RePReL

Integrating Relational Planning and Reinforcement Learning for Effective Abstraction

Sriraam Natarajan

UT DALLAS

Harsha Kokel

UT DALLAS

Balaraman Ravindran

IIT MADRAS

Arjun Manoharan

IIT MADRAS

Prasad Tadepalli

Oregon State University

RL and Planning

- Reinforcement Learning
  - + *Proven successful in complex games*
  - + *Adaptive and robust against uncertainitities*
  - – *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

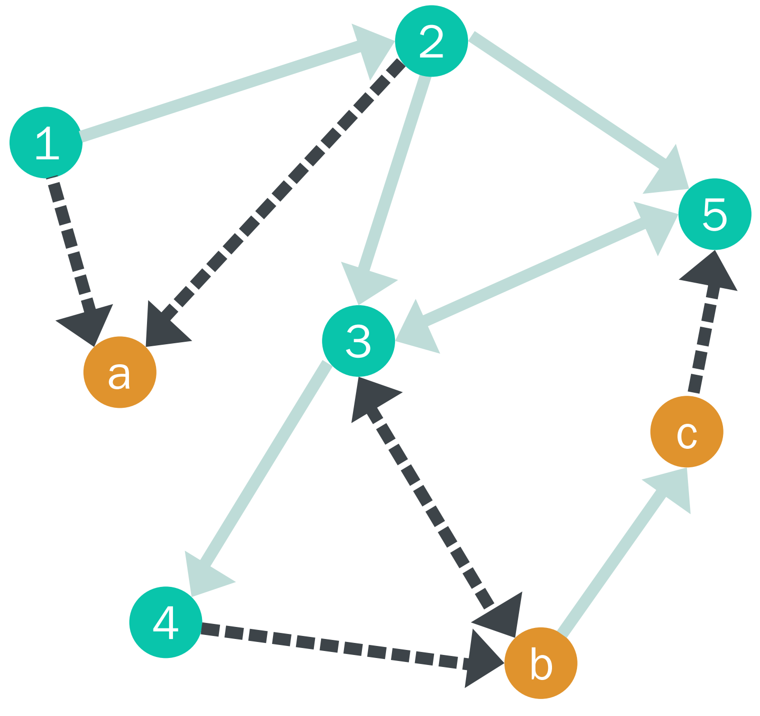
- Policy Sketches[1] for modular policies that address multi-task RL
- PLANQ-learning[2] uses planner to shape reward function that guides the Q-learner.
- PEORL[3] (Planning–Execution–Observation–Reinforcement-Learning) framework uses symbolic planner to guide the exploration and learning in RL
- TMP-RL[5] 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]Andreas, Klein, and Levine ICML 2017

[2]Grounds and Kudenko, AAMAS 2008; [3]Yang, Lyu, Liu, and Gustafson IJCAI 2018; [5]Jiang, Yang, Zhang, and Stone, IROS 2019;
[6]Eppe, Nguygen, and Wermter, Front. in Rob. and AI 2019; 7Illanes, Yan, Icarte, and McIlraith ICAPS 2020

