RePReL
Integrating Relational Planning and Reinforcement Learning for Effective Abstraction
RL and Planning

- Reinforcement Learning
  + Proven successful in complex games
  + Adaptive and robust against uncertainties
  - Relies on huge amount of data
  - Not effective for long-horizon problems
  - Generalization to different task

- Planning
  + Not data but prior-knowledge
  + Better generalization
  - May not capture uncertainties
Planner + RL

- Policy Sketches\(^1\) for modular policies that address multi-task RL
- PLANQ-learning\(^2\) uses planner to shape reward function that guides the Q-learner.
- PEORL\(^3\) (Planning–Execution–Observation–Reinforcement-Learning ) framework uses symbolic planner to guide the exploration and learning in RL
- TMP-RL\(^5\) and PDDL+HER\(^6\) framework use integrated approach for robotic systems with uncertainty and continuous state space
- Taskable-RL\(^7\) formalizes the high-level planner and low-level RL executioner setup

\(^1:\text{Andreas, Klein, and Levine ICML 2017} \)  
\(^2:\text{Grounds and Kudenko, AAMAS 2008} \)  
\(^3:\text{Yang, Lyu, Liu, and Gustafson IJCAI 2018} \)  
\(^5:\text{Jiang, Yang, Zhang, and Stone, IROS 2019} \)  
\(^6:\text{Eppe, Nguyen, and Wermter, Front. in Rob. and AI 2019} \)  
\(^7:\text{Illanes, Yan, Icarte, and McIlraith ICAPS 2020} \)
Motivation

Relational domains
RePReL

- Plan the sequence of high level subgoals and learn to execute each subgoal at lower level
RePReL

Goal directed relational MDP: 
\[\langle S, A, P, R, \gamma, G \rangle\]
**Definition 3.** The subgoal RMDP $M_o$ for each operator $o$ is defined by the 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 $\gamma$. State and Actions remain same as the original RMDP. The reward function $R_o$ and 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 $R(s, a, s')$ indicating the reward function from the original GRMDP definition. $t_R$ is a fixed terminal reward.
RePReL

- Plan the sequence of high level subgoals and learn to execute each subgoal at lower level

- Advantage:
  - Compositionality
  - Task specific state representations

- Dynamic First Order Conditional Influence (D-FOCI) statements to obtain task-specific abstract representations

Kokel et al ICAPS 2021
D-FOCI

First Order Conditional Influence (FOCI) statements

if condition then X1 influence X2

Dynamic FOCI statements

sub-task: $X_1 \xrightarrow{+1} X_2$

Natarajan, Tadepalli, Dietterich, and Fern 2008
Example
Example

task1(x):

```
Y → X +1 → X
```
Example

\[
\text{task1}(x): \quad \begin{align*}
Y & \rightarrow X & \quad +1 & \quad X \\
X & \rightarrow \text{Reward}
\end{align*}
\]
Example

task1(x):

Abstract state for task1(2):
Example

task1(x):

\[ Y \rightarrow X \xrightarrow{+1} X \]

task1(x):

\[ X \xrightarrow{+1} \text{Reward} \]

Abstract state for task1(3):
Example

task1(x):

\[ Y \rightarrow Z \xrightarrow{+1} Y \]

\[ Y \rightarrow X \xrightarrow{+1} X \]

\[ X \xrightarrow{+1} \text{Reward} \]

Abstract state for task1(3):

\[ a \rightarrow 3 \rightarrow 5 \]

\[ b \rightarrow 3 \]
Example

task1(x):

Y \rightarrow Z \quad +1 \quad Y

Y \rightarrow X \quad +1 \quad X

X \quad +1 \quad \text{Reward}

Abstract state for task1(3):

a \rightarrow b

c

2

b

4

5

3

a

2

b

3

5
Abstraction

**Definition 4** (Li, Walsh, and Littman (2006)). A model-agnostic abstraction $\phi(s)$ is such that for any action $a$ and abstract state $\bar{s}$, $\phi(s_1) = \phi(s_2)$ if and only if

\[
\sum_{\{s'_1 | \phi(s'_1) = \bar{s}\}} R_o(s_1, a, s'_1) = \sum_{\{s'_2 | \phi(s'_2) = \bar{s}\}} R_o(s_2, a, s'_2)
\]

\[
\sum_{\{s'_1 | \phi(s'_1) = \bar{s}\}} P_o(s_1, a, s'_1) = \sum_{\{s'_2 | \phi(s'_2) = \bar{s}\}} P_o(s_2, a, s'_2)
\]
Abstraction

**Definition 4** (Li, Walsh, and Littman (2006)). A model-agnostic abstraction $\phi(s)$ is such that for any action $a$ and abstract state $\bar{s}$, $\phi(s_1) = \phi(s_2)$ if and only if

$$
\sum_{\{s'_1 | \phi(s'_1) = \bar{s}\}} R_o(s_1, a, s'_1) = \sum_{\{s'_2 | \phi(s'_2) = \bar{s}\}} R_o(s_2, a, s'_2)
$$

$$
\sum_{\{s'_1 | \phi(s'_1) = \bar{s}\}} P_o(s_1, a, s'_1) = \sum_{\{s'_2 | \phi(s'_2) = \bar{s}\}} P_o(s_2, a, s'_2)
$$
Abstraction

\[
\begin{align*}
\{\text{taxi-at}(L1), \text{move}(\text{Dir})\} & \xrightarrow{+1} \text{taxi-at}(L2) \\
\{\text{taxi-at}(L1), \text{move}(\text{Dir})\} & \xrightarrow{} R \\
\text{pickup}(P): \quad \{\text{taxi-at}(L1), \text{at}(P, L), \text{in-taxi}(P)\} & \xrightarrow{+1} \text{in-taxi}(P) \\
\text{pickup}(P): \quad \text{in-taxi}(P) & \xrightarrow{} R_o
\end{align*}
\]

