# Representation Learning for Acting and Planning: A Top Down Approach 

Tutorial IJCAI 2022

Blai Bonet<br>Universitat Pompeu Fabra, Barcelona<br>Hector Geffner<br>ICREA \& Universitat Pompeu Fabra<br>Wallenberg Guest Professor, Linköping University<br>With inputs from Dominik, Simon, Andrés; RLeap Team (UPF, LiU)

Slides at https://www.dtic.upf.edu/~hgeffner/tutorial-2022.pdf

## TAILOR

## Bottom-up vs. Top-Down Representation Learning (1)

- Deep learning (DL) and Deep Reinforcement Learning (DRL) have revolutionised the landscape of AI, exploiting power of stochastic gradient descent
- Yet DL and DRL struggle with OOD/structural generalization
$\triangleright$ Inductive biases in neural architectures assumed to help but vague, informal
- Alternative: Language-based representation learning
$\triangleright$ Don't choose low-level arch and expect "right representation" to emerge
$\triangleright$ Choose high-level language instead, and learn representations over language
- Separation between what is to be learned and how


## Bottom-up vs. Top-down Representation Learning (2)

- Yoshua Bengio at IJCAI 2021: System 2 Deep Learning: Higher-Level Cognition, Agency, Out-of-Distribution Generalization and Causality:
". . . Systematic generalization hypothesized to arise from efficient factorization of knowledge into recomposable pieces corresponding to reusable factors . . ."
- Language-based representation learning:
$\triangleright$ learn the "recomposable pieces" in a language
$\triangleright$ recombinations and generalization will follow semantics
- Very much in line with traditional AI: just learn from data the representations that have traditionally been crafted by hand
- Potential benefits: meaningful learning bias, semantics, transparency, reasoning


## Example: Minigrid/BabyAI [Chevalier-Boisvert et al., 2019]


$\triangleright$ Task: Pick up grey box behind you, then go to grey key and open door
$\triangleright$ Red triangle is agent at bottom right. Light-grey is field of view
$\triangleright$ Learn controller that accepts goals and obs, and outputs action to do
$\triangleright$ Like a "classical planning problem" but state representation not known, and goals to be achieved reactively (not by planning) with policies that generalize

## DRL vs. Language-based Representation Learning

- Surprise is not that DL and DRL methods struggle in Minigrid, but that they manage to generate meaningful behavior at all, given so little prior knowledge
- Yet methodology largely ad hoc: from intuitions to architectures and experiments using baselines...
- From perspective of language-based representation learning, key questions are:
$\triangleright$ What are the domain-independent languages for representing dynamics?
$\triangleright$ What the languages for representing general reactive policies, subgoals?
$\triangleright$ How can representations over such languages be learned?


## Outline of the Tutorial

- Background 1: Classical planning, planning width
- Languages for
$\triangleright$ representing general dynamics
$\triangleright$ representing general policies
$\triangleright$ representing general subgoal structures (sketches; 'intrinsic rewards")
- Background 2: Qualitative numerical planning problems (QNPs)
- Learning representations over these languages:
$\triangleright$ learning general dynamics
$\triangleright$ learning general policies
$\triangleright$ learning general subgoal structures
- Wrap up; Challenges

Copy of these slides at https://www.dtic.upf.edu/~hgeffner/tutorial-2022.pdf

## Outline of the Tutorial (2)

- Tutorial is not a survey on learning to act and plan; too much for us; too much for $1: 30 \mathrm{~h}$
- Focus is on a coherent research thread that covers a lot of ground:
$\triangleright$ Crisp and simple ideas and formulations for stating, understanding, and addressing key problems
- Many open problems; many opportunities for research


## Background 1:

## Classical Planning and Planning Width

## Background: Model for Classical AI Planning

A (classical) state model is a tuple $\mathcal{S}=\left\langle S, s_{0}, S_{G}, A c t, A, f, c\right\rangle$ :

- finite and discrete state space $S$
- a known initial state $s_{0} \in S$
- a set $S_{G} \subseteq S$ of goal states
- actions $A(s) \subseteq$ Act applicable in each $s \in S$
- a deterministic state-transition function $s^{\prime}=f(a, s)$ for $a \in A(s)$
- positive action costs $c(a, s)$, assumed 1 by default

A solution to the model or plan is a sequence of applicable actions $a_{0}, \ldots, a_{n}$ that maps $s_{0}$ into $S_{G}$
i.e. there must be state sequence $s_{0}, \ldots, s_{n+1}$ such that $a_{i} \in A\left(s_{i}\right), s_{i+1}=f\left(a_{i}, s_{i}\right)$, and $s_{n+1} \in S_{G}$

## A Language for Classical Planning: STRIPS

- A (grounded) problem in STRIPS is a tuple $P=\langle F, O, I, G\rangle$ :
$\triangleright F$ is set of (ground) atoms
$\triangleright O$ is set of (ground) actions
$\triangleright I \subseteq F$ stands for initial situation
$\triangleright G \subseteq F$ stands for goal situation
- Actions $o \in O$ represented by
$\triangleright$ Add list $\operatorname{Add}(o) \subseteq F$
$\triangleright$ Delete list $\operatorname{Del}(o) \subseteq F$
$\triangleright$ Precondition list $\operatorname{Pre}(o) \subseteq F$

A problem $P$ in STRIPS defines state model $S(P)$ in compact form ...