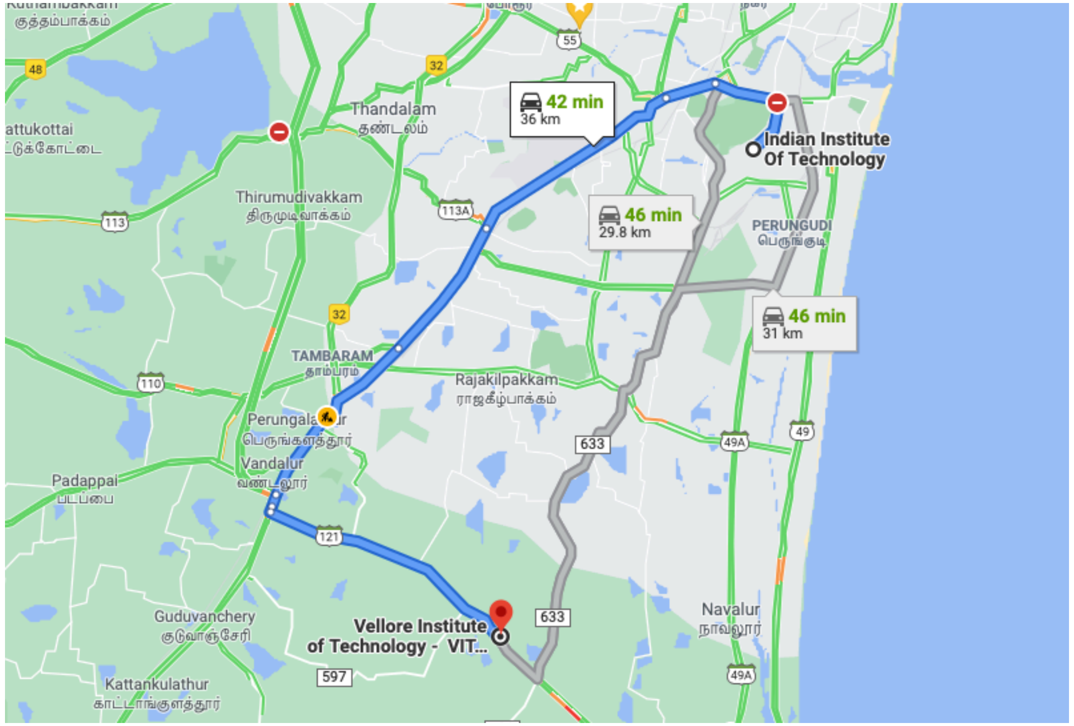
Motivation

Relational domains



Motivation

Abstract Representations

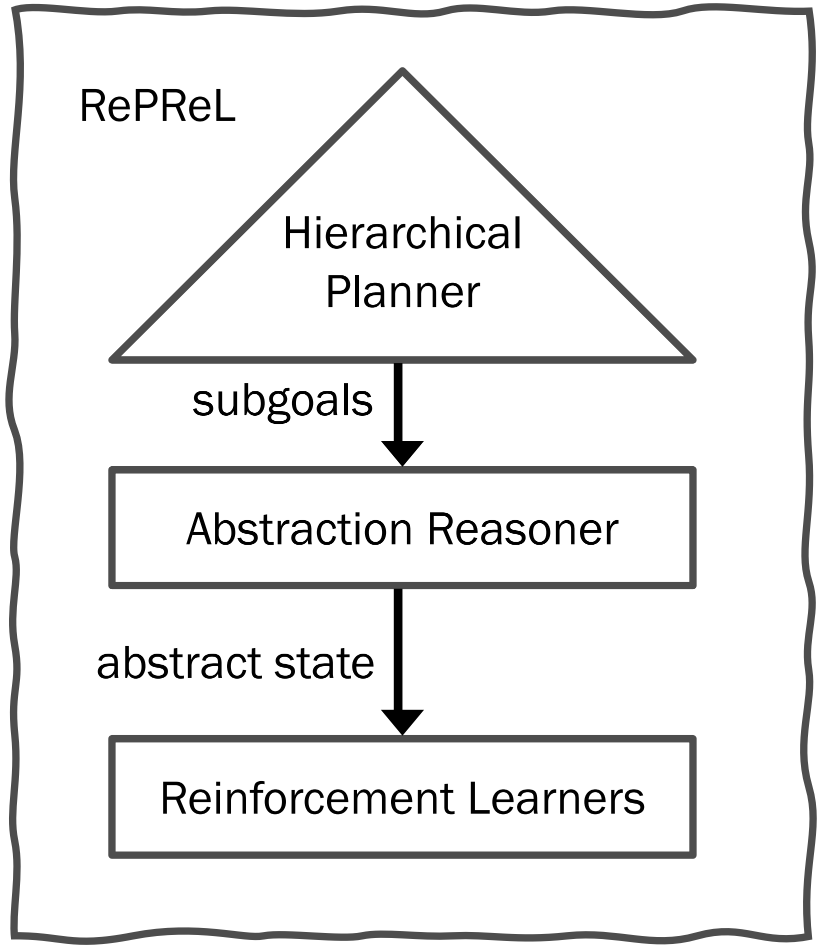


Planning

Execution

RePReL

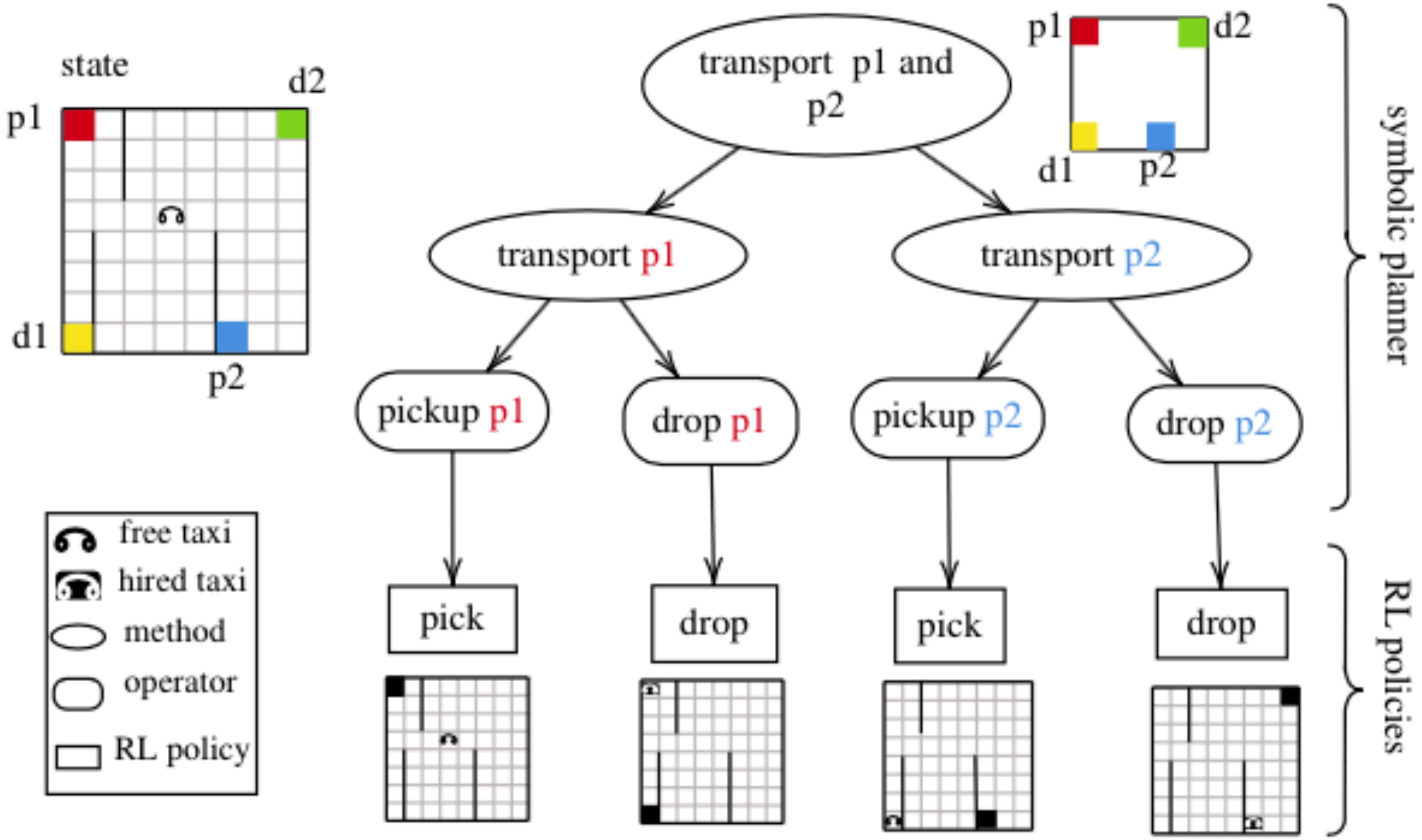
- Plan the sequence of high level subgoals and learn to execute each subgoal at lower level



Kokel et al ICAPS 2021

RePReL

Goal directed relational MDP:
<S, A, P, R, γ, G>



RePReL

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 γ. 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:*

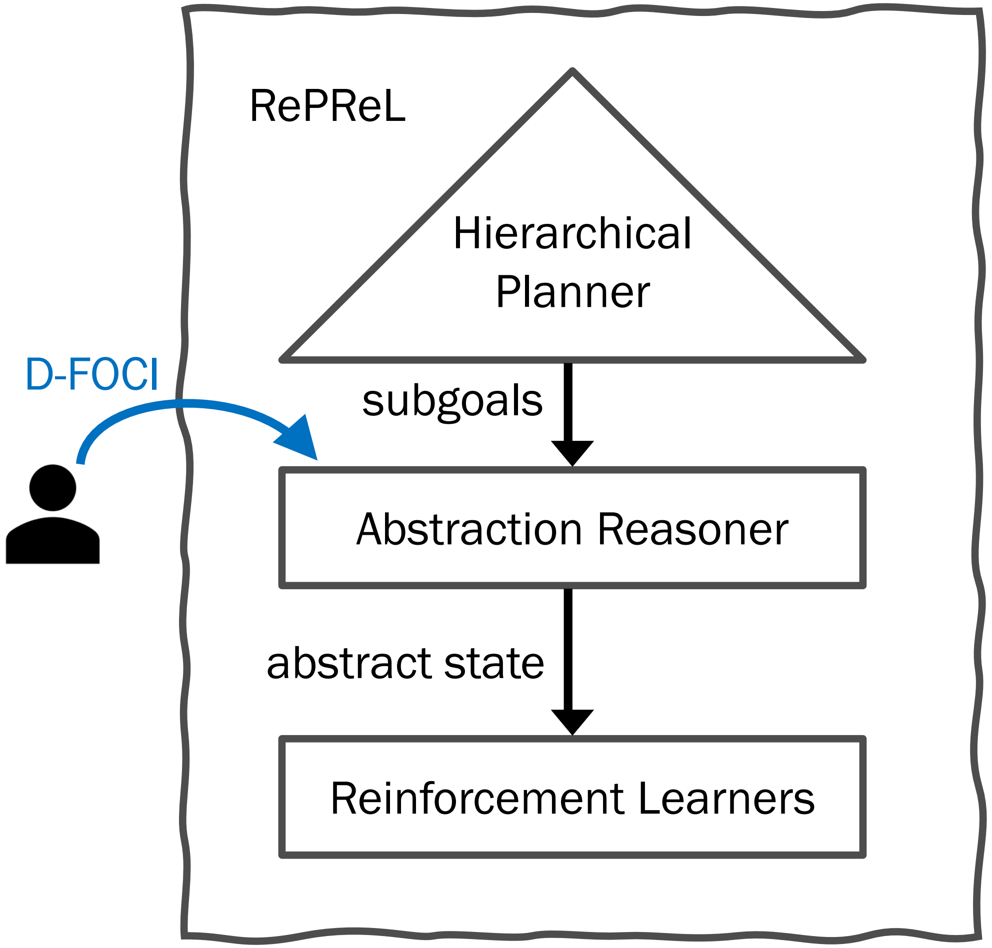
$$R_o(s, a, s') = \begin{cases} t_R + R(s, a, s') & \text{if } s' \in \beta(o) \text{ and } s \notin \beta(o) \\ 0 & \text{if } s' \in \beta(o) \text{ and } s \in \beta(o) \\ R(s, a, s') & \text{otherwise} \end{cases}$$

