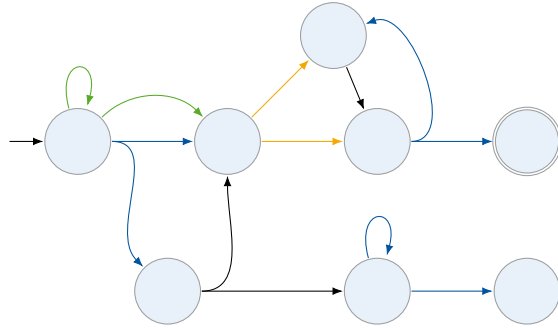


Learning Generalized Policies for Fully Observable Non-Deterministic Planning Domains

IJCAI 2024

Till Hofmann, Hector Geffner

August 6, 2024

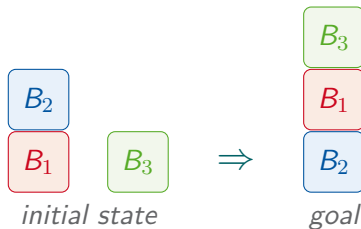


Generalized Policies for Classical Planning

Classical planning:

- ▶ Given: action model + initial state + goal
- ▶ Task: find a (concrete) policy $\pi_P : S \rightarrow 2^A$ that maps each state s to a set of actions $\pi_P(s)$

```
(:action pickup :parameters (?x)
:precondition (and
  (clear ?x) (ontable ?x) (handempty))
:effect (and (not (ontable ?x))
  (not (clear ?x)) (not (handempty))
  (holding ?x)))
(:action putdown ...)
(:action stack ...)
(:action unstack ...)
```



Generalized Policies for Classical Planning

Classical planning:

- ▶ Given: action model + initial state + goal
- ▶ Task: find a (concrete) policy $\pi_P : S \rightarrow 2^A$ that maps each state s to a set of actions $\pi_P(s)$

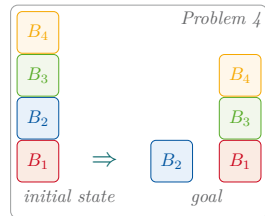
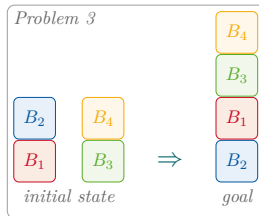
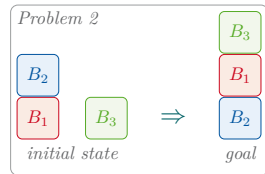
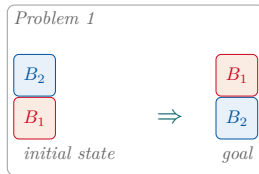
Generalized planning:

- ▶ Given: action model + **class of problems** $\mathcal{Q}$
- ▶ *Each* problem: initial state + goal
- ▶ Task: find a general policy that maps each problem $P \in \mathcal{Q}$ to a concrete policy $\pi(P) = \pi_P$

Example (Blocks World)

General policy:

- 1 Unstack all misplaced blocks
- 2 Stack them in the correct order



Generalized Policies for Classical Planning

Classical planning:

- ▶ Given: action model + initial state + goal
- ▶ Task: find a (concrete) policy $\pi_P : S \rightarrow 2^A$ that maps each state s to a set of actions $\pi_P(s)$

Generalized planning:

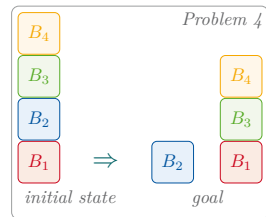
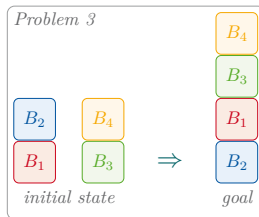
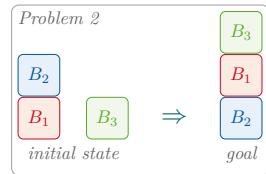
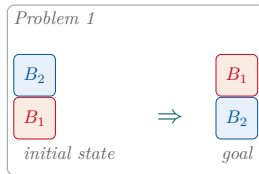
- ▶ Given: action model + **class of problems** $\mathcal{Q}$
- ▶ *Each* problem: initial state + goal
- ▶ Task: find a general policy that maps each problem $P \in \mathcal{Q}$ to a concrete policy $\pi(P) = \pi_P$

assumed
to be closed

Example (Blocks World)

