Hierarchical Reinforcement Learning with AI Planning Models

GenPlan 2023: Seventh Workshop on Generalization in Planning
December 16, 2023

Junkyu Lee junkyu.lee@ibm.com
Michael Katz Michael.katz1@ibm.com
Don Joven Agravante don.joven.r.agravante@ibm.com
Miao Liu miao.liu1@ibm.com
Geraud Nangue Tasse geraudnt@gmail.com
Tim Klinger tklinger@us.ibm.com
Shirin Sohrabi ssohrab@us.ibm.com

Code and data are available at

https://github.com/IBM/parl_agents

https://github.com/IBM/parl_annotations

https://github.com/IBM/parl_minigrid

IBM

Motivation

AI Planning

Given a symbolic action model, finding plans can be done fast per problem basis

Pros: Exploration
Cons: Generalization

Deep RL

Given samples from an RL environment,
fit a neural network that can generalize in high-dimensional state space

Pros: Generalization
Cons: Sample efficiency

Integrating PL/RL

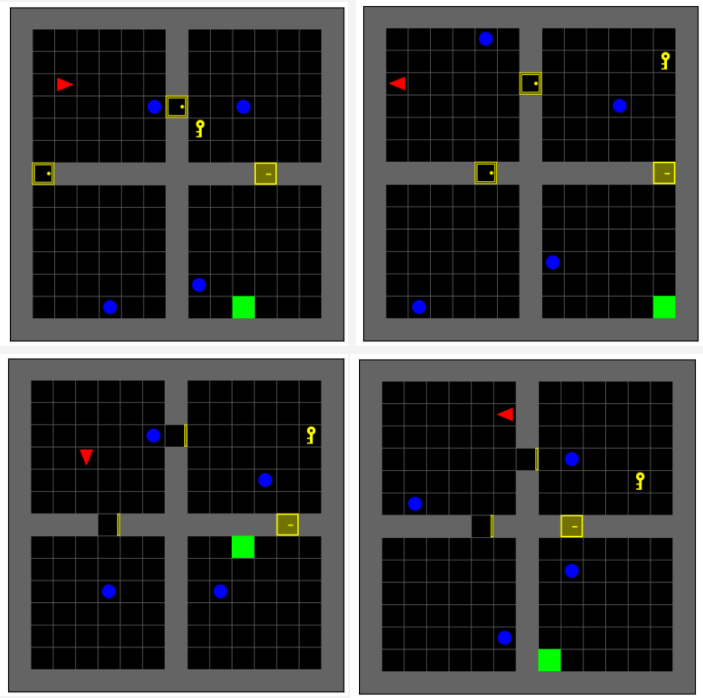
Given many environments shares high-level task structure,
Annotate high-level task as AI planning model.

Fit neural networks, one per action operators in the planning task

Generalization + Sample Efficiency

RL tasks with variations

2 x 2 MiniGrid with a Locked Door



RL State Space
- Image-like representation of MiniGrid

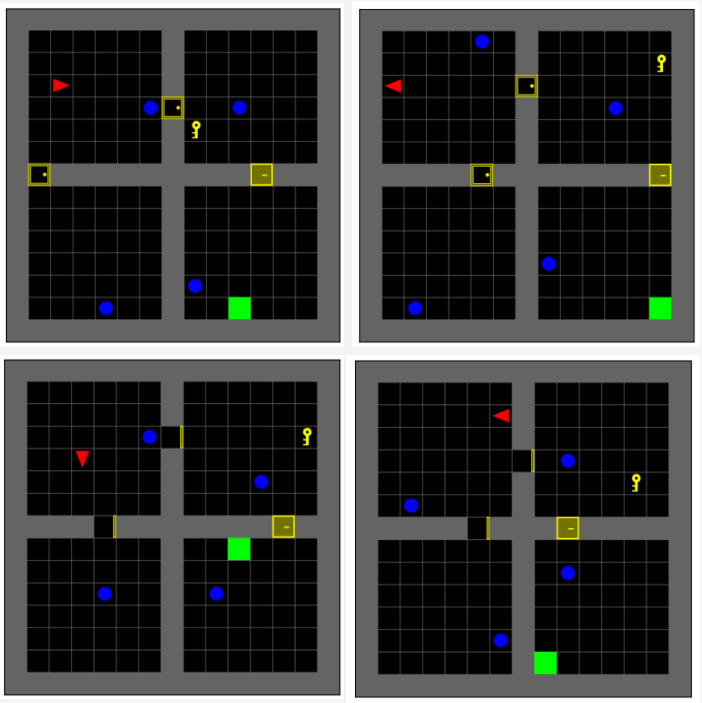
RL Action Space
- Turn left, right
- Move forward
- Interact with object
- Pickup, Drop an object

