Learning models of very large hybrid domains **that are good for planning**

Leslie Pack Kaelbling

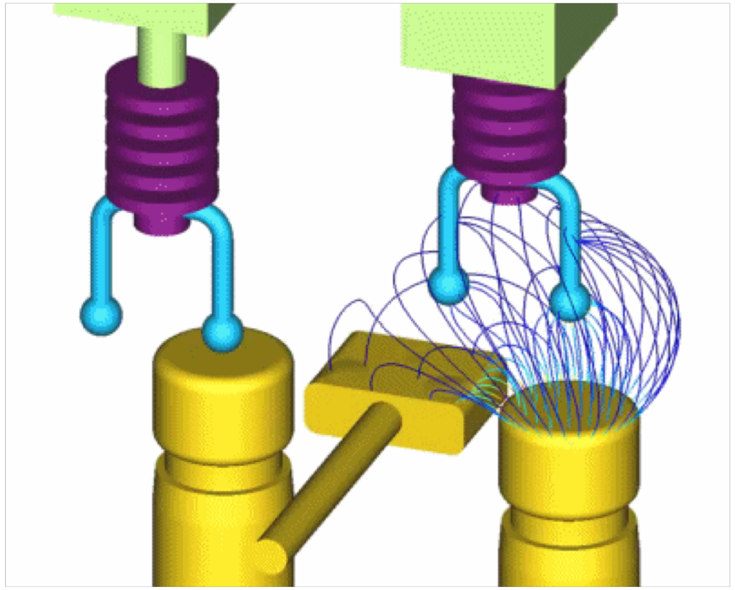
Zi Wang, Victoria Xia

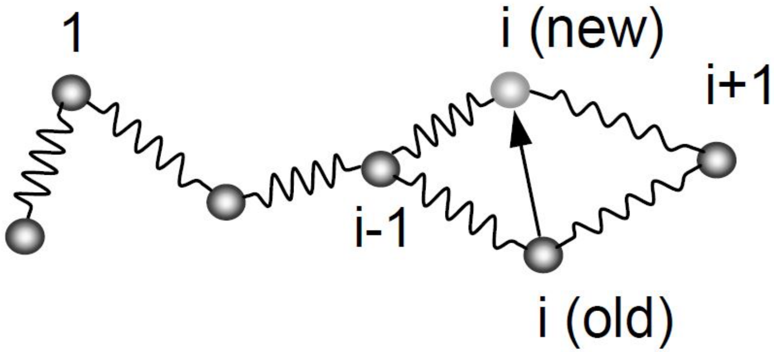
George Konidaris, Tomas Lozano-Perez
Hanna Pasula, Luke Zettlemoyer

