Learning general policies and sketches

NeurIPS Workshop GenPlan'23

Hector Geffner
RWTH Aachen University
Aachen, Germany

Linköping University
Linköping, Sweden

with Blai Bonet, Simon Ståhlberg, Dominik Drexler, and RLeap Team

Learning High-Level Representations: A Key Challenge in AI

- Learn representations that support **reasoning** and **planning**, that **generalize** and are **reusable**, . . .

- **Yoshua Bengio**'s challenges reflected in title of his IJCAI 2021 talk:
  - ▷ *System 2 Deep Learning: Higher-level cognition, agency, out-of-distribution generalization and causality*

- **Yann LeCun**'s three challenges, AAAI 2020:
  - ▷ AI must learn to represent the world
  - ▷ AI must think and plan in ways compatible with gradient-based learning
  - ▷ AI must learn hierarchical representation of action plans

Two Approaches to Learning High-Level Representations

- **Bottom-up approach**
  - ▷ Representations emerge from **architecture**, loss function, and "right" bias
  - ▷ Most common approach in deep (reinforcement) learning

- **Top-down approach**
  - ▷ Representations learned over **language** with "right" syntax and semantics
  - ▷ Reasoning, meaningful learning bias, transparency, **what** vs. **how**
  - ▷ Doesn't assume background knowledge; compatible with deep learning

Our focus: **top-down representation learning to act and plan**

Three concrete learning problems for acting and planning

- Learning **general models**
  - ▷ Language and semantics
  - ▷ Learning: **combinatorial approach**
- Learning **general policies**
  - ▷ Language and semantics
  - ▷ Learning: **combinatorial approach** and **DRL approach**
- Learning **general subgoal structures** (sketches)
  - ▷ Language, semantics, **width**
  - ▷ Learning: **combinatorial approach**

The setting is **classical planning**:

- ▷ factored deterministic MDPs; states given by atoms $p(c_1, \ldots, c_k)$
- ▷ fixed set of **domain predicates** p
- ▷ variable set of objects c_1 that depend on **domain instance**

Learning Problem #1: General Models

- Problems P specified as **instances** $P = \langle D, I \rangle$ of **general domain** D
  - ▷ **Domain** D specified in terms of **action schemas** and **predicates**
  - ▷ **Instance** is $P = \langle D, I \rangle$ where I details **objects**, **init**, **goal**

- E.g., DELIVERY problem where packages to be moved by robot to target cell in grid; **any** number of packages, **any** grid size, captured with domain with **three STRIPS action schemas**. Can they be **learned**, predicates included?

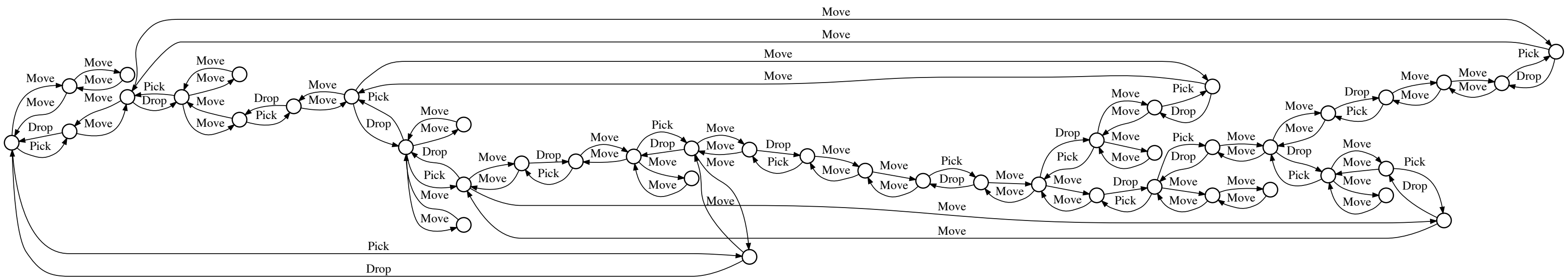
$move(c, c')$
Preconds: $atRobot(c)$, $adjacent(c, c')$
Effects: $atRobot(c')$, $\neg atRobot(c)$

$pick(o, c)$:
Preconds: $atRobot(c)$, $at(o, c)$, $emptyhand$
Effects: $held(o)$, $\neg at(o, c)$, $\neg emptyhand$