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

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

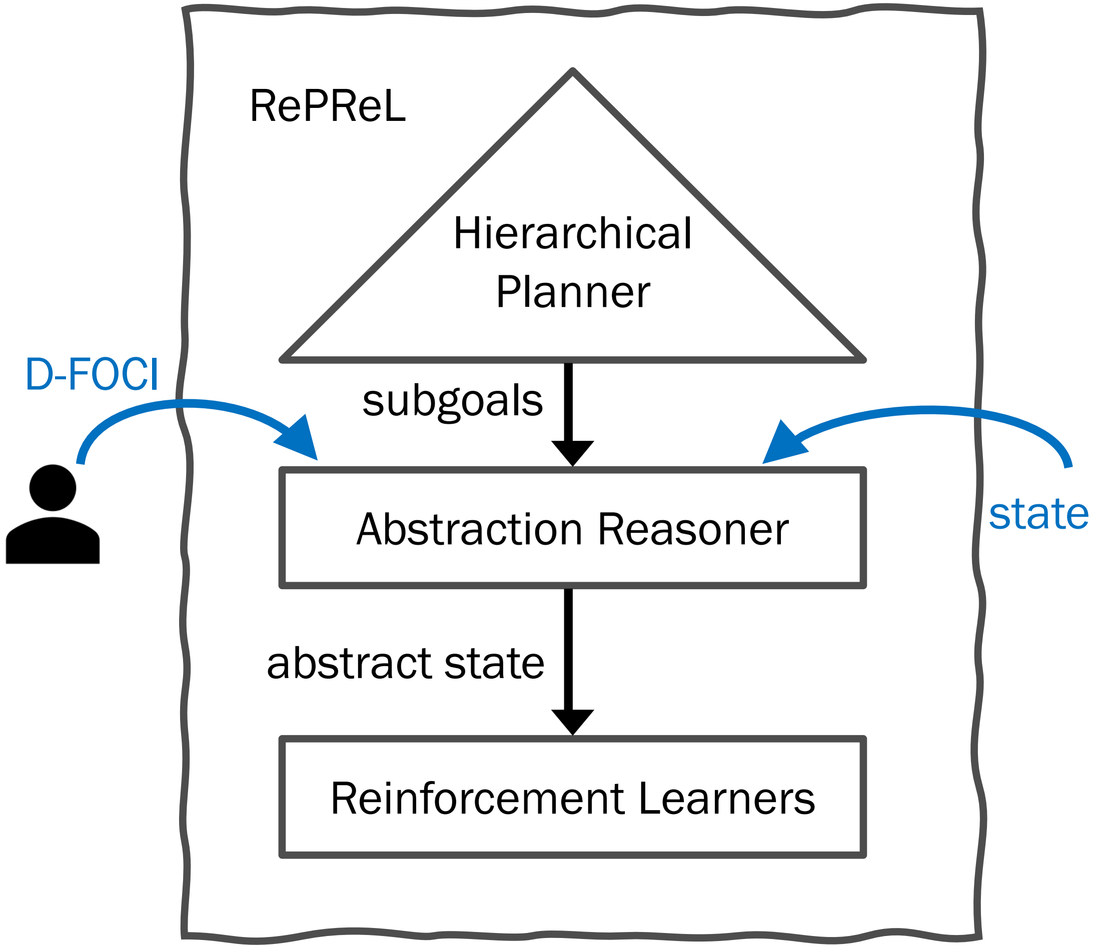
D-FOCI

First Order Conditional Influence (FOCI) statements

if $condition$ then X1 influence X2

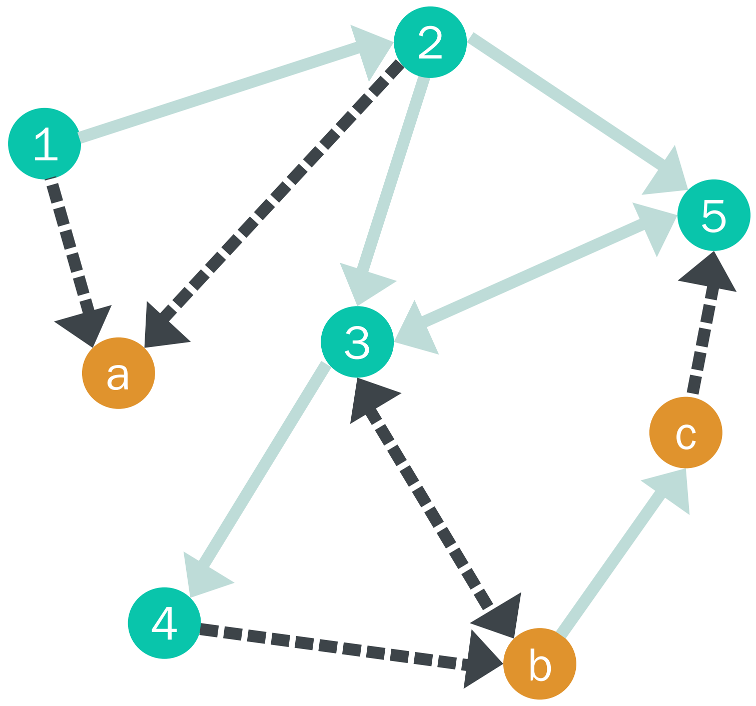
Dynamic FOCI statements

$$\text{sub-task} : X1 \xrightarrow{+1} X2$$

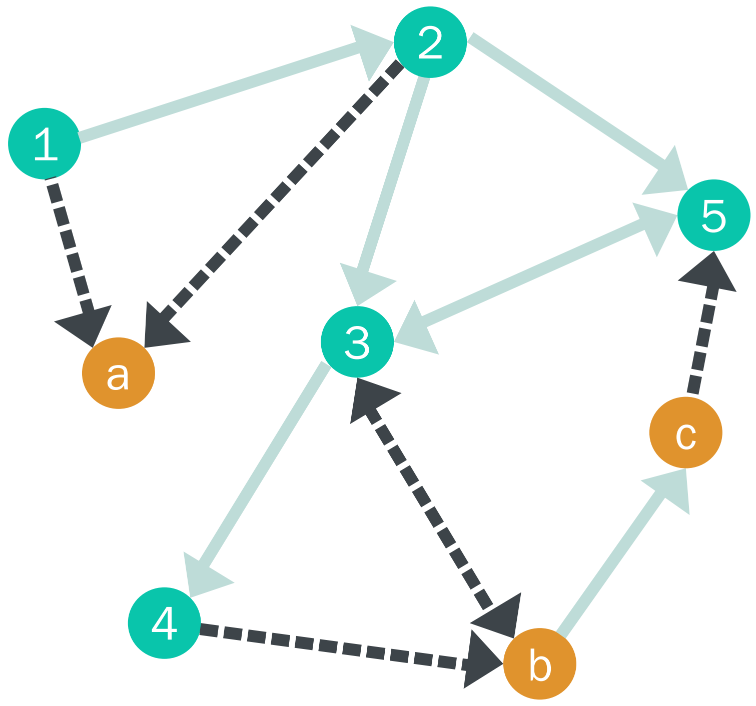


Natarajan, Tadepalli, Dietterich, and Fern 2008

Example

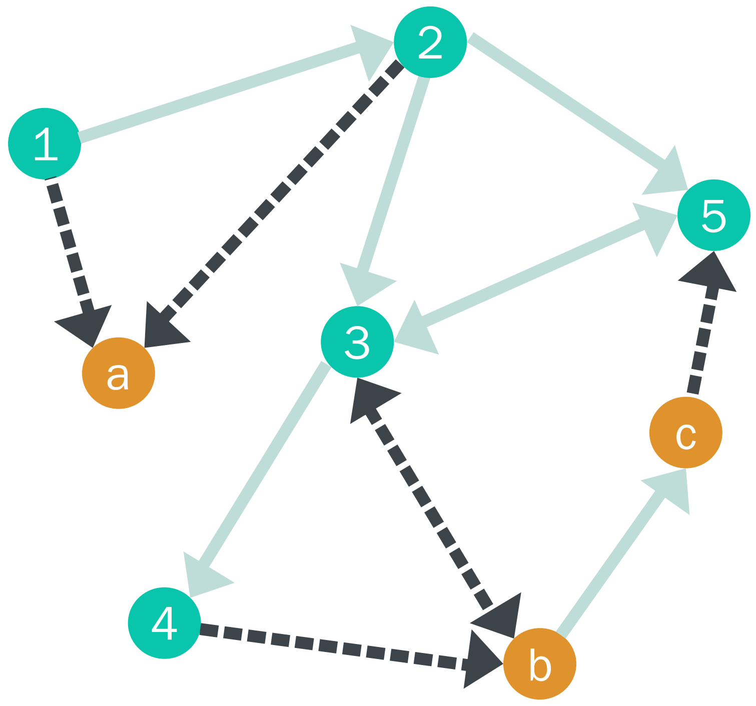


Example



task1(x):