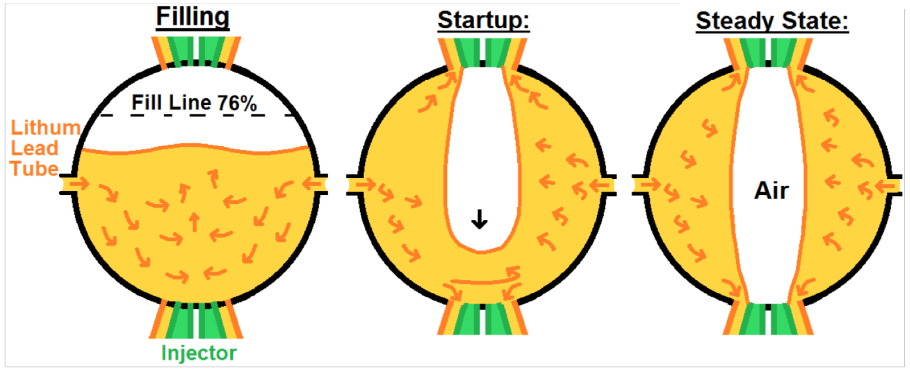
Many models of the physical world

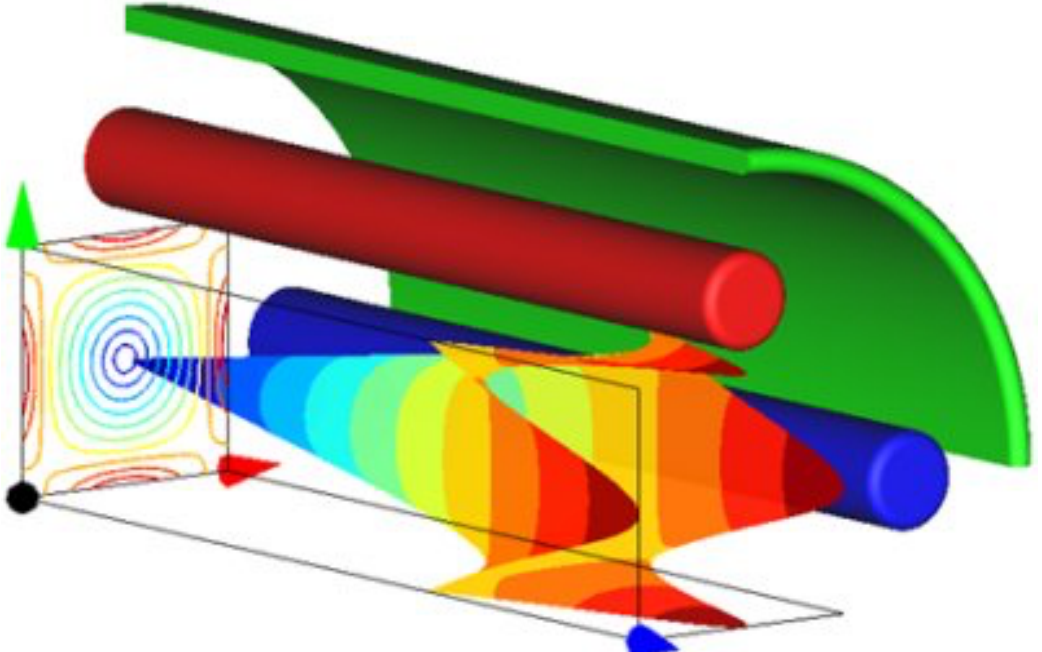
All wrong, some useful.

Useful for what?



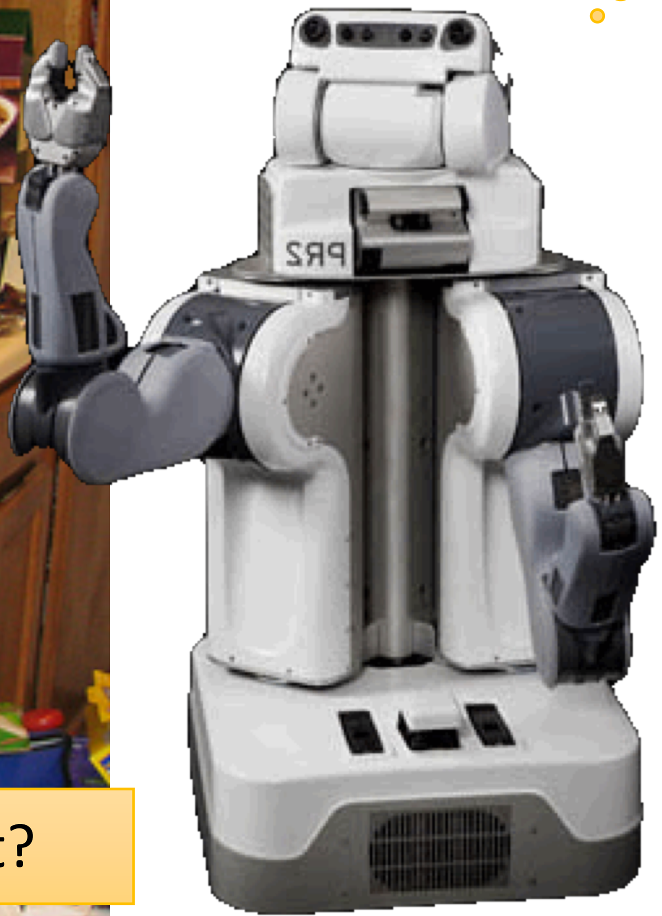






My focus: Models that are useful for robot action selection

WTF? [1]

