Hybrid Deep RePReL: Integrating Relational Planning and Reinforcement Learning for Information Fusion

1st Harsha Kokel
Computer Science
The University of Texas at Dallas
Dallas, Texas, USA
hkokel@utdallas.edu

2nd Nikhilesh Prabhakar
Computer Science
The University of Texas at Dallas
Dallas, Texas, USA
nikhilesh.prabhakar@utdallas.edu

3rd Balaraman Ravindran
Robert Bosch Centre for Data Science and Artificial Intelligence
Indian Institute of Technology
Chennai, India
ravi@cse.iitm.ac.in

4th Erik Blasch
Air Force Research Laboratory
Rome, NY, USA
erik.blasch.1@us.af.mil

5th Prasad Tadepalli
School of EECS
Oregon State University
Corvallis, Oregon, USA
tadepall@engr.orst.edu

6th Sriraam Natarajan
Computer Science
The University of Texas at Dallas
Dallas, Texas, USA
Sriraam.Natarajan@utdallas.edu

***Abstract*—Fusion of high-level symbolic reasoning with lower level signal-based reasoning has attracted significant attention. We propose an architecture that integrates the high-level symbolic domain knowledge using a hierarchical planner with a lower level reinforcement learner that uses hybrid data (structured and unstructured). We introduce a novel neuro-symbolic system, Hybrid Deep RePReL that achieves the best of both worlds—the generalization ability of the planner with the effective learning ability of deep RL. Our results in two domains demonstrate the superiority of our approach in terms of sample efficiency as well as generalization to increased set of objects.**

***Index Terms*—Fusion, Reinforcement Learning, Structured and Unstructured domains**

I. Introduction

Recently, the neurosymbolic architectures are of interest to the information fusion community for physics-based and human-derived information fusion (PHIF) along the directions of emerging interest in ontologies, explainable Artificial Intelligence (AI), and situational awareness. From the Data Fusion Information Group (DFIG) model as related to the Joint Director of the Labs (JDL) definitions, there are seven levels of data processing. Low-level information fusion (LLIF) includes L0 data preprocessing and L1 estimation and classification, which are traditional methods in information fusion. The high-levels of information fusion (HLIF) includes L2 situation assessment, L3 threat assessment, L4 sensor management, L5 user refinement, and L6 mission management [1]. Thus, the emerging developments in machine (ML), deep (DL), and reinforcement learning (RL) align with HLIF.

Neurosymbolic architectures (NSAs) are attractive for human interaction with information fusion systems, especially for question/answering when in the form of semantic queries that include symbolic definitions, such as a dictionary of terms. When using the NSA, the user queries a knowledgebase from which a fusion inference engine produces an output [2]. Using the motivation that information fusion seeks to reduce uncertainty, the international society of information fusion (ISIF) has created the Uncertainty Representation and Reasoning Evaluation Framework (URREF) ontology [3]—a knowledgebase of uncertainty terms and theory of mathematical processing. The use of the NSA for information fusion enhances methods for situation assessment for complex event processing (CEP). Vilamala et al. [4] introduce DeepProbCEP, a hybrid NSA that leverages both a neural architecture, to interpret raw data, and logical rules, to express patterns defining complex events, while allowing for end-to-end learning by fusing videos and audio information for situation assessment. The NSA improves sound and video accuracy [5] that can be integrated into an information fusion architecture for user refinement and assessment.

User refinement (UR), at least from the DFIG/JDL perspective, is an emerging topic for human-machine systems concerning the use of DL/RL techniques [6]. Challenges for UR include controlling many platforms [7], sensor tasks [7], and mission needs. RL has been widely used in sensor management, but recent developments in deep RL (DRL) offers the ability to incorporate DL methods (e.g., for classification from multiple sensors) into control. RL combined with deep symbolic learning supports an emergent challenge for DL methods for explainability (for the user), interpretability (of the machine), and transparency (of the data and ethical decisions). The development of NSA with RL can enhance trust [8], and control for human-agent knowledge systems [9].

Building a two-level neurosymbolic system that combines a higher-order symbolic reasoner with a lower-level fast deep learner is not only a long cherished dream but is one of the more popular research directions inside AI [10]. Particularly, exploring the combination of learning and neurosymbolic processing in the context of sequential decision-making is quite natural [11]–[13] given the recent success of DRL methods on large-scale tasks [14]–[16].

We recently developed a combination of a higher-order symbolic planner with a lower-level RL system called RePReL [17], [18] to effectively construct abstractions to accelerate learning in structured domains (with several interacting objects that cannot be efficiently represented using a fixed-length vector). RL in structured domains is inherently a difficult task and only a small number of solutions exist [19]–[21]. The RePReL system takes a first step in the direction of combining (relational) planning and RL in solving structured problems by using the planner to define a smaller set of (abstract) state-action spaces to allow for efficient learning by the lower level RL agent. The key success of the RePReL method lies in its capability of generalization to a varying number of objects.

