Learning Portable Representations for High-Level Planning

Steven James[1] Benjamin Rosman[1] George Konidaris[2]

[1]University of the Witwatersrand [2]Brown University

June 12, 2020

Authors


(a) Steven James


(b) Benjamin Rosman


(c) George Konidaris

Introduction

- Planning in continuous state space and continuous action space is costly and inefficient.
- Learn symbolic representations of state space.
- Previous work of Konidaris et al. [3] shows a procedure to generate these symbolic representations.
- However, learned symbols are not general.

Contribution

1. Learn symbols in *egocentric* state space with [3].
2. Then, for each newly encountered environment, augment the environment specific representations.

$\rightarrow$ Results in sample efficiency.
$\rightarrow$ More general representations.

Semi Markov Decision Processes

Framing the problem as Semi-Markov Decision Processes. (What is that?)

$$MDP = (States, Actions, Transitions, Rewards)$$

In MDP, time is discrete. In SMDP time is continuous

$$SMDP = (States, Options, Transitions, Rewards)$$

$$o_i \in Options = (\mathcal{I}_i, \pi_i, \beta_i)$$

$\mathcal{I}_i$: initiation set,
π_i: policy of o_i,
β_i: termination probability of o_i.

High-Level Planning

High level planning operates using symbolic states and operators.
A set of propositions:

$$\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$$

A set of operators:

$$\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$$

High level state is obtained by assigning truth values to p_i's. Each operator is described as:

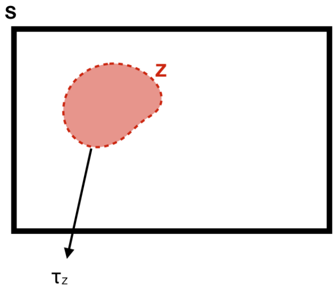
$$\alpha_i = (\mathit{precond}_i, \mathit{effect}_i^+, \mathit{effect}_i^-)$$

Symbols and Plans

Definition

A propositional symbol σ_Z is the name associated with a test τ_Z, and the corresponding set of states $Z = \{s \in S | \tau_Z(s) = 1\}$.

We want our classifiers to efficiently compute:



Symbols and Plans

Definition

A propositional symbol σ_Z is the name associated with a test τ_Z, and the corresponding set of states $Z = \{s \in S | \tau_Z(s) = 1\}$.

We want our classifiers to efficiently compute:

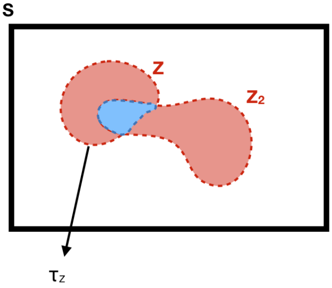


Figure: AND operation

Symbols and Plans

Definition

A propositional symbol σ_Z is the name associated with a test τ_Z, and the corresponding set of states $Z = \{s \in S | \tau_Z(s) = 1\}$.

We want our classifiers to efficiently compute:

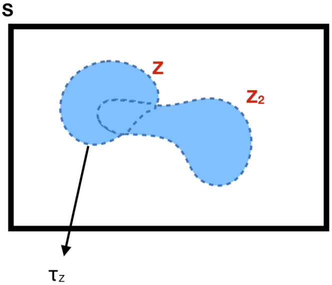


Figure: OR operation

Definition

A propositional symbol σ_Z is the name associated with a test τ_Z, and the corresponding set of states $Z = \{s \in S | \tau_Z(s) = 1\}$.

We want our classifiers to efficiently compute:

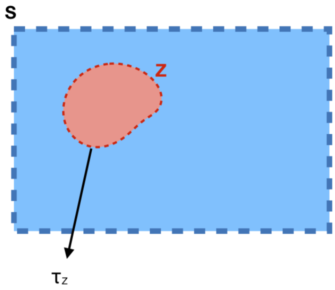
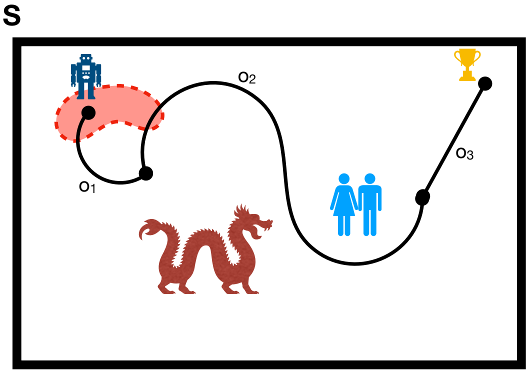