## From Language to Models

STRIPS problem $P=\langle F, O, I, G\rangle$ determines state model $\mathcal{S}(P)$ where

- the states $s \in S$ are collections of atoms from $F$
- the initial state $s_{0}$ is $I$
- the goal states $s_{G}$ are such that $G \subseteq s_{G}$
- the actions $a$ in $A(s)$ are ops in $O$ s.t. $\operatorname{Prec}(a) \subseteq s$
- the next state is $s^{\prime}=[s \backslash \operatorname{Del}(a)] \cup \operatorname{Add}(a)$
- action costs $c(a, s)$ are all 1

Common approach for solving $P$ is using path-finding/heuristic search algorithms over graph defined by $\mathcal{S}(P)$ where nodes are the states $s$, and edges $\left(s, s^{\prime}\right)$ are state transitions caused by an action $a$; i.e., $s^{\prime}=f(a, s)$ and $a \in A(s)$

The source node is the initial state $s_{0}$, and the targets are the goal states $s_{G}$

## Background: Width and Width-based Algorithms

- IW(1) is a breadth-first search that prunes states $s$ that don't make a feature true for first time in the search, given set of Boolean features $F$
$\triangleright$ In classical planning, $F$ is the set of (ground) atoms in problem
- $\operatorname{IW}(k)$ is $\operatorname{IW}(1)$ but over set $F^{k}$ made up of conjunctions of $k$ features from $F$
- Alternatively, $\operatorname{IW}(k)$ is a breadth-first search that prunes $s$ if novelty $(s)>k$
- IW runs $\operatorname{IW}(1), \operatorname{IW}(2), \ldots, \operatorname{IW}(k)$ sequentially until problem solved or $k=N$
- IW is blind like DFS and BFS but diff enumeration; uses state structure
- IW $(k)$ expands up to $N^{k}$ nodes and runs in polytime $\exp (2 k-1)$


## Planning for *Atomic Goals* with IW(1) and IW(2)

| \# | Domain | I | IW(1) | IW(2) | Neither |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 1. | 8puzzle | 400 | 55\% | 45\% | 0\% |
| 2. | Barman | 232 | 9\% | 0\% | 91\% |
| 3. | Blocks World | 598 | 26\% | 74\% | 0\% |
| 4. | Cybersecure | 86 | 65\% | 0\% | 35\% |
| 22. | Pegsol | 964 | 92\% | 8\% | 0\% |
| 23. | Pipes-NonTan | 259 | 44\% | 56\% | 0\% |
| 24. | Pipes-Tan | 369 | 59\% | 37\% | 3\% |
| 25. | PSRsmall | 316 | 92\% | 0\% | 8\% |
| 26. | Rovers | 488 | 47\% | 53\% | 0\% |
| 27. | Satellite | 308 | 11\% | 89\% | 0\% |
| 28. | Scanalyzer | 624 | 100\% | 0\% | 0\% |
| 33. | Transport | 330 | 0\% | 100\% | 0\% |
| 34. | Trucks | 345 | 0\% | 100\% | 0\% |
| 35. | Visitall | 21,859 | 100\% | 0\% | 0\% |
| 36. | Woodworking | 1659 | 100\% | 0\% | 0\% |
| 37. | Zeno | 219 | 21\% | 79\% | 0\% |
| Total/Avgs |  | 37,921 | 37.0\% | 51.3\% | 11.7\% |

$\mathbf{8 8 . 3 \%}$ of the 37,921 instances solved by IW(1) or IW(2) [Lipovetzky and G., 2012]

## Performance of IW is No Accident: Theory

- Width of $P, w(P)$, is min $k$ for which there is a sequence of subgoals (atom tuples) $t_{0}, t_{1}, \ldots, t_{n},\left|t_{i}\right| \leq k$ such that:
$\triangleright t_{0}$ is true in the initial situation
$\triangleright$ the optimal plans for $t_{n}$ are optimal plans for $P$
$\triangleright$ all optimal plans for $t_{i}$ can be extended into optimal plans for $t_{i+1}$ by adding a single action
- Also $w(P)=0$ if goal reachable in 0 or 1 step; $w(P)=N+1$ if no solution, where $N$ is number of atoms in $P$.
- Theorem: If $w(P)=k$, then $\operatorname{IW}(k)$ solves $P$ optimally in $\exp (2 k-1)$ time
- Theorem: Domains like Blocks, Logistics, Gripper, ... have all width 2 independent of problem size provided that goals are single atoms


## Practical Variations of IW

SIW: Serialized iterated width [Lipovetzky and G., 2012]

- Use IW greedily to decrease number of unachieved goals $\# g$; assumes conjunctive top goal (simple goal serialization)

BFWS: Best-first guided by novelty measure $w_{\langle \# g, \# c\rangle}$ and $\# g$

- $\operatorname{BFWS}\left(f_{5}\right)$ : back-end of state-of-the-art Dual-BFWS, \#c from relaxed plans
- $k$ - $\operatorname{BFWS}\left(f_{5}\right)$ : poltytime variant of $\operatorname{BFWS}\left(f_{5}\right)$ used as front-end of Dual-BFWS
- BFWS (R): version that does not use action structure; just PDDL simulator
[Lipovetzky and G., 2017; Francès et al., 2017]


## Understanding Width: Test Your Knowledge!

How to prove in standard encodings that:

- Blocks world instances with goal clear $(x)$ or $h o l d(x)$ have width 1
- Delivery instances with goal hold $(x)$ or $\operatorname{Agent} \operatorname{At}(y)$ have width 1
- Blocks world instances with goal on $(x, y)$ have width 2
- Delivery instances with goal $\operatorname{Pkg} A t(x, y)$ have width 2
- Blocks and Delivery with arbitrary conjunctive goals have no bounded width

Delivery is simplified Logistics: agent in grid, picking up and dropping pkgs
For proving $w(G) \leq k$ :

- Necessary 1: If $a_{1}, \ldots, a_{n}$ is optimal plan for goal $G$, each prefix $a_{1}, \ldots, a_{i}$ must be optimal plan for some $t_{i},\left|t_{i}\right| \leq k$
- Necessary 2: For these $t_{i}{ }^{\prime}$ s, all optimal plans for $t_{i}$ extend into optimal plans for $t_{i+1}$.