RL Rewards
- +1 reaching green cell (goal location)

RL Tasks variations
- location of Balls
- location of doors, door open/closed states
- location of green goal location
- location of keys in a room
- initial location of agent

AI Planning Task

2 x 2 MiniGrid with a Locked Door

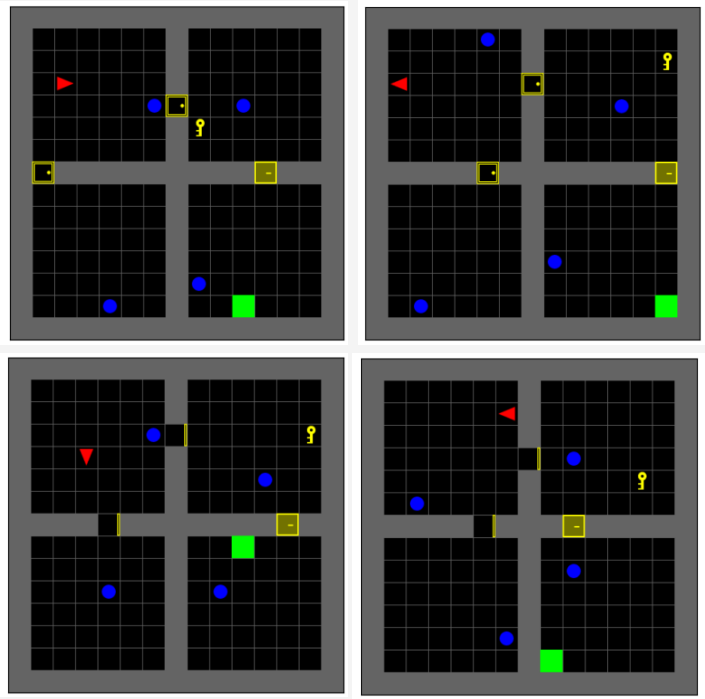


Planning Task: move to a room with a goal location

- Movement between rooms
  - abstract away exact locations
  - blue balls are not relevant, ignore balls
- Interaction with keys and doors
  - pick up key to open locked doors
  - doors can be open/closed/locked
  - planning task only lock/unlock doors
- Objects
  - Room, Key, Door
- Predicate
  - (at-agent ?room), (at ?key ?room)
  - (carry ?key), (empty-hand)
  - (locked ?door), (unlocked ?door)
  - other static predicates encoding room connectivity, key and door related information

AI Planning Model as Annotation to RL Tasks

2 x 2 MiniGrid with a Locked Door



Action schema in PDDL (Planning Domain Definition Language)

```
(:action move-room
    :parameters (?d - door ?r1 - room ?r2 - room)
    :precondition (and
        (CONNECTED-ROOMS ?r1 ?r2)
        (at-agent ?r1)
        (LINK ?d ?r1 ?r2)
        (unlocked ?d)
    )
    :effect (and
        (not (at-agent ?r1))
        (at-agent ?r2)
    )
)
```

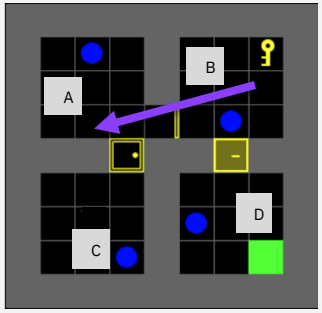
```
(:action pickup
    :parameters (?k - key ?r - room)
    :precondition (and
        (at ?k ?r)
        (at-agent ?r)
        (empty-hand)
    )
    :effect (and
        (not (at ?k ?r))
        (not (empty-hand))
        (carry ?k)
    )
)
```

```
(:action drop
    :parameters (?k - key ?r - room)
    :precondition (and
        (carry ?k)
        (at-agent ?r)
    )
    :effect (and
        (at ?k ?r)
        (empty-hand)
        (not (carry ?k))
    )
)
```

```
(:action lock
    :parameters (?k - key ?d - door ?r1 - room ?r2 - room)
    :precondition (and
        (CONNECTED-ROOMS ?r1 ?r2)
        (at-agent ?r1)
        (LINK ?d ?r1 ?r2)
        (carry ?k)
        (unlocked ?d)
        (KEYMATCH ?k ?d)
    )
    :effect (and
        (locked ?d)
        (not (unlocked ?d))
    )
)
```

```
(:action unlock
    :parameters (?k - key ?d - door ?r1 - room ?r2 - room)
    :precondition (and
        (CONNECTED-ROOMS ?r1 ?r2)
        (at-agent ?r1)
        (LINK ?d ?r1 ?r2)
        (carry ?k)
        (locked ?d)
        (KEYMATCH ?k ?d)
    )
    :effect (and
        (not (locked ?d))
        (unlocked ?d)
    )
)
```

