From Model-free to Model-based AI: Representation Learning for Planning

43rd German Conf. on AI (KI 2020), Bamberg

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, Spain

Wallenberg Guest Professor
Linköping University, Sweden

23/9/2020

Outline

- **Problem of generality** in AI
- Model-free **Learners**
- Model-based **Solvers**
- **Systems 1 and 2?**
- **Integration** of learners and solvers:
  - ▷ Learning **symbolic representations from data**
  - ▷ Learning **from symbolic representations**

Ref: *Model-free, model-based, and general intelligence.* H. G., Proc. IJCAI 2018

AI Programming and Problem of Generality

There was a time (60s, 70s, 80s) when AI was done mostly by **programming**:

- pick up a challenging task and domain X (humor, story understanding, ...)
- analyze/introspect/find out how task is solved
- capture this reasoning in a program

Great ideas and great books on programming and **AI programming** came out from this work, but **methodological problem:**

- Programs written by hand were **not robust or general**

From Programs to Learners and Solvers

- This problem led to **methodological shift**:

  - from writing **programs for ill-defined problems** . . .
  - to designing **algorithms for well-defined mathematical tasks**

- New general programs **learners** and **solvers** have a **crisp functionality**: both can be seen as computing **functions** that map inputs into outputs

$$\textit{Input } x \Longrightarrow \boxed{\textsc{Function } f} \Longrightarrow \textit{Output } f(x)$$

- The algorithms are **general** in the sense that they are not tied to particular examples but to classes of **models** and **tasks** expressed in **mathematical form**

Learners (1)

$$\textit{Input } x \Longrightarrow \boxed{\textsc{Function } f} \Longrightarrow \textit{Output } f(x)$$

- In **deep learning (DL)** and **deep reinforcement learning (DRL)**, training results in function f_θ

- f_θ given by structure of **neural network** and adjustable parameters θ
  - ▷ In DL, **input** x may be an image and **output** $f_\theta(x)$ a classification label
  - ▷ In DRL, **input** x may be state of game, and **output** $f_\theta(x)$, value of state

- Parameters θ learned by **minimizing error function**
  - ▷ In DL, error depends on inputs and target outputs in training set
  - ▷ In DRL, error depends on value of states and successor states

- Most common **optimization algorithm** is **stochastic gradient descent**

Learners (2)

$$\textit{Input } x \Longrightarrow \boxed{\textsc{Function } f} \Longrightarrow \textit{Output } f(x)$$

- Excitement about AI due to **successes in DL and DRL**
  - ▷ Breakthroughs in image understanding, speech recognition, Go, . . .
  - ▷ Superhuman performance in Chess and Go from **self-play** alone
- The basic ideas underlying DL and DRL not new but from 80s and 90s
  - ▷ Recently, more CPU power, more data, deeper nets, attractive problems
- DL and DLR remarkably powerful **yet** they
  - ▷ require lots of training and data
  - ▷ lack understanding
  - ▷ are hard to understand as well
  - ▷ are not trustworthy (self-driving cars?)

Solvers

$$\textit{Input } x \Longrightarrow \boxed{\text{FUNCTION } f} \Longrightarrow \textit{Output } f(x)$$

- **Solvers** derive output $f(x)$ for **given input** x from **model**:
  - ▷ **SAT:** x is a formula in CNF, $f(x) = 1$ if x satisfiable, else $f(x) = 0$
  - ▷ **Classical planner:** x is a planning problem P, and $f(x)$ is plan that solves P
  - ▷ **Bayesian net:** x is a query over Bayes Net and $f(x)$ is the answer
  - ▷ **Constraint satisfaction, Markov decision processes, POMDPs, . . .**

- **Generality:** Solvers not tailored to particular examples
- **Expressivity:** Some models very expressive, "AI-Complete" (POMDPs)
- **Learners are solvers too:** $\text{argmin}_w \sum_{x \in D} L(x, f_w(x))$ (Diff. programming)
- **Complexity:** Computation of $f(x)$ is (NP) hard; $|x|$ **not bounded**
- **Challenge:** Solvers shouldn't break just because x has many variables

Learners vs Solvers

$$\textit{Input } x \Longrightarrow \boxed{\textsc{Function } f} \Longrightarrow \textit{Output } f(x)$$

- **Learners** require **experience over related problems** x but then fast
  - ▷ They compute function f from training, then apply it
- **Solvers** deal with **completely new problems** x but need **to think**
  - ▷ They compute $f(x)$ for each input x from scratch