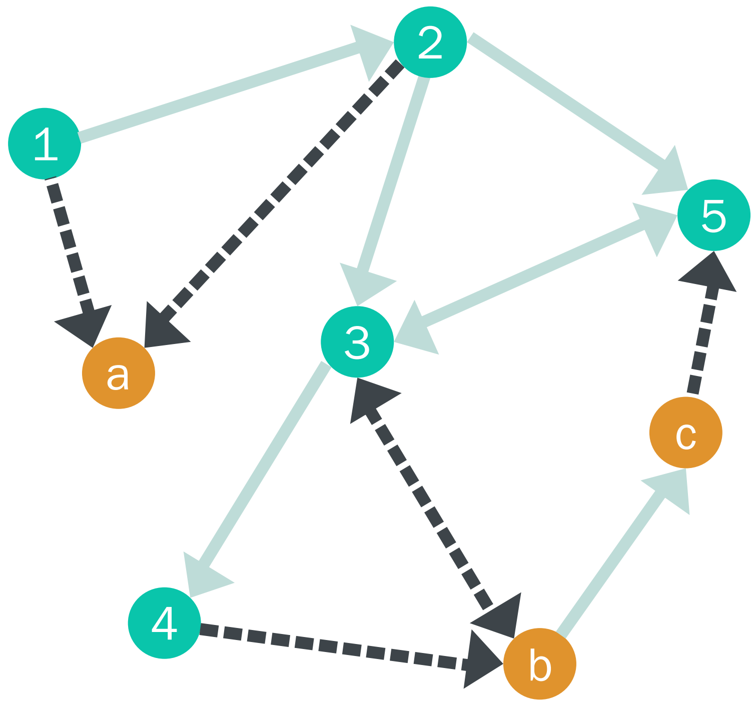
Example



task1(x): Y → X $\xrightarrow{+1}$ X

task1(x): X $\xrightarrow{+1}$ Reward

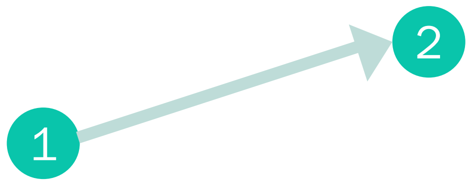
Example



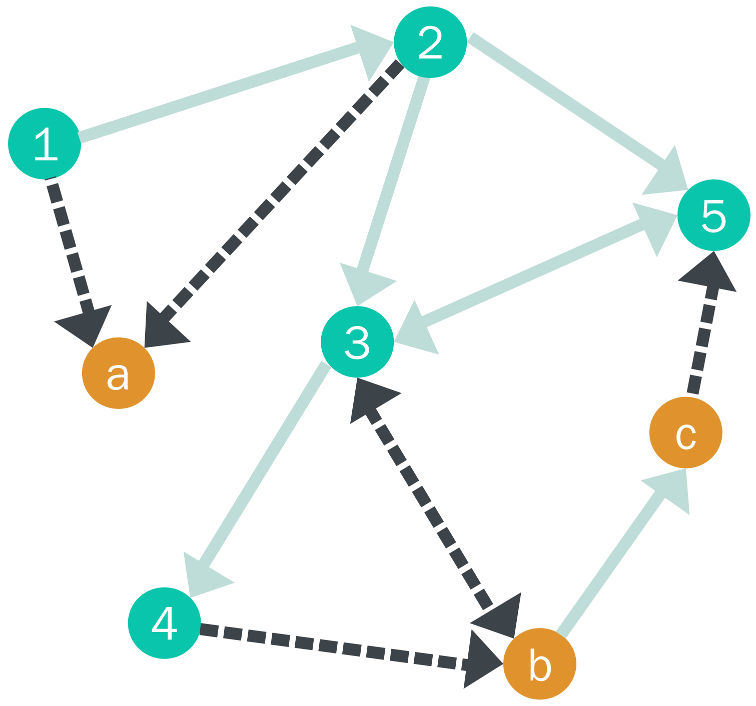
task1(x): $Y \rightarrow X \xrightarrow{+1} X$

task1(x): $X \xrightarrow{+1}$ Reward

Abstract state for task1(2):



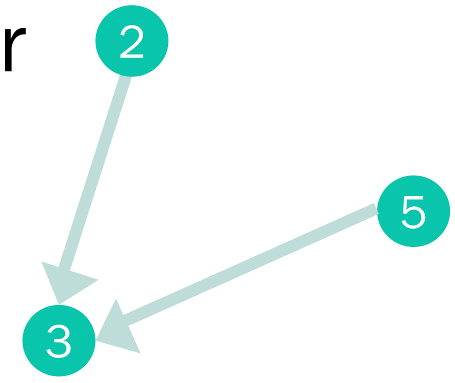
Example



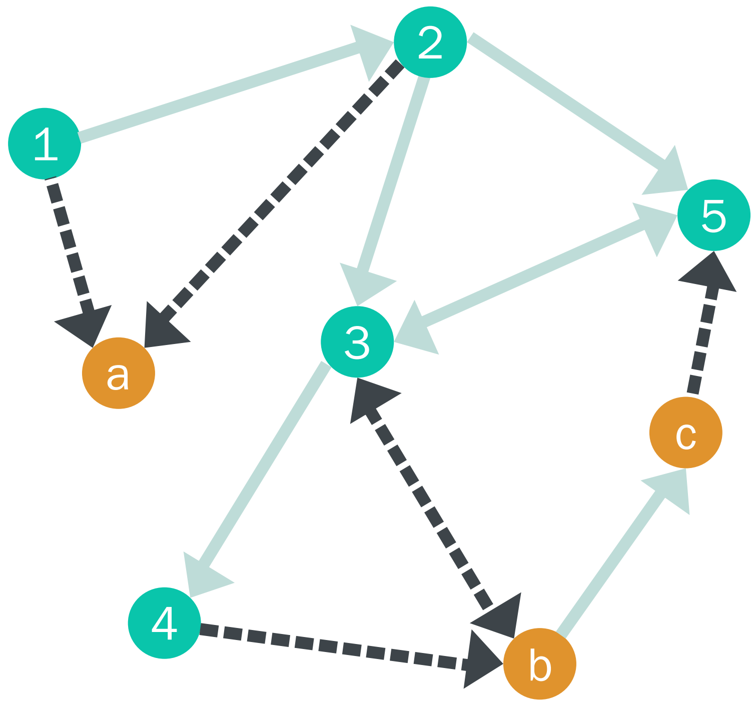
task1(x): Y $\rightarrow$ X $\xrightarrow{+1}$ X

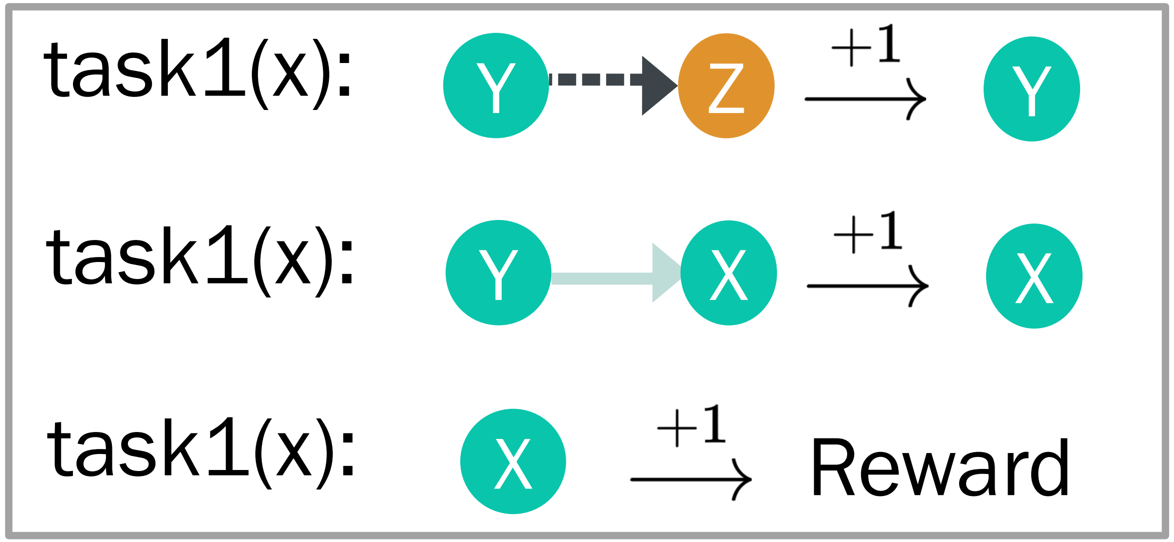
task1(x): X $\xrightarrow{+1}$ Reward

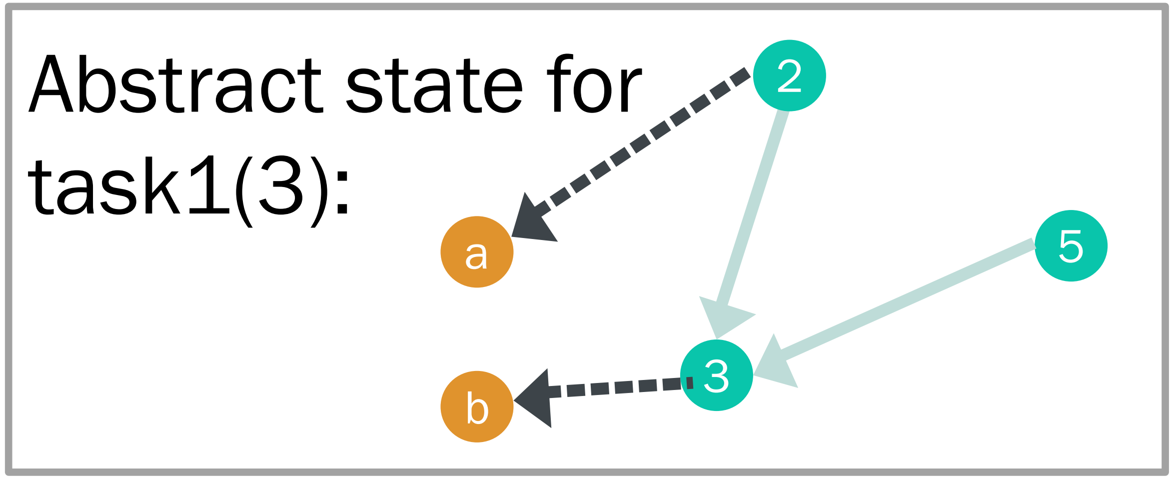
Abstract state for task1(3):