General policy:

- 1 Unstack all misplaced blocks
- 2 Stack them in the correct order



Expressing and Learning Generalized Policies

Rule-based general policies (Bonet and Geffner 2018):

- ▶ Express a policy as a set of rules of the form $C \mapsto E_1 \mid \dots \mid E_k$ over a feature space Φ
- ▶ Features: Boolean and numerical features, e.g., *# clear blocks*
- ▶ Learnable from a set of small problems (Bonet et al. 2019; Francès et al. 2021)

Example

General policy for clearing a one block x :

$$r_1 : \{\neg H, n > 0\} \mapsto \{H, n \downarrow\}$$

$$r_2 : \quad \quad \quad \{H\} \mapsto \{\neg H\}$$

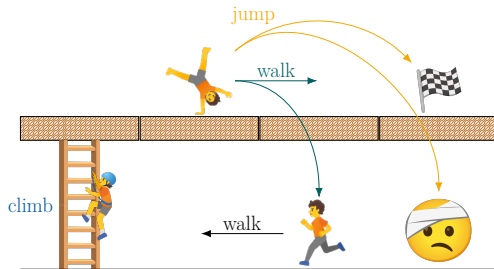
where

$H \triangleq$ true if holding a block

$n \triangleq$ number of blocks on top of x

Generalized Policies for FOND Planning

- ▶ FOND: action a may have multiple outcomes $a_1, a_2, \dots$ not under the agent's control
- ▶ *Fairness*: for an infinitely occurring action a , all outcomes $a_1, a_2, \dots$ occur infinitely often
- ▶ *Strong-cyclic*: all max. fair trajectories reach goal

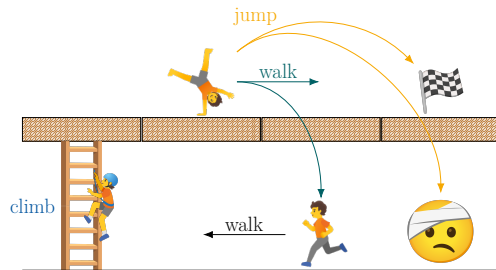


Generalized Policies for FOND Planning

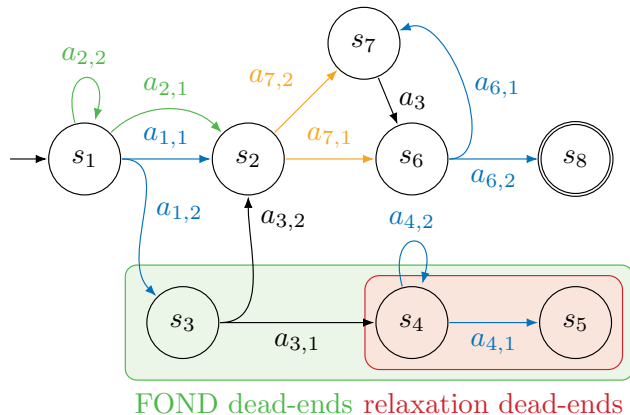
- ▶ FOND: action a may have multiple outcomes $a_1, a_2, \dots$ not under the agent's control
- ▶ *Fairness*: for an infinitely occurring action a , all outcomes $a_1, a_2, \dots$ occur infinitely often
- ▶ *Strong-cyclic*: all max. fair trajectories reach goal

Questions

- 1 How do we express general policies for FOND domains?
- 2 How can we learn FOND general policies?