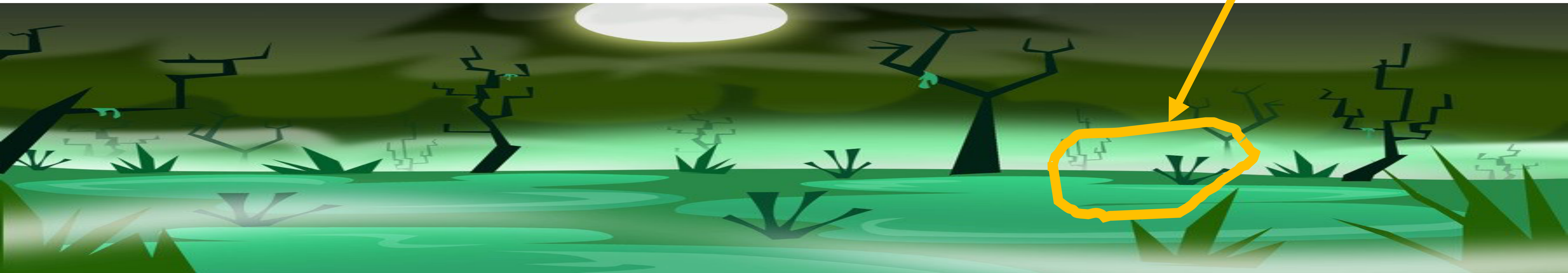


[1] What To do First?

Konidaris's tale of two levels

crisp
- potentially highly inaccurate
- sparse
- easier to learn model
- easier to compute with

swampy: (PDEs)
- potentially very accurate
- hard to know state
- hard to learn model
- hard to compute with