While RePReL was successful, it had an important assumption—the underlying features are discrete. That is to say that the higher-order symbolic planner essentially operated at the level of predicates defined in first-order logic and then constructed appropriate grounded predicates as features for the RL agent. This assumption restricts the RePReL approach in two specific ways. First is that the power of DRL methods is not effectively exploited. Second, the approach cannot fully exploit the fusion of multi-modal data where the data could arrive from multiple sources—images, text and potentially a feature based descriptor.

This paper relaxes the discrete feature assumption and extends the RePReL framework to construct a *Hybrid Deep Relational Planning and Reinforcement Learning (HDRePReL)* architecture that **fuses** multi-modal information from two or more sources. The key contributions of the paper include: (1) We extend the standard RePReL architecture to handle multi-modal data by fusing information from multiple sources. (2) The resulting HDRePReL framework is a neurosymbolic architecture that leverages the deliberative nature of a higher-level planner with the fast learning nature of the lower level deep network. (3) As far as we are aware, this is one of the first works in the direction of combining relational planner with DRL to address the problem of learning in structured, and heterogeneous (discrete/continuous) domains. (4) Finally, our empirical evaluations on two domains clearly demonstrate the effectiveness of learning and the efficiency of generalization and transfer across related tasks.

The rest of the paper is organized as follows. The next section provides a background on RePReL. Then, Section III presents the HDRePReL architecture. Section IV present our empirical evaluations before concluding the paper by presenting the directions for future research.

II. Background

There has been a recent surge in combined planning and reinforcement learning approaches for multi-task RL problems [17], [22]–[28]. Among these, the RePReL architecture proposed by Kokel et al. [17] provides a unique framework to construct task-specific state abstractions. RePReL formulates structured multi-task RL problems as goal directed relational Markov Decision Process (GRMDP).

Definition 1: A GRMDP $\mathcal{M}$ is represented as a tuple $\langle S, A, P, R, \gamma, G\rangle$, where the states S and the actions A are represented by a set of objects E, a set of predicates Q, and action types Y. P is a transition probability function $S \times A \times S \rightarrow [0, 1]$, R is a reward function $S \times A \times S \rightarrow \mathbb{R}$, $\gamma \in [0, 1)$ is the discount factor, and G is the set of goals that the agent may be asked to achieve.

A problem instance for a GRMDP is defined by a pair $\langle s \in S, g \in G\rangle$, where s is a state and g is a goal condition, both represented using sets of literals, i.e., positive and/or negative atoms. A solution is a policy that starts from s and ends in a state satisfying g. The RePReL framework proposes to solve the GRMDPs using a combination of planning and RL. The RePReL architecture, shown in Figure 1, consists of three stacked modules: Symbolic Planner, Abstraction Reasoner, and RL agents.

1) **Symbolic Planner:** The *symbolic planner* uses the high-level planning domain description to decompose the goal into a sequence of temporally extended actions called *options*. Essentially, the planner decomposes the GRMDP into small sub-goal RMDPs.
2) **Abstraction Reasoner:** The *abstraction reasoner* generates a task-specific abstract state representation using the dynamic first-order conditional influence statements provided by a domain expert.
3) **RL Agents:** Finally, multiple *reinforcement learners* at the lowest level learn separate RL policies for each option in the abstract state space.

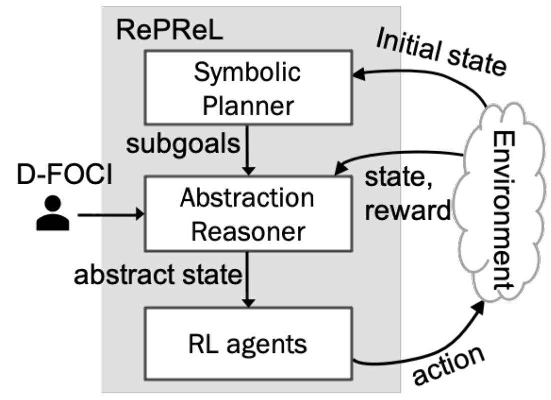


Fig. 1: RePReL architecture.

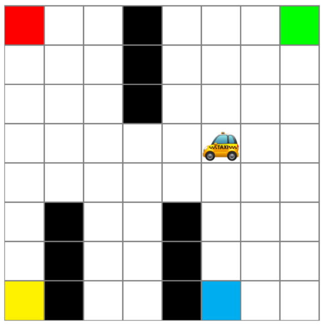
III. Hybrid Deep RePReL

To address multi-task hybrid RL problems, with structured and unstructured data, there is a need to incorporate data from different sources. For convenience, the proposed approach assumes the unstructured data is an image, but it can be extended to other form of unstructured data verbatim.

Given: A combination of structured (object descriptions as predicate logic representations) and unstructured (images or text) data.
To Do: Develop a hybrid architecture that learns to act.