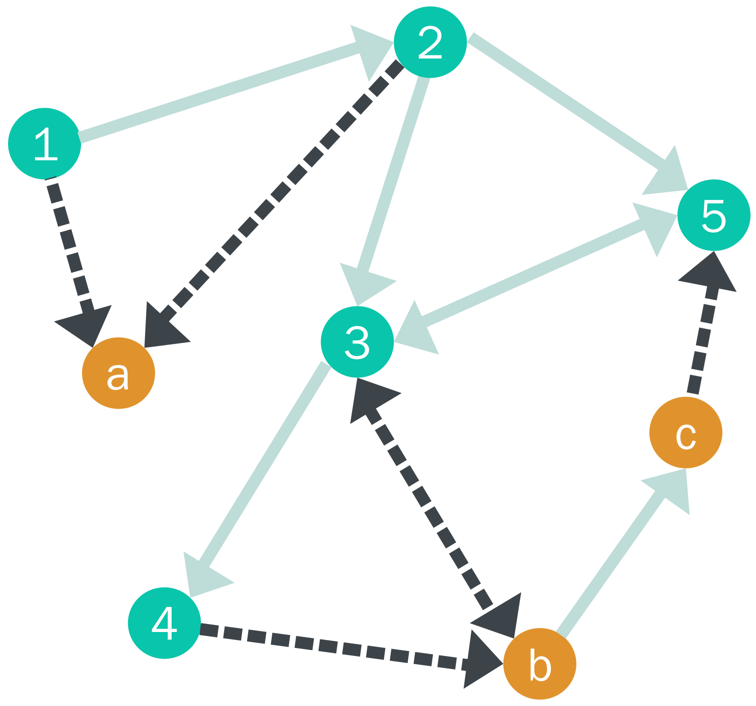
Example


1
2
3
4
5
a
b
c


task1(x): Y Z $\xrightarrow{+1}$ Y
task1(x): Y X $\xrightarrow{+1}$ X
task1(x): X $\xrightarrow{+1}$ Reward


Abstract state for task1(3):
2
a
5
3
b

Example

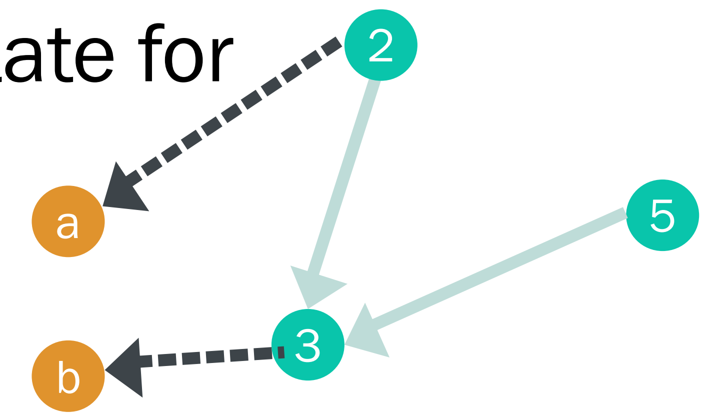


task1(x): Y ⇢ Z $\xrightarrow{+1}$ Y

task1(x): Y → X $\xrightarrow{+1}$ X

task1(x): X $\xrightarrow{+1}$ Reward

Abstract state for task1(3):



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*

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

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

Dietterich NeurIPS 2000; Ravindran and Barto IJCAI 2003;
Givan, Dean, and Greig AI 2003; Li, Walsh, and Littman ISAIM 2006

Abstraction

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

$$\sum_{\{s_1' \mid \phi(s_1') = \overline{s}\}} R_o(s_1, a, s_1') = \sum_{\{s_2' \mid \phi(s_2') = \overline{s}\}} R_o(s_2, a, s_2')$$

$$\sum_{\{s_1' \mid \phi(s_1') = \overline{s}\}} P_o(s_1, a, s_1') = \sum_{\{s_2' \mid \phi(s_2') = \overline{s}\}} P_o(s_2, a, s_2')$$

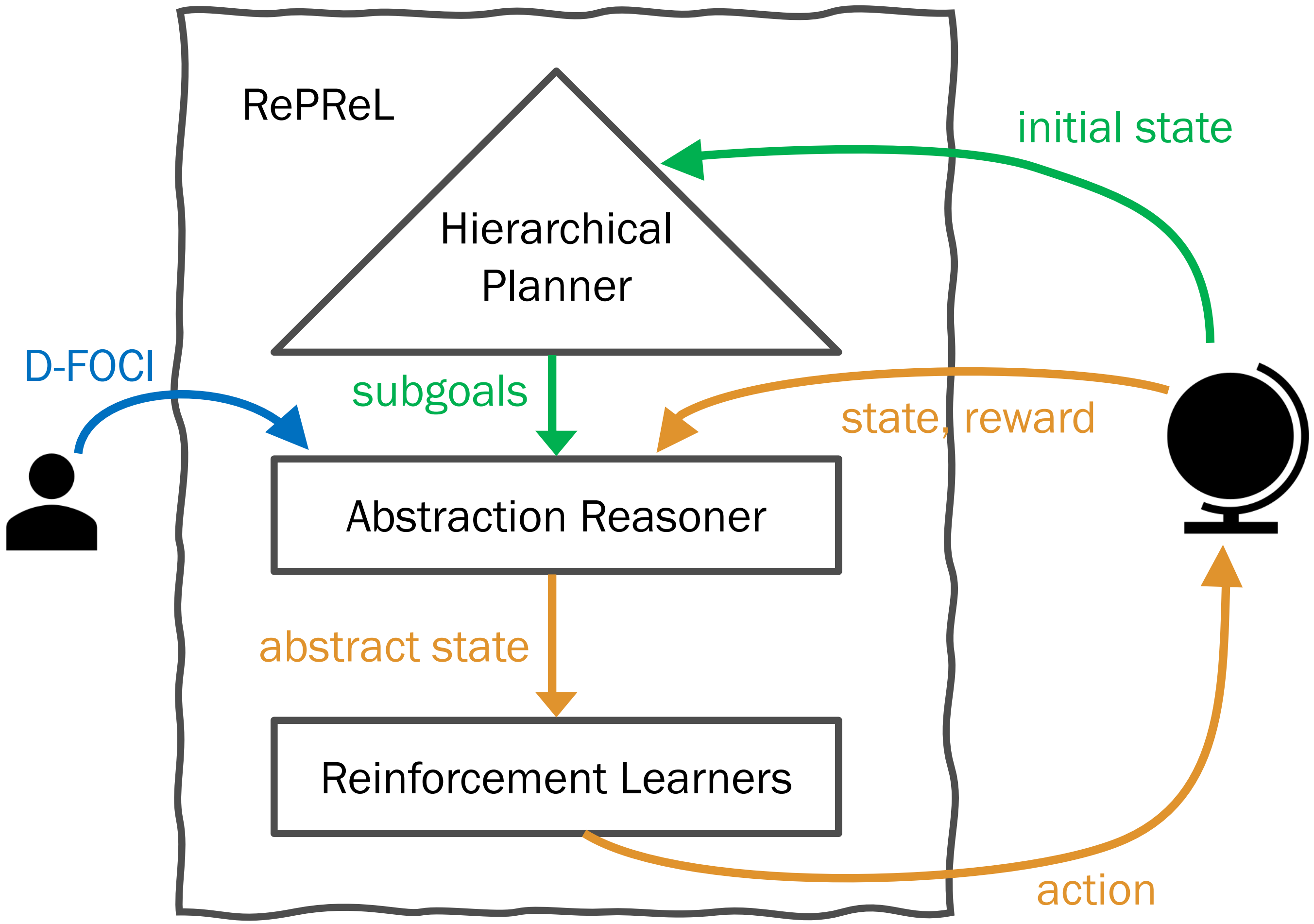
Dietterich NeurIPS 2000; Ravindran and Barto IJCAI 2003;
Givan, Dean, and Greig AI 2003; Li, Walsh, and Littman ISAIM 2006

Abstraction

$$\{\text{taxi-at(L1), move(Dir)}\} \xrightarrow{+1} \text{taxi-at(L2)}$$
$$\{\text{taxi-at(L1), move(Dir)}\} \longrightarrow R$$
$$\text{pickup(P):}$$
$$\quad \{\text{taxi-at(L1), at(P, L), in-taxi(P)}\} \xrightarrow{+1} \text{in-taxi(P)}$$
$$\text{pickup(P): in-taxi(P)} \longrightarrow R_o$$

D-FOCI for Taxi pickup task

RePReL Learning



- Get high level plan
- For each subgoal
  - *Loop till the s*
    - Get the a
    - Get the p
    - Take a ste
      next state
    - Update th
      state