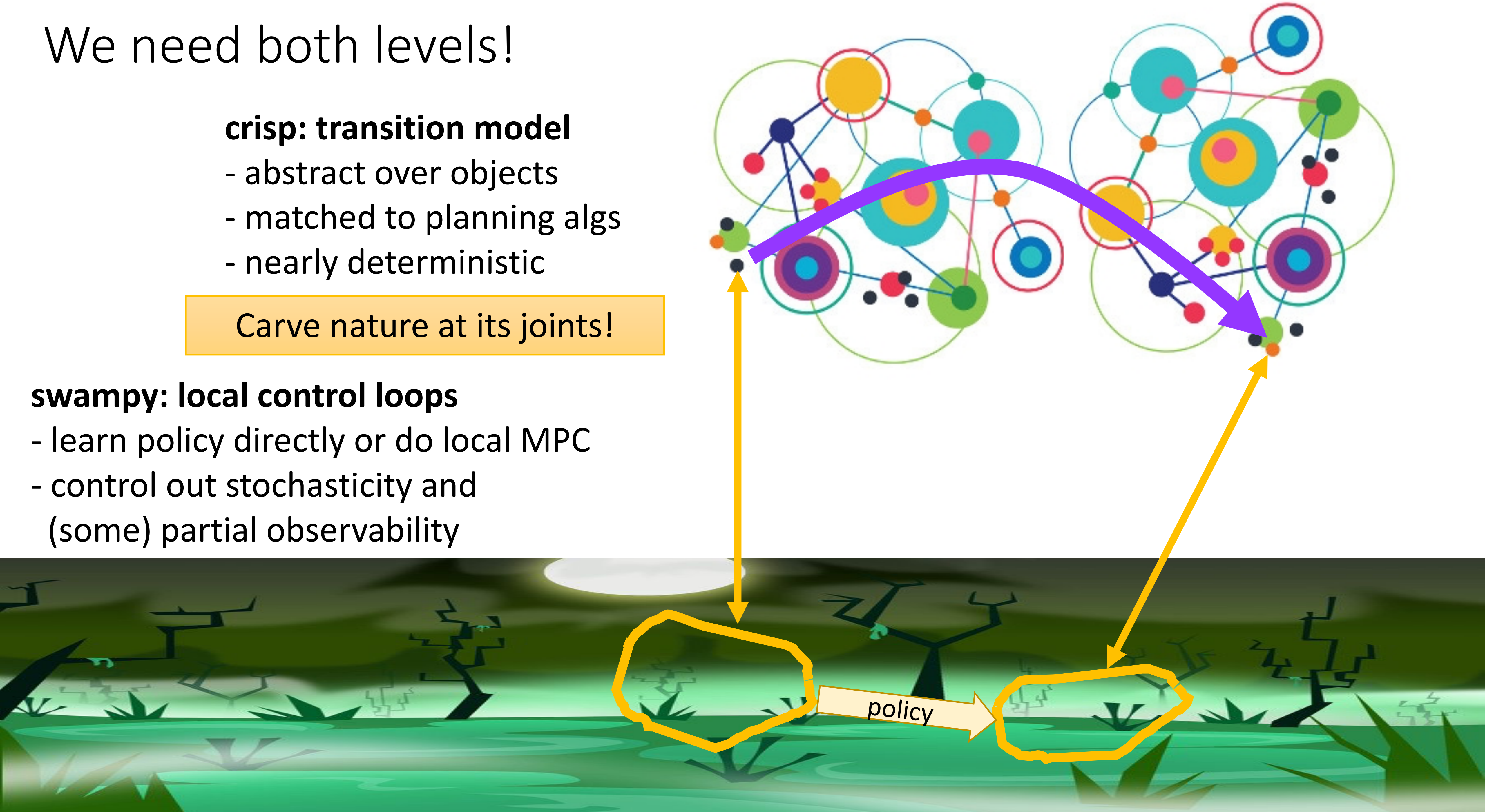


We need both levels!

crisp: transition model
- abstract over objects
- matched to planning algs
- nearly deterministic

Carve nature at its joints!

