displays the predicates, while the second and third levels show the objects.

Step ② (Selecting Predicates and Objects): In this step, users need to select predicates from the cascading menu and choose objects to populate these predicates. The first object selected for the first predicate must be "man," indicating that the rule inference begins with the agent. Additionally, the objects within the same predicate cannot be identical. This ensures diversity in the rules' logic and relevance to the agent's decision-making process.

Step ③ (Forming Rules by Adding Predicates): After selecting predicates, when the "Add Predicate" button is pressed, the chosen predicates are incorporated into the box on the left side of Figure 5, forming the most current rule. This step facilitates the dynamic construction of rules, allowing users to iteratively build complex logic by adding one predicate at a time, thus tailoring the rules to the specific needs of the training environment and the objectives of the hierarchical reinforcement learning model.

Step ④ (Processing Rules): If the currently constructed rule requires adjustments, clicking the "Clear Input" button will empty the content in the box on the left side of Figure 6, and the cascading menu on the left side of Figure 5 will also be cleared. If there are no issues with the currently constructed rule, the "Add Rule" button can be selected to add the well-constructed rule from the box into the model's rule set. This step ensures that users have the flexibility to refine or completely redo their rule construction before committing it to the model.

Once these steps are completed, both the added rules and the rules generated during the model's training can be viewed simultaneously in Figure 7.

REFERENCES

[1] Laure Crochepierre, Lydia Boudjeloud-Assala, and Vincent Barbesant. 2022. Interactive Reinforcement Learning for Symbolic Regression from Multi-Format Human-Preference Feedbacks. In *IJCAI-22*, Lud De Raedt (Ed.). 5900–5903. Demo Track.

[2] Sainyam Galhotra and Udayan Khurana. 2022. Automated relational data explanation using external semantic knowledge. *Proc. VLDB Endow.* 15, 12 (aug 2022), 3562–3565.

[3] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael A. Specter, and Lalana Kagal. 2018. Explaining Explanations: An Approach to Evaluating Interpretability of Machine Learning. *CoRR* abs/1806.00069 (2018). arXiv:1806.00069

[4] Kevin A. Gluck and John E. Laird. 2019. *Interactive Task Learning: Humans, Robots, and Agents Acquiring New Tasks through Natural Interactions.* The MIT Press.

[5] Weiwei Gu, Anant Sah, and Nakul Gopalan. 2024. Interactive Visual Task Learning for Robots. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 21 (Mar. 2024), 23793–23795.

[6] Neil Hulbert, Sam Spillers, Brandon Francis, James Haines-Temons, Ken Gil Romero, Benjamin De Jager, Sam Wong, Kevin Flora, Bowei Huang, and Athirai A. Irissappane. 2020. EasyRL: A Simple and Extensible Reinforcement Learning Framework. *CoRR* abs/2008.01700 (2020). arXiv:2008.01700

[7] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. 2018. Reward learning from human preferences and demonstrations in Atari. In *NeurIPS 2018*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 8022–8034.

[8] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems* 29 (2016).

[9] Aaron Mininger and John E. Laird. 2022. A Demonstration of Compositional, Hierarchical Interactive Task Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 11 (Jun. 2022), 13203–13205. https://doi.org/10.1609/aaai.v36i11.21728

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[11] Vishal Pallagani, Bharath Muppasani, Biplav Srivastava, Francesca Rossi, Lior Horesh, Keerthiram Murugesan, Andrea Loreggia, Francesco Fabiano, Rony Joseph, and Yathin Kethepalli. 2023. Plansformer Tool: Demonstrating Generation of Symbolic Plans Using Transformers. In *IJCAI-23*. 7158–7162. Demo Track.

[12] Thanh Cong Phan, Thanh Tam Nguyen, Matthias Weidlich, Hongzhi Yin, Jun Jo, and Quoc Viet Hung Nguyen. 2022. exRumourLens: Auditable Rumour Detection with Multi-View Explanations. In *ICDE*. 3174–3177.

[13] Bidipta Sarkar, Aditi Talati, Andy Shih, and Dorsa Sadigh. 2021. PantheonRL: A MARL Library for Dynamic Training Interactions. In *AAAI*.

[14] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[15] Tingyu Wang, Yuchao Tao, Amir Gilad, Ashwin Machanavajjhala, and Sudeepa Roy. 2023. Explaining Differentially Private Query Results with DPXPlain. *Proc. VLDB Endow.* 16, 12 (aug 2023), 3962–3965.

[16] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1995–2003.

[17] Haodi Zhang, Zhichao Zeng, Keting Lu, Kaishun Wu, and Shiqi Zhang. 2022. Efficient Dialog Policy Learning by Reasoning with Contextual Knowledge. In *Thirty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press, 11667–11675.