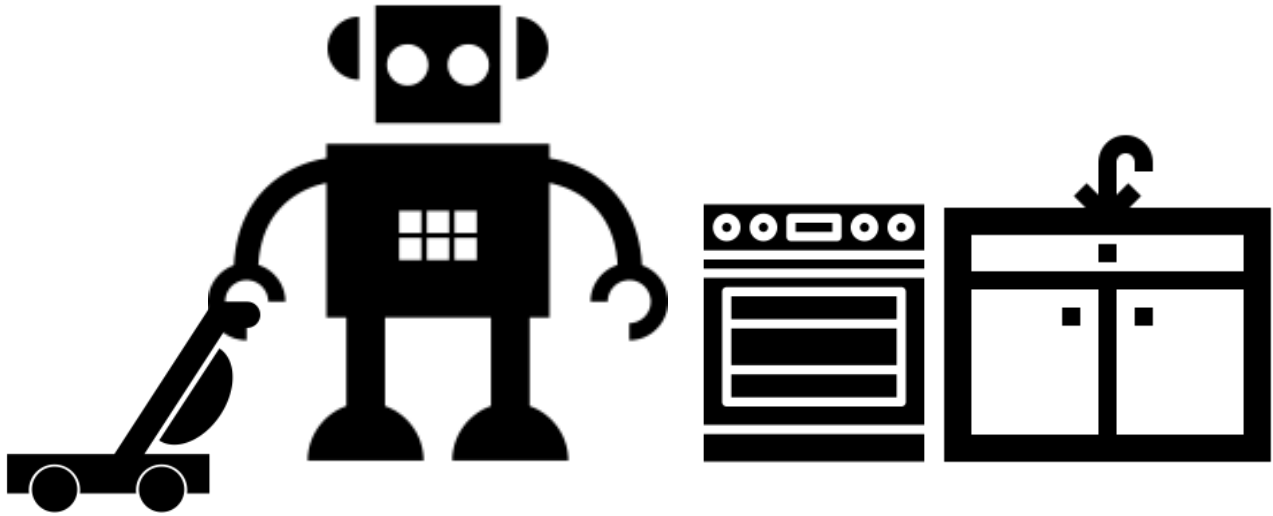
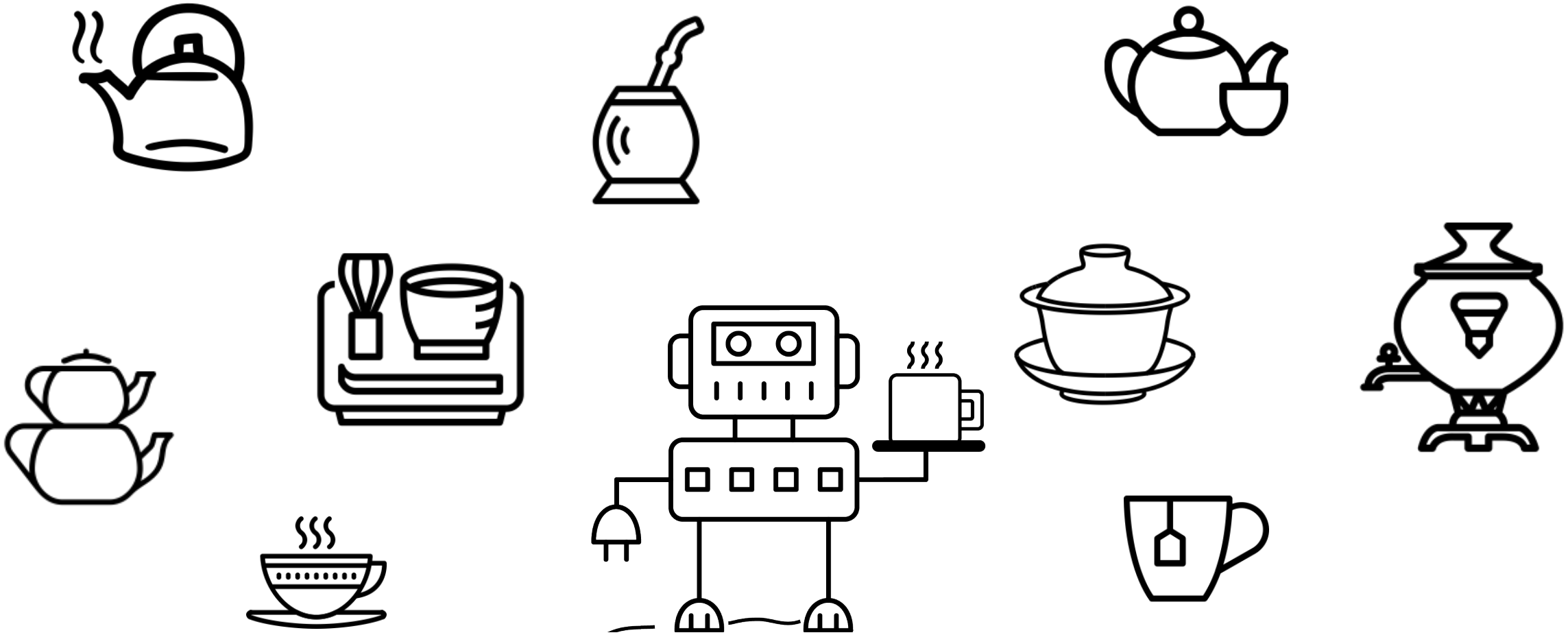
Doing for our robots what nature did for us

Leslie Pack Kaelbling
MIT CSAIL

Research goal:
understand the computational mechanisms necessary to make
a general-purpose intelligent robot

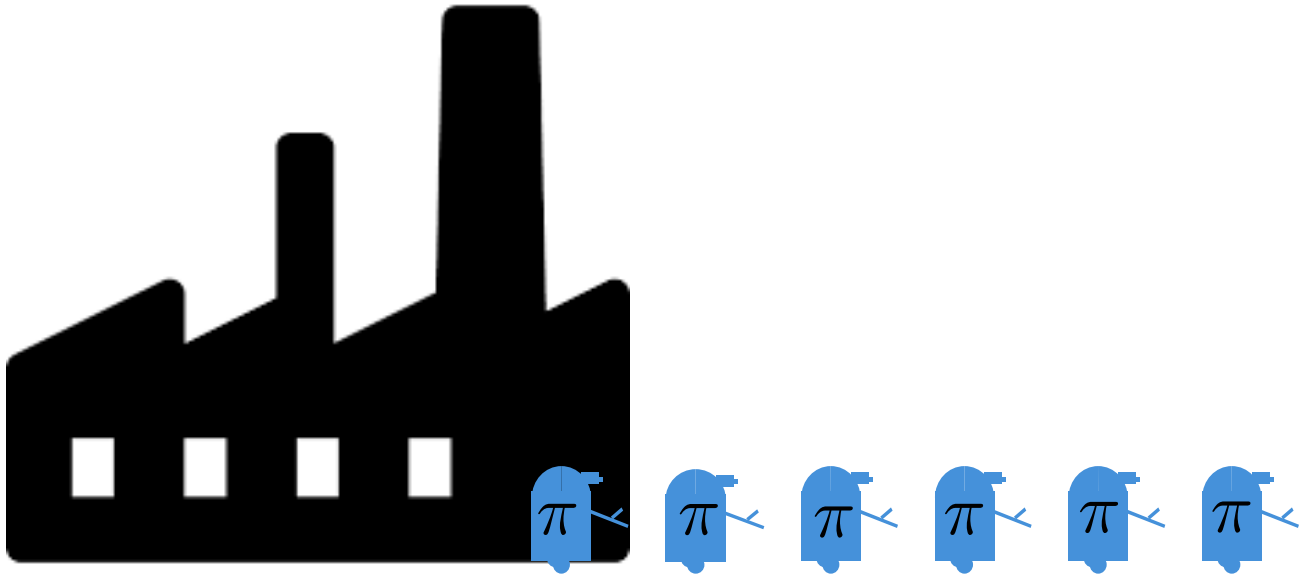


Proxy: Make tea in a kitchen you've never been in before

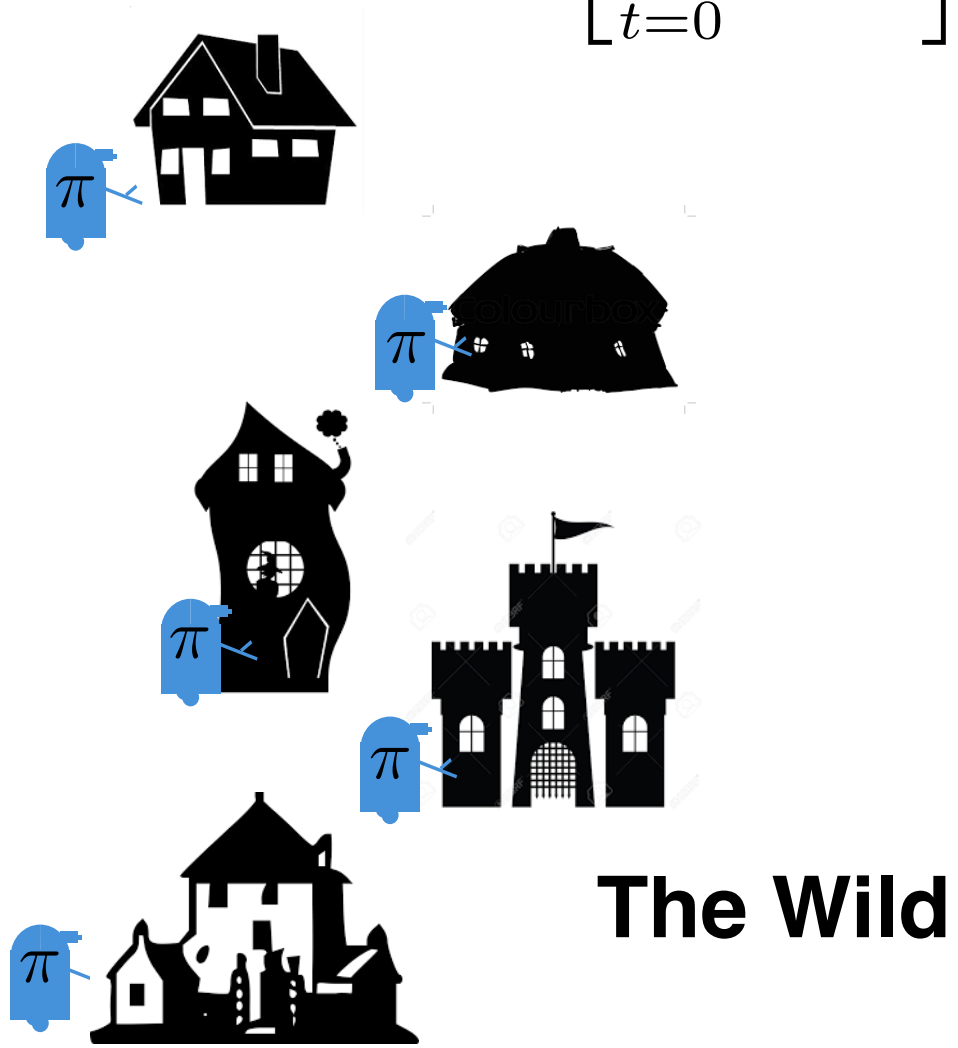


Doing for our robots what nature did for us

Find a mapping $\pi : (o, a)^* \rightarrow a$ that optimizes $E_{\text{env}}\left[\sum_{t=0}^{\infty} r_t \mid \pi\right]$

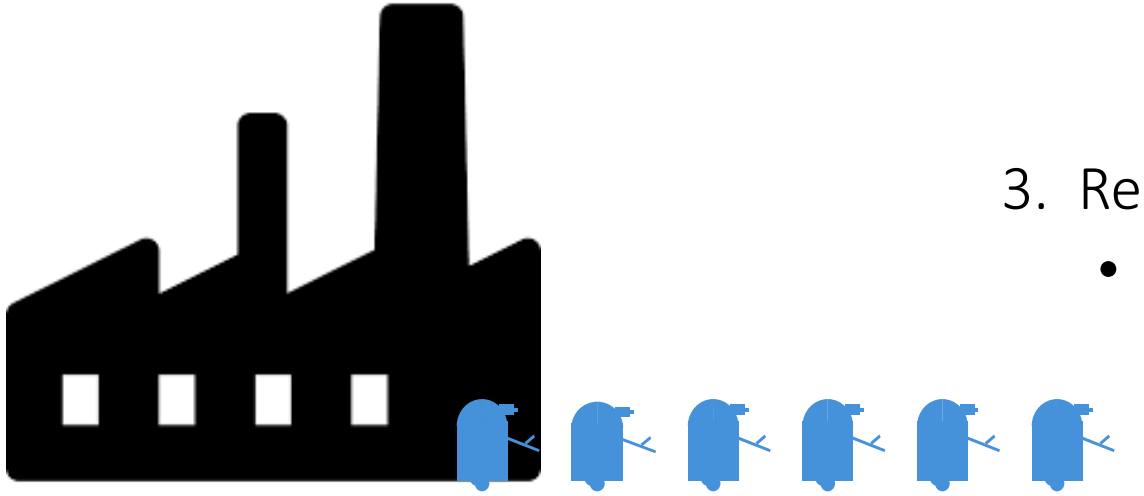


Robot Factory



The Wild

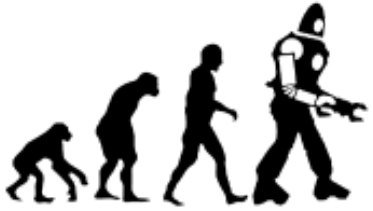
Designing the robot factory: a hard job!!