Figure: NOT operation

Symbols and Plans

Definition

A plan $p = \{o_1, o_2, \ldots, o_{p_n}\}$ from a state set $Z \subseteq S$ is a sequence of options $o_i \in O$, $1 \leq i \leq p_n$, to be executed from some state in Z.



Symbols and Plans

Definition

The plan space for an SMDP is the set of all tuples (Z, p), where $Z \subseteq S$ is a set of states in the SMDP, and p is a plan.

Symbols for Propositional Planning

We need a symbol for the precondition for each option.

Definition

The precondition of option o is the symbol referring to its initiation set: $Pre(o) = \sigma_{I_o}$.

Then, we need a symbol for the effect of each option.

Definition

Given an option o and a set of states $X \subseteq S$, we define the image of o from X as: $Im(X, o) = \{s' | \exists s \in X, P(s'|s, o) > 0\}$.

Symbols for Propositional Planning

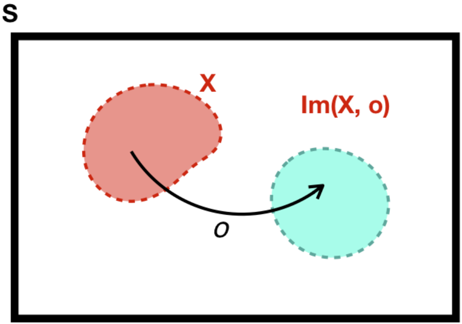


Figure: "Image" of option o from a set of states X.

Symbols for Propositional Planning

Theorem

Given an SMDP, the ability to represent the preconditions of each option and to compute the image operator is sufficient for determining whether any plan tuple (Z, p) is feasible [3].

Proof.

Consider any plan tuple (Z, p), with plan length n. We set $z_0 = Z$ and repeatedly compute $z_{j+1} = Im(z_j, p_j)$, for $j \in \{1, 2, \ldots, n\}$. The plan tuple is feasible if and only if $z_i \subseteq Pre(p_{i+1})$, $\forall i \in \{0, 1, \ldots, n-1\}$. $\square$

Symbols for Propositional Planning

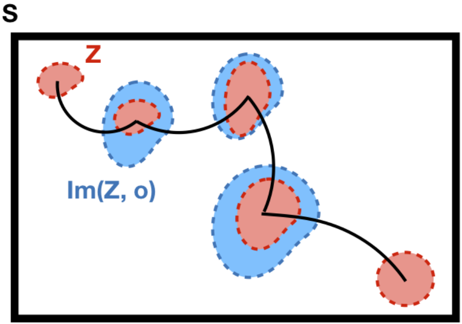


Figure: Informal proof

Symbols for Propositional Planning

- It is the necessary condition.
- Checking feasibility $\implies$ We have precond. and image ops. symbols
- It is the sufficient condition.
- We have precond. and image ops. symbols $\implies$ Checking feasibility.
- Checking feasibility $\iff$ We have precond. and image ops. symbols.

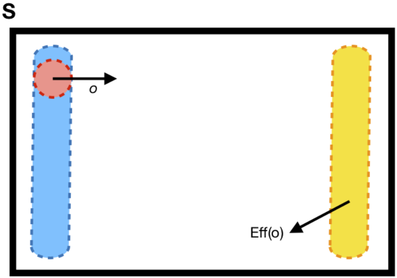
Symbols for Propositional Planning

- However, representing Im(X, o) may be hard.
- There are good alternatives that is appropriate for this framework.
- Subgoal options and abstract subgoal options.

Symbols for Propositional Planning

Definition

The effect set of subgoal option o is the symbol representing the set of all states that an agent can possibly find itself in after executing o: $Eff(o) = \{s' | \exists s \in S, t, P(s', t | s, o) > 0\}$.