Thinking is hard but essential for dealing with new problems

Thinking can be done **effectively** with right computational ideas

Next: Thinking effectively in context of **planning**

Classical Planning: Finding Plans in Huge Mental Mazes

Challenge: find path to goal in graph with # nodes **exponential** in # variables

Old Idea: If you don't know how to solve P, **solve simpler problem** P', and use solution of P' for solving P (Polya, Minsky, Pearl)

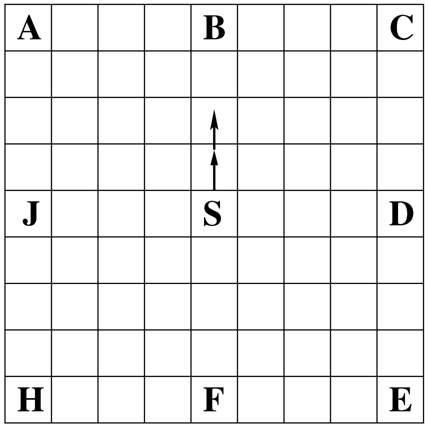
- In **monotonic relaxation** P', effects of actions on variables made **monotonic**
- Monotonicity makes relaxation P' **decomposable** and therefore **tractable**
- **Heuristic** $h(s)$ in P set to **cost of plan from** s **in relaxation** P'

Heuristic obtained and used to solve **any problem** P *from scratch*

No experience required *in problems related to* P

(McDermott 1996, Bonet, Loerincs, G. 1997, . . .)

Goal Recognition



- **Task:** infer **agent goal** $G \in \mathcal{G}$ from **observations** O on behavior
- Bayes' rule: $P(G|O) = P(O|G)\,P(G)/P(O)$, priors $P(G)$ assumed given
- Likelihood $P(O|G)$ set as monotonic function f of **cost difference:**
  - ▷ $c^{-}(G)$: cost of reaching G with plan incompatible with observations
  - ▷ $c^{+}(G)$: cost of reaching G with plan compatible with observations

$P(G|O)$ *computed using* **Bayes' rule** *and* $2|\mathcal{G}|$ **calls to planner**

No experience required *in related problems*

(Ramirez and G. 2009, 2010)

Polynomial Algorithms for Exponential Spaces: Structure

- IW(1) is a **breadth-first search** that **prunes** states s that don't make a feature true for first time in the search, from given **set of boolean features** F

- IW(k) is IW(1) but over set F^k made up of conjunctions of k features from F

  - ▷ Most domains have **small width** $w \leq 2$ when **goals are single atoms**
  - ▷ **Any** such instances solved **optimally** by IW(w) in **low poly time**

- IW(k) can work with **simulators**. No PDDL or goal needed. **Variants:**

  - ▷ BFWS(R): SOTA planning algorithm which doesn't use **action structure**
  - ▷ Rollout IW(1): fast **on-line planner** that plays Atari from **screen pixels**

(Lipovetzky and G. 2012; Lipovetzky, Ramirez, G. 2015; Bandres, Bonet, G. 2018)

Learners vs. Solvers (2)

- Rollout IW(1) **planner** and DQN **learner** perform comparably well in Atari
- They illustrate **key difference between learners and solvers:**
  - ▷ DQN requires lots of training data and time, and then plays very fast
  - ▷ Rollout IW(1) plays out of the box but thinking a bit before each move

This is a general characteristic:

- **Learners** require **experience over related problems** x but then are fast
  - ▷ They compute function f from training, then apply it
- **Solvers** deal with **completely new problems** x but need **to think**
  - ▷ They compute $f(x)$ for each input x from scratch

Learners and Solvers: System 1 and System 2?

Dual process accounts of the human mind assume two processes (D. Kahneman: Thinking, Fast and Slow, 2011; K. Stanovich: The Robot's Rebellion, 2005)

System 1 (Intuitive Mind)	**System 2** (Analytical Mind)
fast	slow
associative	deliberative
unconscious	conscious
effortless	effortful
parallel	serial
specialized	general
...	...
Learners?	Solvers?