## Part II: Languages

- Language for expressing dynamics
- Language for expressing general policies
- Language for expressing general subgoal structures


## Language for Expressing Dynamics: First-Order STRIPS

Problems specified as instances $P=\langle D, I\rangle$ of general planning domain:

- Domain $D$ specified in terms of action schemas and predicates
- Instance is $P=\langle D, I\rangle$ where $I$ details objects, init, goal

Distinction between general domain $D$ and specific instance $P=\langle D, I\rangle$ important for reusing action models, and also for learning them:

- Learning $P_{i}=\left\langle D, I_{i}\right\rangle$ implies learning $D$ that generalizes to other instances

In RL and DRL, there is no notion of domain: generalization to other "instances" analyzed experimentally; closest things are "procedurally generated instances," and "probability distribution over tasks"

## Example: 2-Gripper Problem $P=\langle D, I\rangle$ in PDDL

```
(define (domain gripper)
    (:requirements :typing)
    (:types room ball gripper)
    (:constants left right - gripper)
    (:predicates (at-robot ?r - room)(at ?b - ball ?r - room)(free ?g - gripper)
            (carry ?o - ball ?g - gripper))
    (:action move
        :parameters (?from ?to - room)
        :precondition (at-robot ?from)
        :effect (and (at-robot ?to) (not (at-robot ?from))))
    (:action pick
        :parameters (?obj - ball ?room - room ?gripper - gripper)
        :precondition (and (at ?obj ?room) (at-robot ?room) (free ?gripper))
        :effect (and (carry ?obj ?gripper) (not (at ?obj ?room)) (not (free ?gripper))))
    (:action drop
        :parameters (?obj - ball ?room - room ?gripper - gripper)
        :precondition (and (carry ?obj ?gripper) (at-robot ?room))
        :effect (and (at ?obj ?room) (free ?gripper) (not (carry ?obj ?gripper)))))
(define (problem gripper2)
    (:domain gripper)
    (:objects roomA roomB - room Ball1 Ball2 - ball)
    (:init (at-robot roomA) (free left) (free right) (at Ball1 roomA)(at Ball2 roomA))
    (:goal (and (at Ball1 roomB) (at Ball2 roomB))))
```


## Preview: Learning Dynamics in Lifted STRIPS

- Planning problem $P_{i}=\left\langle D, I_{i}\right\rangle$ defines unique state graph $G\left(P_{i}\right)$
- Learning as inverse problem: from graphs $G_{1}, \ldots, G_{k}$, learn $D, I_{i}$ :

Given graphs $G_{1}, \ldots, G_{k}$, find simplest instances $P_{i}=\left\langle D, I_{i}\right\rangle$ such that graphs $G_{i}$ and $G\left(P_{i}\right)$ are isomorphic, $i=1, \ldots, k$.

- Problem cast and solved as combinatorial optimization task [B. and G., 2020]
- Complexity of $D$ determined by $\#$ and arities of action schemas and predicates
- Variations: missing edges, noisy observations [Rodriguez et al., 2021a]
- Related
$\triangleright$ Learning schemas from ground traces [Cresswell et al., 2013]
$\triangleright$ Deep learning of action schemas from images via autoencoders [Asai, 2019]
$\triangleright$ Learning prop. action models from options [Konidaris et al., 2018]
$\triangleright$ Most work on learning action models assumes domain predicates known


## Second Task: General Policies

- General policy represents strategy for solving multiple domain instances reactively; i.e., without having to search or plan
$\triangleright$ E.g., policy for achieving on $(x, y)$; any \# of blocks, any configuration
- What are good languages for expressing such policies?
- Number of works have addressed the problem [Khardon 1999; Martin and G., 2004; Fern et al., 2006; Srivastava et al., 2011; Hu and De Giacomo, 2011]
- Subtlety: set of (ground) actions change from instance to instance with objects

Learning general policies also a key goal in (Deep) RL

## General Policies: A Language [B. and G., 2018]

- General policies are given by rules $C \mapsto E$ over set $\Phi$ of features
- Features $f$ are state functions that have a well-defined value $f(s)$ on every reachable state of any instance of the domain
$\triangleright$ Boolean features $p: p(s)$ is true or false
$\triangleright$ Numerical features $n$ : $n(s)$ is non-negative integer

Computation of feature values assumed to be "cheap": features assumed to have linear number of values at most, computable in linear time (in $|P|$ ).

## Example: General Policy for $\operatorname{clear}(x)$

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

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

## 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


## Language and Semantics of General Policies: Definitions

- Policy rules $C \mapsto E$ over set $\Phi$ of Boolean and numerical features $p, n$ :
$\triangleright$ Boolean conditions in $C: p, \neg p, n=0, n>0$
$\triangleright$ qualitative effects in $E: p, \neg p, p ?, n \downarrow, n \uparrow, n$ ?
- State transition $\left(s, s^{\prime}\right)$ satisfies rule $C \mapsto E$ if
$\triangleright f(s)$ makes body $C$ true
$\triangleright$ change from $f(s)$ to $f\left(s^{\prime}\right)$ satisfies $E$
- A policy $\pi$ for class $\mathcal{Q}$ of problems $P$ is given by policy rules $C \mapsto E$
$\triangleright$ Transition ( $s, s^{\prime}$ ) in $P$ compatible with $\pi$ if $\left(s, s^{\prime}\right)$ satisfies a policy rule
$\triangleright$ Trajectory $s_{0}, s_{1}, \ldots$ compatible if $s_{0}$ of $P$ and transitions compatible with $\pi$
- $\pi$ solves $P$ if all max trajectories compatible with $\pi$ reach goal of $P$
- $\pi$ solves collection of problems $\mathcal{Q}$ if it solves each $P \in \mathcal{Q}$


## Example: Delivery

- Pick packages spread in $n \times m$ grid, one by one, to target location
- Features $\Phi=\{H, p, t, n\}$ : hold, dist. to nearest pkg \& target, \# undelivered
- Policy $\pi$ that solves class $\mathcal{Q}_{D}$ : any $\#$ of pkgs and distribution, any grid size

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

## General Policies: Three Questions

1. How to prove that general policy solves potentially infinite class of instances $\mathcal{Q}$ ?
2. How to learn policies (and the features involved) to solve $\mathcal{Q}$ ?
3. How to learn policies that are guaranteed to solve infinite $\mathcal{Q}$ ?