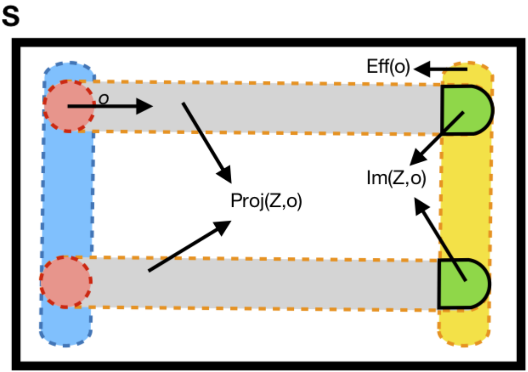
Symbols for Propositional Planning

Definition

Given an option o and a set of states $Z \subseteq S$, we define the projection of Z, with respect to o (denoted $Project(Z, o)$) as: $Project(Z, o) = \{[a, b] | \exists a', [a', b] \in Z\}$.

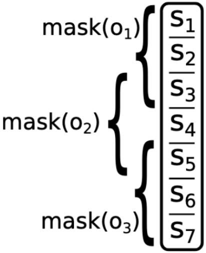
Then, image operator for an abstract subgoal option can be computed as: $Im(Z, o) = Project(Z, o) \cap Eff(o)$.

Symbols for Propositional Planning



Constructing a PDDL Domain Description

So far, we only created symbols for $Pre(o)$. We also need $Im(X, o)$.



Factor	State Variables	Options
f_1	s_1, s_2	o_1
f_2	s_3	o_1, o_2
f_3	s_4	o_2
f_4	s_5	o_2, o_3
f_5	s_6, s_7	o_3

Figure: Reprinted from [3].

Constructing a PDDL Domain Description

- Executing an option o_i projects the factors it changes and intersects with $Eff(o_i)$.
- A future execution of option o_j projects the overlapping factors out of $Eff(o_i)$.
- This may result in a combinatorial explosion if we would not be careful.
- Remember, we need to enumerate these combinations since all we want is $Im(X, o)$.

Problem with Environment Specific State Space

Figure: We cannot use the symbol learned in the left environment for the right environment. Reprinted from [2].

Symbolic Representations in Egocentric Space

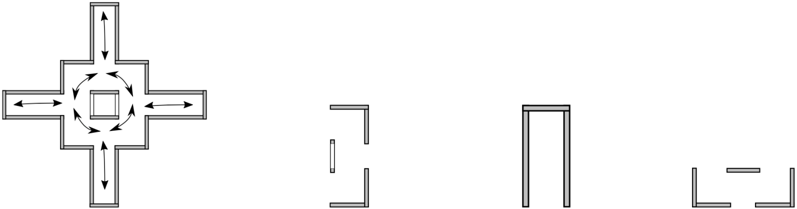

(a) (b) (c) (d)

Figure: (a) Possible actions are shown with arrows. (b-d) Local egocentric observations. Reprinted from [2].

Symbolic Representations in Egocentric Space

Option	Precondition	Effect
`Clockwise1`	`wall-junction`	`window-junction`
`Clockwise2`	`window-junction`	`wall-junction`
`Anticlockwise1`	`wall-junction`	`window-junction`
`Anticlockwise2`	`window-junction`	`wall-junction`
`Outward`	`wall-junction` $\vee$ `window-junction`	`dead-end`
`Inward`	`dead-end`	$\begin{cases} \texttt{window-junction} \text{ w.p. } 0.5 \\ \texttt{wall-junction} \text{ w.p. } 0.5 \end{cases}$

Figure: Subgoal options learned in egocentric space. Reprinted from [2].

Transferring Egocentric Symbols

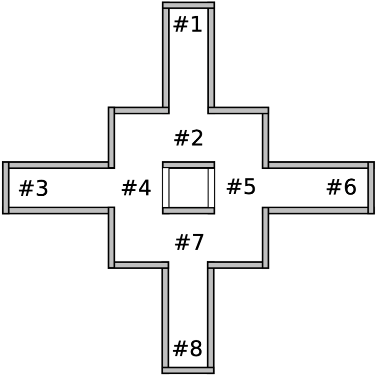


Figure: Environment-specific states are clustered. Reprinted from [2].