Consider the scenario of a ride sharing app, the information required for booking and riding a passenger arrives from different sources. The local geography of the region is obtained from a map, the location of the cab can be obtained from the cab driver's mobile, and the location of the rider and the destination of the ride is obtained from the rider's mobile. In such cases, the information required for a ride can be a conglomeration of many varied types of data coming from different sources.

As a more concrete example, consider a hybrid taxi domain, shown in Figure 2. There are two passengers in this domain and one taxi. Six actions available in this domain are: `east`, `west`, `north`, `south`, `pick`, `drop`. The task is to transport passenger(s) from their current location to their destination location. Only *one* passenger can hire the taxi at a time. We consider **three different sources of information** in this domain, the taxi location and the geography of the region is available as an image from one source (cf Figure 2a); the current location and the destination location of the passenger `p1` is available from `p1`'s mobile as state predicates; and similarly the current location and the destination of the passenger `p2` is available from `p2`'s mobile (cf Figure 2b). Hence, the state or the observation from the environment is hybrid—consisting of the structured data from passengers' mobile and unstructured image data with taxi location.



(a)

$at(p1, l1), dest(p1, d1)$

$at(p2, l2), dest(p1, d2)$

(b)

Fig. 2: (a) Taxi location and geography information available as an image. (b) Passenger location and destination information as state predicates from a passenger's mobile.

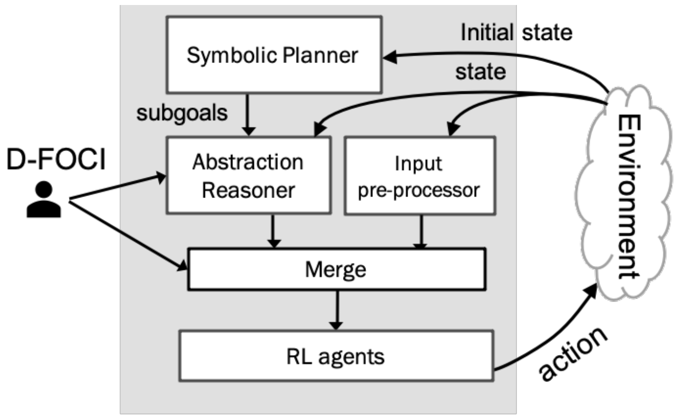


Fig. 3: Proposed HDRePReL architecture.

For such hybrid RL domains, we extend the RePReL framework and introduce hybrid deep RePReL (HDRePReL). In HDRePReL, shown in Figure 3, we introduce an input preprocessing module and a merge module to handle the combination of structured and unstructured information. In addition, there is an underlying DRL layer that allows for learning with these hybrid inputs. The next section explains each of these modules in greater detail.

A. Symbolic Planner

Given a problem instance of a GRMDP $\langle s, g \rangle$, the symbolic planner provides a high-level plan $\Pi = [o_1, o_2, ..., o_n]$ which is a sequence of options (or temporally extended operators). Each option o has predefined termination condition $\beta(o)$, necessary effects of the operator. A subgoal RMDP M_o is defined for each option o, which is solved to obtain the option policy π_o.

Definition 2: The **subgoal RMDP** M_o for an option o is defined as a tuple $\langle S, A, P_o, R_o, \gamma \rangle$ consisting of states S, actions A, transition function P_o, reward function R_o, and discount factor γ. The state and action spaces remain same as the original GRMDP. The reward function R_o and the transition probability distribution function P_o are defined as follows:

$$R_o(s,a,s') = \begin{cases} t_R + R(s,a,s') & \text{if } s' \in \beta(o) \text{ and } s \notin \beta(o) \\ 0 & \text{if } s' \in \beta(o) \text{ and } s \in \beta(o) \\ R(s,a,s') & \text{otherwise} \end{cases}$$

$$P_o(s,a,s') = \begin{cases} 0 & \text{if } s \in \beta(o) \text{ and } s' \notin \beta(o) \\ 1 & \text{if } s \in \beta(o) \text{ and } s' \in \beta(o) \\ P(s,a,s') & \text{otherwise} \end{cases}$$

with a fixed terminal reward t_R, and the reward function $R(s,a,s')$ and the transition function $P(s,a,s')$ from the original GRMDP.

Essentially, the reward function in the original GRMDP would correspond to the step cost function, which applies to all options, and reward R_o is the only goal-specific reward. These subgoal RMDPs are solved by the RL agents at lower level in an abstract state space.

B. Abstraction Reasoner

State abstraction defines a smaller set of states for the MDP state space, enabling efficient learning by an RL agent. The abstraction reasoner takes domain knowledge and constructs the state abstraction. In the taxi domain example, to drop a passenger that is already in the taxi, the location of the other passenger and their destinations are irrelevant. The reasoner component precisely identifies such irrelevant features given the domain knowledge and infers a state abstraction.

