Learning Generalized Policies for Fully Observable Non-Deterministic Planning Domains

Till Hofmann and Hector Geffner

RWTH Aachen University

At a Glance

- **Generalized planning:** solve 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

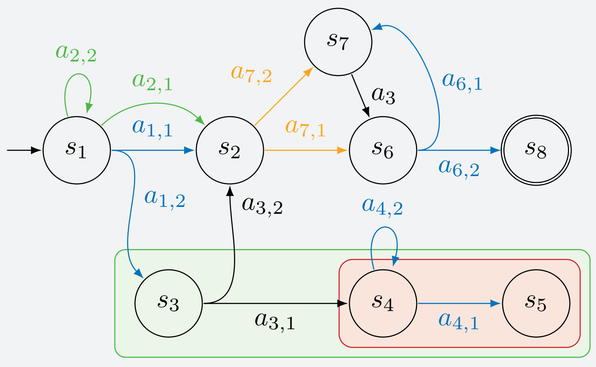
Generalized Classical Planning

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

- We assume $\mathcal{Q}$ is *closed*: if s is a solvable state in $P \in \mathcal{Q}$, then $P[s] \in \mathcal{Q}$ (i.e., P with initial state s)
- *Concrete policy:* partial mapping $\pi_P : S \to 2^A$ from states s to sets of actions $\pi_P(s) \subseteq A(s)$
- Policy π_P solves problem P if all maximal π-trajectories end in a goal state
- *General policy:* partial mapping $\pi : \mathcal{Q} \to \Pi$ from problem instance $P \in \mathcal{Q}$ to concrete policy $\pi(P) = \pi_P$
- General policy π solves $\mathcal{Q}$ if every $\pi(P)$ solves $P \in \mathcal{Q}$
- *Policy language:* policy consists of a set R of QNP-like rules $C \mapsto E$ over a collection Φ of features
- *Feature pool:* Boolean and numerical description logic features over domain predicates
- *Semantics:* state transition (s, s') satisfies a rule $C \mapsto E$ 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: $a \in \pi_D(s)$ if $f(a, s) = s'$ and (s, s') satisfies a rule of π

Generalized FOND Planning

- Actions may have multiple (non-deterministic) outcomes, chosen by the environment
- *Fairness:* for an infinitely occurring action a, all outcomes $a_1, a_2, \ldots$ occur infinitely often
- *Strong-cyclic:* all max. fair trajectories reach goal
- *Deterministic relaxation:* non-deterministic action a is replaced by deterministic actions $a_1, a_2, \ldots$



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

→ Learn a classical general policy + state constraints

→ Encode state constraints B like conditions C, but describing *bad* states

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

Paper Poster Code

Learning Generalized FOND Policies

Approach:

1. Sample closed class of small FOND problems $\mathcal{Q}' \subset \mathcal{Q}$
2. Generate feature pool over domain predicates, using a description logic feature language
3. Learn general policy that solves the classical problems $\mathcal{Q}'_D$ and avoids FOND dead-ends
4. Learn a set of constraints B for the dead-end states

⇒ Encode policy and constraints as SAT problem

Propositional variables

- $Good(s, s')$ is true if the transition (s, s') is good
- $Select(f)$ is true if the feature f is selected