$drop(o, c)$:
Preconds: $atRobot(c)$, $held(o)$
Effects: $at(o, c)$, $\neg held(o)$, $emptyhand$

Example: Learning $P = \langle D, I \rangle$ from Single Graph G

Input: State graph G of agent in $1{\times}3$ grid, moving/picking/dropping 2 pkgs



Output: **Simplest** STRIPS representation $P = \langle D, I \rangle$ that **generates** G

```
Move(?to,?from):
  Pre: neq(?to,?from), p5(?to,?from)
  Pre:  p2(?from), -p2(?to)
  Eff: -p2(?from),  p2(?to)

Pick(?p,?x):
  Pre:  p2(?x),  p1, -p3(?p),  p4(?p,?x)
  Eff:          -p1,  p3(?p), -p4(?p,?x)

Drop(?p,?x):
  Pre:  p2(?x), -p1,  p3(?p), -p4(?p,?x)
  Eff:           p1, -p3(?p),  p4(?p,?x)
```

Interpretation of learned predicates:

– p_1: gripper empty
– $p_2(x)$: agent at cell x,
– $p_3(p)$: agent holds pkg p,
– $p_4(p, x)$: pkg p in cell x
– $p_5(x, y)$: cell x adj to y

- Domain D correct for **any** grid, **any** $\#$ of packages. Structure of nodes uncovered.

Learning Problem #1: General Models

- $P = \langle D, I \rangle$ defines unique **state graph** $G(P)$
- Learning as **inverse task:** from graphs $G_1, \ldots, G_k$, learn problems $P = \langle D, I_i \rangle$:

> Given graphs $G_1, \ldots, G_k$, find **simplest** instances $P_i = \langle D, I_i \rangle$ such that graphs G_i and $G(P_i)$ are isomorphic, $i = 1, \ldots, k$.

- **Problem** cast/solved as **combinatorial optimization task** [Bonet and G., 2020]
- **Complexity** of P_i determined by # and arities of action schemas and predicates
- **Variations:** noisy graphs, gray-box states [Rodriguez *et al.*, 2021, Occhipinti *et al.*, 2022]

[**Open:** How to solve (a version of) this problem using **DL/gradient descent?**]

Learning Problem #2: General Policies [Bonet, G., 2018]

General policy for achieving $clear(x)$ in Blocks; **any instance**

- **Features** $\Phi = \{H, n\}$: 'holding' and 'number of blocks above x'
- **Policy** π for class $\mathcal{Q}_{clear}$ of problems with goal $clear(x)$ given by two rules:

$$\{\neg H, n > 0\} \mapsto \{H, n\downarrow\} \qquad ; \qquad \{H, n > 0\} \mapsto \{\neg H\}$$

- **Meaning:**
  - ▷ if $\neg H$ & $n > 0$, move to successor state where H holds and n **decreases**
  - ▷ if H & $n > 0$, move to successor state where $\neg H$ holds, n **doesn't change**

- **Semantics of policy** π specified by rules $C_i \mapsto E_i$:
  - ▷ state transition (s, s') is π-**transition** iff $s \models C_i$, and $[s, s'] \models E_i$ for some i
  - ▷ π-**trajectories** made up of π-transitions, starting at s_0
  - ▷ π **solves** class of problems $\mathcal{Q}$ if, in every $P \in Q$, all π-trajectories end in goal

Example 2: Delivery

- **Domain:** Move packages in $n \times m$ grid, one by one, to target location
- **Features** $\Phi = \{H, p, t, n\}$: hold, dist. to nearest pkg & target, $\#$ undelivered
- **General policy** π: **any** $\#$ of pkgs and distribution, **any** grid size

$\{\neg H, p > 0\} \mapsto \{p\downarrow, t?\}$	go to nearest package
$\{\neg H, p = 0\} \mapsto \{H, p?\}$	pick it up
$\{H, t > 0\} \mapsto \{t\downarrow, p?\}$	go to target cell
$\{H, t = 0\} \mapsto \{\neg H, n\downarrow, p?\}$	drop package

Policy can be shown to be correct, solving **any instance**.