We consider idea of learning first and move then to 1 . Not much to say about 3 .

## Preview: Learning General Policies

Given a known domain $D$, training instances $P_{1}, \ldots, P_{k}$, over $D$, and a finite pool of domain features $\mathcal{F}$, each with a cost, find the cheapest policy $\pi$ over $\mathcal{F}$ such that $\pi$ solves all $P_{i}, i=1, \ldots, k$

- Problem cast and solved as combinatorial opt. task [Francès et al., 2021]
- Pool of features $\mathcal{F}$ generated from domain predicates using 2-variable (description) logic grammar; feature cost given by syntax tree size
- Deep learning approaches [Toyer et al., 2018; Garg et al., 2020] do not need $\mathcal{F}$ but not $100 \%$ correct in general
- Recent DL approach also avoids $\mathcal{F}$ and nearly $100 \%$ correct when 2 -variable logic features suffice; exploits relation between GNNs and 2-variable logic [Ståhlberg et al., 2022a and 2022b]


## Proving that a General Policy Solves Class of Instances $\mathcal{Q}$

How to prove that this policy $\pi$ achieves $\operatorname{clear}(x)$ in all Block problems?

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

- Soundness: policy $\pi$ applies in every non-goal state $s$
$\triangleright$ for any such $s$, there is $\left(s, s^{\prime}\right)$ compatible with $\pi$
- Acyclicity: no sequence of transitions $\left(s_{i}, s_{i+1}\right)$ compatible with $\pi$ cycle

Theorem: If $\pi$ is sound and acyclic in $\mathcal{Q}$, and no dead-ends, $\pi$ solves $\mathcal{Q}$

Exercise: Show that policy for clear $(x)$ is sound and acyclic in Blocks

## Acyclicity, Termination, and QNPs

- Termination: criterion that ensures that policy is acyclic over any domain
- A policy $\pi$ is terminating if for all infinite trajectories $s_{0}, \ldots, s_{i}, \ldots$ compatible with $\pi$, there is a numerical feature $n$ such that:
$\triangleright n$ is decremented in some recurrent transition $\left(s, s^{\prime}\right)$; i.e., $n\left(s^{\prime}\right)<n(s)$
$\triangleright n$ is not incremented in any recurrent transition $\left(s, s^{\prime}\right)$; i.e., $n\left(s^{\prime}\right) \ngtr n(s)$
- Every such trajectory deemed impossible or unfair ( $n$ can't decrement below 0 ), thus if $\pi$ terminates, $\pi$-trajectories terminate
- Termination notion is from QNPs; verifiable in time $O\left(2^{|\Phi|}\right)$ by SIEVE algorithm [Srivastava et al., 2011], where $\Phi$ is set of features involved in the policy

More about QNPs later on . . .

## Third Task: Subgoal Structure

Subgoal structure important in planning and RL ("intrinsic rewards", hierarchies)
Sketches powerful language for expressing subgoal structure [B. and G., 2021]

- Goal serialization and full policies expressible as sketches
- Semantics in terms of subgoals to be achieved; not so with HTNs, LTL
- Sketches split problems into subproblems

If subproblems have a bounded width, problems solved in polytime

## Example: Sketches for Delivery

- Width=0 Sketch (full policy)

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

- Width=2 Sketch:

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

- Width=1 Sketch:

$$
\begin{aligned}
& \{\neg H\} \mapsto\{H\} \\
& \{H\} \mapsto\{\neg H, n \downarrow\}
\end{aligned}
$$

go and pick package
go and deliver package
Features: Holding ( $H$ ); Dist. to nearest Pkg $(p)$, Target $(t)$; \# Undeliv Pkgs ( $n$ )

## Syntax and Semantics of Sketch Rules

- Syntax: For Boolean and numerical features $p$ and $n$ :
$\triangleright p, \neg p, n>0, n=0$ can appear in $C$
$\triangleright p, \neg p, n \uparrow, n \downarrow, n$ ? can appear in $E$
- Semantics: State pair $\left(s, s^{\prime}\right)$ satisfies sketch rule $C \mapsto E$ if
$\triangleright f(s)$ satisfies $C$
$\triangleright\left(f(s), f\left(s^{\prime}\right)\right)$ satisfies $E$

Syntax of sketches and policies the same, and so with semantics, except that ( $s, s^{\prime}$ ) is not a 1 -step state transition necessarily

Interpretation: When in state $s$, the set of subgoal states $G_{R}(s)$ to aim at is:

$$
G_{R}(s)=\left\{s^{\prime} \mid\left(s, s^{\prime}\right) \text { satisfies sketch rule or } s^{\prime} \text { is goal }\right\}
$$

## Sketch Width

- Sketch $R$ splits problems $P$ in $\mathcal{Q}$ into collection of subproblems $P\left[s, G_{R}(s)\right]$ :
$\triangleright$ Initial state $s$ : reachable state $s$ in $P$
$\triangleright($ Sub $)$ goal states $G_{R}(s)=\left\{s^{\prime} \mid\left(s, s^{\prime}\right)\right.$ satisfies sketch rule or $s^{\prime}$ is goal $\}$
- Width of sketch $R$ over $\mathcal{Q}=\max _{s, P \in \mathcal{Q}}$ width $\left(P\left[s, G_{R}(s)\right]\right)$
$\triangleright$ for definition in presence of dead-ends, see refs
Theorem: Any $P$ in $\mathcal{Q}$ is solvable in $O\left(b \cdot N^{|\Phi|+2 k-1}\right)$ time by $\operatorname{SIW}_{\mathrm{R}}$ algorithm if sketch $R$ is terminating and has width over $\mathcal{Q}$ bounded by $k$ [B. and G., 2021]
$\triangleright N$ : Number of atoms in problem $P$; $\Phi$ : Set of features in sketch
$\operatorname{SIW}_{\mathrm{R}}$ is like SIW but subgoal to achieve next given by sketch
$\triangleright$ SIW is $\operatorname{SIW}_{\mathrm{R}}$ with sketch $R$ with single rule: $\{\# g>0\} \mapsto\{\# g \downarrow\}$