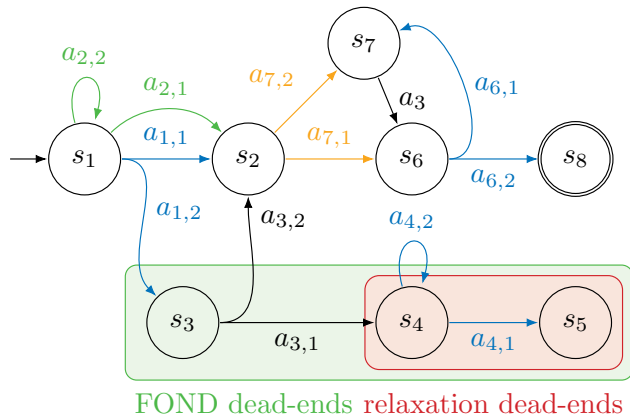


From Classical to FOND General Policies

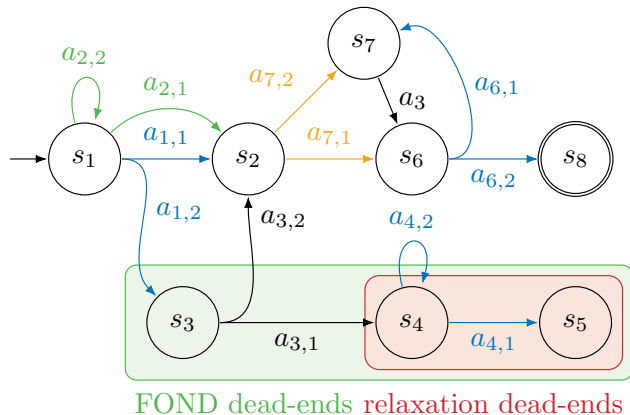


From Classical to FOND General Policies

All-outcome relaxation: agent can choose action outcome $\rightarrow$ classical planning



From Classical to FOND General Policies

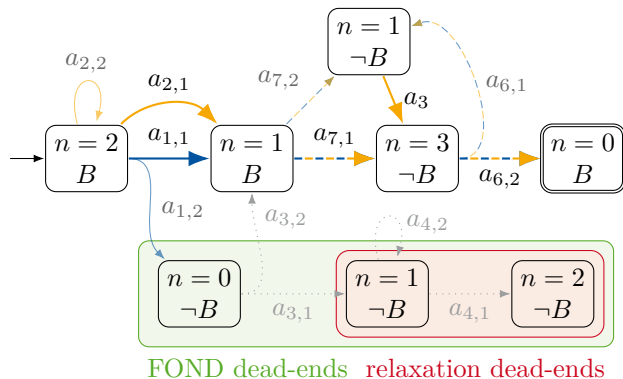


All-outcome relaxation: agent can choose action outcome $\rightarrow$ classical planning

Observation

A general policy for the *all-outcome relaxation* is also a general policy for the original FOND problem if it avoids FOND dead-ends

From Classical to FOND General Policies

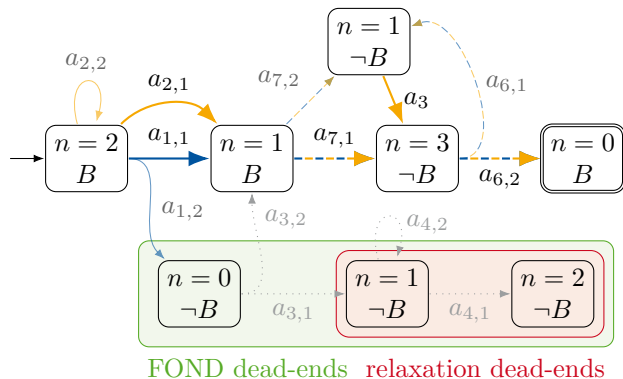


All-outcome relaxation: agent can choose action outcome $\rightarrow$ classical planning

Observation

A general policy for the *all-outcome relaxation* is also a general policy for the original FOND problem if it avoids FOND dead-ends

From Classical to FOND General Policies



All-outcome relaxation: agent can choose action outcome $\rightarrow$ classical planning

Observation

A general policy for the *all-outcome relaxation* is also a general policy for the original FOND problem if it avoids FOND dead-ends

Learning:

- Select features that describe the FOND dead-ends
 - Learn a general policy for the relaxation that avoids dead-ends
- $\Rightarrow$ Combinatorial optimization problem

more details on poster

Example: Learned Policy for Acrobatics

Selected features:

- 1 Distance $d \equiv \text{dist}(\text{position}, \text{next-fwd}, \text{position}_G)$ between the current position and the goal position
- 2 Boolean feature $U \equiv |\text{up}|$ which is true if the agent is currently on the beam
- 3 Boolean feature $B \equiv |\text{broken-leg}|$ which is true if the agent's leg is broken

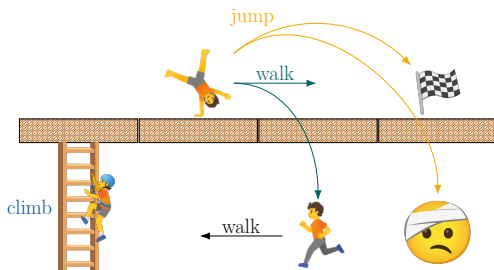
Learned policy rules:

$$r_1 : \{U, d > 0, \neg B\} \mapsto \{d \downarrow\}$$

$$r_2 : \{\neg B, \neg U\} \mapsto \{U\} \mid \{d \uparrow\}$$

Learned state constraint:

$$b_1 : \{B, \neg U\}$$



Evaluation

$\mathcal{Q}$	# problems	# solved	# training p.	# obj (train)	# obj (all)	t_{solve}/s	# features	# sel. features	# constraints	max complex.	policy cost
acrobatics	18	18	3	3	9	0.12	23	3	1	4	6
beam-walk	9	9	2	3	9	0.05	22	2	0	4	5
blocks3ops	95	95	4	4	20	3:44	194	3	0	5	11
blocks-clear	95	95	2	3	20	1.5	34	2	0	4	6
blocks-on	190	190	2	3	20	1:56	704	3	0	6	11
doors	19	19	5	7	33	1:18	625	4	1	10	19
islands	300	300	4	32	83	1h	1182	4	1	7	13
first-resp	99	15^M	2	5	36	33:40	332	5	2	7	20
miner	69	13^I	2	9	184	297h	1073	8	4	6	28
spiky-tire	170	36^I	3	6	23	1h	479	8	5	8	36
tireworld	980	7^C	1	3	100	0.11	27	5	4	4	12
triangle-tire	10	1^I	1	6	231	0.29	27	3	1	4	9

Evaluation

$\mathcal{Q}$	# problems	# solved	# training p.	# obj (train)	# obj (all)	t_{solve}/s	# features	# sel. features	# constraints	max complex.	policy cost
acrobatics	18	18	3	3	9	0.12	23	3	1	4	6
beam-walk	9	9	2	3	9	0.05	22	2	0	4	5
blocks3ops	95	95	4	4	20	3:44	194	3	0	5	11
blocks-clear	95	95	2	3	20	1.5	34	2	0	4	6
blocks-on	190	190	2	3	20	1:56	704	3	0	6	11
doors	19	19	5	7	33	1:18	625	4	1	10	19
islands	300	300	4	32	83	1h	1182	4	1	7	13
first-resp	99	15^M	2	5	36	33:40	332	5	2	7	20
miner	69	13^I	2	9	184	297h	1073	8	4	6	28
spiky-tire	170	36^I	3	6	23	1h	479	8	5	8	36
tireworld	980	7^C	1	3	100	0.11	27	5	4	4	12
triangle-tire	10	1^I	1	6	231	0.29	27	3	1	4	9

Evaluation

$\mathcal{Q}$	# problems	# solved	# training p.	# obj (train)	# obj (all)	t_{solve}/s	# features	# sel. features	# constraints	max complex.	policy cost
acrobatics	18	18	3	3	9	0.12	23	3	1	4	6
beam-walk	9	9	2	3	9	0.05	22	2	0	4	5
blocks3ops	95	95	4	4	20	3:44	194	3	0	5	11
blocks-clear	95	95	2	3	20	1.5	34	2	0	4	6
blocks-on	190	190	2	3	20	1:56	704	3	0	6	11
doors	19	19	5	7	33	1:18	625	4	1	10	19
islands	300	300	4	32	83	1h	1182	4	1	7	13
first-resp	99	15^M	2	5	36	33:40	332	5	2	7	20
miner	69	13^I	2	9	184	297h	1073	8	4	6	28
spiky-tire	170	36^I	3	6	23	1h	479	8	5	8	36
tireworld	980	7^C	1	3	100	0.11	27	5	4	4	12
triangle-tire	10	1^I	1	6	231	0.29	27	3	1	4	9