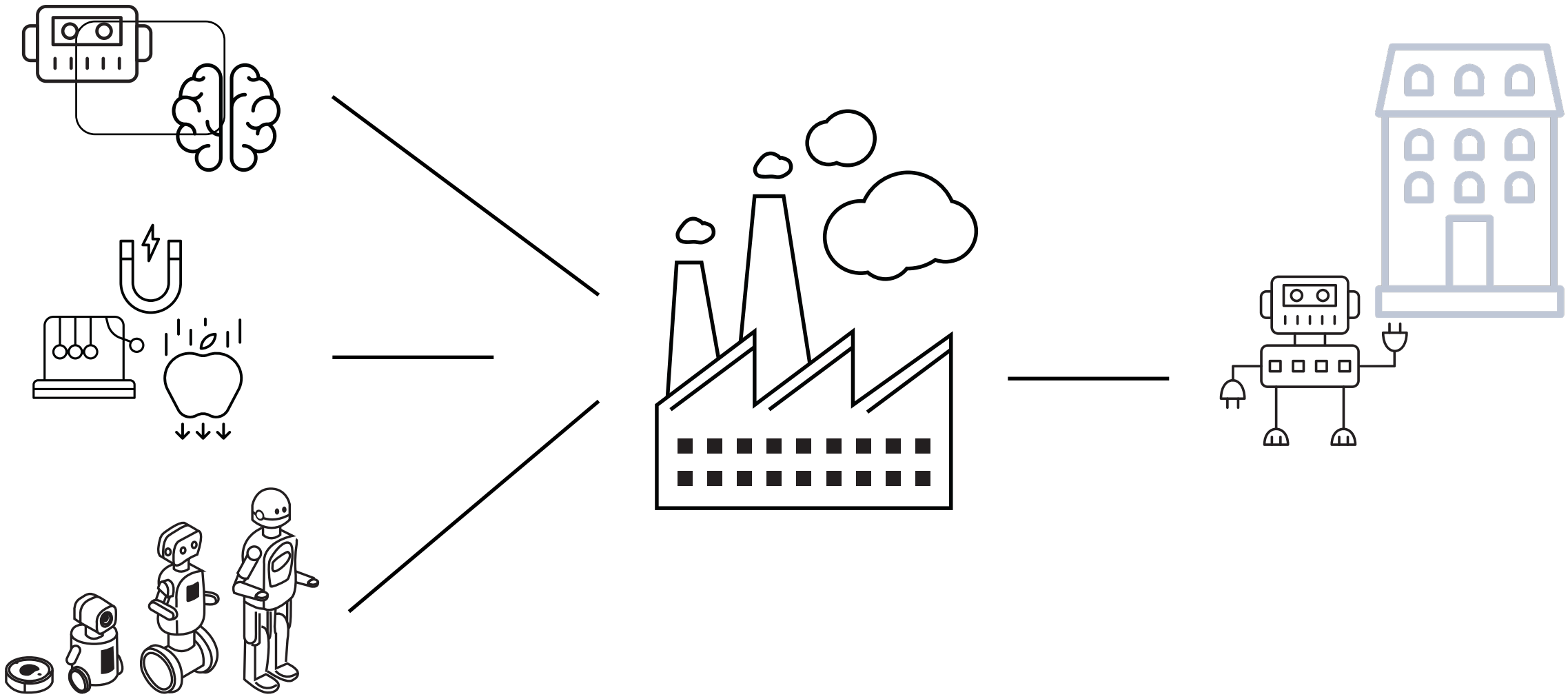
1. Reverse-engineer humans
   - hard biology!
   - limited applicability

2. Engineers write pi
   - hard engineering!

3. Recapitulate evolution
   - slow!

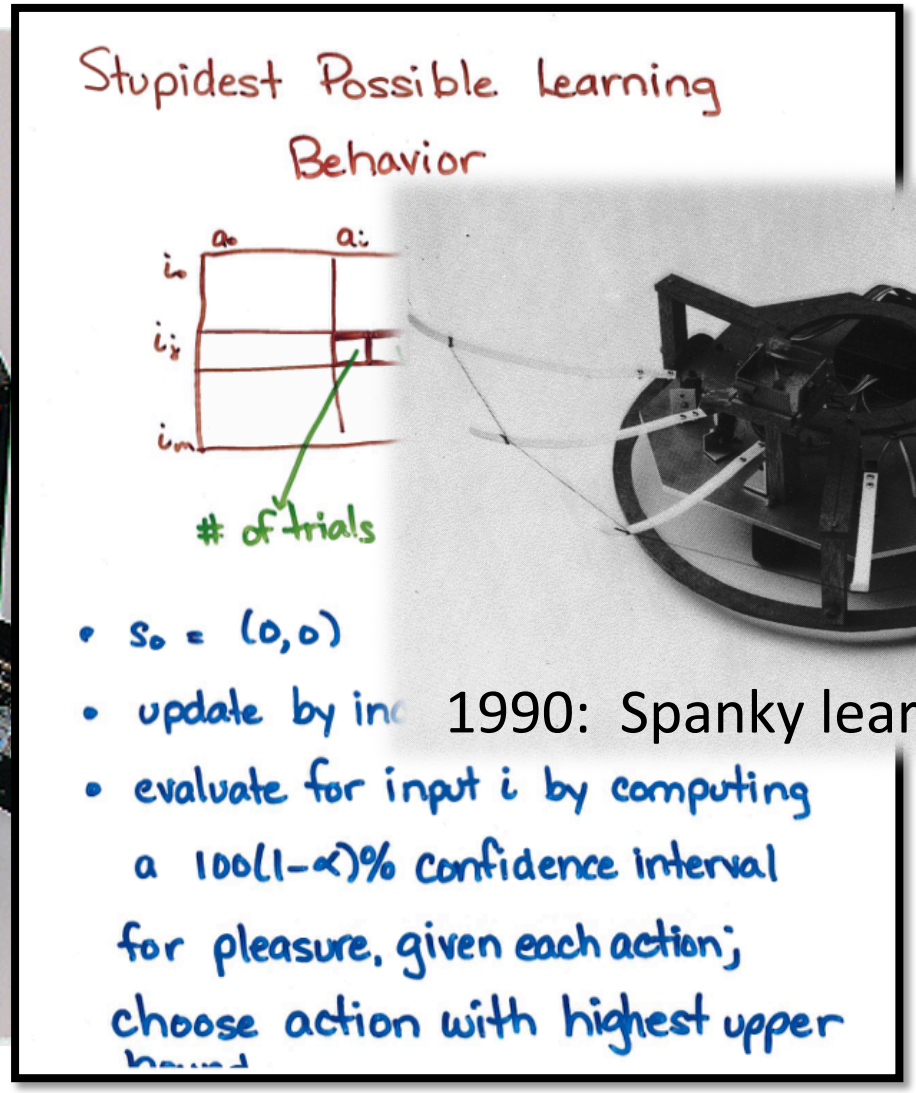


Use insights from neuroscience, physics, engineering to bias learning in the factory to learn in the wild



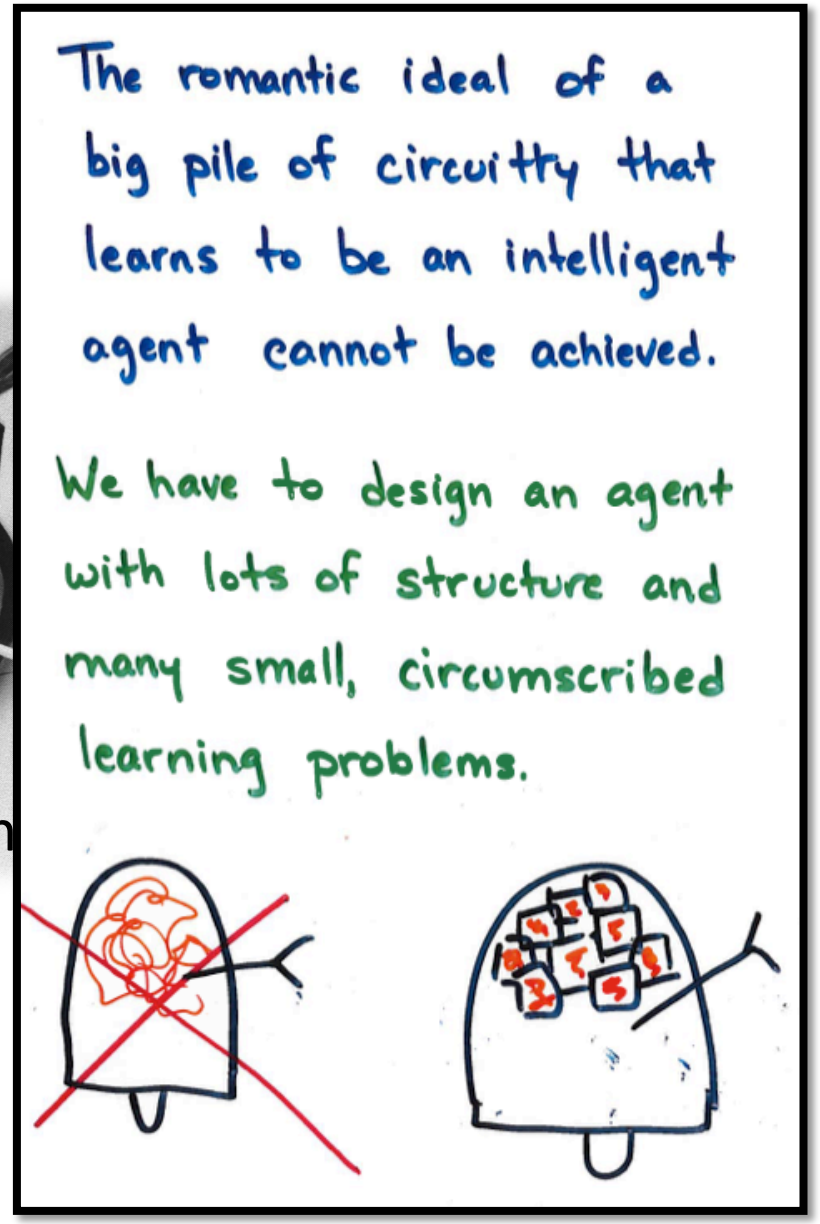
A quick bit of lpk history

1985: Flakey
lpk learns to navigate



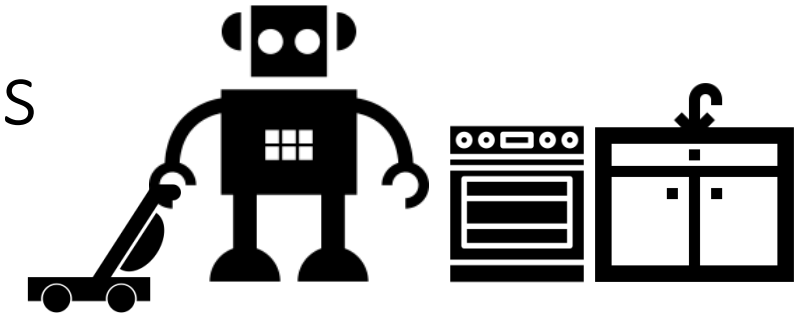
1988: reinventing the RL wheel

1990: Spanky learns to n



1995: disillusionment

Build in general algorithms, learn models



Domain-dependent

- transition models
- inference rules
- search control

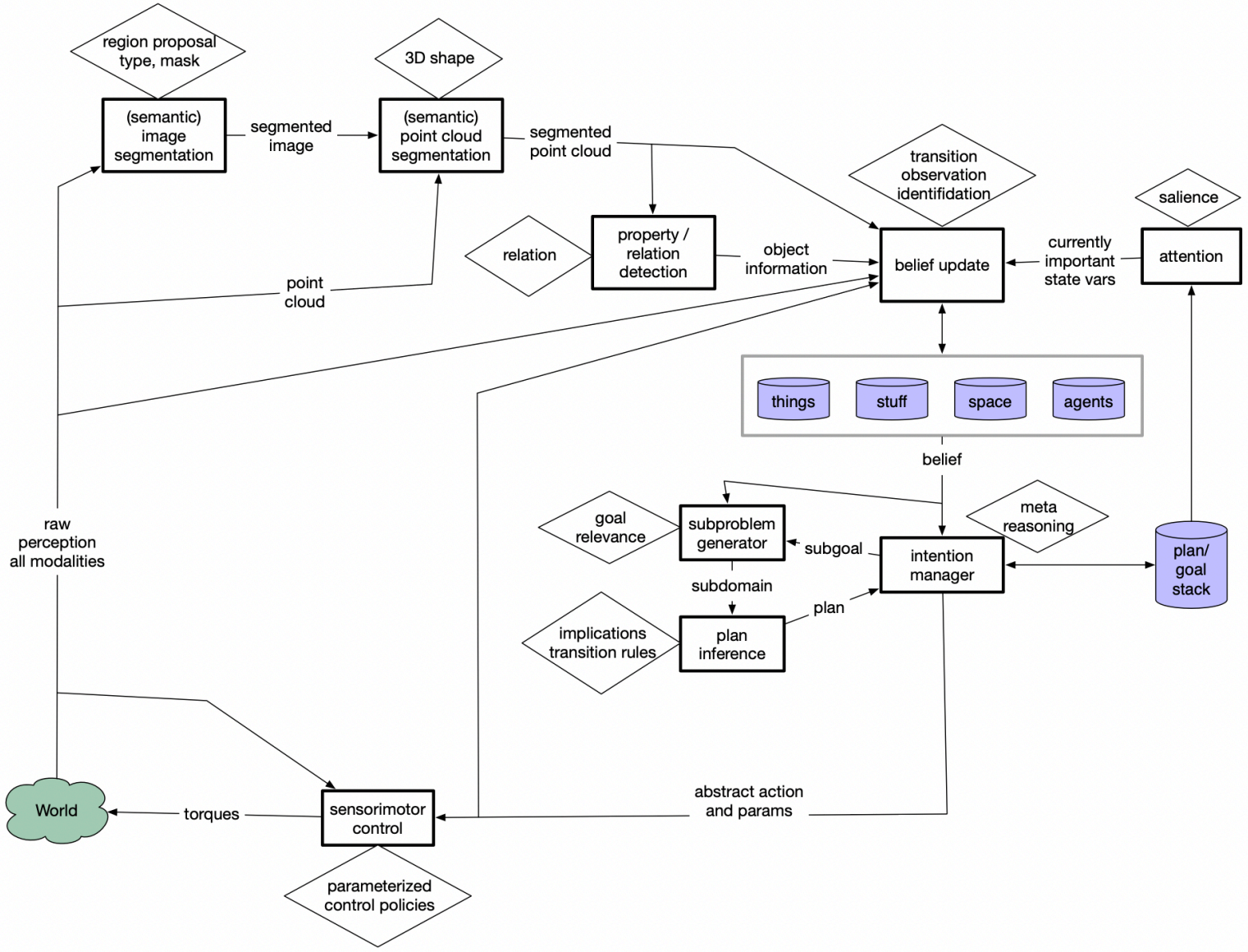
General representation and inference mechanisms:

- convolution in space and time
- kinematics
- path planning
- forward/backward causal inference
- abstraction over objects
- state abstraction/aggregation
- temporal abstraction
- state estimation, data association