Learning Problem #2: Min-SAT Encoding $T(\mathcal{S},\mathcal{F})$

- **Inputs** to formula $T = T(\mathcal{S},\mathcal{F})$:
  - ▷ **Feature pool** $\mathcal{F}$ defined from domain predicates and **C2 logic**
  - ▷ **State transitions** $\mathcal{S}$ from the training instances $P_1,\ldots,P_k$
- **Variables** in T: $Good(s,s')$, $V(s,d)$, $Select(f)$: $(s,s')\in\mathcal{S}$, $f\in\mathcal{F}$, $d\leq d_{max}$
- **Formulas** in $T = T(\mathcal{S},\mathcal{F})$:

1. Policy **closed**: $\bigvee_{(s,s')\in\mathcal{T}} Good(s,s')$ *[non-goal, non dead-end s]*
2. Policy **acyclic**: $Good(s,s'), V(s,d), V(s',d') \supset d' < d$, *[non-goal, non-dead s]*
3. Policy **safe**: $\neg Good(s,s')$ *[non-dead-end, dead-end]*
4. **Exactly-1** $\{V(s,d) : d\leq d_{max}\}$ *[$V(s,d)$ iff $V(s)=d$, all s]*
5. Features **distinguishes** good from bad transitions: *[(s,s') and (t,t') in $\mathcal{S}$]*

$$Good(s,s')\wedge\neg Good(t,t')\rightarrow\bigvee_{f:\Delta_f(s,s')\neq\Delta_f(t,t')} Select(f)$$

Theorem: Theory $T(\mathcal{S},\mathcal{F})$ is SAT iff there is a policy π over features $\Phi\subseteq\mathcal{F}$ that solves $P_1,\ldots,P_n$. Policy rules from SAT assignment; capture feature changes in the transitions (s,s') labeled good.

Experimental Results over Classical Domains

	$\mathcal{S}$	$\mathcal{S}/\!\sim$	d_{max}	$\|\mathcal{F}\|$	$vars$	$clauses$	t_{all}	t_{SAT}	c_Φ	$\|\Phi\|$	k^*	$\|\pi_\Phi\|$
$\mathcal{Q}_{clear}$	1,161	55	7	532	7.9K	243.7K(242.3K)	6	1	8	3	4	3
$\mathcal{Q}_{on}$	1,852	329	10	1,412	17.3K	376.6K(281.5K)	231	153	13	5	5	7
$\mathcal{Q}_{grip}$	1,140	61	12	835	6.5K	102.6K(100.8K)	2	1	9	3	4	4
$\mathcal{Q}_{rew}$	432	361	15	514	5.5K	214.9K(98.9K)	8	1	7	2	6	2
$\mathcal{Q}_{deliv}$	42,473	5442	56	1,373	753.4K	38.2M(23.5M)	3071	2902	30	4	14	6
$\mathcal{Q}_{visit}$	2,396	310	8	188	13.9K	244.5K(160.6K)	3	1	7	2	5	1
$\mathcal{Q}_{span}$	10,777	96	19	764	85.0K	2.2M(2.2M)	46	2	9	3	6	2
$\mathcal{Q}_{micon}$	4,706	4,636	14	1,073	23.8K	23.6M(2.4M)	184	104	11	4	5	5
$\mathcal{Q}_{bw}$	2,136	2,136	7	1,766	10.9K	4.6M(180.1K)	252	64	10	3	6	1

Classes of problems $\mathcal{Q}_D$ for different planning domains D

Some theories $T(\mathcal{S}, \mathcal{F})$ are very large (38M clauses) but solved

Learned policies for each of the domains **can be proved to be correct**; but learning doesn't guarantee it

Learning Problem #3: Subgoal Structures [Bonet and G. 2021]

- **Sketch** of width=2 for Delivery:

$$\{n > 0\} \mapsto \{n\downarrow\} \quad \text{deliver package}$$

- **Sketch** of width=1:

$$\begin{array}{ll} \{\neg H\} \mapsto \{H\} & \text{go and pick package} \\ \{H\} \mapsto \{\neg H, n\downarrow\} & \text{go and deliver package} \end{array}$$

- **Sketch** of width=0 (**full policy**)

$$\begin{array}{ll} \{\neg H, p > 0\} \mapsto \{p\downarrow, t?\} & \text{go to nearest package} \\ \{\neg H, p = 0\} \mapsto \{H, p?\} & \text{pick it up} \\ \{H, t > 0\} \mapsto \{t\downarrow, p?\} & \text{go to target cell} \\ \{H, t = 0\} \mapsto \{\neg H, n\downarrow, p?\} & \text{drop package} \end{array}$$