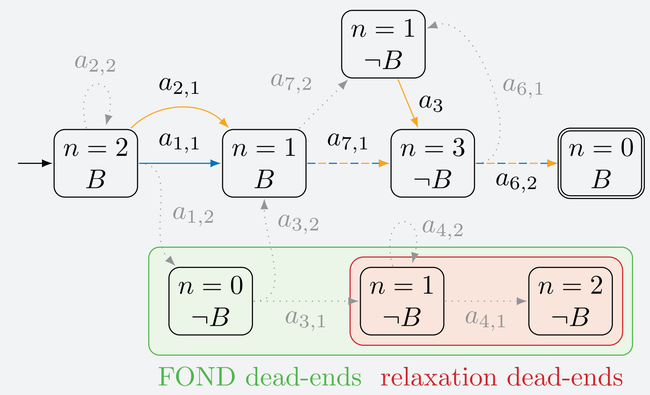
Formulas

1. For every alive state s, select at least one good transition from the safe actions:
$$\bigvee_{a \in Safe(s)} \bigvee_{s' \in F(a,s)} Good(s, s')$$
where $a \in Safe(s)$ if no $s' \in F(a, s)$ is a dead-end.
2. Goal states have distance 0, i.e., for every goal state s:
$$V(s, 0)$$
3. For every alive state s, choose a goal distance:
$$\text{Exactly-1} : \{V(s, d)\}_{d \in \mathbb{N}}$$
4. For every good transition (s, s'), there must be one outcome leading towards the goal:
$$Good(s, s') \wedge V(s, d) \to \bigwedge_{\substack{a \in A(s): \\ s' \in F(a,s)}} \bigvee_{s'' \in F(a,s)} V(s'', d'') \to d'' < d$$
5. There cannot be any good transitions to dead states, i.e., for every alive state s and dead state s':
$$\neg Good(s, s')$$
6. For every goal state s and non-goal state s':
$$\bigvee_{f : [\![f(s)]\!] \neq [\![f(s')]\!]} Select(f)$$
7. For every alive state s and dead state s':
$$\bigvee_{f : [\![f(s)]\!] \neq [\![f(s')]\!]} Select(f)$$
8. Every pair of a good transition (s_1, s'_1) and non-good transition (s_2, s'_2) must be distinguishable by a feature:
$$Good(s_1, s'_1) \wedge \neg Good(s_2, s'_2) \to D(s_1, s_2) \vee D2(s_1, s'_1, s_2, s'_2)$$
where
$$D(s_1, s_2) = \bigvee_{f : [\![f(s_1)]\!] \neq [\![f(s_2)]\!]} Select(f)$$
and
$$D2(s_1, s'_1, s_2, s'_2) = \bigvee_{f : \Delta_f(s_1, s'_1) \neq \Delta_f(s_2, s'_2)} Select(f)$$

Implementation

- SAT problem solved with `clingo` (Gebser et al. 2011)
- Optimizations:
  - 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 1 solves the deterministic relaxation, but does not avoid FOND dead-ends

Policy 2 solves the determinstic relaxation and avoids FOND dead-ends ⇒ general FOND policy

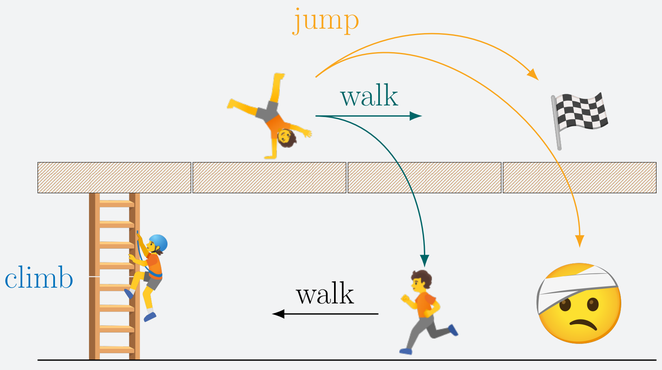
The extracted policy has the following rules:

$$r_1 : \quad \{n > 0, B\} \mapsto \{n\downarrow\}$$
$$r_2 : \quad \{n > 0, B\} \mapsto \{\neg B, n\uparrow\}$$
$$r_3 : \quad \{n > 0, \neg B\} \mapsto \{n\uparrow\} \mid \{n\downarrow, B\}$$

It has the following state constraint:

$$b_1 : \quad \{n = 0, \neg B\}$$

Example: Acrobatics



The learned policy π_{acro} uses three features:

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

The learned policy $\pi_{\text{acro}} = \pi_{R,B}$ consists of the following rules R:

$$r_1 : \quad \{U, d > 0, \neg B\} \mapsto \{d\downarrow\}$$
$$r_2 : \quad \{\neg B, \neg U\} \mapsto \{U\} \mid \{d\uparrow\}$$

It has a single constraint $B = \{b_1\}$:

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

Proposition. *The general policy π_{acro} solves the class $\mathcal{Q}_{acro}$ of solvable* `acrobatics` *problems.*

Evaluation

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

International Joint Conference on Artificial Intelligence (IJCAI), Aug 3–Aug 9, 2024, Jeju, South Korea