BHPN: Belief-space Hierarchical Planning in the Now

Commitments

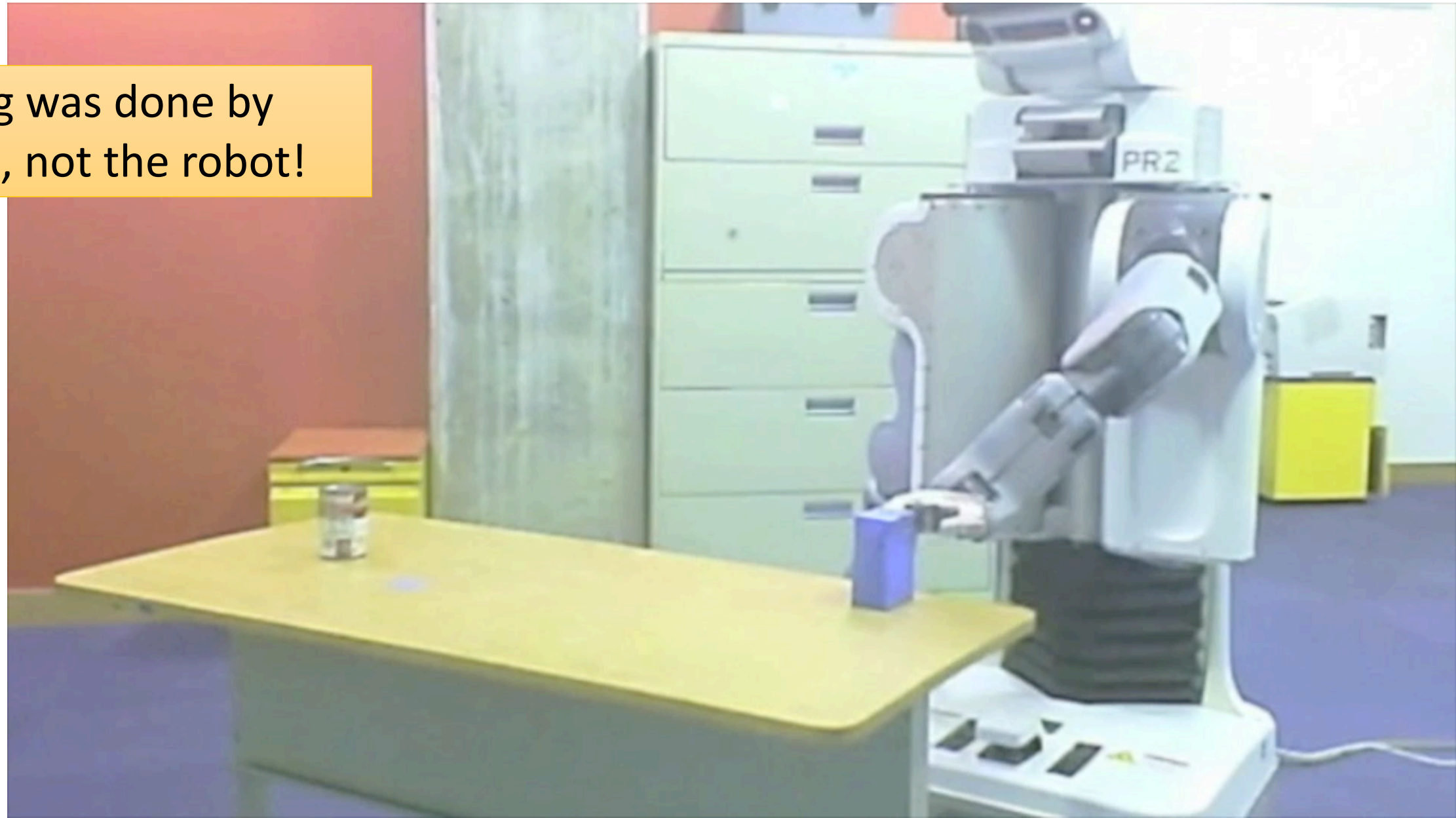
- objects
- properties
- 3D space
- compositional motor primitives
- model-predictive control
- temporal intention hierarchy
- explicit representation of uncertainty



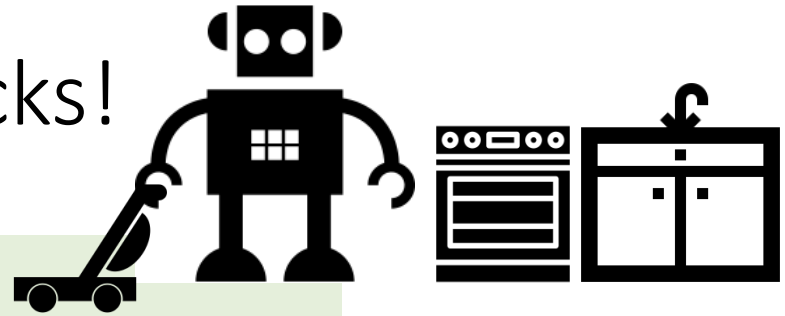
Kaelbling, Lozano-Perez: ICRA11, ISRR11, IJRR13,

BHPN: Same code (almost), different start, goal, domain, robot

All learning was done by
lpk and tlp, not the robot!



An old robot **can** learn new tricks!



Learned

- transition models
- inference rules
- search control

Build in general representation and inference mechanisms:

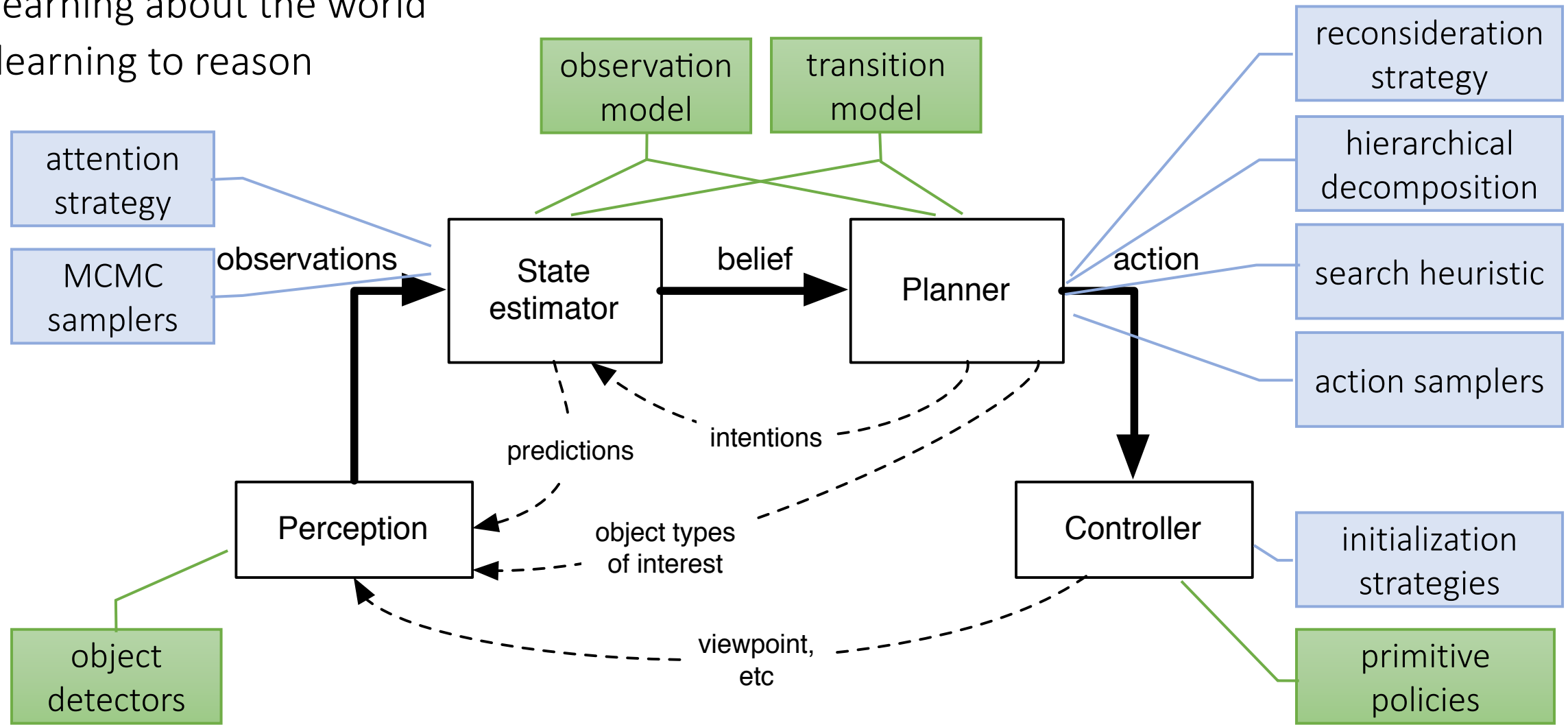
- convolution in space and time
- kinematics
- path planning
- forward/backward causal inference

- abstraction over objects
- state abstraction/aggregation
- temporal abstraction
- state estimation, data association

Varieties of learning

learning about the world

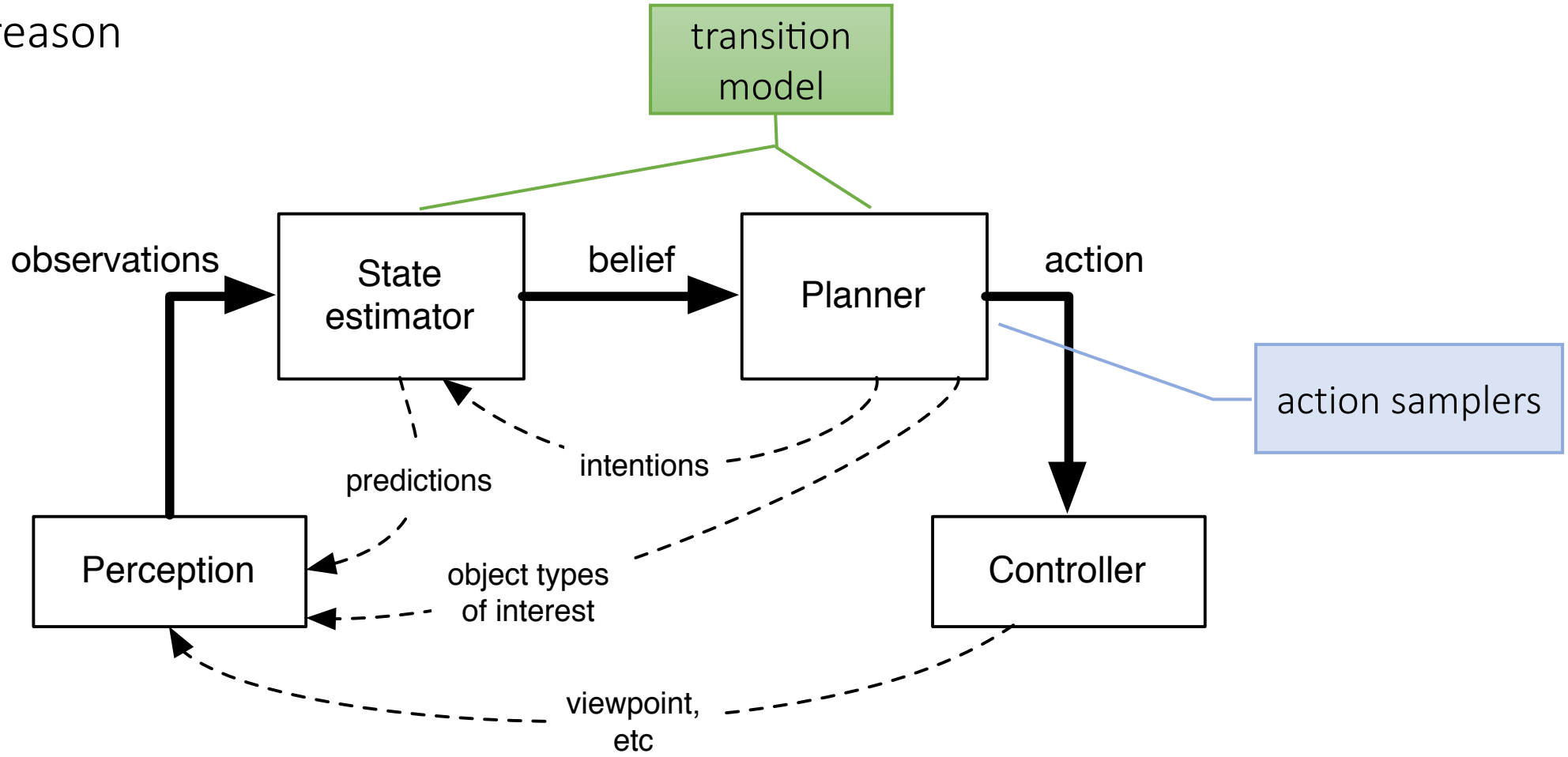
learning to reason



Learning transition model for planning

learning about the world

learning to reason



Dynamics modeling in large hybrid domains

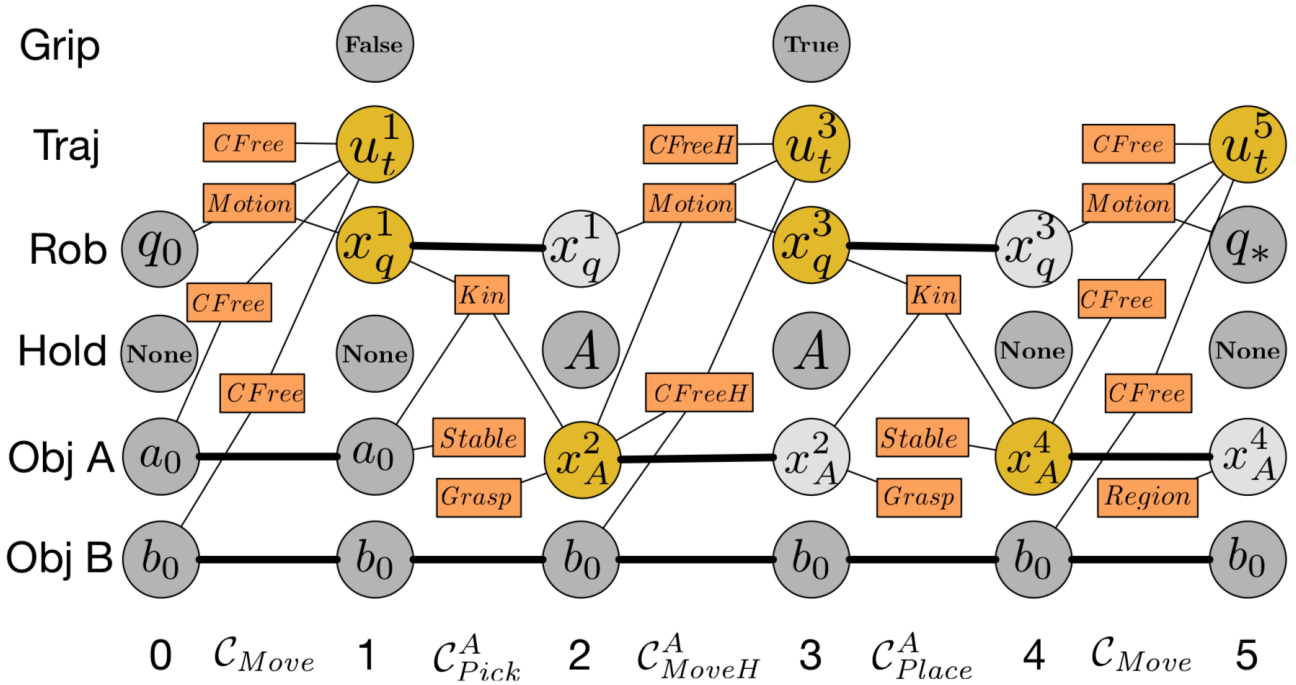
Need factored, sparse, constraint-based model of action effects

Pure forward search (MCTS, IW) unlikely to work

- infinite branching factor
- dimensionality-reducing (or tight) constraints

Modern TAMP (task and motion planning) strategies integrate structure and parameter search

- constrained optimization (Toussaint)
- pre-image back-chaining (Kaelbling and Lozano-Perez)
- sampling guided by task-level plans (Garrett)



How can a **competent** robot acquire a new ability?