- **Language** of sketches is **same language** of policies: rules $C_i \mapsto E_i$
- **Semantics** of sketches slightly different:
  - ▷ In state s_i where C_i holds, reach **subgoal** s_{i+1} s.t. $[s_i, s_{i+1}] \models E_i$
  - ▷ If sketch is **terminating** and **subproblems** have width $\leq k$, problems solvable by SIW_R algorithm in time $\exp(k)$

Learning Sketches [Drexler *et al.*, 2022]

- Given a **domain** D, training **instances** $P_1, \ldots, P_n$, a **pool** of features $\mathcal{F}$, and **bound** k, find **min-cost** sketch R over $\mathcal{F}$ such that
  - ▷ Subproblems induced by R on each P_i have all **width bounded** by k,
  - ▷ Sketch R is **terminating** (structurally acyclic)

- Learning task model and solved as combinatorial optimization problem in Clingo [Gebser, Kaufmann, Schaub 2012]

- E.g., sketch given by rules $\{\neg H\} \mapsto \{H\}$ and $\{H\} \mapsto \{\neg H, n\downarrow\}$ learned in this way

Last Twist: General policies via DRL and GNNs

- Can we learn general policies using **deep (reinforcement) learning**?

- Before, they were learned them by solving **Min-SAT problem** $T(\mathcal{S}, \mathcal{F})$:
  - ▷ $\mathcal{S}$: set of state transitions (s, s') over small instances
  - ▷ $\mathcal{F}$: **pool of features** derived from domain predicates and **C2 logic**

- **C2** is fragment of first-order logic that uses **two variables only**

- Interestingly, **tight correspondence** known between **C2** and **GNNs**

- **Idea:** Represent policy $\pi(s'|s; w)$ with GNN; learn w parameters with RL
  - ▷ **Before:** explicit pool of features; now, GNN takes care of features
  - ▷ **Before:** π constrained to solve training instances; now, penalize π if it doesn't

GNN + Actor-Critic for Gen Policies [Ståhlberg *et al.*, 2023]

Domain	Coverage (%)	Domain	Coverage (%)
Blocks	**100%**	Delivery	**100%**
Gripper	**100%**	Miconic	**100%**
Visitall	**100%**	Grid	70%
Logistics	36%	Spanner	68%

- Nearly **perfect general policies** obtained in several domains (100%)
- But the interesting part is the **failure** in three marked domains, as it has **nothing to do with RL algorithm:**
  - ▷ **C2/GNN expressivity not enough**: binary relations need to be composed
  - ▷ **Generality-optimality tradeoff**: can't have both in some domains
- By addressing these two problems, **100% coverage over all domains** obtained

(unlikely to get similar results without **understanding** these problems)

Summary: Top-down representation learning to act and plan

- **Three learning problems** in planning:
  - ▷ Learning general models
  - ▷ Learning general policies
  - ▷ Learning general subgoal structures (sketches)
- **Two methods**:
  - ▷ Combinatorial optimization: Min-SAT, Clingo
  - ▷ Continuous optimization: deep (reinforcement) learning
- **Potential benefits** of top-down approaches (vs. bottom-up):
  - ▷ Transparency, structural **generalization**; distinction what/how, **reuse**, . . .

References

[Barceló et al., 2020] Barceló, P., Kostylev, E. V., Monet, M., Pérez, J., Reutter, J., and Silva, J. P. (2020). The logical expressiveness of graph neural networks. In *ICLR*.

[Bonet and Geffner, 2018] Bonet, B. and Geffner, H. (2018). Features, projections, and representation change for generalized planning. In *Proc. IJCAI*, pages 4667–4673.

[Bonet and Geffner, 2020a] Bonet, B. and Geffner, H. (2020a). Learning first-order symbolic representations for planning from the structure of the state space. In *Proc. ECAI*.

[Bonet and Geffner, 2020b] Bonet, B. and Geffner, H. (2020b). Qualitative numeric planning: Reductions and complexity. *Journal of AI Research*, 69:923–961.

[Bonet and Geffner, 2021] Bonet, B. and Geffner, H. (2021). General policies, representations, and planning width. In *Proc. AAAI*, pages 11764–11773.

[Drexler et al., 2021] Drexler, D., Seipp, J., and Geffner, H. (2021). Expressing and exploiting the common subgoal structure of classical planning domains using sketches. In *Proc. KR*, pages 258–268.

[Drexler et al., 2022] Drexler, D., Seipp, J., and Geffner, H. (2022). Learning sketches for decomposing planning problems into subproblems of bounded width. In *Proc. ICAPS*.