Pipeline

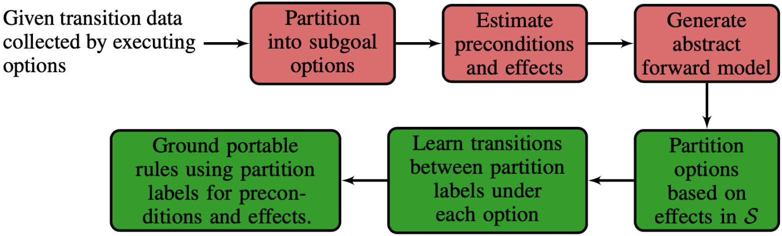


Figure: General pipeline for learning portable representations. Reprinted from [2].

Experiments

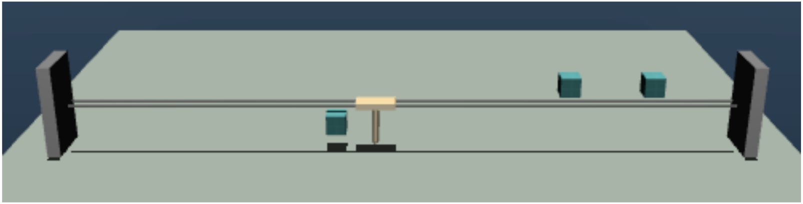
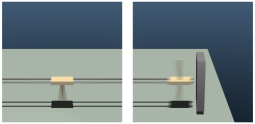
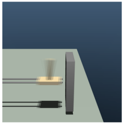


Figure: The rod-and-block domain. Available high-level options are `GoLeft`, `GoRight`, `RotateUp`, and `RotateDown`. Reprinted from [2].

Experiments



(a)



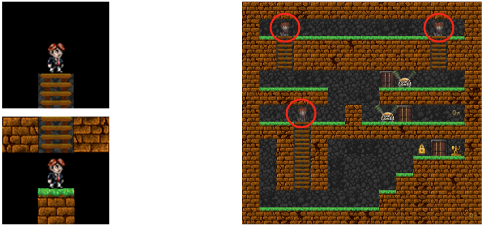
(b)

```
(:action Up Clockwise_1
 :parameters()
 :precondition (and (symbol_18)
                    (symbol_11)
                    (notfailed))
 :effect (and (symbol_12)
              (not symbol_18))
)
```

(c)

Figure: An example learned rule in PDDL. (a) Preconditions, (b) effect. Reprinted from [2].

Experiments



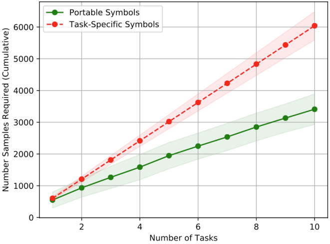
(a) (b)

```
(:action DownLadder_1
 :parameters()
 :precondition (and (symbol_80)
                    (notfailed))
 :effect (and (symbol_622)
              (not symbol_80))
)
```

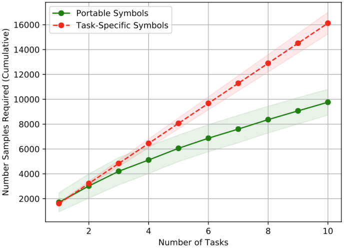
(c)

Figure: An example learned rule in PDDL. (a) Up, precondition. Down, effect. (b) States in which this option can be executed. Reprinted from [2]. Also, the character is taken from the amazing game Braid [1].

Experiments



(a) Results for the *Rod-and-Block* domain.



(b) Results for the *Treasure Game* domain.

Figure: The usage of egocentric state space reduces the necessary sample size. Reprinted from [2].

[1] Jonathan Blow. Braid. https://en.wikipedia.org/wiki/Braid_(video_game), 2008.

[2] Steven James, Benjamin Rosman, and George Konidaris. Learning portable representations for high-level planning. *arXiv preprint arXiv:1905.12006*, 2019.

[3] George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.