- Learn new primitive skill
  - Examples: Cutting, pushing, stirring, pouring, throwing
  - Closed-loop low-level policy intended to achieve some objective, possibly parameterized

Most robot learning

- Add that skill to existing skill set to accomplish new goals!
  - For flexibility, use a general-purpose planner
  - Learn description of skill's preconditions and effects
  - Representation should generalize over objects, locations, etc.

Our focus

Factored, sparse dynamics model: when will **skill** achieve **result**?

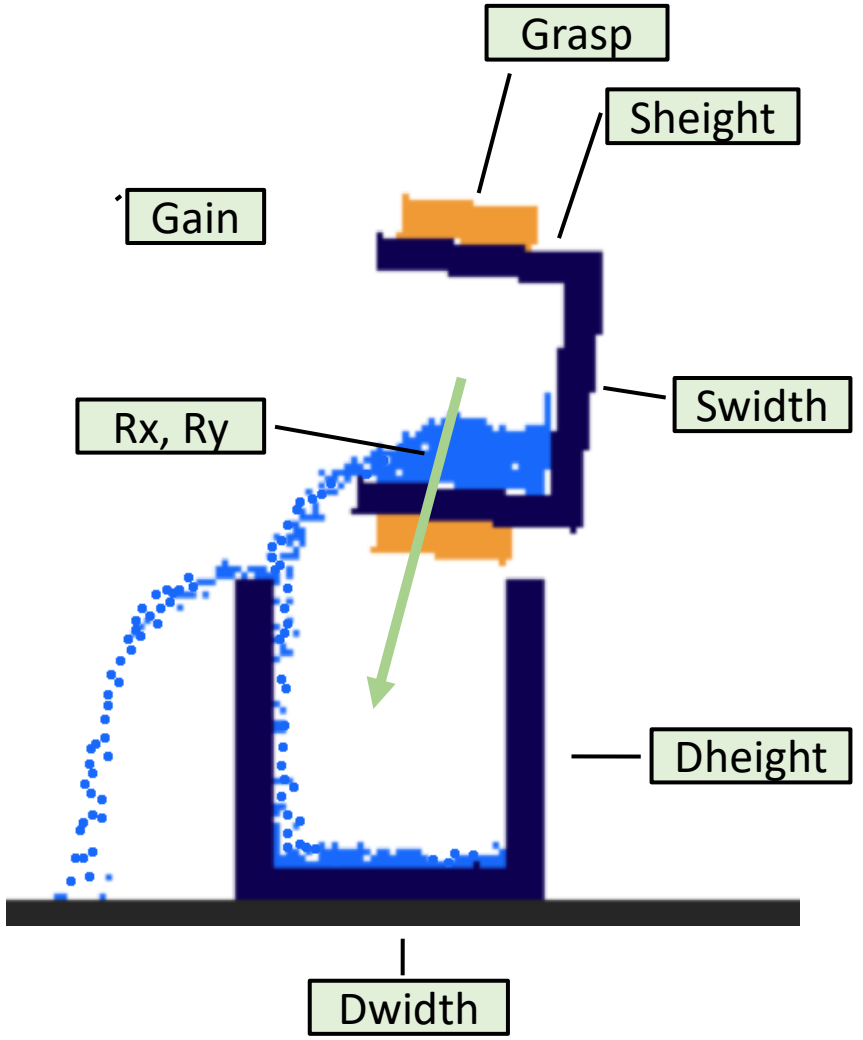
Learn efficiently to plan efficiently

Result: Contains(Dest, Liquid)

Skill: Pour(**Gain**)

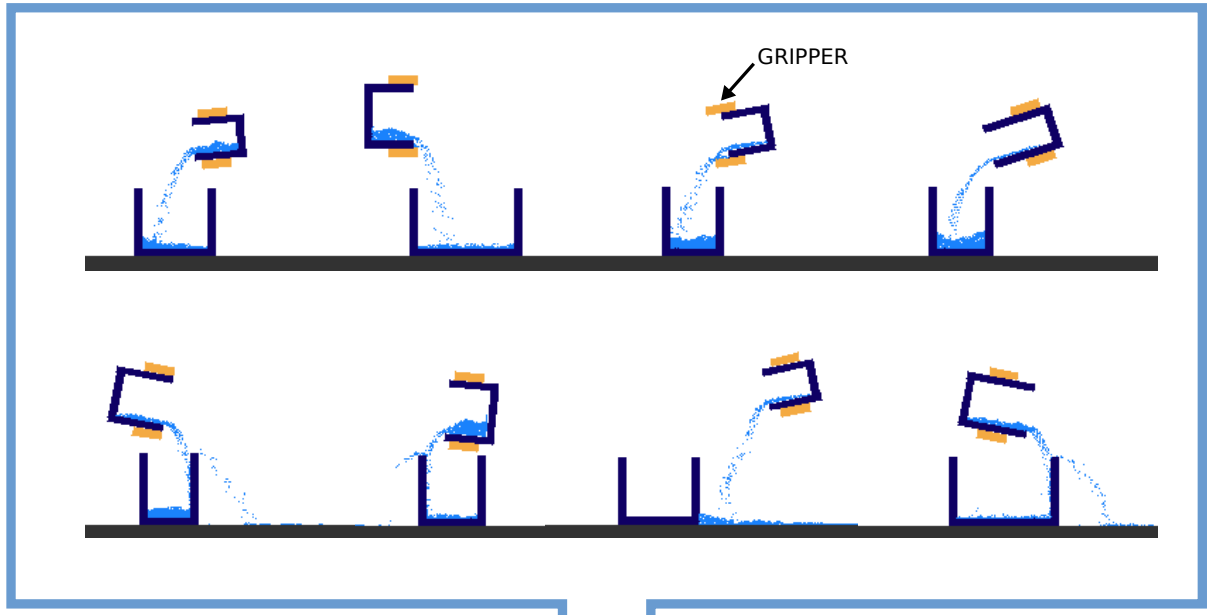
Preimage:

- Contains(Source, Liquid)
- Holding(Source, **Grasp**)
- Shape(Source) = (**Swidth, Sheight**)
- Shape(Dest) = (**Dwidth, Dheight**)
- RelPose(Source, Dest) = (**Rx, Ry**)
- **Constraint(Sw, Sh, Dw, Dh, Rx, Ry, Grasp, Gain)**



Learning the operator constraint: supervised training

labeled training data



$$\text{constraint}(\theta) \equiv g(\theta) > 0$$

Sw, Sh, Dw, Dh, Rx, Ry, Grasp, Gain

θ

Gaussian Process Regression

score

$g(\theta)$

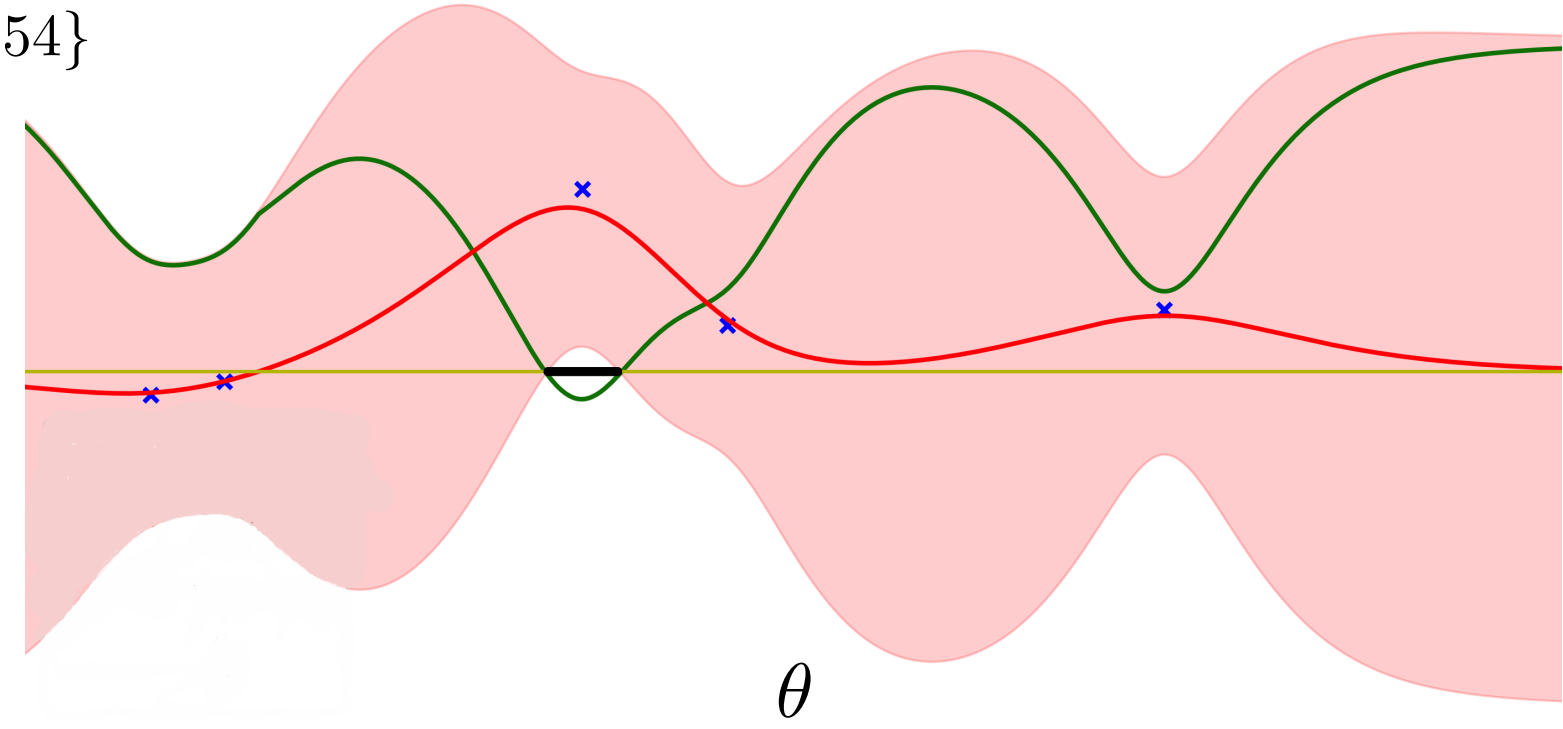
Gaussian process regression

Represent distribution over functions!

- ✕ : observations $(\theta_i, g(\theta_i))$
- — : mean $\mu(\theta)$
- ▭ : stdev $\mu(\theta) \pm 2\sigma(\theta)$
- — : high probability super level set

$$\{\theta \mid P(g(\theta) > 0) > 0.954\}$$

$$\text{constraint}(\theta) \equiv g(\theta) > 0$$



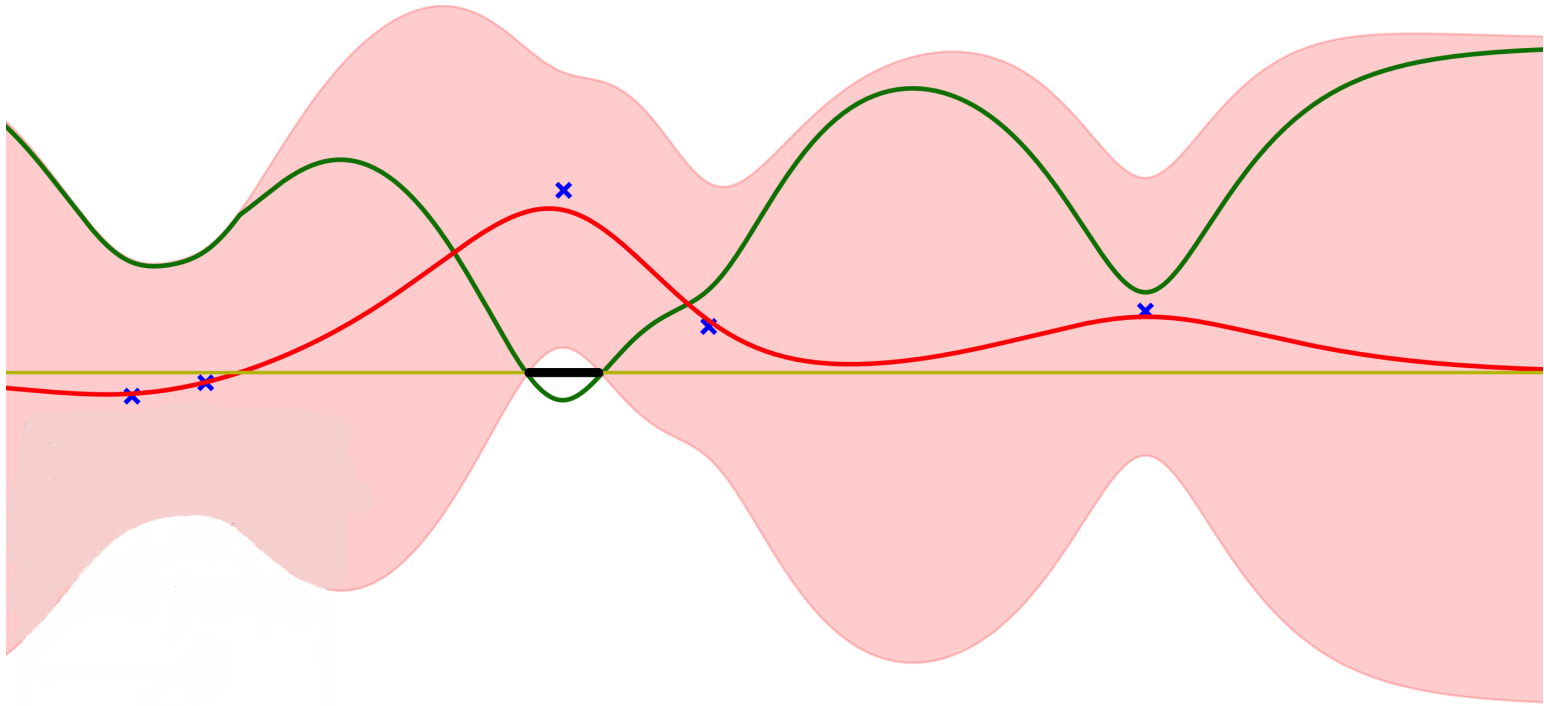
Active learning: robot experience is expensive!