Evaluation

$\mathcal{Q}$	# problems	# solved	# training p.	# obj (train)	# obj (all)	t_{solve}/s	# features	# sel. features	# constraints	max complex.	policy cost
acrobatics	18	18	3	3	9	0.12	23	3	1	4	6
beam-walk	9	9	2	3	9	0.05	22	2	0	4	5
blocks3ops	95	95	4	4	20	3:44	194	3	0	5	11
blocks-clear	95	95	2	3	20	1.5	34	2	0	4	6
blocks-on	190	190	2	3	20	1:56	704	3	0	6	11
doors	19	19	5	7	33	1:18	625	4	1	10	19
islands	300	300	4	32	83	1h	1182	4	1	7	13
first-resp	99	15^M	2	5	36	33:40	332	5	2	7	20
miner	69	13^I	2	9	184	297h	1073	8	4	6	28
spiky-tire	170	36^I	3	6	23	1h	479	8	5	8	36
tireworld	980	7^C	1	3	100	0.11	27	5	4	4	12
triangle-tire	10	1^I	1	6	231	0.29	27	3	1	4	9

Evaluation

$\mathcal{Q}$	# problems	# solved	# training p.	# obj (train)	# obj (all)	t_{solve}/s	# features	# sel. features	# constraints	max complex.	policy cost
acrobatics	18	18	3	3	9	0.12	23	3	1	4	6
beam-walk	9	9	2	3	9	0.05	22	2	0	4	5
blocks3ops	95	95	4	4	20	3:44	194	3	0	5	11
blocks-clear	95	95	2	3	20	1.5	34	2	0	4	6
blocks-on	190	190	2	3	20	1:56	704	3	0	6	11
doors	19	19	5	7	33	1:18	625	4	1	10	19
islands	300	300	4	32	83	1h	1182	4	1	7	13
first-resp	99	15^M	2	5	36	33:40	332	5	2	7	20
miner	69	13^I	2	9	184	297h	1073	8	4	6	28
spiky-tire	170	36^I	3	6	23	1h	479	8	5	8	36
tireworld	980	7^C	1	3	100	0.11	27	5	4	4	12
triangle-tire	10	1^I	1	6	231	0.29	27	3	1	4	9

Evaluation

proven to generalize
to all instances

$\mathcal{Q}$	# problems	# solved	# training p.	# obj (train)	# obj (all)	t_{solve}/s	# features	# sel. features	# constraints	max complex.	policy cost
acrobatics	18	18	3	3	9	0.12	23	3	1	4	6
beam-walk	9	9	2	3	9	0.05	22	2	0	4	5
blocks3ops	95	95	4	4	20	3:44	194	3	0	5	11
blocks-clear	95	95	2	3	20	1.5	34	2	0	4	6
blocks-on	190	190	2	3	20	1:56	704	3	0	6	11
doors	19	19	5	7	33	1:18	625	4	1	10	19
islands	300	300	4	32	83	1h	1182	4	1	7	13
first-resp	99	15^M	2	5	36	33:40	332	5	2	7	20
miner	69	13^I	2	9	184	297h	1073	8	4	6	28
spiky-tire	170	36^I	3	6	23	1h	479	8	5	8	36
tireworld	980	7^C	1	3	100	0.11	27	5	4	4	12
triangle-tire	10	1^I	1	6	231	0.29	27	3	1	4	9

LEARNING GENERALIZED POLICIES FOR FULLY OBSERVABLE NON-DETERMINISTIC PLANNING DOMAINS

Till Hofmann and Hector Geffner

RWTH Aachen University

At a Glance

- Generalized planning: solves a class of problems with a single policy
- Learning generalized policies has shown to be feasible for classical planning
- Observation: FOND planning is classical planning + dead-end detection
- Key Idea: Learn a classical policy on the deterministic relaxation while also avoiding FOND dead-ends
- Result: For many domains, the learned policy can be proven to generalize correctly