Dietterich [29] defines *irrelevant state variables* for an MDP as variables that never influence the reward function or the relevant state variables. Formally,

Definition 3: State variables Y are **irrelevant** in an MDP, if state variables can be partitioned into two disjoint subsets X and Y such that

1) $P(x', y'|x, y, a) = P(x'|x, a)P(y'|x, y, a)$
2) $R(s, a, s') = R(\langle x, y\rangle, a, \langle x', y'\rangle) = R(x, a, x')$

Given a Dynamic Bayesian Network (DBN) representation [30] of the transition function of an MDP, the set of relevant variables can be identified by starting at the reward variable and collecting all the variables that influence the collected variables. This set of relevant variables form a "model-agnostic state abstraction". Such abstractions ensure that the reward distribution and the transition dynamics in the abstract MDP and the original MDP are same. Hence, model-agnostic abstraction is safe.

The *transition function of a GRMDP is first-order/relational* and hence a simple DBN does not suffice. Previous work have used a 2-timeslice Probabilistic Relational Model (PRM) [31] to capture the transition function in relational MDP. However, PRMs are not suitable for capturing GRMDPs. Hence, RePReL framework uses an extension of the First-Order Conditional Influence (FOCI) language [32] called Dynamic FOCI (D-FOCI).

D-FOCI language consist of statements of the form,

$$\texttt{<option>}:\texttt{<influents>} \xrightarrow{+1} \texttt{<resultant>} \qquad (1)$$

where `influents` is a finite set of literals, `resultant` is a single literal, and the `option` is a temporally extended operator from the symbolic planner. Additionally, action variables are allowed in `influents` and reward variables are allowed in `resultant`. A D-FOCI statement states that when executing the given `option`, the `resultant` literal in time-step $t+1$ is influenced by literals in `influents` in time-step t. D-FOCI encodes the context-specific influence information where the context is the `option` being executed. The $+1$ on top of the arrow indicates the influence on the next time-step, for same time-step influence the $+1$ can be skipped. To encode perpetual influence between literals, `<option>` in the D-FOCI statement can be omitted.

A task-specific model-agnostic abstraction is derived for each `option` by regressing through the set of *applicable* D-FOCI statements and collecting the relevant literals that influence the relevant literals, starting with the literals that influence the reward variables. A D-FOCI statement is *applicable* for a given option o if the `<option>` in the D-FOCI statement matches o or the `<option>` is empty. Inspired by the work of Manhaeve et al. [33], we *extend the first-order language used by D-FOCI to include latent predicates for hybrid, and heterogeneous domains*. Latent predicate literals are allowed in the `influents` as well as the `resultant`.

For example, in the hybrid taxi domain, the location of the taxi is available from the image so a latent predicate is introduced to represent the taxi location—`Img:taxi_at`. The passenger location and destination are available as state predicates, so they are represented using the standard first-order logic notation. The location of the taxi is influenced by its previous location and the action performed, which is captured in the D-FOCI statement in Equation 2a. Further, when executing the task of picking up a passenger, if one assumes that the taxi is empty, then it can be safely inferred that the passenger's location, the passenger in the taxi, and the taxi's location are the only influents for the task. This influence is captured in Equations 2b and 2c. Similarly, the influence information while dropping passenger P is captured in Equation 2d–2f.

$$\{\texttt{action}, \texttt{Img:taxi\_at}(X)\} \xrightarrow{+1} \texttt{Img:taxi\_at}(X) \qquad (2a)$$

$$\texttt{pick}(P):\{\texttt{action}, \texttt{in\_taxi}(P), \texttt{at}(P, Y), \texttt{Img:taxi\_at}(X)\} \xrightarrow{+1} \texttt{in\_taxi}(P) \qquad (2b)$$

$$\texttt{pick}(P):\{\texttt{in\_taxi}(P)\} \longrightarrow Reward \qquad (2c)$$

$$\texttt{drop}(P):\{\texttt{at\_dest}(P)\} \longrightarrow Reward \qquad (2d)$$

$$\texttt{drop}(P):\{\texttt{at}(P, X), \texttt{dest}(P, D), \texttt{at\_dest}(P)\} \longrightarrow \texttt{at\_dest}(P) \qquad (2e)$$

$$\texttt{drop}(P):\{\texttt{action}, \texttt{Img:taxi\_at}(X), \texttt{at}(P, Y), \texttt{in\_taxi}(P)\} \xrightarrow{+1} \texttt{at}(P, K) \qquad (2f)$$

The process of identifying the abstract state representations from the D-FOCI statements with latent predicates remains the same. Hence, the extracted abstract state might have the latent predicates.

C. Input Pre-processor

While the structured part of the state observation is processed by the abstraction reasoner, the unstructured part of the state space is passed through the input pre-processing module. This module processes the unstructured part of the state observation and provides latent state embeddings to the *merge* module. The input pre-processing module can be a Convolutional Neural Network (CNN) for image data, a transformer for text data or a combination of both (we employ CNNs in our experiments). The neural network of the input pre-processor can be chosen based on the type of the unstructured data.