Try pouring in situations that will give us the **most useful** information

- sample to maximize **acquisition function**

$$\phi(\theta) = 2\sigma(\theta) - |\mu(\theta)|$$

- high values when
  - mean is close to 0 (near a boundary)
  - stdev is high (uncertain about the value)
- —: $\phi(\theta)$



Straddle algorithm, Bryan et al, NIPS 2016

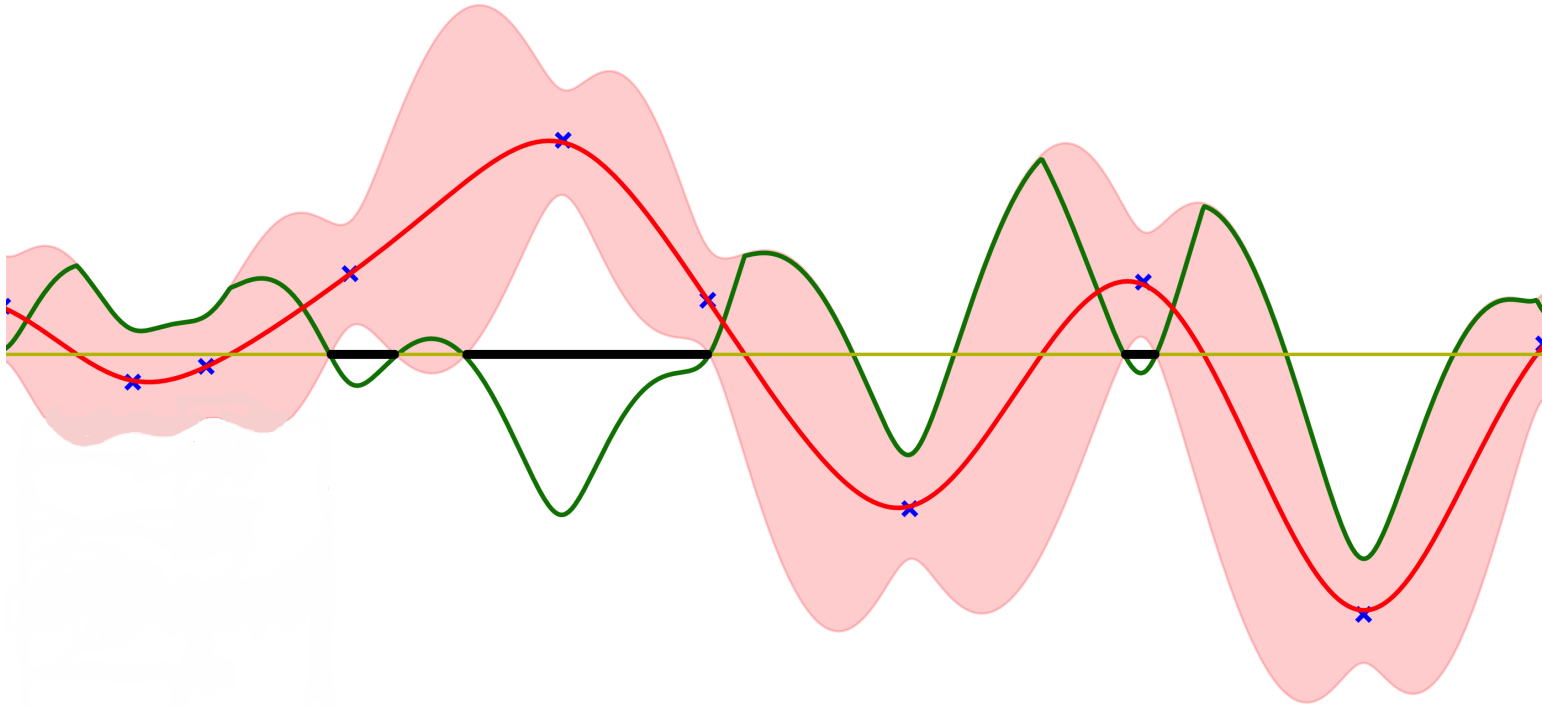
Active learning: robot experience is expensive!

Try pouring in situations that will give us the **most useful** information

- sample to maximize **acquisition function**

$$\phi(\theta) = 2\sigma(\theta) - |\mu(\theta)|$$

- high values when
  - mean is close to 0
    (near a boundary)
  - stdev is high
    (uncertain about the value)
- —: $\phi(\theta)$



Straddle algorithm, Bryan et al, NIPS 2016

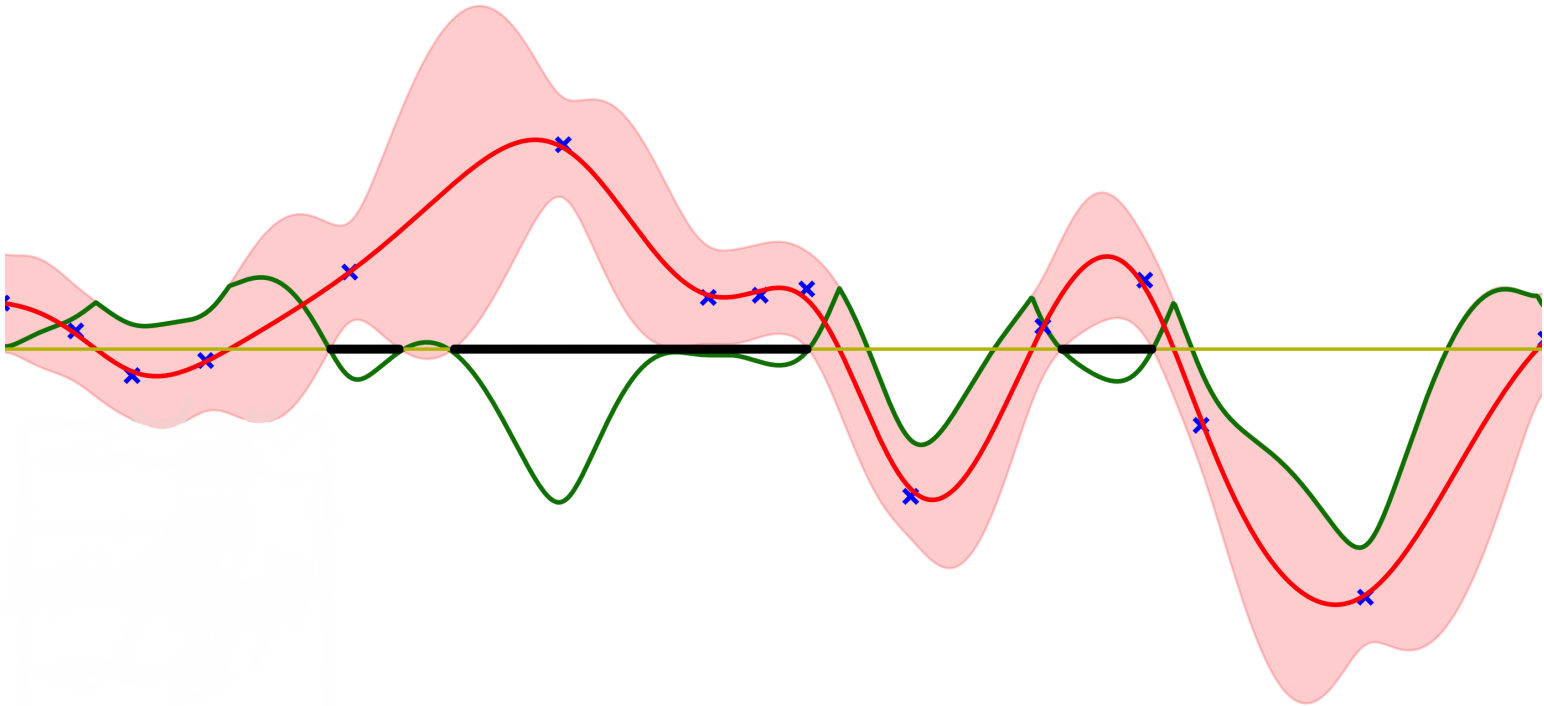
Active learning: robot experience is expensive!

Try pouring in situations that will give us the **most useful** information

- sample to maximize **acquisition function**

$$\phi(\theta) = 2\sigma(\theta) - |\mu(\theta)|$$

- high values when
  - mean is close to 0
    (near a boundary)
  - stdev is high
    (uncertain about the value)
- —: $\phi(\theta)$

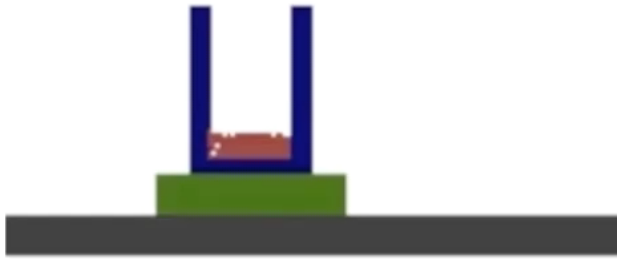


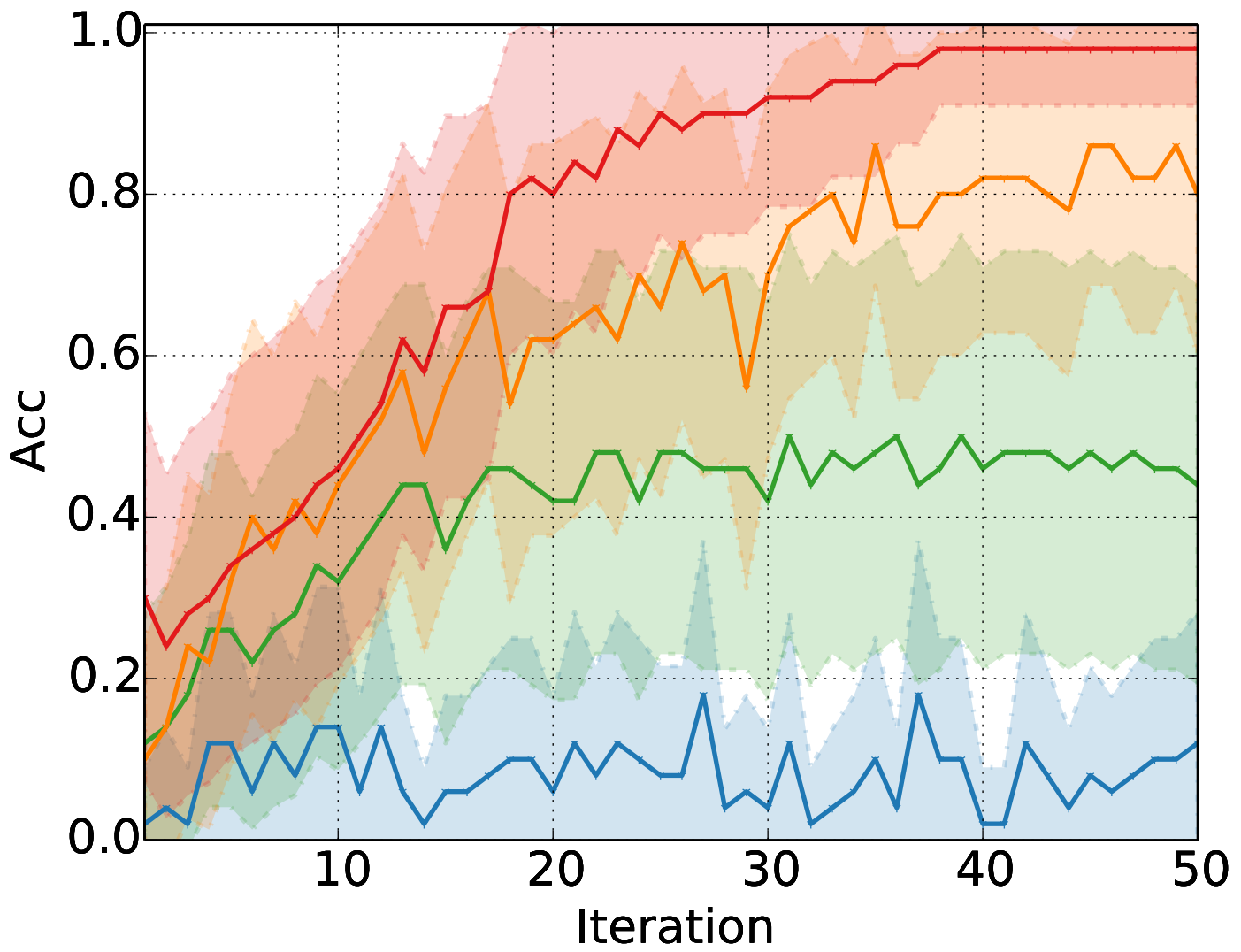
Straddle algorithm, Bryan et al, NIPS 2016

Active learning is data efficient!