Generalized Classical Planning

Given a class $\mathcal{Q} = \{P_1, P_2, \dots\}$ of (classical) planning problems over the same domain:

- We assume $\mathcal{Q}$ is closed: If s is a subgoal state in $P \in \mathcal{Q}$, then $P[s] \in \mathcal{Q}$ (i.e., P with initial state s)
- Classical policy: partial mapping $\pi_P: S \rightarrow 2^S$ from states s to sets of actions $\pi_P(s) \subseteq A(s)$
- Policy π solves problem P if all maximal π -trajectories end in a goal state
- General policy: partial mapping $\pi: \mathcal{Q} \rightarrow \Pi$ from problem instances $P \in \mathcal{Q}$ to classical policies $\pi(P) = \pi_P$
- General policy π solves $\mathcal{Q}$ if every $\pi(P)$ solves $P \in \mathcal{Q}$
- Policy language: policy consists of a set $\mathcal{R}$ of QSP-like rules $C \Rightarrow E$ over a collection Φ of features
- Feature goal: Boolean and numerical description logic formulas over domain predicates
- Semantics: state transition (s, s') satisfies a rule $C \Rightarrow E$ if all feature conditions in C are true in s and the feature values change from s to s' according to E
- For $P \in \mathcal{Q}$, the general policy π defines π_P as follows: $s \in \pi_P(s)$ if $f(s, s') = s'$ and (s, s') satisfies a rule of π

Generalized FOND Planning

- Actions may have multiple (non-deterministic) outcomes, chosen by the environment
- Reference: for an infinitely recurring action a , all outcomes $a_1, a_2, \dots$ occur infinitely often
- Strong-safety: all max. fat trajectories reach goal
- Deterministic relaxation: non-deterministic action a is replaced by deterministic actions $a_1, a_2, \dots$



Observation: A policy π_P for the deterministic relaxation P_P is a policy for the original FOND problem P if it avoids all FOND dead-ends

- Learn a classical general policy π state constraints
- Exclude state constraints $\mathcal{B}$ the conditions C , but describing bad states

Theorem (simplified): If rules $\mathcal{R}$ encode a general classical policy for $\mathcal{Q}_D$ that avoids FOND dead-ends and state constraints $\mathcal{B}$ describe the dead-ends, then π_P consisting of rules $\mathcal{R}$ and state constraints $\mathcal{B}$ is a general FOND policy for $\mathcal{Q}$



Paper



Poster



Code

Learning Generalized FOND Policies

Approach:

- Sample closed class of small FOND problems $\mathcal{C} \subseteq \mathcal{Q}$
- Generate feature goal over domain predicates, using a description logic feature language
- Learn general policy that solves the classical problem $\mathcal{Q}_D$ and avoids FOND dead-ends
- Learn a set of constraints $\mathcal{B}$ for the dead-end states to exclude policy and constraints as SAT problem

Propositional variables

- $\text{Goal}(f, s')$ is true if the transition (s, s') is goal
- $\text{Satisf}(f)$ is true if the feature f is satisfied

Formulas

- For every state transition (s, s') select at least one goal transition from the sub actions:

$$\bigvee_{a \in \text{Sub}(s)} \bigvee_{a' \in \text{Sub}(s')} \text{Goal}(s, s')$$

where $a \in \text{Sub}(s)$ if $s \in a'$ and $P(s, a)$ is a dead-end.