Learners and Solvers: Challenges

- **Top goal:** General **two-way integration** of System 1 and System 2 inference in AI systems; i.e. **learners** and **solvers**

- **Challenge: Learn representation of models used by solvers from data**
  - ▷ symbols and state variables, first-order models, abstractions

- **Two dimensions in representation learning** for planning:
  - ▷ Learning from what: symbolic, non-symbolic, or black-box states
  - ▷ Learning for what: model-free control, model-based control, generalized model

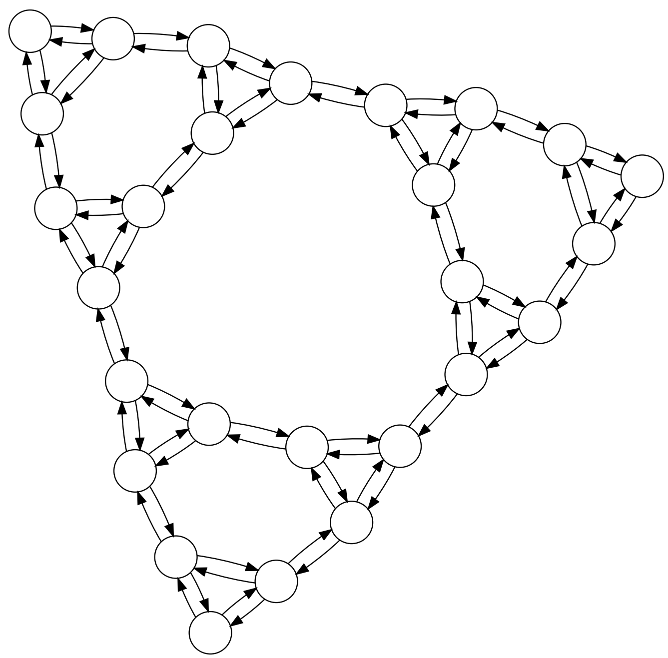
- **Next:** We address **two points** in this space:
  - ▷ Learning **first-order symbolic action model** from black-box states
  - ▷ Learning **generalized planning models** from symbolic action models

Learning first-order models from the structure of state space

Can we learn this . . .

```
Move(fr,to,d): Move disk $d$ from disk $fr$ to disk $to$
 Static: LARGER(fr,d),LARGER(to,d) NEQ(fr,to)
 Pre: clear(to),clear(d), on(d,fr),-on(d,to)
 Eff: clear(fr),-clear(to),-on(d,fr),on(d,to)
```

. . . from this?



Formulation: Target Language

- Planning instance in PDDL is $P = \langle D, I \rangle$ where D is **first-order domain** (relations, action schemas) and I provides **instance information** (objects and relations they satisfy initially)

- A planning instance P defines a **state graph** G

- **Question:**

> ▷ Can we **learn** $P = \langle D, I \rangle$ back from the graph G?
>
> ▷ Can we **learn** $P_i = \langle D, I_i \rangle$, $i = 1, \ldots, k$ from graphs $G_1, \ldots, G_k$?
>
> (This means **learning action schemas and relations from graphs**)

Learned domain D can be used then to plan over **any** domain instances

Formulation: From State Graph to First-Order PDDL

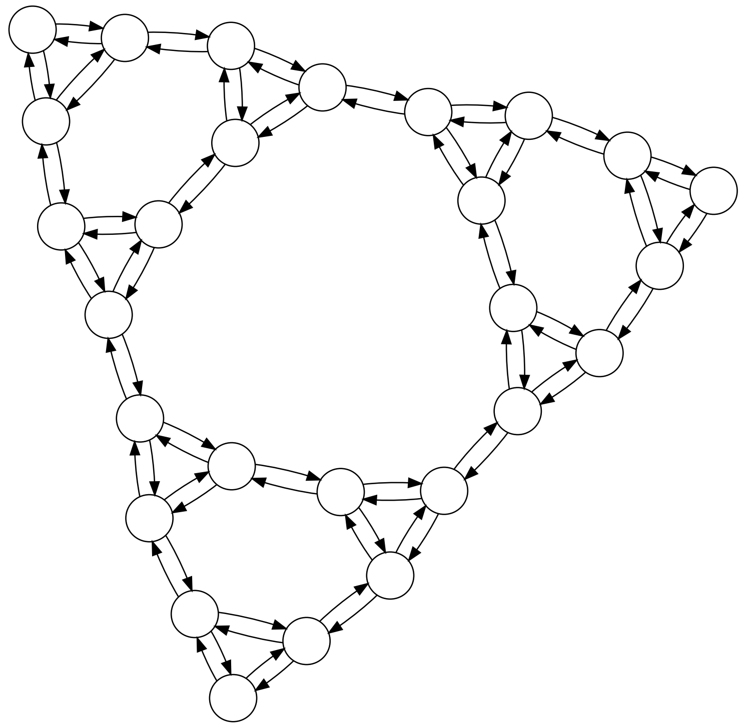
- **Task:** Find **simplest** instances $P_i = \langle D, I_i \rangle$ that account for **input** labeled graphs G_i, $i = 1, \ldots, k$, **without knowing anything about D or the I_i's**

- Space of possible domains D bounded by **small values** of a **small number of hyerparameters**: number of action schemas, predicates, arities.