D. Merge module

The relevant state variables obtained from the abstraction reasoner and the latent predicates obtained from the input pre-processor serve as inputs to the merge module. The relevant state predicates from the abstraction reasoner would include the latent predicates. The merge module would replace the latent predicates with the respective latent embedding using the D-FOCI statements. The derived abstract state then serves as the input to the RL agent.

E. RL agent

To allow for heterogeneous data, we replace the tabular RL agents in the RePReL framework by the DRL agents. To this effect, we modify the RePReL learning algorithm from Kokel et al. [17] to batch setting. That is, the $\langle s, a, r, s' \rangle$ (state, action, reward, next-state) tuples are stored in buffers during exploration. Then after fixed number of exploration steps, the RL agents and the input pre-processing module are trained on a batch sampled from the buffer.

In summary, the key extensions to the original RePReL work includes the use of multi-modal information, an additional merge module along with the extended reasoner module to generate latent features, the use of Deep RL in the lowest level, and updating the learning to be performed in the batch mode. Given, these extensions, we now present our empirical evaluations.

IV. Experiments

We evaluation HDRePReL in two domains, Taxi and Craft World. We design our experiments to explicitly answer the following questions.

Q1: Sample Efficiency: Do the abstractions induced in HDRePReL improve sample efficiency?
Q2: Generalization: Does HDRePReL efficiently generalize to varying number of objects?

Domains: The first domain that we consider is the **hybrid taxi domain**. This is an 8×8 grid, shown in Figure 2a, with one taxi and 4 special locations: $\langle R, G, B, Y \rangle$. There can be more than one passenger at a time in the grid; but the taxi can be hired only by a single passenger. In every episode, the pickup and drop location of each passenger are sampled from the 4 special locations. The agent can perform 6 actions: *up, down, right, left, pick, drop*. The taxi location and the grid is

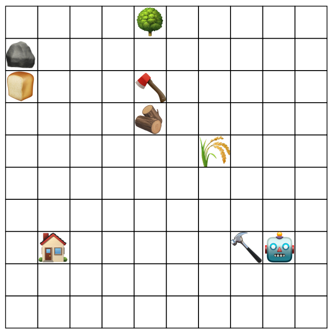


Fig. 4: Craftworld domain.

provided as a gray scale image and the passenger information is provided as vector. The environment provides a reward of -0.1 for every step and -1 for pick or drop action in wrong locations. We consider the following three tasks in this domain: *task* 1, drop a passenger to their destination; *task* 2, drop two passengers to the respective destinations; and *task* 3, drop three passengers to their destination. The set of D-FOCI statements used in this domain are presented in Equation 2.

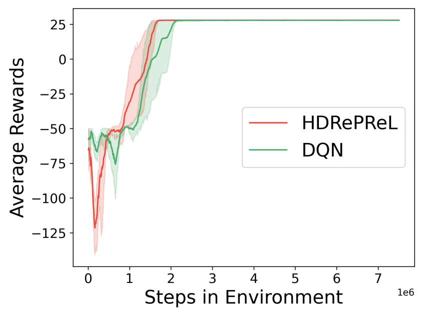
The second domain is a Minecraft-inspired grid-world domain called Craftworld [34]. This environment contains a 10x10 grid, as shown in Figure 4. This domain has 7 different object, some of which are holdable. The six actions available in this domain are the same as that of the hybrid Taxi domain. In this domain, the agent's location and the map is available as unstructured data–as image, whereas all the objects' location are available as state predicates. We consider three tasks in this domain: *Task* 1, make bread by bringing an axe to the wheat's location; *Task* 2, build a house by bringing a hammer to the wood's location; *Task* 3, break a rock by bringing a hammer to the rock's location. The set of D-FOCI statements used in this domain are presented in Equation 3.

Baselines: We evaluate our HDRePReL agent against a double *Deep Q-Network (DQN)* [35], a state-of-the-art DRL method. The DQN agent learns a single end-to-end policy for completing the task. The network architecture of the DQN agent includes a CNN module—equivalent to the input pre-processor. The CNN module receives the unstructured part of the state observation. Then, the output of the CNN module is concatenated with the structured part of the state representation and passed through a multi-layered perceptron. The network parameters are summarized in the Table I. For honest comparison, we employed the same network parameters in HDRePReL and DQN. All of our code is available online[1].