- Goal state has distance 0, i.e., for every goal state s :

$$V(s, 0)$$

- For every alive state s , choose a goal distance:

$$\text{Reach}(s) = \bigvee_{a \in \text{Sub}(s)} V(s, a)$$

- For every goal transition (s, s') , there must be one outcome leading towards the goal:

$$\text{Goal}(s, s') \wedge s \in a' \Rightarrow \bigvee_{a' \in \text{Sub}(s')} V(s', a') < d$$

- There cannot be any goal transitions to dead states, i.e., for every alive state s and dead state s' :

$$\neg \text{Goal}(s, s')$$

- For every goal state s and non-goal state s' :

$$\bigvee_{f \in \text{Sub}(s)} \text{Satisf}(f)$$

- For every alive state s and dead state s' :

$$\bigvee_{f \in \text{Sub}(s')} \text{Satisf}(f)$$

- Every pair of a goal transition (s, s') and non-goal transition (s, s'') must be distinguishable by a feature:

$$\text{Goal}(s, s') \wedge \neg \text{Goal}(s, s'') \Rightarrow D(s, s') \vee D(s, s'')$$

- where

$$D(s, s') = \bigvee_{f \in \text{Sub}(s)} \text{Satisf}(f)$$

- and

$$D(s, s') = \bigvee_{f \in \text{Sub}(s)} \text{Satisf}(f)$$

- Implementation

- SAT problem solved with C4.5 (Geffner et al. 2011)

- Optimization:

- Ranking $V(s, d)$ replaced by labeling safe states

- (safe: all children are safe, goal states always safe)

- Only distinguish critical states from alive states

- (critical: dead-end but alive parent)

Extracting a General FOND Policy



Policy π solves the deterministic relaxation, but does not avoid FOND dead-ends

Policy π solves the deterministic relaxation and avoids FOND dead-ends on general FOND policy

The extracted policy has the following rules:

$$r_1: \{s > 0, B\} \Rightarrow \{a\}$$

$$r_2: \{s > 0, B\} \Rightarrow \{a, a'\}$$

$$r_3: \{s > 0, B\} \Rightarrow \{a\} \mid \{a, B\}$$

B has the following state constraint:

$$b_1: \{s > 0, B\}$$

Example: Acrobatics



The learned policy $\pi_{\text{Acrobatics}}$ uses three features:

Distance d in d steps, next goal position, between the current position and the goal position.

Boolean feature B is $\{a\}$ which is true if the agent is currently on the beam.

Boolean feature B is $\{a\}$ which is true if the agent's leg is broken.

The learned policy $\pi_{\text{Acrobatics}} = \pi_{\text{Acrobatics}}$ consists of the following rules $\mathcal{R}$:

$$r_1: \{d > 0, B\} \Rightarrow \{a\}$$

$$r_2: \{d > 0, B\} \Rightarrow \{a\} \mid \{a\}$$

B has a single constraint $f = \{b_1\}$

$$b_1: \{d, B\}$$

Proposition: The general policy $\pi_{\text{Acrobatics}}$ solves the class $\mathcal{Q}_{\text{Acrobatics}}$ of acrobatics problems.

Evaluation

$\mathcal{Q}$	# policies	# policies	# policies	# policies	# policies	# policies	# policies	# policies	# policies
acrobatics	18	18	3	0.12	21	3	1	4	6
beam-walk	9	9	3	0.05	22	2	0	5	5
blockstack	95	95	4	244	194	2	0	5	11
blocksworld	65	65	2	1	1	2	2	4	6
blocksworld	100	100	2	1	1	2	2	4	6
blocks	19	19	5	7	125	625	4	10	19
blocksworld	60	125	5	234	125	2	2	7	10
blocks	300	300	4	32	18	182	4	7	13
blocks	60	125	5	234	125	2	2	7	10
blocksworld	120	120	5	6	36	478	8	5	10
blocksworld	980	980	5	1	3	611	27	5	4
blocksworld	10	10	1	6	625	27	3	1	4

Come visit our poster!



Paper

Appendix

Bibliography (I)

-  Bonet, Blai, Guillem Francès, and Hector Geffner (2019). “Learning features and abstract actions for computing generalized plans”. In: *AAAI*, pp. 2703–2710.
-  Bonet, Blai and Hector Geffner (2018). “Features, Projections, and Representation Change for Generalized Planning”. In: *IJCAI*, pp. 4667–4673.
-  Francès, Guillem, Blai Bonet, and Hector Geffner (2021). “Learning general planning policies from small examples without supervision”. In: *AAAI*, pp. 11801–11808.

Till Hofmann, Hector Geffner

IJCAI, Jeju Island, South Korea, August 6, 2024