- **Target language** and **bounds** provide **strong structural priors** and make task **combinatorial**, expressed and solved via **SAT**

Learning first-order symbolic representations from the structure of the state space, B. Bonet, H. G., ECAI 2020

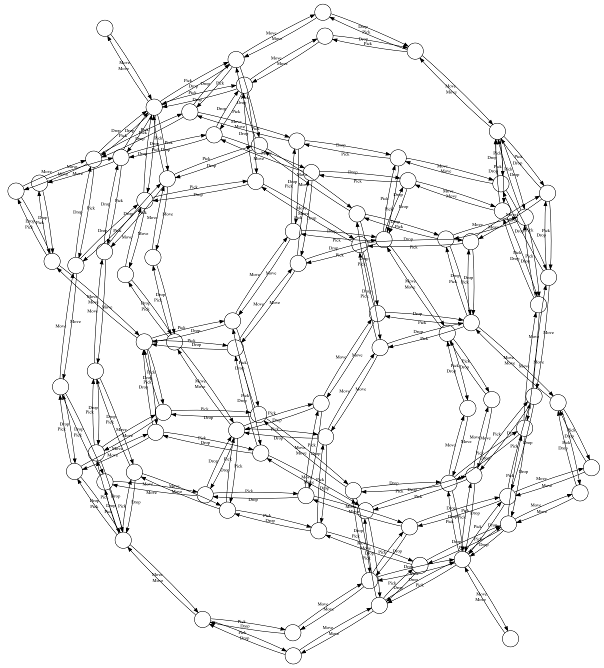
Example: Hanoi. Input and Output



<u>Move(fr,to,d):</u>

```
 Static: LARGER(fr,d),LARGER(to,d) NEQ(fr,to)
 Pre: -clear(fr),clear(to),clear(d),Non(fr,d),-Non(d,fr),Non(d,to)
 Eff: clear(fr),-clear(to),Non(d,fr),-Non(d,to)
```