RePReL Learning

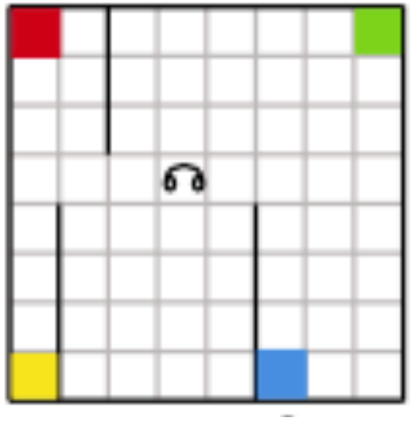
- Get high level plan
- For each subgoal
  - *Loop till the subgoal is achieved*
    - Get the abstract state
    - Get the policy for that subgoal
    - Take a step and observe reward, next state
    - Update the policy using **abstract state**

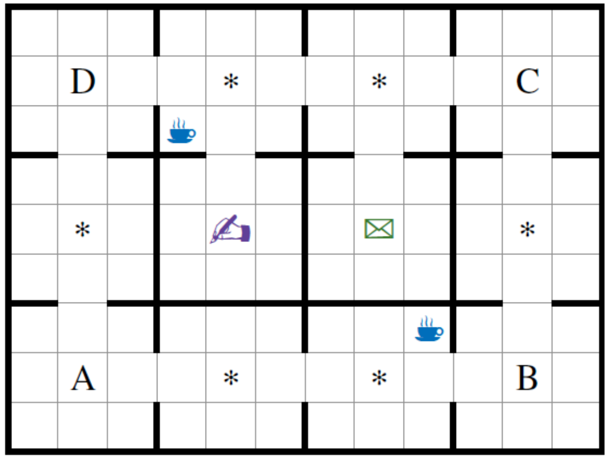
Algorithm 1 RePReL Learning Algorithm

INPUT: $\mathfrak{P}(O, M)$, goal set g, env, t_R, F
OUTPUT: RL policies $\pi_o, \forall o \in O$

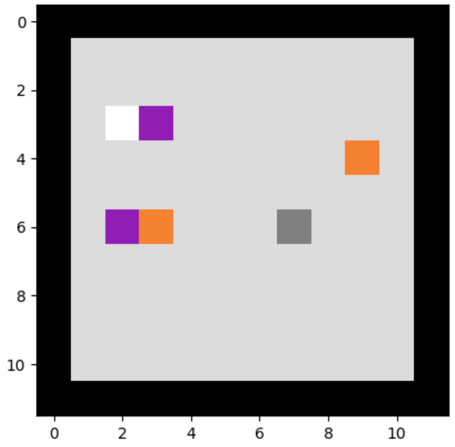
$\pi_o \leftarrow 0, \forall o \in O$ ▷ initialize RL policy for each operator
for each episode **do**
  $s \leftarrow$ get state from env
  $\Pi \leftarrow \mathfrak{P}(s, g)$ ▷ get high-level plan
  for o_g in Π **do**
    $\pi \leftarrow \pi_o$ ▷ get resp. RL policy
    $\hat{s} \leftarrow \text{GetAbstractState}(s, o_g, F)$
    done $\leftarrow \hat{s} \in \beta(o_g))$ ▷ check terminal state
    while not done **do**
      $a \leftarrow \pi(\hat{s})$ ▷ get action
      $s' \leftarrow \text{env.step}(a)$ ▷ take step in env
      $r \leftarrow R(s, a, s')$ ▷ get step reward
      $\hat{s}' \leftarrow \text{GetAbstractState}(s, o_g, F)$
      done $\leftarrow \hat{s}' \in \beta(o_g)$ ▷ check terminal next state
      if done **then**
        $r = r + t_R$ ▷ add terminal reward
      end if
      $\pi.update(\hat{s}, a, \hat{s}', r)$ ▷ update policy
      $s, \hat{s} \leftarrow s', \hat{s}'$
    end while
  end for
end for
return $\pi_o, \forall o \in O$

Experiments



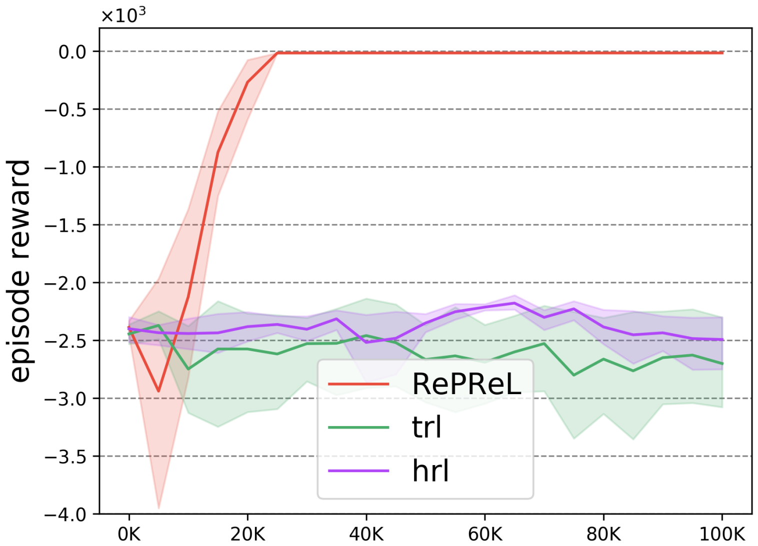


- Domains
  - *Office World*
  - *Craft World*
  - *Relational Taxi*
  - *Relational Box World*
- Baselines
  - *HRL (options framework)*
  - *TRL (Taskable RL, Illanes et al. 2020)*

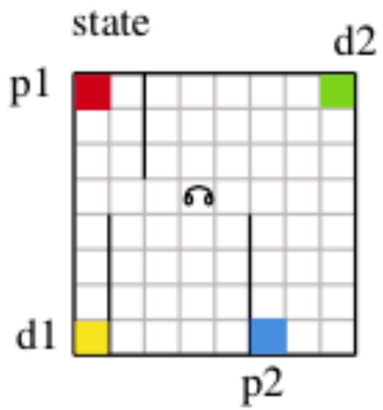


Experiments

- Sample efficiency
- Transfer across task
- Generalization across objects



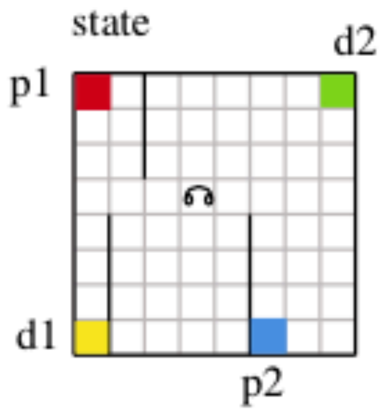
transport p1

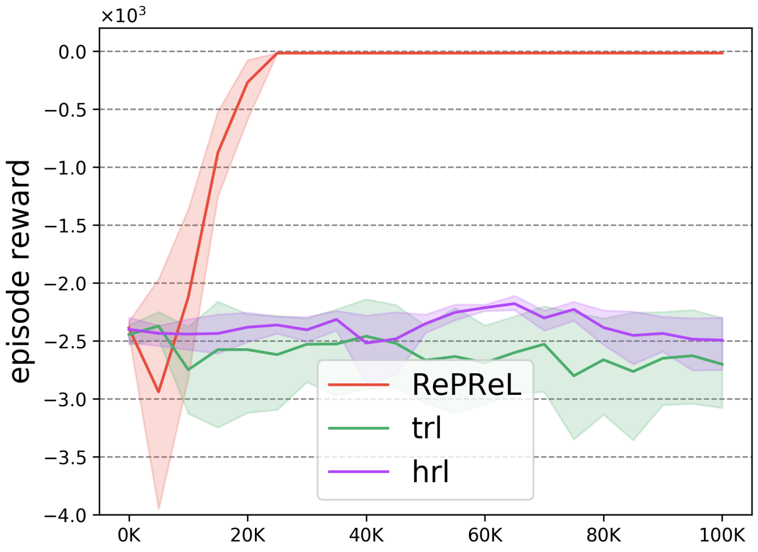


trl: Taskable RL (Illanes et al. ICAPS 2020)