## Another Example: IPC Grid [Drexler et al., 2021]

This sketch is terminating and has width 1 for IPC domain Grid (pick and deliver keys spread in grid where cells can be locked and opened with other keys):

- Sketch:
$\begin{aligned} & \triangleright r_{1}:\{l>0\} \mapsto\{l \downarrow, k ?, o ?, t ?\} \\ & \triangleright r_{2}:\{l=0, k>0\} \mapsto\{k \downarrow, o ?, t ?\} \\ & \triangleright r_{3}:\{l>0, \neg o\} \mapsto\{o, t ?\} \\ & \triangleright r_{4}:\{l=0, \neg t\} \mapsto\{o ?, t\} \quad \text { (if }\end{aligned}$
- Features:
$\triangleright l$ is the number of unlocked grid cells
$\triangleright k$ is the number of misplaced keys
$\triangleright o$ is true iff robot holds key for which there is a closed lock
$\triangleright t$ is true iff robot holds key that must be placed at some target grid cell


## Preview: Learning Sketches [Drexler et al., 2022]

Given a known domain $D$, training instances $P_{1}, \ldots, P_{n}$, and non-negative integer $k$, find simplest sketch $R$ over a pool of features $\mathcal{F}$ such that

- Subproblems induced by $R$ on each $P_{i}$ have all width bounded by $k$,
- Sketch $R$ is terminating

Possibly first approach for learning subgoal structure based on crisp principles

Many threads that come together:

- Planning width
- Language of general policies
- Termination notion from QNPs
- Semantics of sketches


## Exercise: Test Your Knowledge! (Not trivial)

In the 1985 AIJ paper, Macro-Operators: A Weak Method for Learning, Rich Korf provides macro-tables for puzzles like Rubik Cube, 8-puzzle, and other hard puzzles that encode policies $\pi(s)$ for solving them from any initial state

- Can these compact policies be replaced by even more compact sketches of bounded width?
- Can these sketches be general? That is, applicable to Rubik cubes and $n$-sliding puzzles of different sizes?
- Can such sketches be learned with current method? Expressivity? Scalability? Other methods?


## Background 2: <br> Qualitative Numerical Planning Problems (QNPs)

## Language for QNPs

- Language for planning involving propositional and numerical variables
- QNPs [Srivastava et al. 2011] different than numerical planning:
$\triangleright$ Numerical vars in QNPs are non-negative, real-valued
$\triangleright$ Effects on numerical variables: just qualitative increments/decrements
$\triangleright$ Numerical literals: whether variable is zero or positive only
- These differences make plan-existence for QNPs decidable
- QNPs provide language for general policies and sketches:
$\triangleright$ QNP actions similar to policy/sketch rules but features replaced by variables
- We follow [B. and G., 2020b]


## Syntax for QNPs

A qualitative numerical problem (QNP) is tuple $Q=\langle F, V, I, O, G\rangle$ :

- $F$ and $V$ are sets of propositional and numerical variables (not features!)
- $I$ and $G$ denote initial and goal states
- $O$ : actions $a$ with precs, and prop. and numeric effects $\operatorname{Pre}(a), \operatorname{Eff}(a), N(a)$ :
$\triangleright F$-literals may appear in $I, G, \operatorname{Pre}(a)$ and $\operatorname{Eff}(a)$
$\triangleright V$-literals may appear in $I, G$ and $\operatorname{Pre}(a)$
$\triangleright N(a)$ can only have expressions of the form $X \uparrow$ and $X \downarrow$ for var $X$ in $V$
- $V$-literal is either $X=0$ or $X>0$ for variable $X$ in $V$
- Example: QNP $Q_{\text {clear }}=\langle\{H\},\{n\}, I, O, G\rangle$
$\triangleright I=\{n>0, \neg H\}$
$\triangleright G=\{n=0\}$
$\triangleright O=\{a, b\}$ where $a=\{\neg H, n>0\} \mapsto\{H, n \downarrow\}$ and $b=\{H\} \mapsto\{\neg H\}$
- QNP actions like policy rules above but $H$ and $n$ not features but variables


## Semantics and Solutions of QNPs

- Policy $\pi$ for a QNP is partial map from state $s$ into actions such that:
$\triangleright \pi(s)=\pi\left(s^{\prime}\right)$ if $s$ and $s^{\prime}$ qualitatively similar: same $F$ and $V$ true literals
- $\pi$ solves QNP if all maximal QNP-fair $\pi$-trajectories reach the goal
$\triangleright$ QNP fairness: trajectory unfair if numerical variable decremented infinite number of times and incremented finite number of times.

Theorem [Srivastava et al., 2011]: $\pi$ solves QNP $Q$ iff $\pi$ is strong cyclic solution of the FOND problem $T_{D}(Q)$ obtained from $Q$ that terminates

- $T_{D}(Q)$ replaces numerical $X$ by Boolean variable " $X>0$ " (" $X=0$ " is negative literal)
- Qualitative effects $X \uparrow$ replaced by effect $X>0$
- Qualitative effects $X \downarrow$ replaced by non-deterministic effect " $X>0 \mid X=0$ "
- Strong-cyclic: every reachable state is connected to goal state by $\pi$

Polytime reduction from QNPs to FOND, but more complex than $T_{D}$ [B. and G., 2020b]