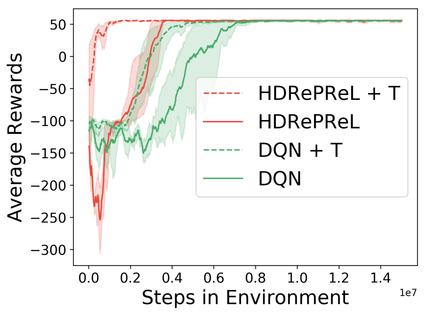
Sample Efficiency: To evaluate the efficiency of the HDRePReL architecture, we compare it against the DQN agent. We compare the learning curves of these two agents on three tasks of the hybrid taxi domain in Figure 5 and on the three tasks of the craftworld in Figure 6. Ignore the dashed lines with '+T' for now. While both the HDRePReL and the DQN agent achieves the optimal reward after 200K steps in Taxi *task* 1; their performance significantly differs in the five remaining tasks. In these tasks, it can be clearly seen that the efficiency of HDRePReL is significantly better than the baseline thus demonstrating that the abstractions defined by the reasoner enables efficient learning. Hence, we answer **Q1** affirmatively in that HDRePReL significantly improves the *sample efficiency* when compared to an end-to-end RL agent that does not use any domain-specific knowledge.

Generalization: We evaluate generalization capability of the HDRePReL agent in the hybrid taxi domain. We transfer the HDRePReL and the DQN agent trained on *task* 1 and train them on *task* 2. Subsequently, we transfer the agents from *task* 2 to *task* 3. Figures 5b and 5c present the learning curves of these transferred agents indicated by '+T'. While

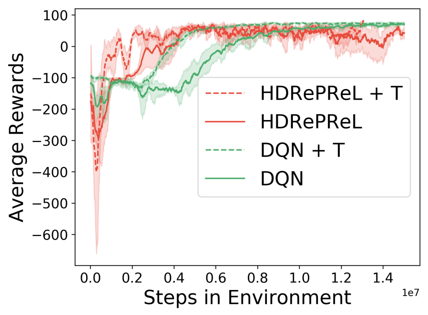
[1]https://starling.utdallas.edu/papers/HybridDeepRePReL



(a) Task 1

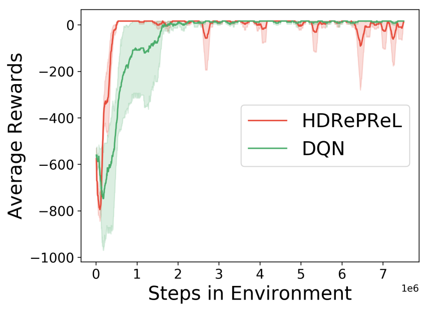


(b) Task 2

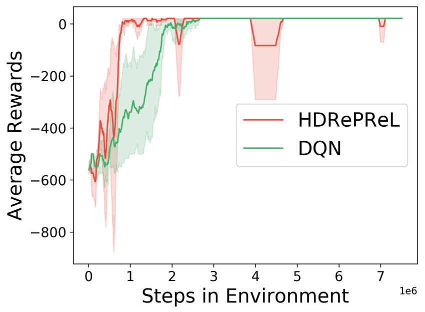


(c) Task 3

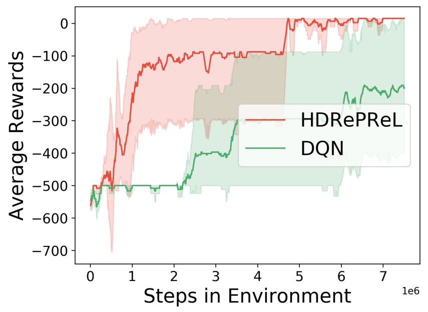
Fig. 5: Comparing learning curves of hybrid deep RePReL with DQN in Hybrid Taxi World. **(a)***Task 1* is to drop passenger p1,**(b)** *Task 2* is to drop p1 and p2, **(c)***Task 3* is to drop p1, p2, p3.



(a) Task 1



(b) Task 2



(c) Task 3

Fig. 6: Comparing learning curves of HDRePReL with DQN in Craft World. **(a)** *task* 1 is to make bread; **(b)** *task* 2 is to build house; **(c)** *task* 3 is to break rock.

both the transferred agents have steeper learning curves than their respective base models, the transferred RePReL agent converges significantly faster than the transferred DQN agent. In many of the tasks, this is achieved with no learning in the new domain. This is due to the inherent generalization capabilities of the HDRePReL agent. These transfer results in hybrid taxi domain allows us to answer **Q2** affirmatively in that HDRePReL allows for successful generalization across varying number of objects and is best suited for relational domains.

Summary: To summarize, the experiments conclusively demonstrates the most important observation about the HDRePReL agent – that it leverages the generalization power of the symbolic planner with the efficient learning ability of the underlying DRL agent. The resulting combination is a powerful neurosymbolic system that not only learns efficiently but *generalizes to a larger number of objects by bootstrapping on its prior learned policies*. The generalization ability is particularly important when learning occurs with different starting states, differing number of objects, different domain configurations or different target states. In real world, assuming that the data arrives only from a single source can lead to disastrous results. Hence, using architectures that support heterogeneous data that have the ability to generalize to different numbers of objects is crucial for a system to be deployed in real-time. HDRePReL takes a first step towards this direction.