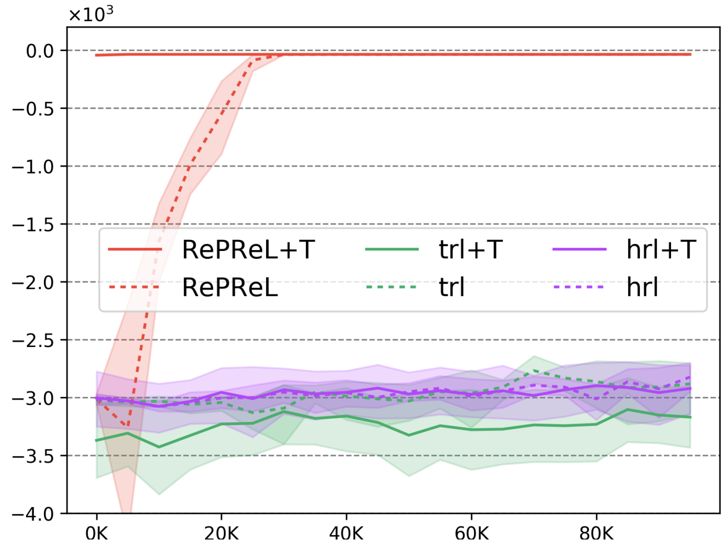
Experiments

- Sample efficiency
- Transfer across task
- Generalization across objects





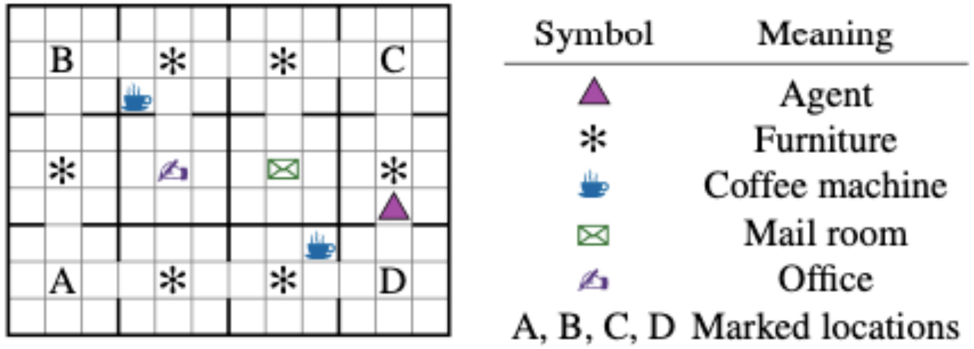
transport p1

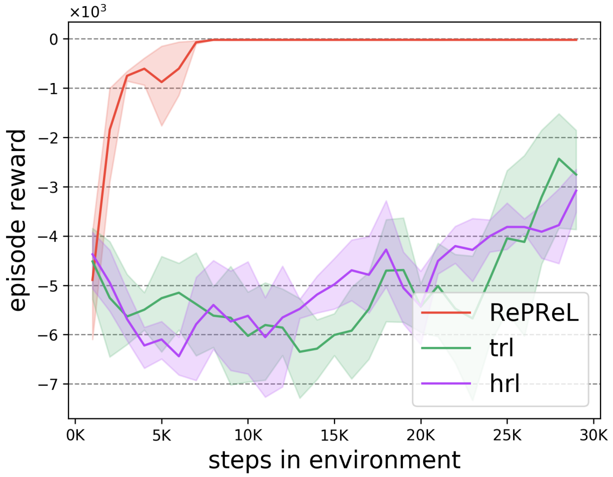


transport p1 and p2

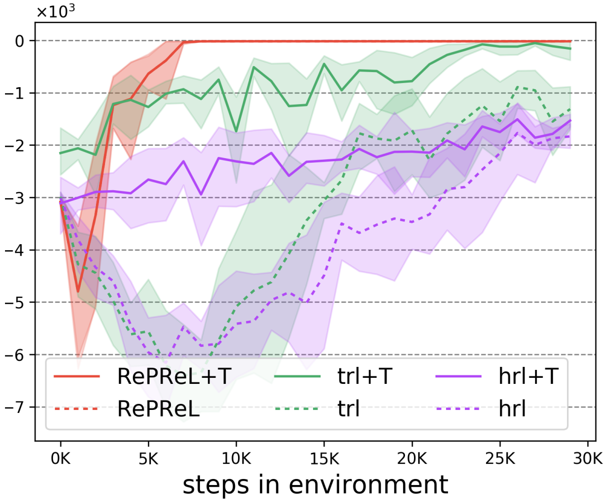
RePReL

Office World

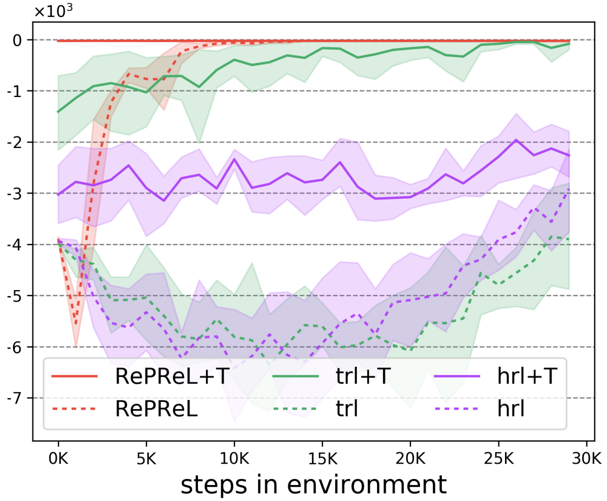




Deliver mail



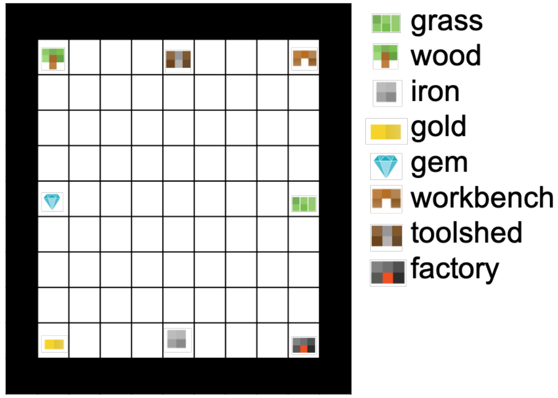
Deliver coffee

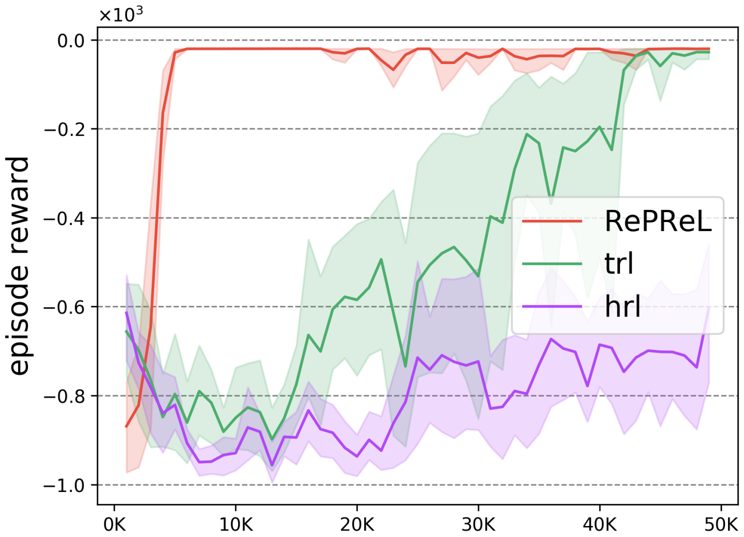


Deliver mail and coffee

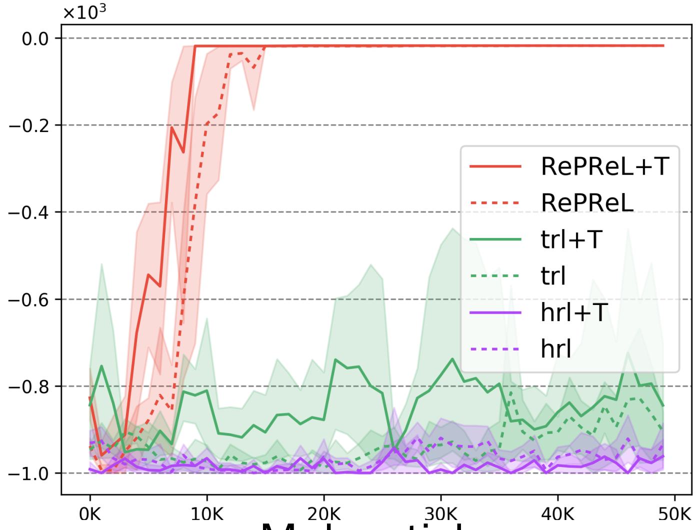
RePReL

CRAFT WORLD





Get wood and iron

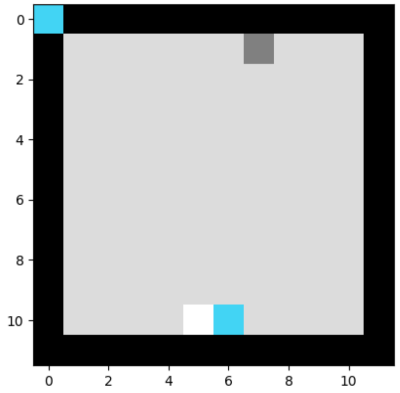


Make stick
(at workbench, needs wood)