Mapping HRL Options and Planning Action



Grounded PDDL action

```
(:action move-room
    :parameters (door-AB, room-B, room-A)
    :precondition (and
        (CONNECTED-ROOMS room-B room-A)
        (LINK door-AB room-B room-A)
        (at-agent room-B)
        (unlokced door-AB))
    :effect (and
        (not (at-agent room-B)
        (at-agent room-A))
```

HRL option from planning action

$$\text{Init} = \{s \mid L(s) \in \text{pre}(o_{PL})\ \forall s \in S\}$$

$$\text{Term} = \{s \mid L(s) \in \text{post}(o_{PL})\ \forall s \in S\}$$

PL state:
{(at-agent room-B),
(unlocked door-AB),
(empty-hand),
(at k1 room-B)}

L(s) maps from RL state to PL state

Option transition

$L(s)$

PL state:
{(at-agent room-A),
(unlocked door-AB),
(carry k1)}

Intrinsic reward penalizing side-effects

PL action: move-room (door-AB, room-B, room-A)

Initiate HRL Option

Terminate HRL Option

PL state:
{(at-agent room-A),
(unlocked door-AB),
(empty-hand),
(at k1 room-B)}

Rotate right

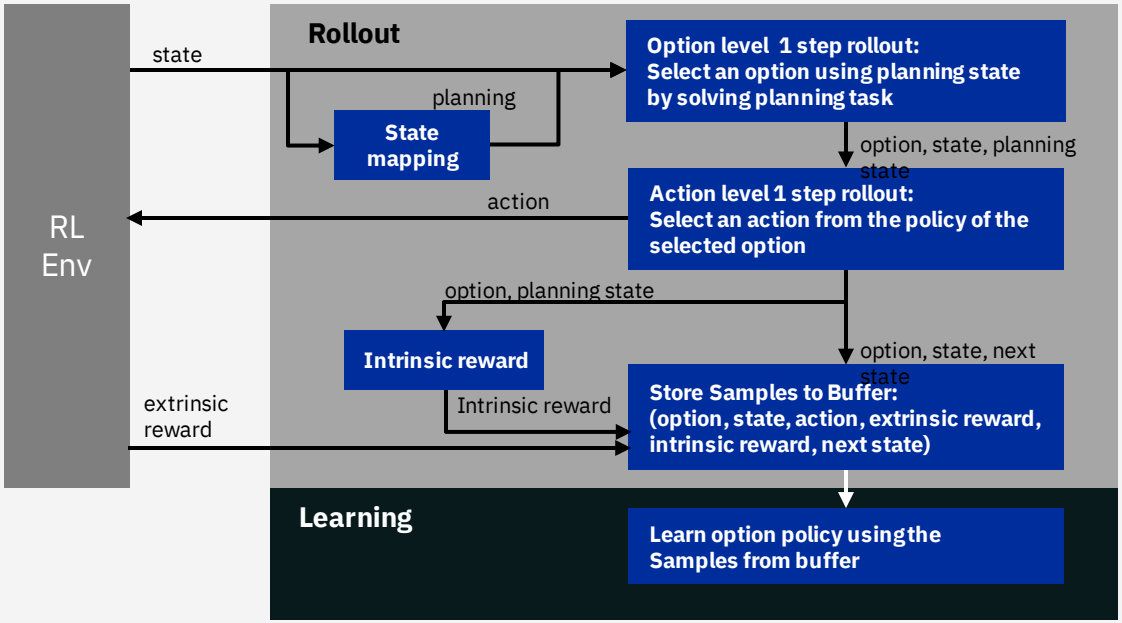
forward forward

Plan Option Learning Algorithm

Algorithm 1 Online Option Learning

Require: Planning task II, Goal-oriented MDP $\mathcal{M}$, State mapping function L
Ensure: Option policies $\pi_{O^*{}_o}(a|s)$.