V. Conclusion

The paper presented a novel neurosymbolic system that is capable of learning in the presence of heterogeneous (discrete and continuous), hybrid (structured and unstructured) and relational (objects and relations) data. Specifically, the paper introduced HDRePReL; an architecture that combines the advantage of a deliberate relational planner with a fast DRL agent. The resulting combination demonstrated both effective learning and efficient generalization across differing numbers of objects. More rigorous evaluation of the system on larger problems is an immediate future direction. Allowing for the DRL agent to communicate back to the planner in order to refine the planner based on new and interesting observation is a high-impact direction that could allow for a fully differentiable end-to-end system. Finally, given the use of a symbolic planner, the resulting decompositions and abstractions are explainable. Thus allowing for richer human interaction with the given system remains an interesting direction for future research.

$$
\begin{aligned}
&\{\texttt{action}, \texttt{Img:agent\_at}(X), \texttt{holdable}(Y),\\
&\texttt{holding}(Y)\} \xrightarrow{+1} \texttt{Img:agent\_at}(X)\\
&\{\texttt{holdable}(Y), \texttt{holding}(Y), \texttt{at}(Y, L)\}\\
&\quad \xrightarrow{+1} \texttt{holding}(Y)\\
&\{\texttt{at(rock,L1)}, \texttt{at(tree,L2)}\}\\
&\quad \xrightarrow{+1} \texttt{Img:agent\_at}(X)\\
&\{\texttt{at(tree,L)}, \texttt{Img:agent\_at}(L), \texttt{holding}(hammer)\}\\
&\quad \longrightarrow \texttt{at(tree,L)}, \texttt{at(wood, L)}\\
&\{\texttt{at(wheat,L)}, \texttt{Img:agent\_at}(L), \texttt{holding}(axe)\}\\
&\quad \longrightarrow \texttt{at(wheat,L)}, \texttt{at(bread, L)}\\
&\{\texttt{at(wood, L)}, \texttt{Img:agent\_at}(L), \texttt{holding}(hammer)\}\\
&\quad \longrightarrow \texttt{at(wood, L)}, \texttt{at(house, L)}\\
&\{\texttt{at(rock,L)}, \texttt{Img:agent\_at}(L), \texttt{holding}(hammer)\}\\
&\quad \longrightarrow \texttt{at(rock,L)}\\
&\texttt{pick}(P) : \texttt{holding}(P) \longrightarrow R_o\\
&\texttt{go\_to}(P) : \{\texttt{at}(P, L), \texttt{Img:agent\_at}(L)\} \longrightarrow R_o
\end{aligned}
\tag{3}
$$

Hyperparameters	**Values**
Learning rate	0.003
Batch size	128
Max steps	$1e6$
Max buffer size	$1e5$
Discount rate	0.99
Intrinsic reward on subgoals	30
Number of CNN Layers	2
CNN Kernel Size	4
CNN Stride	1
CNN Activation Function	$relu$
Epsilon decay	True
Output Size (# of Actions)	6
RL Hidden layers	2
RL Hidden units	256
Craftworld	
Image size	10x10x1
Structured Input Size	48
Max episode length	500
Hybrid Taxi Domain	
Image Size	8x8x1
Structured Input Size	27
Max episode length (Task 1)	500
Max episode length (Task 2)	1000
Max episode length (Task 3)	1000

TABLE I: Summary of the network hyperparameters

ACKNOWLEDGMENT

HK, NP, and SN gratefully acknowledge the support of ARO award W911NF2010224 and AFOSR award FA9550-18-1-0462. PT acknowledges support of DARPA contract N66001-17-2-4030 and NSF & USDA-NIFA under grant 2021-67021-35344. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the AFOSR, ARO, NSF, DARPA or the U.S. government.

REFERENCES

[1] E. Blasch, É. Bossé, and D. A. Lambert, *High-Level Information Fusion Management and System Design*. Artech House, Norwood, MA, 2012.

[2] E. Blasch and A. Aved, "URREF for veracity assessment in query-based information fusion systems," in *FUSION*, 2015, pp. 58–65.

[3] P. Costa, A.-L. Jousselme *et al.*, "URREF: Uncertainty representation and reasoning evaluation framework for information fusion," *Journal of Advances in Information Fusion*, vol. 13, no. 2, pp. 137–157, Dec. 2018.

[4] M. R. Vilamala, L. Hiley *et al.*, "A pilot study on detecting violence in videos fusing proxy models," in *FUSION*, 2019, pp. 1–8.

[5] M. R. Vilamala, H. Taylor *et al.*, "A hybrid neuro-symbolic approach for complex event processing," *CoRR*, vol. abs/2009.03420, 2020.

[6] E. P. Blasch and D. Braines, "Scalable information fusion trust," in *FUSION*, 2021, pp. 1–8.

[7] R. Cruise, E. Blasch, S. Natarajan, and A. Raz, "Cyber-physical command guided swarm," in *DSIAC*, vol. 5, no. 2, 2018, pp. 24–30.