## Termination, Sieve Algorithm [Srivastava et al., 2011]

## Policy for QNP $Q$ terminates if no infinite QNP-fair $\pi$-trajectories

Sieve provides sound and complete polynomial termination test

- State $s$ terminates if either
$\triangleright$ there is no cycle on state $s$, or
$\triangleright$ every cycle on $s$ contains a state $s^{\prime}$ that terminates, or
$\triangleright \pi(s)$ decrements a variable $X$, and every cycle on $s$ that contains a state $s^{\prime}$ such that $\pi\left(s^{\prime}\right)$ increments $X$, contains another state $s^{\prime \prime}$ that terminates
- Policy $\pi$ terminates iff every state reached by $\pi$ terminates

Recent FOND ${ }^{+}$planner handles strong FOND, strong cyclic FOND, QNPs, and hybrids by stating fairness assumptions explicitly [Rodriguez et al. 2021b]

## Part III: Learning Dynamics, Policies, Sketches

- Learning action models:

Given graphs $G_{1}, \ldots, G_{k}$, find simplest instances $P_{i}=\left\langle D, I_{i}\right\rangle$ such that graphs $G_{i}$ and $G\left(P_{i}\right)$ are isomorphic, $i=1, \ldots, k$.

- Learning general policies:

Given known domain $D$, training instances $P_{1}, \ldots, P_{k}$, over $D$, and finite pool of domain features $\mathcal{F}$, each with a cost, find the cheapest policy $\pi$ over $\mathcal{F}$ such that $\pi$ solves all $P_{i}, i=1, \ldots, k$

- Learning sketches:

Given known domain $D$, training instances $P_{1}, \ldots, P_{n}$, and non-negative integer $k$, find simplest sketch $R$ over a pool of features $\mathcal{F}$ such that
$\triangleright$ Subproblems induced by $R$ on each $P_{i}$ have all width bounded by $k$,
$\triangleright$ Sketch $R$ is terminating

## Learning Action Models: Encoding [Rodriguez et al., 2021a]

- Construct answer set program, bounding number of objects, preds, and action/pred. arities:
$\triangleright$ Given $G_{1}, \ldots, G_{n}$ as input graphs over black-box states, with edge labels,
$\triangleright$ Check whether there is STRIPS model $D$ and instances $I_{1}, \ldots, I_{n}$ such that graphs $G\left(P_{i}\right)$ and $G_{i}$ are isomorphic, $i=1, \ldots, n$, where $P_{i}=\left\langle D, I_{i}\right\rangle$
$\triangleright$ Optimize: sum of action and predicate arities, etc
- (Basic) choice variables:
$\triangleright$ Lifted atom is pair ( $\mathrm{P}, \mathrm{T}$ ) where P is int and T is tuple of ints
$\triangleright \operatorname{prec}(\mathrm{A},(\mathrm{P}, \mathrm{T}), \mathrm{V})$ and eff(A, $\mathrm{P}, \mathrm{T}), \mathrm{V})$
(lifted atoms in precs/effects)
$\triangleright$ p_arity $(\mathrm{P}, \mathrm{N})$ and a_arity (A,N) (arities for predicate and action)
$\triangleright \operatorname{val}(S,(P, O), V)$ where 0 is tuple of objs and $V$ is $0 / 1 \quad$ (value of ground atoms at states)
$\triangleright \operatorname{appl}(\mathrm{A}, \mathrm{O}, \mathrm{S})$ and next (A,O,S,T) (ground action A(0) appl/assigned to (S,T))
- (Basic) constraints:
$\triangleright:-\operatorname{state}(S), \operatorname{state}(T), S<T, \operatorname{val}(T,(P, O), V): \operatorname{val}(S,(P, O), V) . \quad$ (diff. states)
$\triangleright\{\operatorname{next}(\mathrm{A}, \mathrm{O}, \mathrm{S}, \mathrm{T}): \operatorname{label}((\mathrm{S}, \mathrm{T}), \mathrm{A})\}=1:-\operatorname{appl}(\mathrm{A}, \mathrm{O}, \mathrm{S})$. (assign edges to actions)
$\triangleright:-\operatorname{state}(\mathrm{S}), \operatorname{action}(\mathrm{A}), \mathrm{N}=\{\operatorname{label}((\mathrm{S}, \mathrm{T}), \mathrm{A})\},\{\operatorname{appl}(\mathrm{A}, \mathrm{O}, \mathrm{S})\}!=\mathrm{N}$. (matching)
$\triangleright$ Compliance of precs/effects of assigned grounded actions to edges
- Clingo program $\sim 400$ lines [Rodriguez et al. 2021a]; more complex in SAT [B. and G., 2020a]


## Learning General Policies: Encoding [Francès et al., 2021]

- Input is set of transitions $\mathcal{S}$ from small instances, pool of features $\mathcal{F}$, parameter (int) $\delta$
- Output is policy: rules obtained from selected features and ("good") transitions
- Combinatorial opt. task $T(\mathcal{S}, \mathcal{F}, \delta)$ : Solve constraints minimizing feature complexity
- Choice variables:

```
\triangleright select(F)
\triangleright good(S,T) (transition (S,T) is "compatible" with policy)
| V(S,N)
(features that define rules)
    (distance from S to goal is N)
```

- Constraints:

```
\triangleright 1 \{ \operatorname { g o o d } ( \mathrm { S } , \mathrm { T } ) \} ~ : - ~ s t a t e ( S ) , ~ n o t ~ t e r m i n a l ( S ) . ~ ( g o o d ~ t r a n s i t i o n s ~ a t ~ n o n - t e r m i n a l s )
\triangleright :- good(S,T), deadend(T). (no good tr. reaches dead-end T)
\triangleright 1 \{ \operatorname { s e l e c t ( F ) : \operatorname { d i f f ( F , S , T ) \} ~ : - ~ g o a l ( S ) , ~ n o t ~ g o a l ( T ) . ~ ( d i s t i n g u i s h ~ g o a l s ) } }
\triangleright {V(S,D):V V
    (set distances)
\triangleright :- good(S,T), V(S,D1), V(T,D2), D1 <= D2.
    (distances avoid cycles)
\triangleright 1 { select(F): diff(F,S1,T1,S2,T2)} :- good(S1,T1), not good(S2,T2).
```