Initialize trajectory buffer B
Initialize a set D for storing options
while $iter < N$ **do**
  // rollout samples from the current option policy
  while $iter_{\text{rollout}} < N_{\text{rollout}}$ **do**
    $s \leftarrow$ current state
    Select an option $O^*{}_o$ from a plan starting in $L(s)$
    if $O^*{}_o \notin D$ **then**
      Create plan option $O^*{}_o$
      Initialize $\pi_{O^*{}_o}$ and Add $O^*{}_o$ to D
    while $s \notin \beta_{O^*{}_o}$ **do**
      Sample (s, a, r_e, u) from RL environment
      Compute intrinsic reward r_i
      Store $(O^*{}_o, s, a, r_e, r_i)$ to buffer B
      $s \leftarrow u$
  // train option policies with model-free algorithms
  for each option $O^*{}_o \in D$ **do**
    Train option policy function $\pi_{O^*{}_o}$ with RL



Intrinsic Rewards

Definition 7 (frame penalized option sub-MDP) Given a goal-oriented MDP $\mathcal{M}$ and an operator option derived from a planning task $O^*_o := \langle \mathcal{I}_{O^*_o}, \pi_{O^*_o}, \beta_{O^*_o} \rangle$, a *frame penalized option sub-MDP* is a sub-MDP of $\mathcal{M}$. $\overline{\mathcal{M}}_{o,s_0} := \{\mathcal{S}, \mathcal{A}, P, \overline{r}, \overline{s}_0, \beta_{O^*_o}, \gamma\}$, where we replace the reward function in $\mathcal{M}$ with an intrinsic reward $\overline{r} := \mathcal{S} \times \mathcal{S} \to \mathbb{R}$,

$$\overline{r}(s, s') = \sum_{v \in \mathcal{V}\left(\mathcal{F}_{O^*_o}(s)\right)} c_1 \cdot \mathbb{I}\left(L(s')[v] \neq \mathcal{F}_{O^*_o}[v]\right) + c_2 \cdot \mathbb{I}\left(s' \notin \beta_{O^*_o}\right), \quad (1)$$

where $\mathbb{I}$ is an indicator function, c_1 and c_2 are negative rewards, and the initial state and the goal with a $\overline{s}_0 \in \mathcal{I}_{O^*_o}$, and $\beta_{O^*_o}$, respectively.

- Penalty per violation of “frame constraints” at symbolic planning level
- Cost per not reaching the termination set of an option

Experiments

Algorithms

- HRL with AI planning and PPO: Hplan PPO
- HRL with reward machines: HRM (Icarte et al 2022)
- Flat RL
  - DDQN, PPO
  - Rainbow, PPO with curiosity

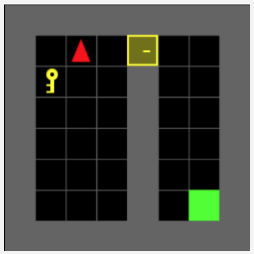
Multi-task evaluation

- Training set: 10^6 RL tasks
- Test set: Unseen 10^3 RL tasks
- Report test set performance (evaluate unseen tasks)

Test Environments

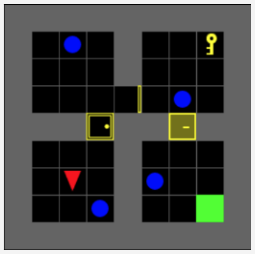
More difficult

Door Key



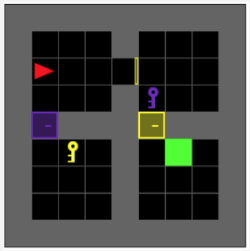
- One of task in gym-minigrid
- No side-effects

2 x2 Locked



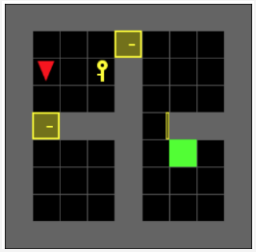
- Side-effects introduced
- Distracting balls
- Plan len 4~5