[8] G. Pavlin, J. P. de Villiers, J. Ziegler *et al.*, "Relations between explainability, evaluation and trust in ai-based information fusion systems," in *FUSION*, 2021, pp. 1–9.

[9] D. Braines, A. Preece, C. Roberts, and E. Blasch, "Supporting agile user fusion analytics through human-agent knowledge fusion," in *FUSION*, 2021, pp. 1–8.

[10] G. Booch, F. Fabiano *et al.*, "Thinking fast and slow in AI," in *AAAI*, 2021, pp. 15 042–15 046.

[11] M. Nilsson and T. Ziemke, "Information fusion: a decision support perspective," in *FUSION*, 2007, pp. 1–8.

[12] G. Anderson, A. Verma, I. Dillig, and S. Chaudhuri, "Neurosymbolic reinforcement learning with formally verified exploration," in *NeurIPS*, vol. 33, 2020, pp. 6172–6183.

[13] L. Mitchener, D. Tuckey, M. Crosby, and A. Russo, "Detect, understand, act: A neuro-symbolic hierarchical reinforcement learning framework," in *IJCLR*, 2021.

[14] D. Silver, A. Huang *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[15] D. Silver, J. Schrittwieser *et al.*, "Mastering the game of go without human knowledge," *Nat.*, vol. 550, no. 7676, pp. 354–359, 2017.

[16] D. Silver, T. Hubert *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[17] H. Kokel, A. Manoharan, S. Natarajan, R. Balaraman, and P. Tadepalli, "RePReL: Integrating relational planning and reinforcement learning for effective abstraction," *ICAPS*, vol. 31, no. 1, pp. 533–541, May 2021.

[18] H. Kokel, A. Manoharan, S. Natarajan, B. Ravindran, and P. Tadepalli, "Dynamic probabilistic logic models for effective abstractions in RL," *CoRR*, vol. abs/2110.08318, 2021.

[19] S. Sanner and K. Kersting, "Symbolic dynamic programming for first-order pomdps," in *AAAI*, 2010.

[20] C. Wang, S. Joshi, and R. Khardon, "First order decision diagrams for relational mdps," *JAIR*, vol. 31, pp. 431–472, 2008.

[21] S. Das, S. Natarajan, K. Roy, R. Parr, and K. Kersting, "Fitted q-learning for relational domains," *CoRR*, vol. abs/2006.05595, 2020.

[22] M. Grounds and D. Kudenko, "Combining reinforcement learning with symbolic planning," in *AAMAS III*, vol. 4865, 2005, pp. 75–86.

[23] F. Yang, D. Lyu, B. Liu, and S. Gustafson, "Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making," *IJCAI*, pp. 4860–4866, 2018.

[24] D. Lyu, F. Yang, B. Liu, and S. Gustafson, "SDRL: Interpretable and data-efficient deep reinforcement learning leveraging symbolic planning," in *AAAI*, 2019, pp. 2970–2977.

[25] Y. Jiang, F. Yang, S. Zhang, and P. Stone, "Task-motion planning with reinforcement learning for adaptable mobile service robots," in *IROS*, 2019, pp. 7529–7534.

[26] M. Eppe, P. D. H. Nguyen, and S. Wermter, "From semantics to execution: Integrating action planning with reinforcement learning for robotic causal problem-solving," *Frontiers in Robotics and AI*, vol. 6, p. 123, 2019.

[27] C. Gehring, M. Asai, R. Chitnis, T. Silver, L. P. Kaelbling, S. Sohrabi, and M. Katz, "Reinforcement learning for classical planning: Viewing heuristics as dense reward generators," *ICAPS*, 2022.

[28] L. Illanes, X. Yan, R. T. Icarte, and S. A. McIlraith, "Symbolic plans as high-level instructions for reinforcement learning," *ICAPS*, pp. 540–550, 2020.

[29] T. G. Dietterich, "An overview of hierarchical reinforcement learning," in *International Symposium on Abstraction, Reformulation, and Approximation*. Springer, 2000.

[30] K. P. Murphy, "Dynamic bayesian networks: Representation, inference and learning," Ph.D. dissertation, 2002.

[31] C. Guestrin, D. Koller *et al.*, "Generalizing plans to new environments in relational mdps," in *IJCAI*, 2003, pp. 1003–1010.

[32] S. Natarajan, P. Tadepalli *et al.*, "Learning first-order probabilistic models with combining rules," *Ann. Math. Artif. Intell.*, vol. 54, no. 1-3, pp. 223–256, 2008.

[33] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt, "Deepproblog: Neural probabilistic logic programming," in *NeurIPS*, vol. 31, 2018.

[34] C. Devin, D. Geng, P. Abbeel, T. Darrell, and S. Levine, "Compositional plan vectors," *NeurIPS*, vol. 32, 2019.

[35] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *AAAI*, 2016, pp. 2094–2100.