Example: Gripper. Input and Output

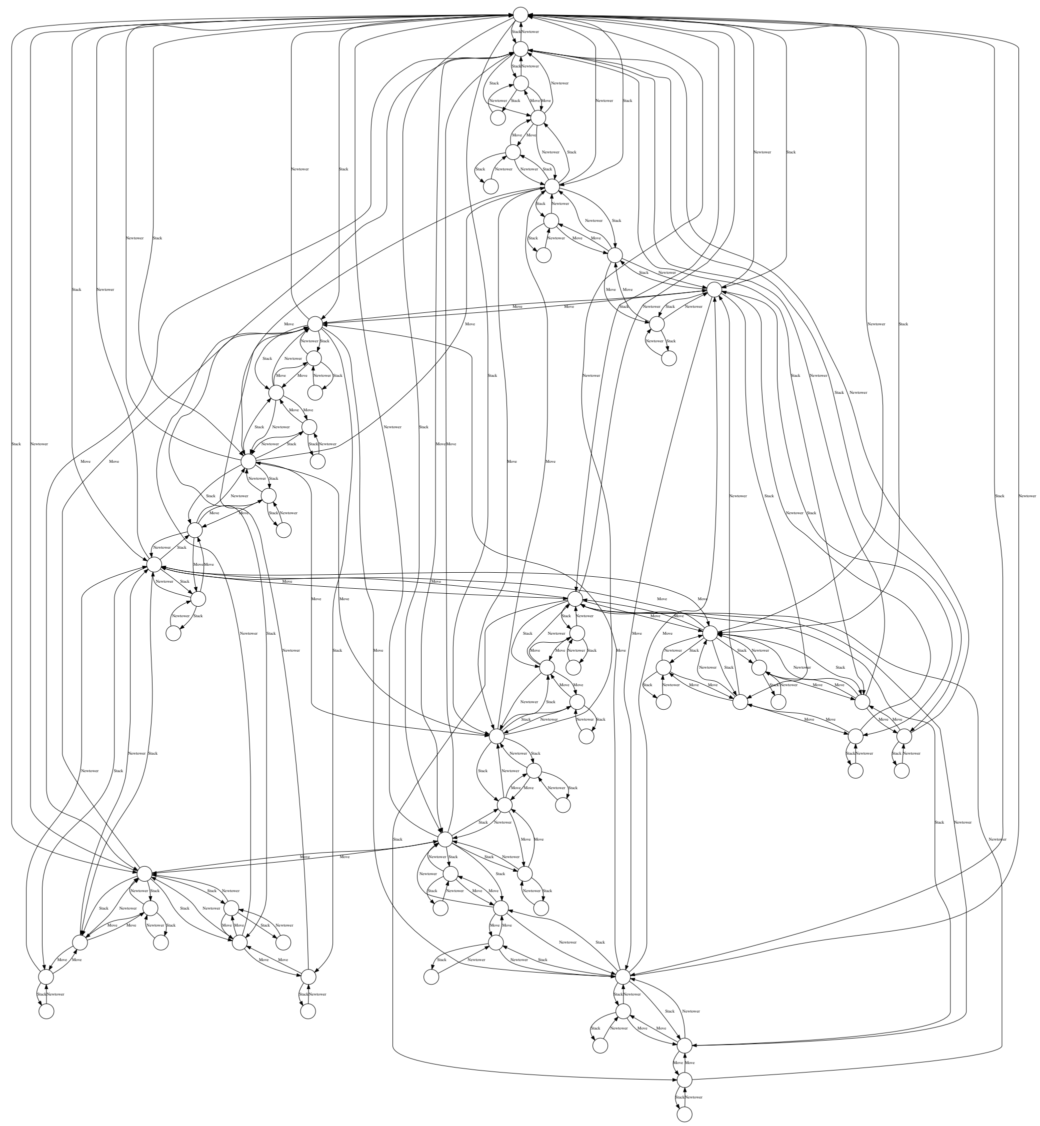


Move(from,to):
```
 Static: CONN(from,to)
 Pre: at(from),-at(to)
 Eff: -at(from),at(to)
```

Drop(ball,room,gripper):
```
 Static: PAIR(room,gripper)
 Pre: at(room),Nfree(gripper),hold(gripper,ball),Nat(room,ball)
 Eff: -Nfree(gripper),-hold(gripper,ball),-Nat(room,ball)
```

Pick(ball,room,gripper):
```
 Static: PAIR(room,gripper)
 Pre: at(room),-Nfree(gripper),-hold(gripper,ball),-Nat(room,ball)
 Eff: Nfree(gripper),hold(gripper,ball),Nat(room,ball)
```

Example: Blocks. Input



Example: Blocks. Output

```
MovetoTable(x,y):
 Static: NEQ(x,y)
 Pre: -Nclear(x),Nclear(y),-Ntable-OR-Non(x,y),Ntable-OR-Non(x,x)
 Eff: -Nclear(y),-Ntable-OR-Non(x,x),Ntable-OR-Non(x,y)

MoveFromTable(x,y,d):
 Static: NEQ(x,y),EQ(y,d)
 Pre: -Nclear(x),-Nclear(d),-Ntable-OR-Non(x,x),Ntable-OR-Non(x,y)
 Eff: Nclear(d),Ntable-OR-Non(x,x),-Ntable-OR-Non(x,y)

Move(x,z,y):
 Static: NEQ(x,z),NEQ(z,y),NEQ(x,y)
 Pre: -Nclear(x),Nclear(y),-Nclear(z),Ntable-OR-Non(x,x),
      Ntable-OR-Non(x,z),-Ntable-OR-Non(x,y)
 Eff: Nclear(z),-Nclear(y),Ntable-OR-Non(x,y),-Ntable-OR-Non(x,z)
```

Learning generalized planning models from action models

- **General policies** are for solving **multiple** planning instances at once
  - ▷ **General policy/strategy** for solving **any** instance of Blocks world
  - ▷ **General policy** for solving other domains or fragments
- **Subtlety:**
  - ▷ different # and configs of objects, diff (ground) actions, diff state spaces
- **Questions:**
  - ▷ How to **represent** general policies?
  - ▷ How to **derive** and **learn** them?
- Questions relevant to planning, learning, and program synthesis; addressed in recent work in **generalized planning**

Generalized planning: Formulation using QNPs

- QNPs stand for **qualitative numerical planning problems**
- QNPs are propositional STRIPS problems extended with **numerical variables** n that can be decreased $n{\downarrow}$ and increased $n{\uparrow}$
- QNPs are decidable and solvable with FOND planners, unlike numerical planning
- E.g., general policy for achieving $clear(x)$ in Blocks world:

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

where H and $n(x)$ for "holding a block" and "$\#$ blocks above x"

How to get these **features** and **policies** in general?