2x2 Two Keys



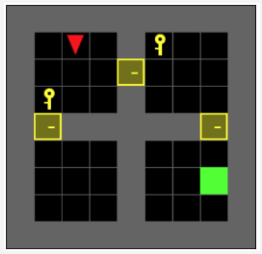
- Plan len 11

2x2 One Disposable Key



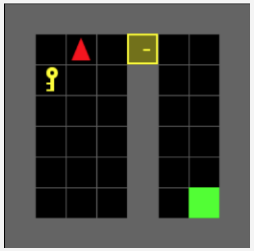
- Dead end introduced
- Plan len 4

2x2 Two Disposable Keys

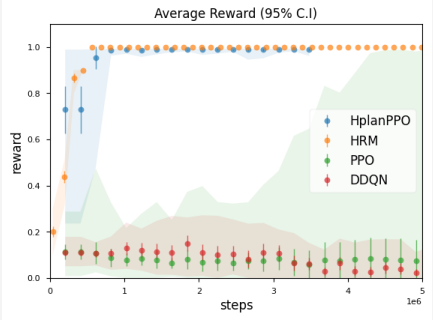


- Dead end with Keys
- Plan len 7

Door Key

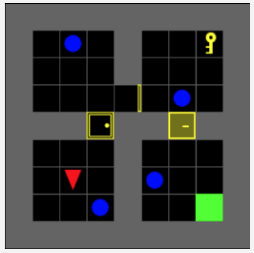


```
(pickup k-yellow-0 r-0-0)
(unlock k-yellow-0 d-yellow-0-0-1-0 r-0-0 r-1-0)
(move-room d-yellow-0-0-1-0 r-0-0 r-1-0)
```

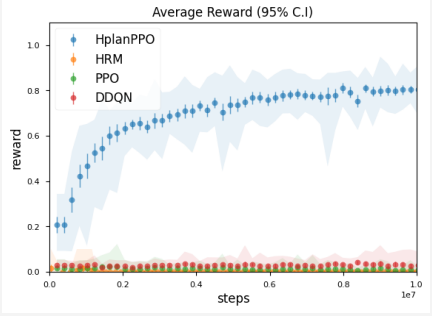


- Flat RL starts to fail
- Rainbow in RLLIB: 0.31

2 x2 Locked

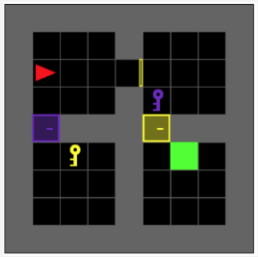


```
(move-room d-yellow-0-1-0-0 r-0-1 r-0-0)
(move-room d-yellow-0-0-1-0 r-0-0 r-1-0)
(pickup k-yellow-0 r-1-0)
(unlock k-yellow-0 d-yellow-1-0-1-1 r-1-0 r-1-1)
(move-room d-yellow-1-0-1-1 r-1-0 r-1-1)
```

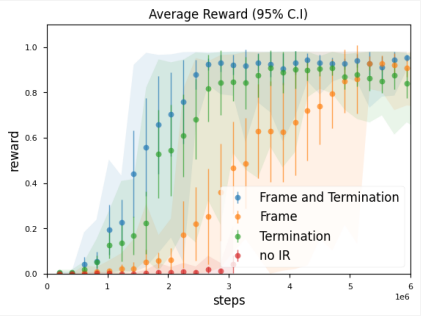


- Hierarchical reward machine (Icarte et al. 2022) starts to fail
- Rainbow in RLLIB: 0.09

2x2 Two Keys

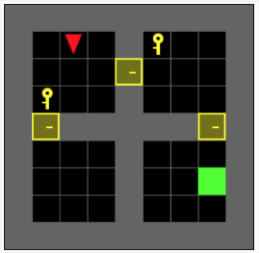


```
(move-room d-yellow-0-0-1-0 r-0-0 r-1-0)
(pickup k-purple-0 r-1-0)
(move-room d-yellow-0-0-1-0 r-1-0 r-0-0)
(unlock k-purple-0 d-purple-0-0-0-1 r-0-0 r-0-1)
(drop k-purple-0 r-0-0)
(move-room d-purple-0-0-0-1 r-0-0 r-0-1)
(pickup k-yellow-1 r-0-1)
(move-room d-purple-0-0-0-1 r-0-1 r-0-0)
(move-room d-yellow-0-0-1-0 r-0-0 r-1-0)
(unlock k-yellow-1 d-yellow-1-0-1-1 r-1-0 r-1-1)
(move-room d-yellow-1-0-1-1 r-1-0 r-1-1)
```