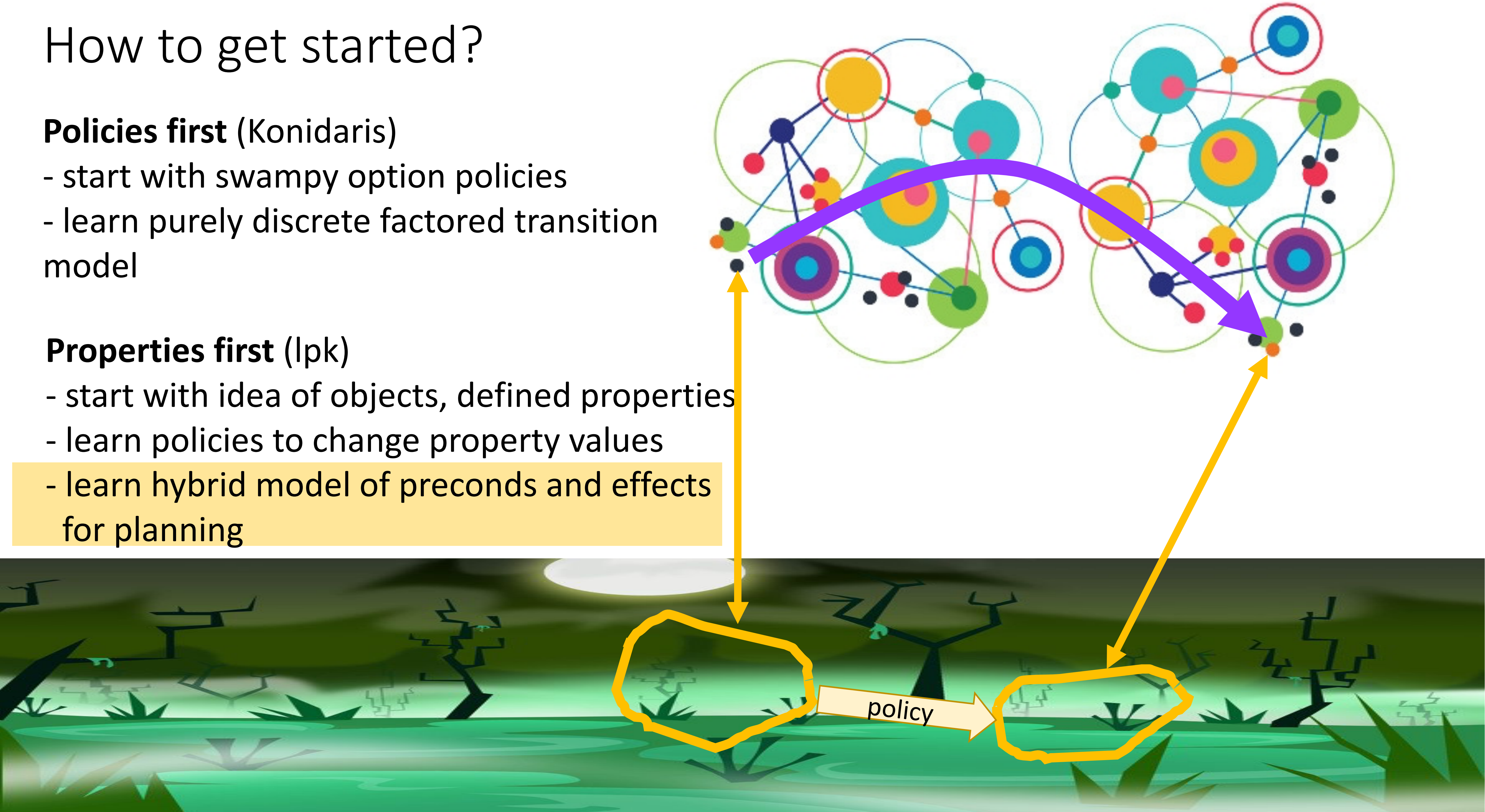
swampy: local control loops
- learn policy directly or do local MPC
- control out stochasticity and (some) partial observability



How to get started?

Policies first (Konidaris)
- start with swampy option policies
- learn purely discrete factored transition model

Properties first (lpk)
- start with idea of objects, defined properties
- learn policies to change property values
- learn hybrid model of preconds and effects for planning