[Fern et al., 2006] Fern, A., Yoon, S., and Givan, R. (2006). Approximate policy iteration with a policy language bias: Solving relational markov decision processes. *Journal of Artificial Intelligence Research*, 25:75–118.

[Francès et al., 2021] Francès, G., Bonet, B., and Geffner, H. (2021). Learning general planning policies from small examples without supervision. In *Proc. AAAI*, pages 11801–11808.

[Grohe, 2020] Grohe, M. (2020). The logic of graph neural networks. In *Proc. of the 35th ACM-IEEE Symp. on Logic in Computer Science*.

[Khardon, 1999] Khardon, R. (1999). Learning action strategies for planning domains. *Artificial Intelligence*, 113:125–148.

[Martín and Geffner, 2000] Martín, M. and Geffner, H. (2000). Learning generalized policies from planning examples using concept languages. In *Proc. KR*.

[Ståhlberg et al., 2023] Ståhlberg, S., Bonet, B., and Geffner, H. (2023). Learning general policies with policy gradient methods. In *Proc. KR*.

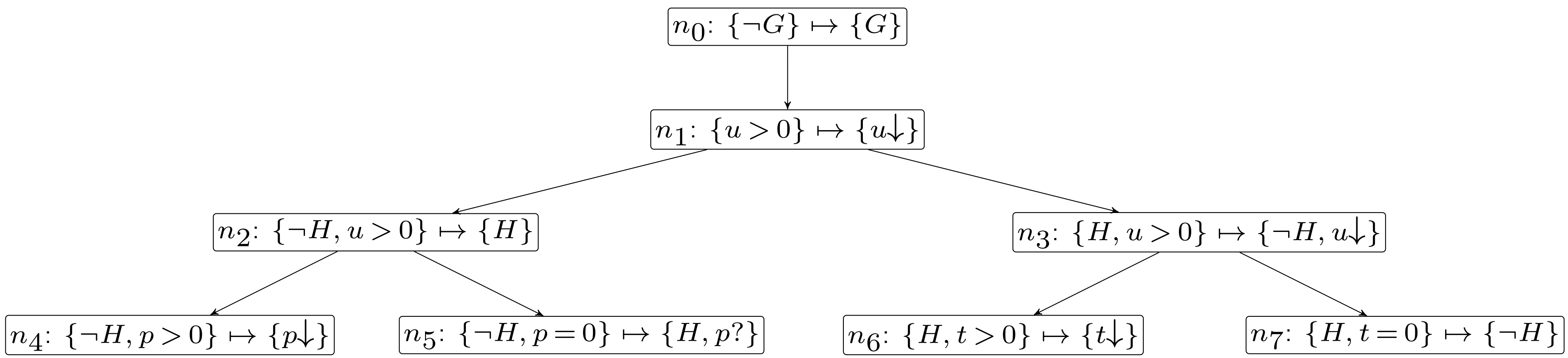
[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

[Toenshoff et al., 2021] Toenshoff, J., Ritzert, M., Wolf, H., and Grohe, M. (2021). Graph neural networks for maximum constraint satisfaction. *Frontiers in artificial intelligence*, 3:98.

Pool of Features $\mathcal{F}$: From Domain predicates and C2 Logic

- **Fixed grammar** generates new **predicates** from **domain** and **goal** predicates p, p_G
- **Unary predicates** called concepts C; **binary predicates**, roles (description logics)
- **Denotation (extension)** of concept C in state s, $C(s)$: **objects** "in" C
- **Features** from concepts C: $n_C(s) = |C(s)|$; $p_C(s)=\top$ iff $|C(s)| > 0$
- **Complexity** of unary predicate ("concept") given by number of grammar rules used
- **Pool** $\mathcal{F}$ obtained from concepts of complexity bounded by a parameter
- **Grammar:** borrowed from "description logics", a **C2 logic**
  - ▷ Primitive: C_p given by domain predicates p and "goal predicates" p_G (p in goal)
  - ▷ Universal: C_u contains all objects
  - ▷ Negation: $\neg C$ contains $C_u \setminus C$
  - ▷ Intersection: $C \sqcap C'$
  - ▷ Quantified: $\exists R.C = \{x : \exists y[R(x,y) \wedge C(y)]\}$ and $\forall R.C = \{x : \forall y[R(x,y) \wedge C(y)]\}$
  - ▷ Roles (binary predicates): R_p, R_p^{-1}, R_p^{+}, and $[R_p^{-1}]^{+}$
- Additional **distance features** $dist(C_1, R, C_2)$ for concepts C_1 and C_2 and role R that evaluates to d in state s iff minimum R-distance between object in C_1 to object in C_2 is d