D-FOCI for Taxi pickup task
RePReL Learning

- Get high level plan
- For each subgoal
  - Loop till the subgoal is achieved
    - Get the abstract state
    - Get the policy for that subgoal
    - Take a step and observe reward, next state
    - Update the policy using state

Diagram:
- RePReL
- Hierarchical Planner
- Abstraction Reasoner
- Reinforcement Learners
- Get high level plan
- For each subgoal
  - Loop till the subgoal is achieved
    - Get the abstract state
    - Get the policy for that subgoal
    - Take a step and observe reward, next state
    - Update the policy using state
RePReL Learning

- Get high level plan
- For each subgoal
  - Loop till the subgoal is achieved
    - Get the abstract state
    - Get the policy for that subgoal
    - Take a step and observe reward, next state
    - Update the policy using abstract state

---

**Algorithm 1** RePReL Learning Algorithm

```
INPUT: \(\Psi(O, M)\), goal set \(g\), env, \(t_R, F\)
OUTPUT: RL policies \(\pi_o, \forall o \in O\)

1: \(\pi_o \leftarrow 0, \forall o \in O\) \quad \triangleright \text{initialize RL policy for each operator}
2: \textbf{for} each episode \textbf{do}
3: \(s \leftarrow \text{get state from env}\)
4: \(\Pi \leftarrow \Psi(s, g)\) \quad \triangleright \text{get high-level plan}
5: \textbf{for} \(o_g\) in \(\Pi\) \textbf{do}
6: \(\pi \leftarrow \pi_o\) \quad \triangleright \text{get resp. RL policy}
7: \(\hat{s} \leftarrow \text{GetAbstractState}(s, o_g, F)\)
8: \(\text{done} \leftarrow \hat{s} \in \beta(o_g)\) \quad \triangleright \text{check terminal state}
9: \textbf{while} not done \textbf{do}
10: \(a \leftarrow \pi(\hat{s})\) \quad \triangleright \text{get action}
11: \(s' \leftarrow \text{env.step}(a)\) \quad \triangleright \text{take step in env}
12: \(r \leftarrow R(s, a, s')\) \quad \triangleright \text{get step reward}
13: \(s' \leftarrow \text{GetAbstractState}(s, o_g, F)\)
14: \(\text{done} \leftarrow s' \in \beta(o_g)\) \quad \triangleright \text{check terminal next state}
15: \textbf{if} done \textbf{then}
16: \(r = r + t_R\) \quad \triangleright \text{add terminal reward}
17: \textbf{end if}
18: \(\pi\text{-update}(\hat{s}, a, s', r)\) \quad \triangleright \text{update policy}
19: \(s, \hat{s} \leftarrow s', \hat{s}'\)
20: \textbf{end while}
21: \textbf{end for}
22: \textbf{end for}
23: \textbf{return} \(\pi_o, \forall o \in O\)
```
Experiments

- **Domains**
  - Office World
  - Craft World
  - Relational Taxi
  - Relational Box World

- **Baselines**
  - HRL (options framework)
  - TRL (Taskable RL, Illanes et al. 2020)
Experiments

- Sample efficiency
- Transfer across task
- Generalization across objects

trl: Taskable RL (Illanes et al. ICAPS 2020)
Experiments

- Sample efficiency
- Transfer across task
- Generalization across objects
RePReL

Office World

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>△</td>
<td>Agent</td>
</tr>
<tr>
<td>*</td>
<td>Furniture</td>
</tr>
<tr>
<td>☕️</td>
<td>Coffee machine</td>
</tr>
<tr>
<td>💌</td>
<td>Mail room</td>
</tr>
<tr>
<td>🏡</td>
<td>Office</td>
</tr>
</tbody>
</table>

A, B, C, D Marked locations

Deliver mail
Deliver coffee
Deliver mail and coffee
RePReL

CRAFT WORLD

Get wood and iron

Make stick
(at workbench, needs wood)
RePReL

Relational Box World

Generalization across objects
For human-level general intelligence, the ability to detect compositional structure in the domain and form task-specific abstractions are necessary.
Other relevant work

- Learning the high-level planner [*Ludovico et al. ICLR 2021*]
- Modify the plan based on RL agents capability [*Lyu et al. AAAI 2019*]
- Automating task termination condition [*Lee et al. 2021*]
- Learning task-specific state representation [*Abdulhai et al. 2021*]
- Learning to plan and act simultaneously [*Patra et al. AI 2021*]
- Improving Robot Navigation [*Wöhlke et al. ICRA 2021*]
- Extending the RePReL framework to Deep RL setting (under prep)
QUESTIONS?
THANKS
Abstraction

Reasoner

Reinforcement Learners

initial state

state, reward

action

RePReL

Symbolic Planner

subgoals

abstract state

D-FOCI

Environment
Abstraction

Given

State
{at(p1, r), taxi-at(13), dest(p1, y), ¬at-dest(p1), ¬in-taxi(p1),
at(p2, b), dest(p2, g), ¬at-dest(p2), ¬in-taxi(p1)}

subtask
⟨ pickup(P), {P/p1, L/r} ⟩

D-FOCI
{taxi-at(L1), move(Dir)} \xrightarrow{+1} taxi-at(L2)
{taxi-at(L1), move(Dir)} \rightarrow R
pickup(P):
{taxi-at(L1), at(P, L), in-taxi(P)} \xrightarrow{+1} in-taxi(P)
pickup(P): in-taxi(P) \rightarrow R_o

Get
Abstract state
{at(p1, r), taxi-at(13), ¬in-taxi(p1), move(Dir)}