Planning in large hybrid (discrete + continuous) domains

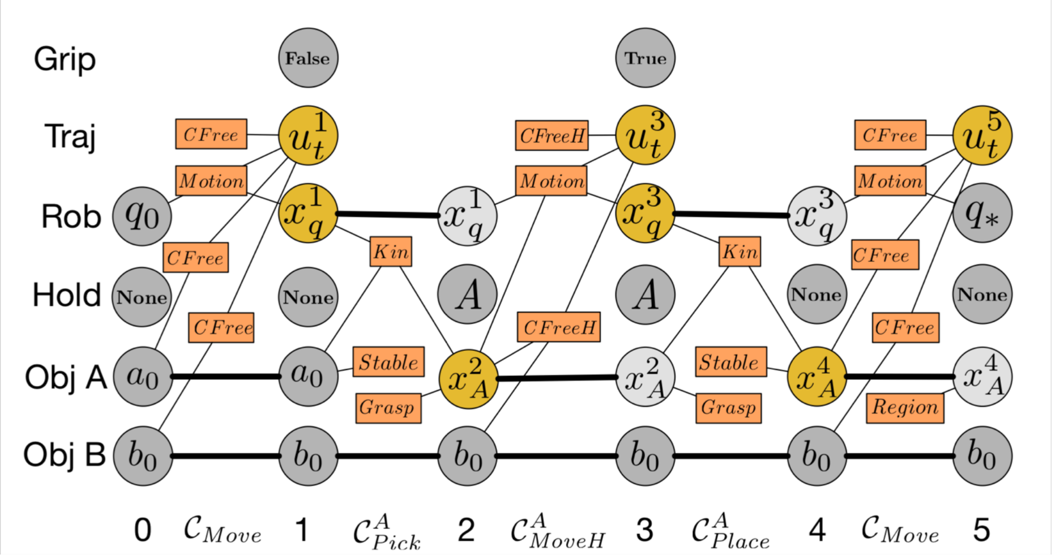
Pure forward search unlikely to work

- infinite branching factor
- unbiased sampling unlikely to satisfy downstream requirements

Represent constraints among discrete and continuous values

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: assume given

- 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

Learning a new operator

1. Determine which other objects may affect or be affected by this policy
   - Old approach (Pasula, Zettlemoyer, K)
   - Preliminary new approach (Xia, Wang, K)

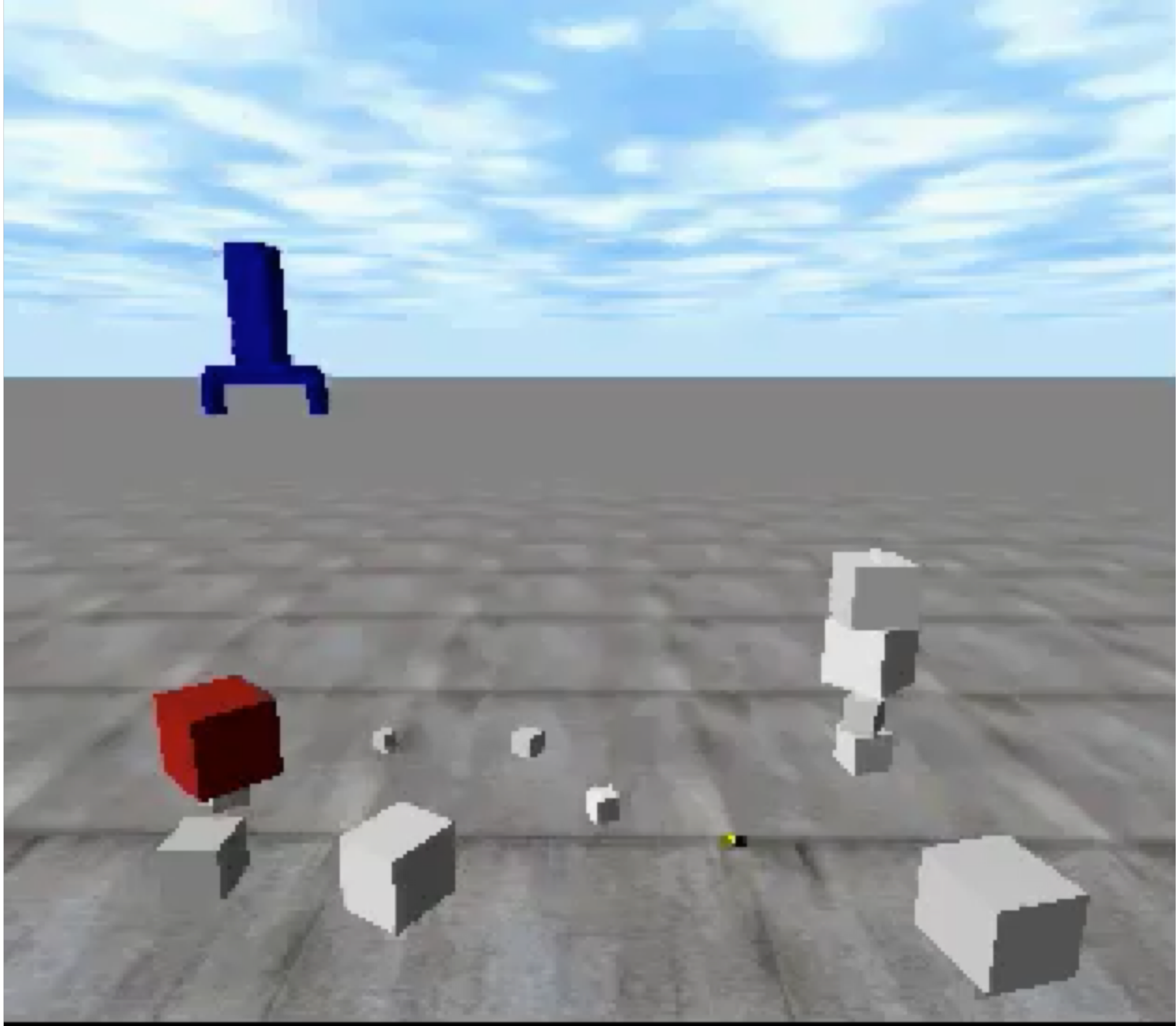
2. Learn detailed relation on properties of all these objects that predicts when the policy will have its intended effect
   - Gaussian process method also supports planning (Wang, Garrett, K, Lozano-Perez)