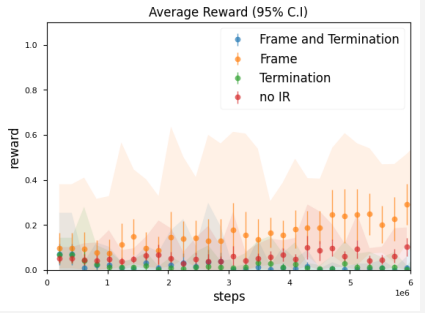


- ~ 2.5 x 10^6 steps needed to learn all necessary options
- Penalty (intrinsic reward) plays important role

2x2 One disposable Key



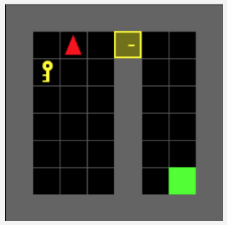
```
(pickup k-yellow-0 r-0-0)
(unlock k-yellow-0 d-yellow-0-0-1-0 r-0-0 r-1-0)
(move-room d-yellow-0-0-1-0 r-0-0 r-1-0)
(move-room d-yellow-1-0-1-1 r-1-0 r-1-1)
```



- HRL with PPO requires more than 6x 10^6 samples

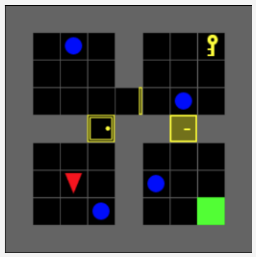
Problem Specific Models

Door Key



```
(define (problem MazeRooms-8by8-DoorKey)
    (:domain MazeRooms)
    (:objects
        R-0-0 R-1-0 -  room
        K-yellow-0 - key
        D-yellow-0-0-1-0 - door
    )
    (:init
        (LINK D-yellow-0-0-1-0 R-0-0 R-1-0)
        (LINK D-yellow-0-0-1-0 R-1-0 R-0-0)
        (KEYMATCH K-yellow-0 D-yellow-0-0-1-0)
        (at-agent R-0-0)
        (at K-yellow-0 R-0-0)
        (locked D-yellow-0-0-1-0)
        (empty-hand)
    )
    (:goal (and
        (at-agent R-1-0))
    )
)
```

2 x2 Locked



```
(define (problem MazeRooms-2by2-LockedSmall)
        (:domain MazeRooms)
        (:objects
            R-0-0 R-0-1 R-1-0 R-1-1 -  room
            K-yellow-0 - key
            D-yellow-0-0-1-0 D-yellow-1-0-1-1 D-yellow-0-0-0-1 - door
        )
        (:init
(CONNECTED-ROOMS R-0-0 R-0-1)
(CONNECTED-ROOMS R-0-0 R-1-0)
(CONNECTED-ROOMS R-0-1 R-0-0)
(CONNECTED-ROOMS R-1-0 R-0-0)
(CONNECTED-ROOMS R-1-0 R-1-1)
(CONNECTED-ROOMS R-1-1 R-1-0)
(LINK D-yellow-0-0-0-1 R-0-0 R-0-1)
(LINK D-yellow-0-0-0-1 R-0-1 R-0-0)
(LINK D-yellow-0-0-1-0 R-0-0 R-1-0)
(LINK D-yellow-0-0-1-0 R-1-0 R-0-0)
(LINK D-yellow-1-0-1-1 R-1-0 R-1-1)
(LINK D-yellow-1-0-1-1 R-1-1 R-1-0)
(KEYMATCH K-yellow-0 D-yellow-0-0-0-1)
(KEYMATCH K-yellow-0 D-yellow-0-0-1-0)
(KEYMATCH K-yellow-0 D-yellow-1-0-1-1)
(at-agent R-0-0)
(at K-yellow-0 R-1-0)
(locked D-yellow-1-0-1-1)
(unlocked D-yellow-0-0-0-1)
(unlocked D-yellow-0-0-1-0)
(empty-hand)
        )
        (:goal
            (and
(at-agent R-1-1)
            )
        )
)
```

FSM for Reward machines or similar