Percent successful **pouring** actions as a function of the **number of training examples**

- : randomly chosen
- : neural network classifier
- : neural network regression
- : GP active learning





Sampling for planning: quality and diversity

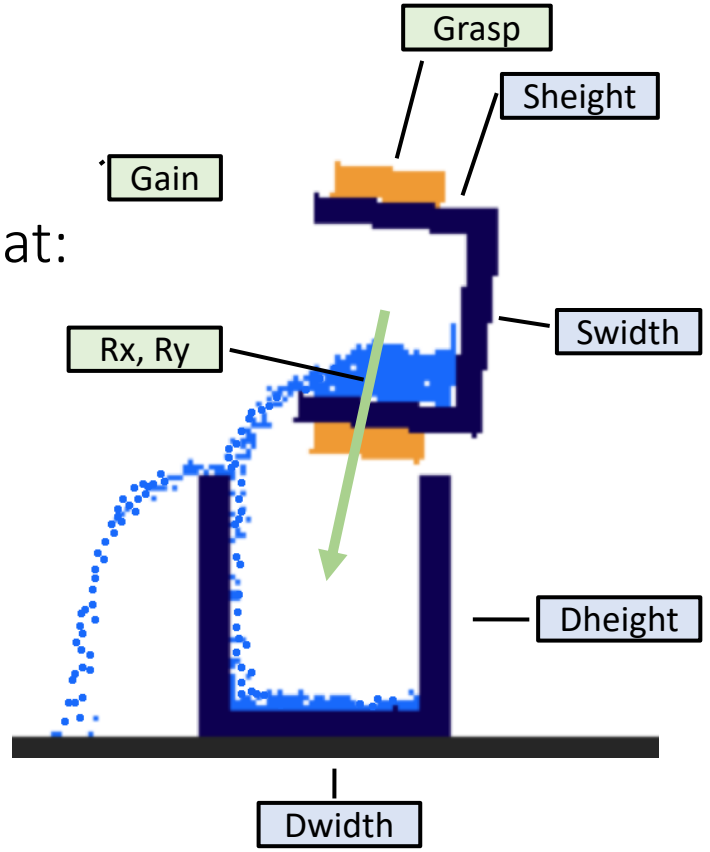
Given values for some parameters, sample values of the others so that:

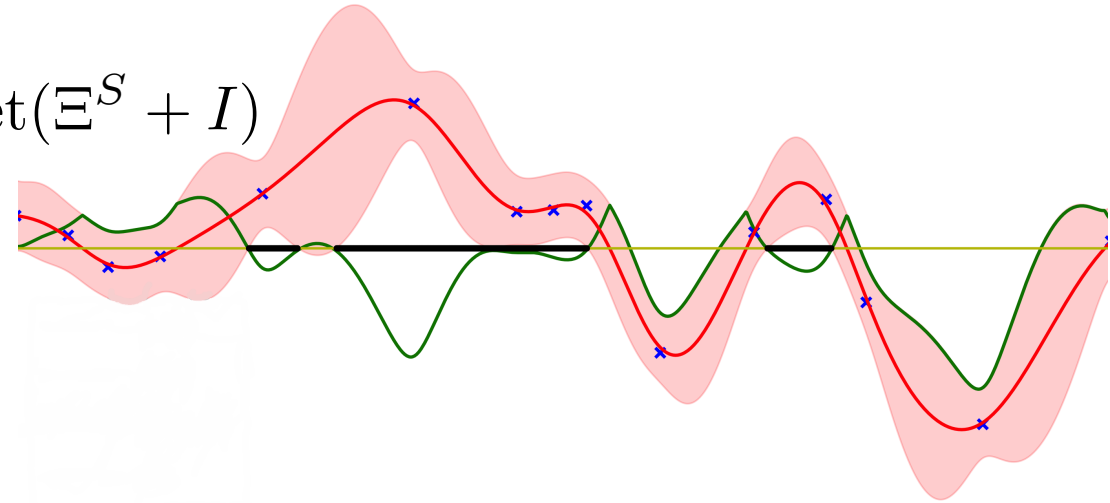
- action is likely to be successful
  - start with most likely to succeed: $\arg\max_\theta \frac{\mu(\theta)}{\sigma(\theta)}$
  - continue with rejection sampling in

$$\{\theta \mid P(g(\theta) > 0) > 0.954\}$$

  - make more efficient by careful weighted selection of samples near previously successful ones
- action differs from previous attempts

$$D(S) = \log\det(\Xi^S + I)$$





Plan to serve coffee, with cream and sugar stirred in!

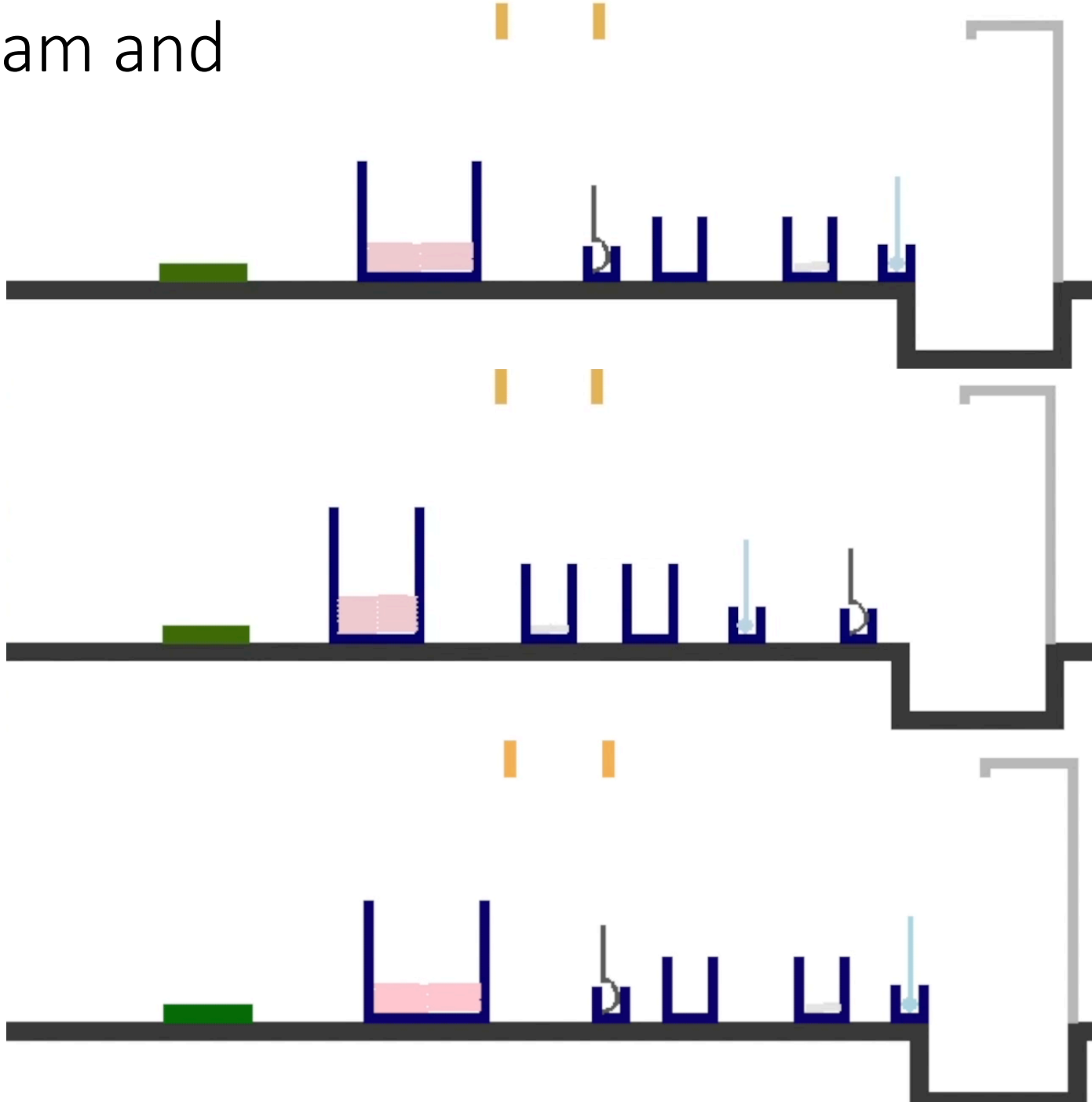
Note substantial variability in object layout and resulting plans

Pre-existing operators

- dispense coffee
- pick up, place object
- move robot
- stir
- dump spoon

Learned operators

- push
- pour
- scoop



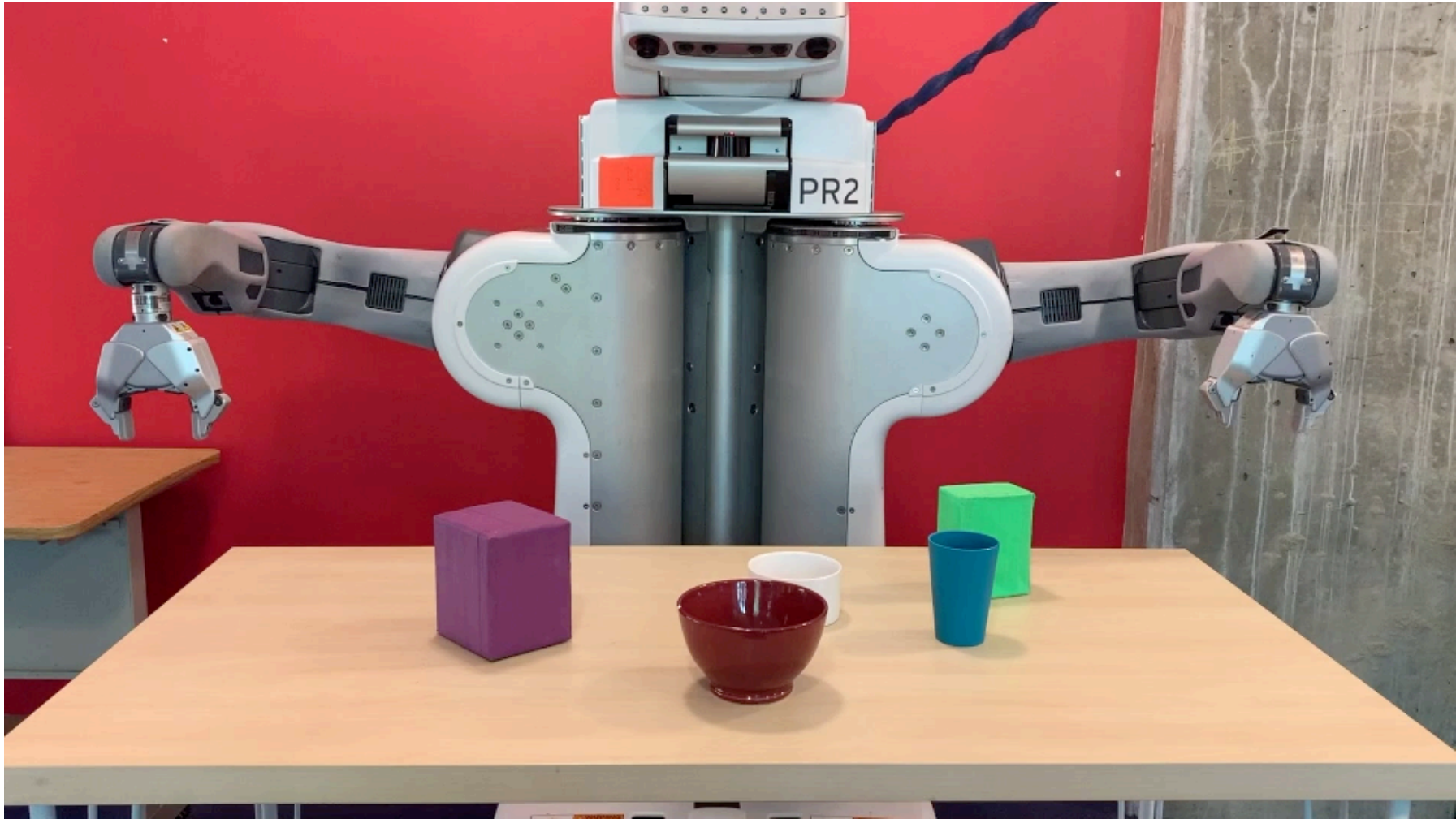
Preliminary results on real robot

Substantial variability in

- starting arrangement
- goal

Given pick/place operators

Learned pour and push



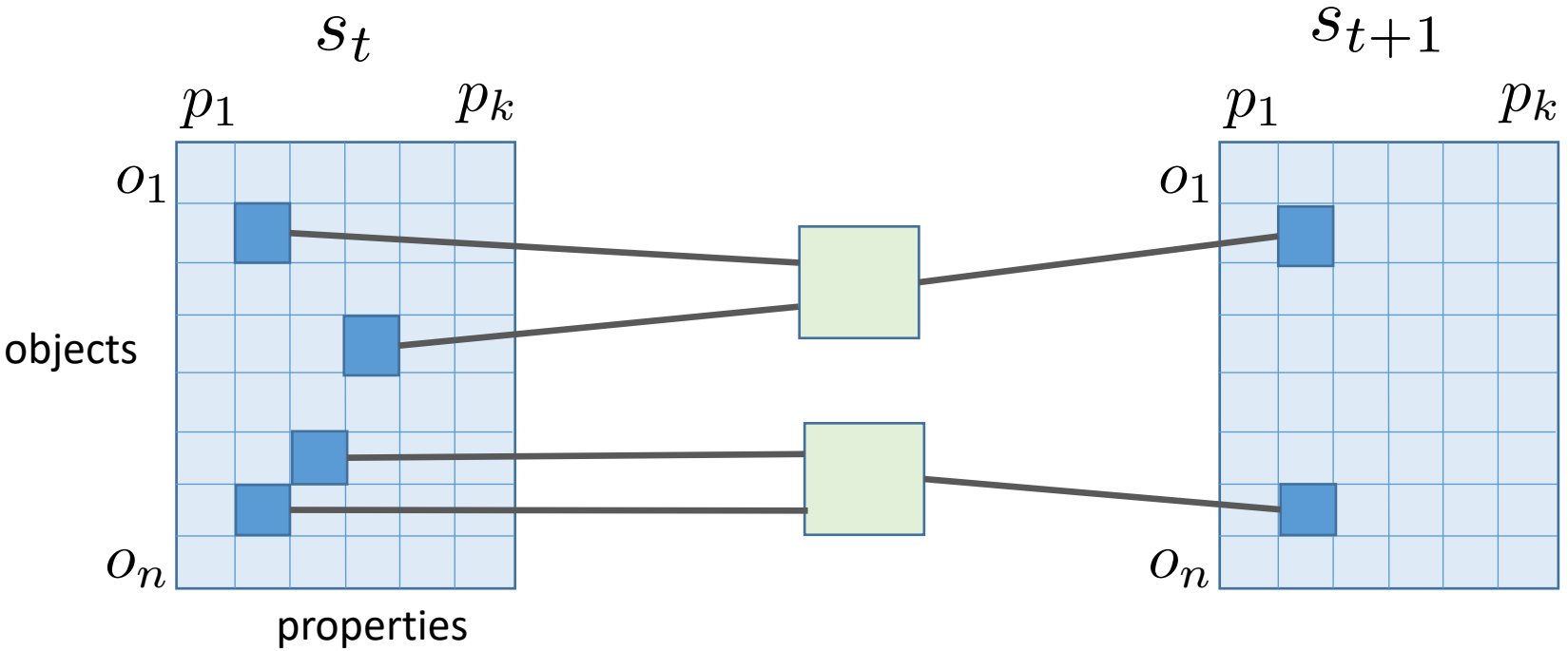
Learning a sparse, factored dynamics rule

1. Learn a sensori-motor policy that can change a property of an object

2. Learn detailed relation on properties of all these objects that predicts when the policy will have its intended effect
   - GP active learning, PDDLStream planning (Wang, Garrett, Lozano-Perez, K; IROS 18)

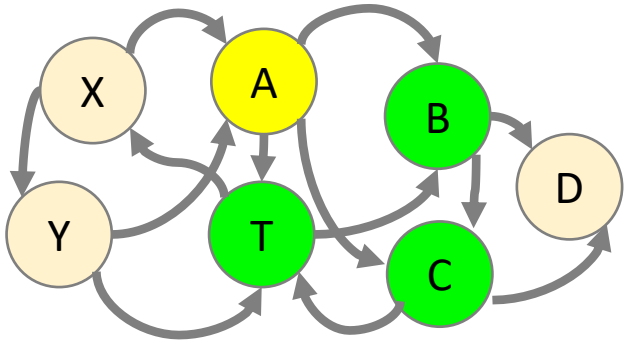
3. Determine which other objects may affect or be affected by this policy
   - Old approach (Pasula, Zettlemoyer, K; JAIR 07)
   - Preliminary new approach (Xia, Wang, K; ICLR 19)