Learning a new operator

1. Determine which other objects may affect or be affected by this policy
   - Old approach (Pasula, Zettlemoyer, K)
   - Preliminary new approach (Xia, Wang, K)

2. Learn detailed relation on properties of all these objects that predicts when the policy will have its intended effect
   - Gaussian process method also supports planning (Wang, Garrett, K, Lozano-Perez)

Blocks with (wonky) physics



Pasula, Zettlemoyer, K; JAIR 2007

Probabilistic dynamic rules

Combine logic and probability to
model effects of actions in complex, uncertain domains

deictic reference:
Y = the object X is on

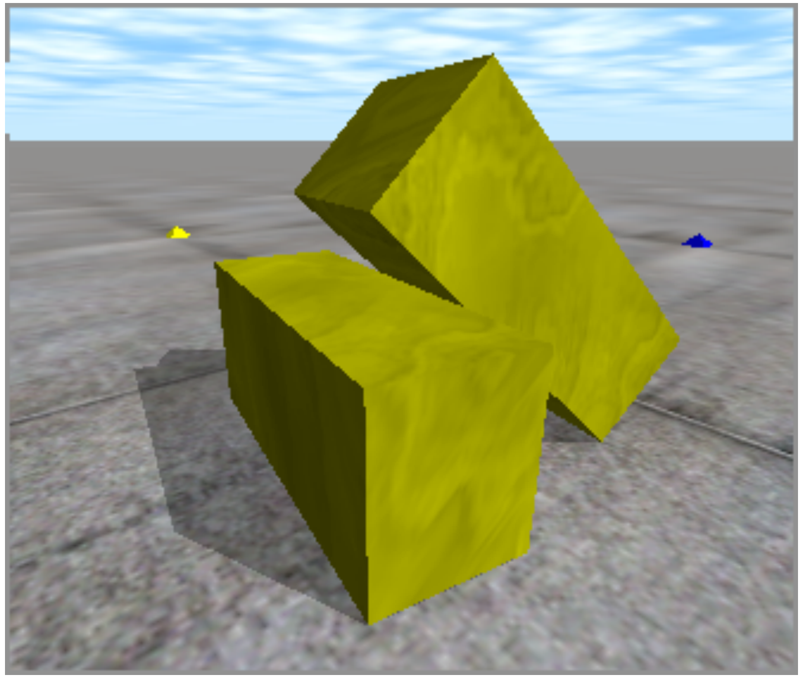
```
pickup(X): {Y: on(X,Y)}
    clear(X), inhand-nil, size(X)>2, size(X)<7 →
          0.803 :¬on(X,Y)
          0.093 : no change
```