Learning Hierarchical Policies [Drexler, Seipp, G. 2023]



Hierarchical policy for $\mathcal{Q} = \mathcal{Q}_{Delivery}$:

- Every **node** n has a **sketch rule** $r(n)$ and a **class** $\mathcal{Q}_n$ of subproblems
- $\mathcal{Q}_n$ determined by **rule** $r(n)$ and **parent** $\mathcal{Q}_{n'}$. For root, $\mathcal{Q}_n = \mathcal{Q}$.
  - ▷ $\mathcal{Q}_n$ forced to have **smaller width** than parent $\mathcal{Q}_{n'}$
  - ▷ $\mathcal{Q}_n$ has **width zero** iff n is a **leaf**

GNN-like net that maps STRIPS states s into $f^s(o) = f^s_L(o)$

1. **Input:** State s (set of atoms true in s), set of objects
2. **Output:** Embeddings $f_L(o)$ for each object o
3. $f_0(o) \sim \mathbf{0}^k$ for each object $o \in s$
4. For $i \in \{0, \ldots, L-1\}$
5. For each **atom** $q := p(o_1, \ldots, o_m)$ true in state s:
6. $m_{q,o} \;:=\; [\mathbf{MLP}_p(f_i(o_1), \ldots, f_i(o_m))]_j$
7. For each **object** o in state s:
8. $f_{i+1}(o) \;:=\; \mathbf{MLP}_U\big(f_i(o), agg(\{\!\{m_{q,o} | o \in q\}\!\})\big)$

- **Value** and **policy** learned from embeddings $f^s(o) = f_L(o)$ for each object o
- Objects o change from instance to instance but **domain predicates** p fixed
- One $\mathbf{MLP}_p$, for each **domain predicate** p; single $\mathbf{MLP}_U$
- **Relational** GNN-like architecture as **STRIPS states** not graphs but rel structures
- Messages exchanged among objects o through the atoms where they appear in s

From the Object Embeddings $f_L^s(o)$ to $V(s)$ and $\pi(s'|s)$

Value function $V(s)$ and policy $\pi(s'|s)$ from embeddings $f^s(o) = f_L(o)$:

- **Value function** $V(s) = V(s; w)$ outputs single scalar through $\mathbf{MLP}$ as:

$$V(s) \; = \; \mathbf{MLP}\big(\textstyle\sum_{o \in O} f^s(o)\big)$$

- **Stochastic policy** $\pi(s'|s) = \pi(s'|s; w)$ selects successor states s' by computing *logits* for pairs (s, s') and passing them through *softmax*:

$$\begin{aligned} \text{logit}(s'|s) &= \; \mathbf{MLP}_1\big(\textstyle\sum_{o \in O} \mathbf{MLP}_2(f^s(o), f^{s'}(o))\big)\,, \\ \pi(s'|s) &\propto \; \exp\big(\text{logit}(s'|s)\big) \end{aligned}$$

Training the General Policy Functions by DRL – Actor-Critic

Input: Training MDPs $\{M_i\}_i$, each with state priors p_i
Input: Value function $V(s)$ with parameter ω
Input: Policy $\pi(s'|s)$ with parameter θ
Input: Differentiable policy $\pi(s|s')$ with parameter θ
Input: Diff. value function $V(s)$ with parameter ω
Parameters: Step sizes $\alpha, \beta > 0$, discount factor γ

Initialize parameters θ and ω
Loop forever:
  Sample MDP index $i \in \{1, \ldots, N\}$
    Sample non-goal state S in M_i with probability p_i
    Sample successor state S' with probability $\pi(S'|S)$
    Let $\delta = 1 + \gamma V(S') - V(S)$
    $\omega \leftarrow \omega + \beta\delta\nabla V(S)$
    $\theta \leftarrow \theta - \alpha\delta\nabla \log \pi(S'|S)$
    If S' is a goal state, $\omega \leftarrow \omega - \beta V(S')\nabla V(S')$

Standard **Actor-Critic RL** algorithm, baseline $V(S)$, where policy does not select actions but **state transitions**, and **action costs** are all 1