    (distinguish good/bad transitions)
    where diff/3 and diff/5 computed from pool at pre-processing

## Learning General Sketches: Encoding [Drexler et al., 2022]

- Input: transitions $\mathcal{S}$ in small instances, pool $\mathcal{F}$, width bound $k$, max \# sketch rules $m$
- Output: sketch of width $\leq k$, acyclic in given instances, with up to $m$ rules
- Combinatorial opt. task $T(\mathcal{S}, \mathcal{F}, k, m)$ : solve constraints min complexity of selected features
- (Basic) variables:

```
\triangleright rule(I)
\triangleright select(F)
\triangleright cond(I,F,V) and eff(I,F,E)
\triangleright subgoal(S,T)
\triangleright (Implied) subgoal(S1,T,S2)
\triangleright (Implied) satis(S1,S2,I)
```

(sketch rule I)
(features that define sketch rules)
(conditions and effects for rule I)
(tuple T of width $k$ is subgoal for S )
(subgoal T for S 1 may lead to S 2 )
(pair (S1, S2) satisfies rule I)

- (Basic) constraints:
$\triangleright$ Well formed rules: atoms cond/3 and eff/3 are consistent and imply select (F)
$\triangleright 1\{\operatorname{subgoal}(\mathrm{~S}, \mathrm{~T}):$ tuple( T$)\}:-\operatorname{state}(\mathrm{S})$, not goal(S). (width $k$ subgoal for S$)$
$\triangleright$ subgoal (S1, T, S2) :- subgoal (S1,T), found (S1,T,S2). (subgoal T may lead to S 2 )
$\triangleright:-\operatorname{subgoal}(\mathrm{S} 1, \mathrm{~T}, \mathrm{~S} 2)$, not satis(S1,S2,I): rule(I). ((S1, S2) satisfies some rule)
$\triangleright$ :-satis(S1,S2,I), not subgoal (S1,T) : $d(\mathrm{~S} 1, \mathrm{~T})<d(\mathrm{~S} 1, \mathrm{~S} 2)$. (dead-end S 2 is farther)
$\triangleright:-\operatorname{satis}(\mathrm{S} 1, \mathrm{~S} 2, \mathrm{I})$, not subgoal $(\mathrm{S} 1, \mathrm{~T}): d(\mathrm{~S} 1, \mathrm{~T}) \leq d(\mathrm{~S} 1, \mathrm{~S} 2) . \quad$ (subgoals optimal)
$\triangleright$ Collection of rules is terminating (approx'ed by testing acyclicity)


## About the Pool of Features $\mathcal{F}$ [B. et al., 2019]

- Description logic grammar allows generation of concepts and roles from domain predicates
- Complexity of concept/role given by size of its syntax tree
- Pool $\mathcal{F}$ obtained from concepts of complexity bounded by parameter
- Denotation of concept $C$ in state $s$ is subset of objects
- Each concept $C$ defines num and Bool features $n_{C}(s)=|C(s)| ; p_{C}(s)=$ T iff $|C(s)|>0$
- Grammar:
$\triangleright$ Primitive: $C_{p}$ given by unary predicates $p$ and unary "goal predicates" $p_{G}$
$\triangleright$ Universal: $C_{u}$ contains all objects
$\triangleright$ Nominals: $C_{a}=\{a\}$ for constants/parameter $a$
$\triangleright$ Negation: $\neg C$ contains $C_{u} \backslash C$
$\triangleright$ Intersection: $C \sqcap C^{\prime}$
$\triangleright$ Quantified: $\exists R . C=\{x: \exists y[R(x, y) \wedge C(y)]\}$ and $\forall R . C=\{x: \forall y[R(x, y) \wedge C(y)]\}$
$\triangleright$ Roles (for binary predicate $p$ ): $R_{p}, R_{p}^{-1}, R_{p}^{+}$, and $\left[R_{p}^{-1}\right]^{+}$
- Additional distance features $\operatorname{dist}\left(C_{1}, R, C_{2}\right)$ 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 $d$


## General Policies By Deep Learning [Ståhlberg et al., 2022a,b]

- Exploits correspondence between graph neural networks (GNNs) and twovariable $\operatorname{logic} \mathcal{C}_{2}$ to learn policy without requiring pool of $\mathcal{C}_{2}$ features $\mathcal{F}$
- Value function $V$ learned that yields general policy $\pi_{V}$ greedy in $V$
- For generalization, based on GNN arch. for $\operatorname{MaxCSP}(\Gamma)$ [Toenshoff et al., 2021]
$\triangleright$ Input given by the states $s$ extended with "goal predicates" $p_{G}$
$\triangleright$ Output $V(s)$ is non-linear aggregation of object embeddings
$\triangleright$ Loss: $\left|V^{*}(s)-V(s)\right|$ for supervised learning of optimal policies
$\triangleright$ Loss: $\max \left\{0,\left[1+\min _{s^{\prime} \in N(s)} V\left(s^{\prime}\right)\right]-V(s)\right\}$ unsupervised/non-optimal
- Nearly as good as policies based on explicit pool $\mathcal{F}$ of $\mathcal{C}_{2}$ features
- Complexity of "latent features" not explicitly bounded