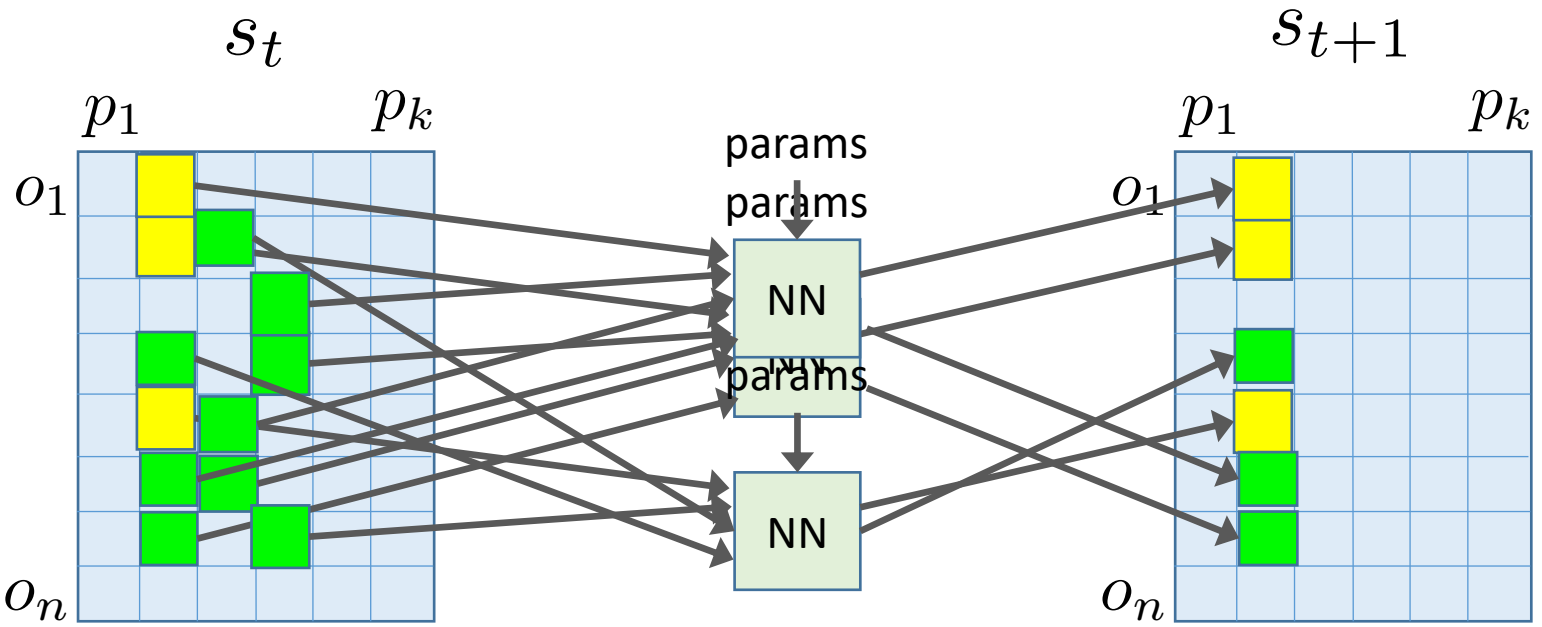
Sparse relational transition model: structure learning



Deictic rule

- motor primitive: Push(Obj, params)
- deictic references: select other objects: Obj2 = Above(Obj), ...
- neural network: predicts distribution on new property values for objects based on old property values
  - **fixed length input and output**
  - mechanism for handling sets





Deictic references name related objects

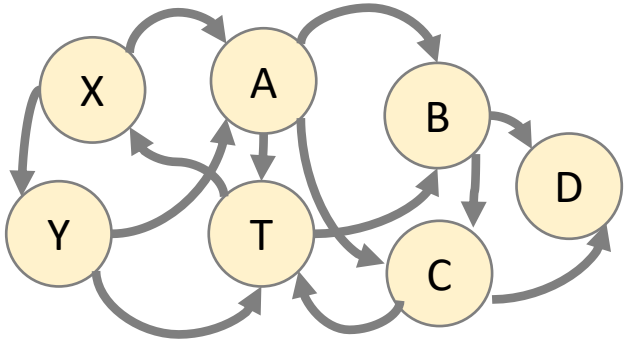
Include objects named in action: Push(Obj)

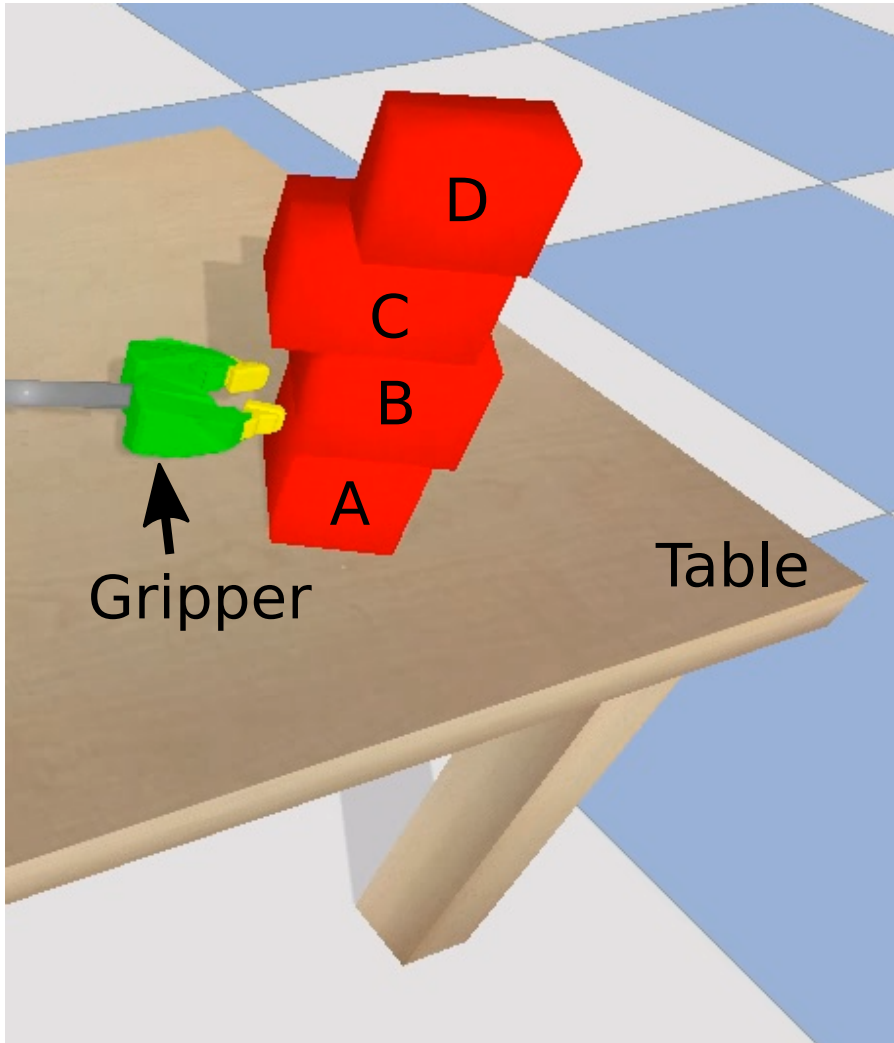
Refer to additional objects via **deictic references:**

- examples: **above, below, near, nearest**
- return an object or a set
- can be applied to an object that has already been "recruited"

Push(O1)

- O2 = Above(O1)
- O3 = Above(O2)
- O4 = Below(O1)





Deictic references name related objects

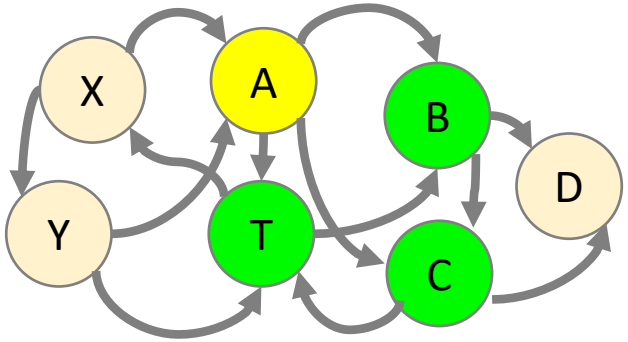
Include objects named in action: Push(Obj)

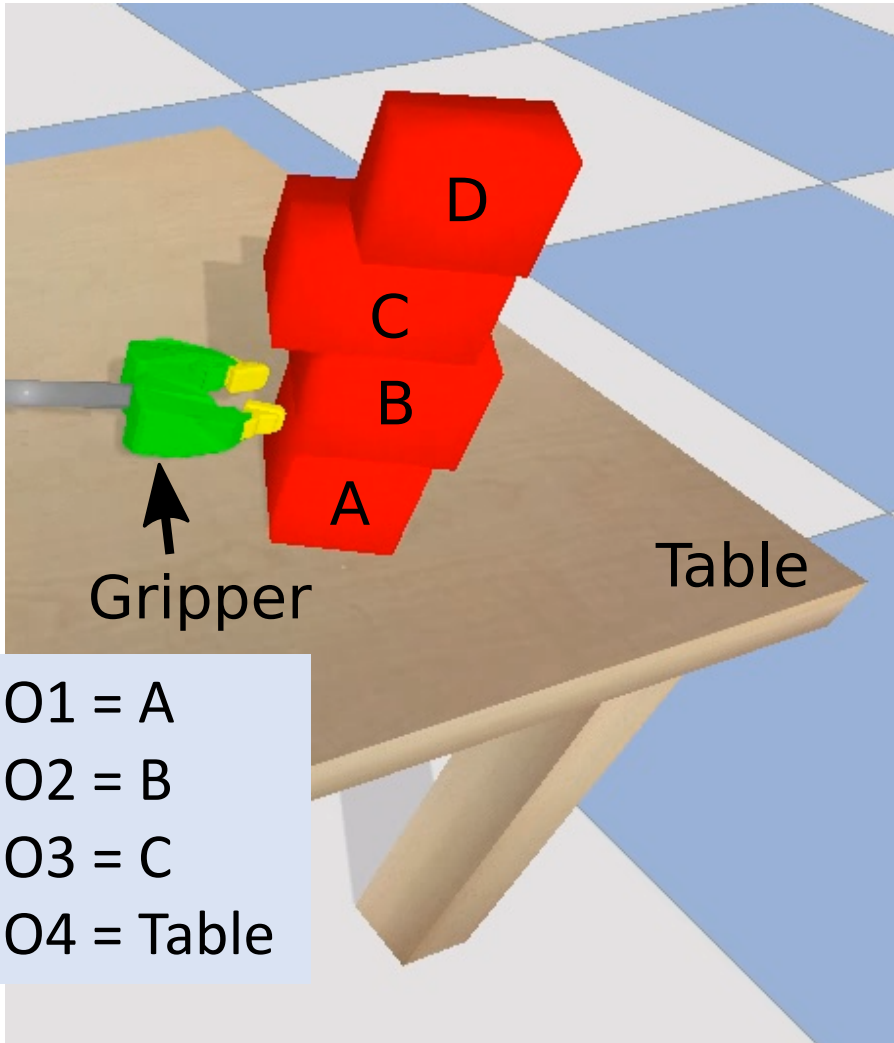
Refer to additional objects via **deictic references:**

- examples: **above, below, near, nearest**
- return an object or a set
- can be applied to an object that has already been "recruited"

Push(O1)

- O2 = Above(O1)
- O3 = Above(O2)
- O4 = Below(O1)





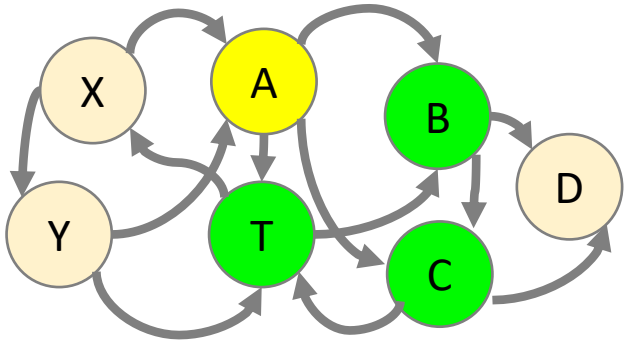
Rule learning: training data is (s, a, s')

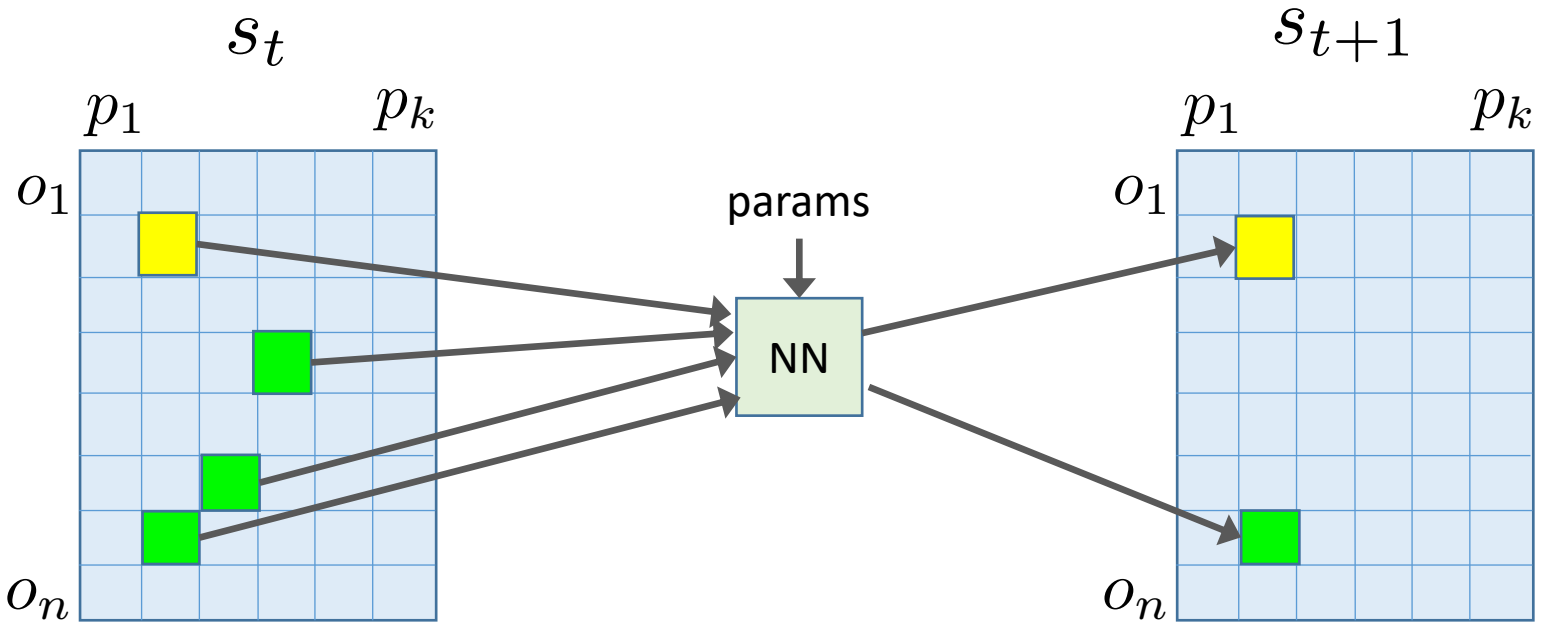
Outer loop over set of rules

- Greedily add best next deictic reference to generate new rule

Inner loop

- EM-method for deciding which rule(s) account for which training examples
- Predict mean and variance for each property
- Gradient descent on NN that predicts next values, minimize conditional log likelihood