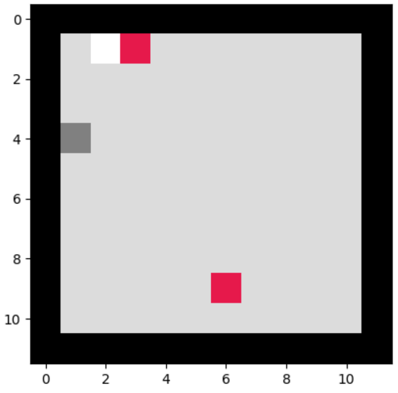
RePReL

Relational Box World

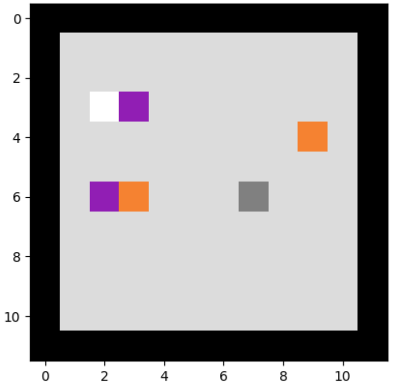
Generalization across objects



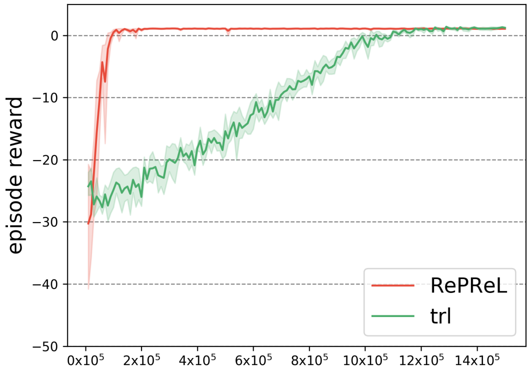
Open lock

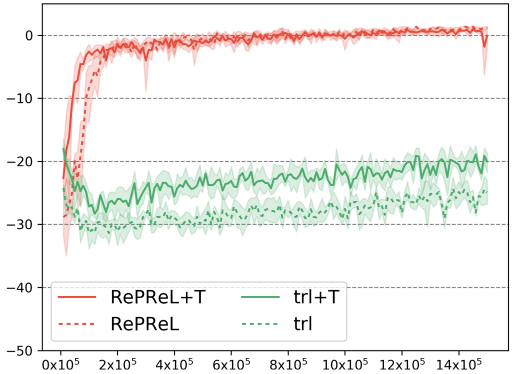


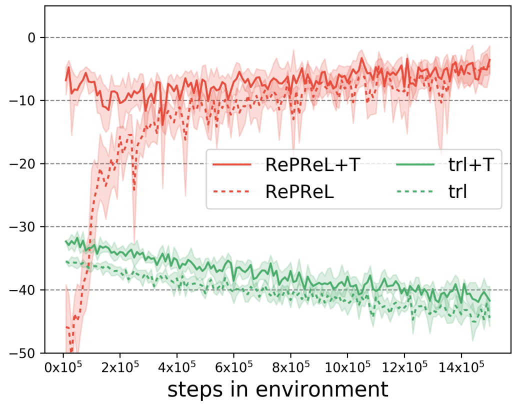
Collect key and open lock



Collect key and
open 2 locks







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 IJCLR 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)

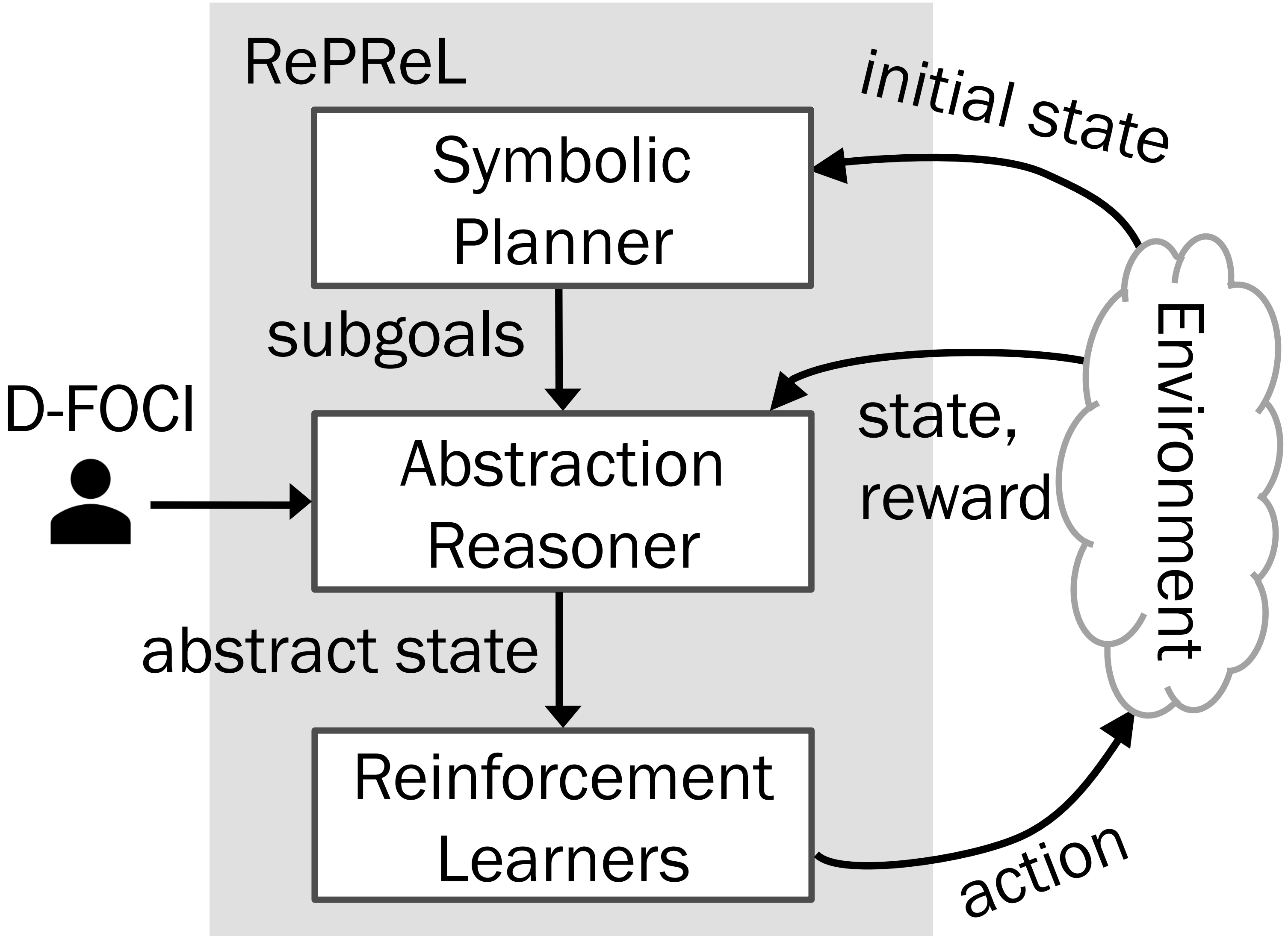
QUESTIONS?

StARLinG LAB

THANKS

starling.utdallas.edu
harshakokel.com

@starling_lab
@harsha_kokel


RePReL
initial state
Symbolic Planner
subgoals
D-FOCI
Abstraction Reasoner
state, reward
Environment
abstract state
Reinforcement Learners
action

Abstraction

Given

State

$$\{\text{at}(\text{p1},\text{r}), \text{taxi-at}(13), \text{dest}(\text{p1},\text{y}), \neg\,\text{at-dest}(\text{p1}), \neg\,\text{in-taxi}(\text{p1}),$$
$$\text{at}(\text{p2},\text{b}), \text{dest}(\text{p2},\text{g}), \neg\,\text{at-dest}(\text{p2}), \neg\,\text{in-taxi}(\text{p1})\}$$

subtask

$$\langle\ \text{pickup}(P), \{P/p1, L/r\}\ \rangle$$

D-FOCI

$\{\text{taxi-at(L1), move(Dir)}\} \xrightarrow{+1} \text{taxi-at(L2)}$

$\{\text{taxi-at(L1), move(Dir)}\} \longrightarrow R$

pickup(P):

$\{\text{taxi-at(L1), at(P, L), in-taxi(P)}\} \xrightarrow{+1} \text{in-taxi(P)}$

pickup(P): $\text{in-taxi(P)} \longrightarrow R_o$

Get

Abstract state

$$\{\text{at}(\text{p1},\text{r}), \text{taxi-at}(13), \neg\,\text{in-taxi}(\text{p1}), \text{move}(\text{Dir})\}$$