## GNN Architecture [Ståhlberg et al., 2022a,b]

```
Algorithm 1: GNN maps state \(s\) into scalar \(V(s)\)
    Input: State \(s\) : set of atoms true in \(s\), set of objects
    Output: V(s)
\(1 f_{0}(o) \sim \mathbf{0}^{k / 2} \mathcal{N}(0,1)^{k / 2}\) for each object \(o \in s\);
2 for \(i \in\{0, \ldots, L-1\}\) do
3 for each atom \(q:=p\left(o_{1}, \ldots, o_{m}\right)\) true in \(s\) do
\(/ / \mathrm{Msgs} q \rightarrow o\) for each \(o=o_{j}\) in \(q\)
\(m_{q, o}:=\left[\mathbf{M L P}_{p}\left(f_{i}\left(o_{1}\right), \ldots, f_{i}\left(o_{m}\right)\right)\right]_{j} ;\)
        for each o in \(s\) do
                        // Aggregate, update embeddings
                    \(f_{i+1}(o):=\operatorname{MLP}_{U}\left(f_{i}(o), \operatorname{agg}\left(\left\{\left\{m_{q, o} \mid o \in q\right\}\right)\right) ;\right.\)
    Final Readout
\({ }_{7} V:=\mathbf{M L P}_{2}\left(\sum_{o \in s} \operatorname{MLP}_{1}\left(f_{L}(o)\right)\right)\)
```


## Wrap Up: Representation Learning for Acting and Planning

- Background 1: Classical planning, planning width
- Languages for
$\triangleright$ representing general dynamics
$\triangleright$ representing general policies
$\triangleright$ representing general subgoal structures (sketches; 'intrinsic rewards")
- Background 2: Qualitative numerical planning problems (QNPs)
- Learning representations over these languages:
$\triangleright$ learning general dynamics
$\triangleright$ learning general policies
$\triangleright$ learning general subgoal structures
- Wrap up; Challenges


## Wrap Up

- To learn representations that generalize due to structure, don't play with low-level neural architecture; choose suitable (domain-independent) target language and learn representations over it:
$\triangleright$ generalization
$\triangleright$ transparency
$\triangleright$ powerful, meaningful bias
$\triangleright$ distinction between what and how
- Examples of learning language-based representations to act and plan:
$\triangleright$ general action dynamics
$\triangleright$ general policies
$\triangleright$ general subgoal structures (sketches)


## Challenges: Language-based Representation Learning

- Scalability of combinatorial optimization approaches
- Use of deep learning (learning lifted dynamics, policies, sketches).
- Alternative target languages for learning (e.g., vs. lifted STRIPS)
- Continuous domains, space, time
- Stochastic and non-deterministic domains
- States in the input: black-box, parsed images, images, videos
- Grounded vs. ungrounded representations
- Learning and reusing "skills", hierarchies
- . .
https://www.dtic.upf.edu/~hgeffner/tutorial-2022.pdf
Plenty to do; if seriously interested, reach us


## References

[Asai, 2019] Asai, M. (2019). Unsupervised grounding of plannable first-order logic representation from images. In Proc. ICAPS.
[Bonet et al., 2019] Bonet, B., Francès, G., and Geffner, H. (2019). Learning features and abstract actions for computing generalized plans. In Proc. AAAI, pages 2703-2710.
[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. AAAl, pages 11764-11773.
[Chevalier-Boisvert et al., 2019] Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. (2019). Babyai: A platform to study the sample efficiency of grounded language learning. In ICLR.
[Cresswell et al., 2013] Cresswell, S. N., McCluskey, T. L., and West, M. M. (2013). Acquiring planning domain models using LOCM. The Knowledge Engineering Review, 28(2):195-213.
[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. AAAl, pages 11801-11808.
[Garg et al., 2020] Garg, S., Bajpai, A., and Mausam (2020). Symbolic network: generalized neural policies for relational mdps. In International Conference on Machine Learning, pages 3397-3407.
[Geffner and Bonet, 2013] Geffner, H. and Bonet, B. (2013). A Concise Introduction to Models and Methods for Automated Planning. Morgan \& Claypool Publishers.
[Hu and De Giacomo, 2011] Hu, Y. and De Giacomo, G. (2011). Generalized planning: Synthesizing plans that work for multiple environments. In Proc. IJCAI, pages 918-923.
[Khardon, 1999] Khardon, R. (1999). Learning action strategies for planning domains. Artificial Intelligence, 113:125-148.
[Konidaris et al., 2018] Konidaris, G., Kaelbling, L. P., and Lozano-Perez, T. (2018). From skills to symbols: Learning symbolic representations for abstract high-level planning. Journal of Artificial Intelligence Research, 61:215-289.
[Lipovetzky and Geffner, 2012] Lipovetzky, N. and Geffner, H. (2012). Width and serialization of classical planning problems. In Proc. ECAI, pages 540-545.
[Lipovetzky and Geffner, 2017a] Lipovetzky, N. and Geffner, H. (2017a). Best-first width search: Exploration and exploitation in classical planning. In Proc. AAAI.
[Lipovetzky and Geffner, 2017b] Lipovetzky, N. and Geffner, H. (2017b). A polynomial planning algorithm that beats lama and ff. Proc. ICAPS.
[Martín and Geffner, 2004] Martín, M. and Geffner, H. (2004). Learning generalized policies from planning examples using concept languages. Applied Intelligence, 20(1):9-19.
[Rodriguez et al., 2021a] Rodriguez, I. D., Bonet, B., Romero, J., and Geffner, H. (2021a). Learning first-order representations for planning from black-box states: New results. In KR. arXiv preprint arXiv:2105.10830.
[Rodriguez et al., 2021b] Rodriguez, I. D., Bonet, B., Sardina, S., , and Geffner, H. (2021b). Flexible fond planning with explicit fairness assumptions. In Proc. ICAPS, pages 290-298.
[Srivastava et al., 2011] Srivastava, S., Zilberstein, S., Immerman, N., and Geffner, H. (2011). Qualitative numeric planning. In AAAI.
[Ståhlberg et al., 2022a] Ståhlberg, S., Bonet, B., and Geffner, H. (2022a). Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits. In Proc. ICAPS.
[Ståhlberg et al., 2022b] Ståhlberg, S., Bonet, B., and Geffner, H. (2022b). Learning generalized policies without supervision using gnns. In Proc. KR.
[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.
[Toyer et al., 2018] Toyer, S., Trevizan, F., Thiébaux, S., and Xie, L. (2018). Action schema networks: Generalised policies with deep learning. In AAAI.