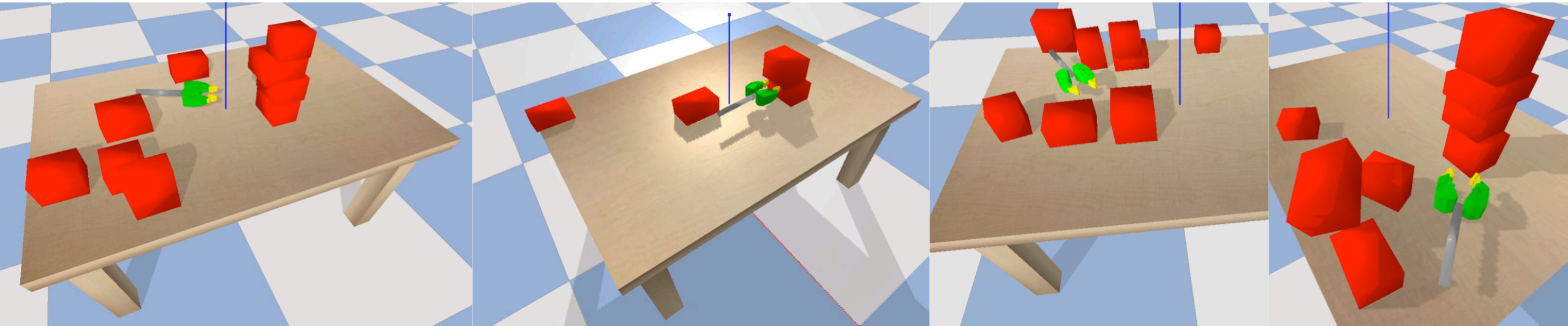




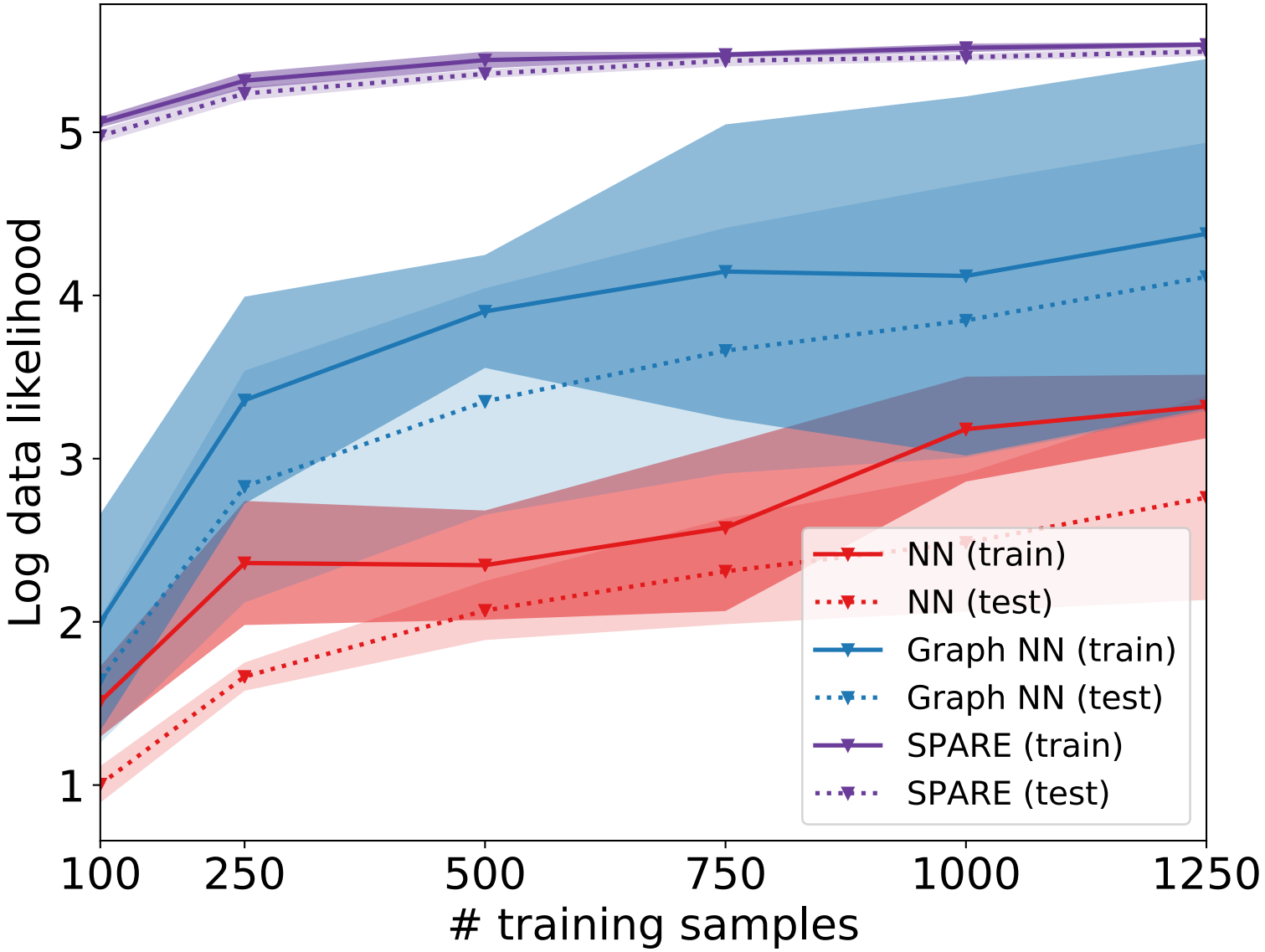
Preliminary results: pushing objects on crowded table

Compare likelihood on held-out data

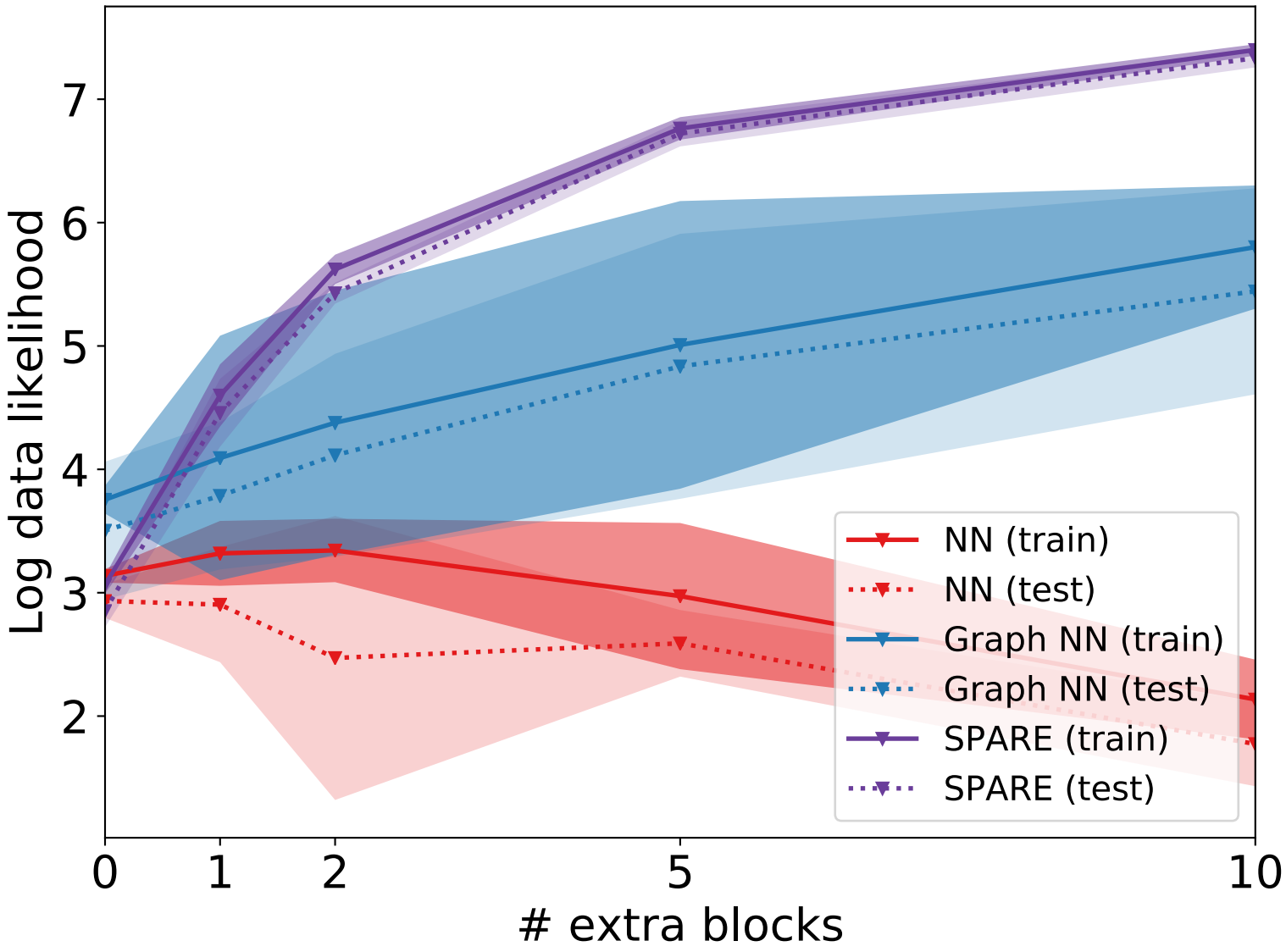
- Learned rule-based model
- Neural network trained on vector of attributes of all objects
  - Pushed object always first
  - Other objects sorted by distance from first
- Graph neural network, fully pairwise connected



Sparse rules learn with less data (3 objects in all scenes)



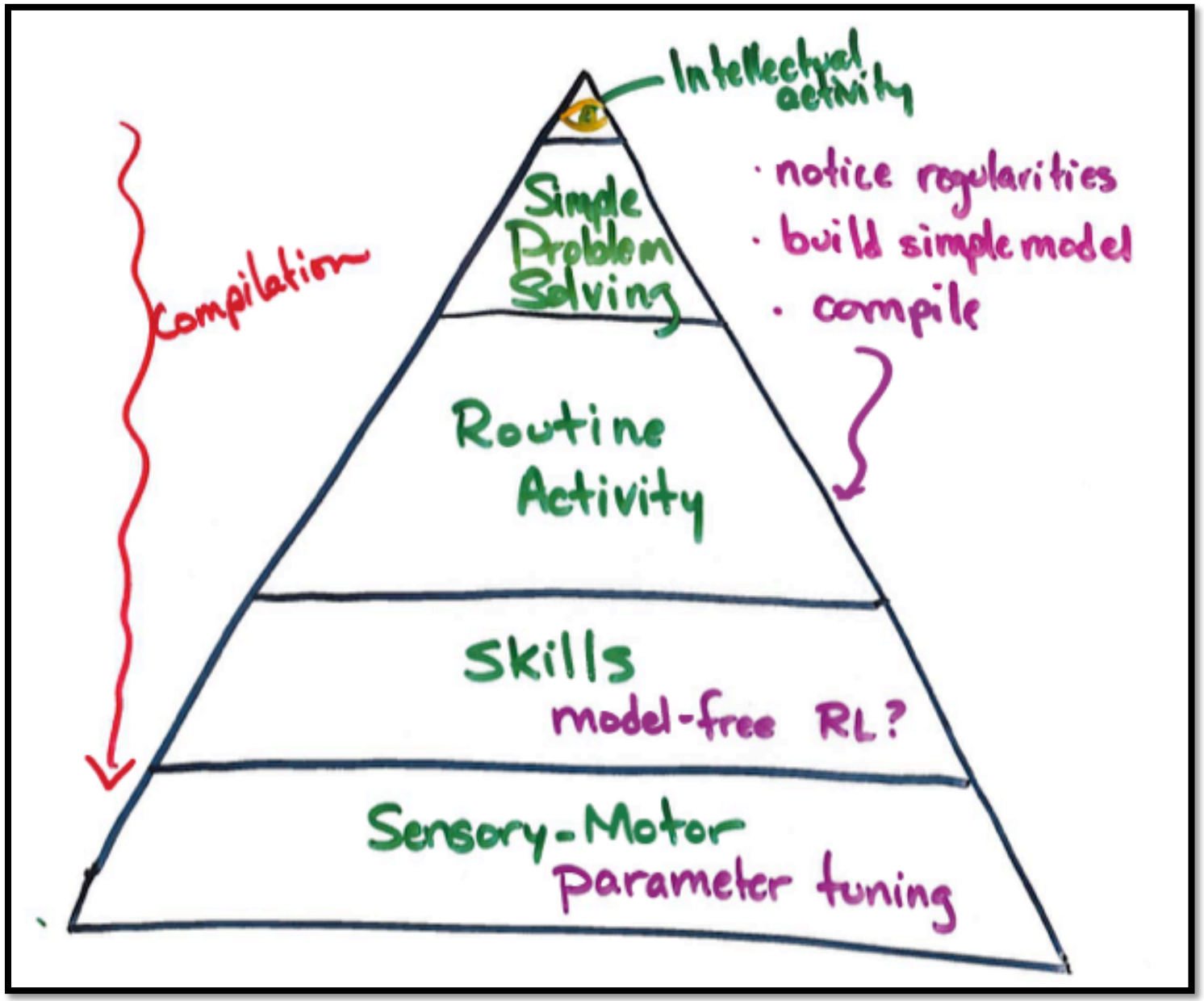
Sparse rules unaffected by clutter



So much more to do!

- Learn perceptual models for state estimation
- Find integrated way of combining learned policies and online reasoning
- Find integrated way of combining end-to-end with local learning signals
- Integrate human interaction systematically as part of the environment
-
-
-
-

To move upward in the pyramid of intelligent robotics



1995: LPK post RL

Thanks to lots of people!

Tomas Lozano-Perez

Caelan Garrett, Zi Wang, Victoria Xia

Summer robot hackers:

- Alex LaGrassa, Skye Thompson, Nishad Gothoskar, Jingxi Xu, Kevin Chen

Kelsey Allen, Tom Silver

Thanks. And out-takes to watch during questions