Learning the features and “abstract actions” using SAT Solver

- **Inputs:**
  - ▷ **CNF formula** $T(\mathcal{S}, \mathcal{F})$ encoding requirements over desired **features**
  - ▷ $\mathcal{S}$: **sampled state transitions**
  - ▷ $\mathcal{F}$: **pool of features** computed from primitive predicates and general grammar
- **Variables:**
  - ▷ $selected(f)$ for each $f \in \mathcal{F}$, true iff $f \in F$, $F \subseteq \mathcal{F}$
  - ▷ $D_1(s,t)$ true iff selected features distinguish s from t; p or $n = 0$ true in one
  - ▷ $D_2(s,s',t,t')$ true iff selected features f distinguish transitions (s,s'), (t,t')
- **Formulas:**
  - ▷ $D_1(s,t) \;\Leftrightarrow\; \bigvee_f selected(f)$
  - ▷ $D_2(s,s',t,t') \;\Leftrightarrow\; \bigvee_f selected(f)$
  - ▷ $\neg D_1(s,t) \;\Rightarrow\; \bigvee_{t'} \neg D_2(s,s',t,t')$
  - ▷ $D_1(s,t)$, when one of s and t is a goal state

Theorem (Bonet, Frances, G. 2019) $T(\mathcal{S}, \mathcal{F})$ is SAT iff $\exists$ set of features $F \subseteq \mathcal{F}$ and actions A over F such that A is **sound and complete** relative to $\mathcal{S}$.

Example: General Policy for Achieving $on(x, y)$

- **Data:** 3 STRIPS instances, 420 state transitions in $\mathcal{S}$, 657 features in $\mathcal{F}$
- **Features learned** X (x held), H (other held), $on(x, y)$; counters $n(x)$, $n(y)$
- **Abstract actions learned:** E abbreviates $\neg X \wedge \neg H$
  - ▷ Pick-x : $E, n(x) = 0 \;\mapsto\; X$,
  - ▷ Pick-above-x : $E, n(x) > 0 \;\mapsto\; H, n(x)\downarrow$,
  - ▷ Pick-above-y : $E, n(y) > 0 \;\mapsto\; H, n(y)\downarrow$,
  - ▷ Put-x-on-y : $X, n(y) = 0 \;\mapsto\; \neg X, on(x, y), n(y)\uparrow$,
  - ▷ Put-aside : $H \;\mapsto\; \neg H$.
- Policy that solves **all instances** found with off-the-shelf (FOND) planner
  - ▷ **If** $E, n(x) > 0, n(y) > 0$ **do** Pick-above-x,
  - ▷ **If** $H, \neg X, n(x) > 0, n(y) > 0$ **do** Put-aside,
  - ▷ **If** $H, \neg X, n(x) = 0, n(y) > 0$ **do** Put-aside,
  - ▷ **If** $E, n(x) = 0, n(y) > 0$ **do** Pick-above-y,
  - ▷ **If** $H, \neg X, n(x) = 0, n(y) = 0$ **do** Put-aside,
  - ▷ **If** $E, n(x) = 0, n(y) = 0$ **do** Pick-above-x,
  - ▷ **If** $X, \neg H, n(x) = 0, n(y) = 0$ **do** Put-x-on-y.

Wrap Up

- **True breakthroughs in DL and DRL**
- **DL** and **DRL**, however, deliver **System 1** boxes only
- **Main challenge** is tight, two-way integration of **learners** and **solvers**
- **Key problem** is learning representation of models used by solvers from data
  - ▷ Learning **from** what: symbolic, non-symbolic, or black-box states
  - ▷ Learning **for** what: model-free control, model-based, generalized models
- Looked at two points in this space
  - ▷ Learning **first-order symbolic planning representations** from state graphs
  - ▷ Learning **abstract models** and **general plans** from small examples
- Plenty to do at the intersection of **planning, representations, and learning**

AI and Social Impact

- **System 2** not only necessary for AI systems; essential for people and **societies**
- AI far from human-level intelligence, yet it can be used for **good** or **ill**
- **Ethical committees** and **AI principles** good but not sufficient
- **Markets and politics** play our **System 1**, focused on the **bottom line**
- If we want **good AI**, we need a **good and decent society**, that engages our **System 2** and cares about the common good

"Need AI for social good 'cause natural intelligence is busy in other pursuits" :-)