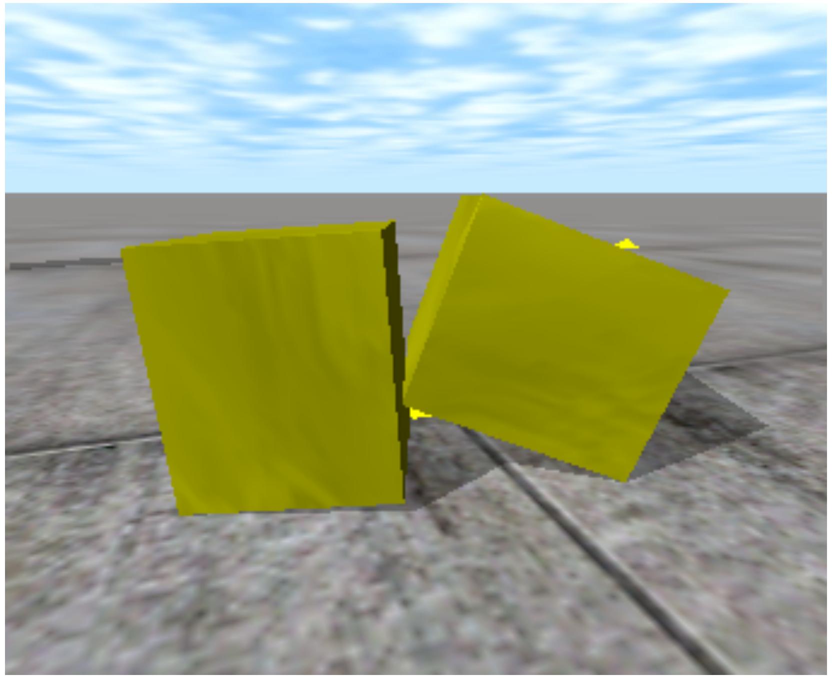
STRIPS assumption:
no other relations change

sparse:
few dependencies

noise outcome:
probs don't sum to 1

For now, assume some definition of "On"





Useful symbolic vocabulary should be learned

Neoclassical learning

Given experience, $\{\langle s_t, a_t, s_{t+1} \rangle\}$

Find rule set that optimizes

$$\mathrm{score}(R) = \sum_t \log \Pr(s_{t+1} \mid s_t, a_t, R) - \alpha |R|$$

Start with one default rule: "stuff happens"

- Symbolic: add, delete rule; change rule conditions
  - Greedy: choose set of outcomes
    - Convex optimization: find maximum likelihood probabilities

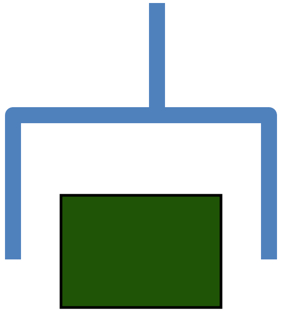
Concept invention

- New concepts allow predictive theory to be expressed more compactly and learned from less data
- Add outer loop with logical operations, quantification, transitive closure, counting
- Definitions subject to complexity penalty

`p1(X) :- ¬∃Y. on(X,Y)`	X is in the hand
`p2() :- ¬∃Z. p1(Z)`	nothing is in the hand
`p3(X) :- ¬∃Y. on(Y,X)`	X is clear
`p4(X,Y) :- on(X,Y)*`	X is above Y
`p5(X,Y) :- p3(X) ∧ p4(X,Y)`	X is on the top of the stack containing Y
`f6(X) :- #Y. p4(X,Y)`	the height of X

Rules learned from data

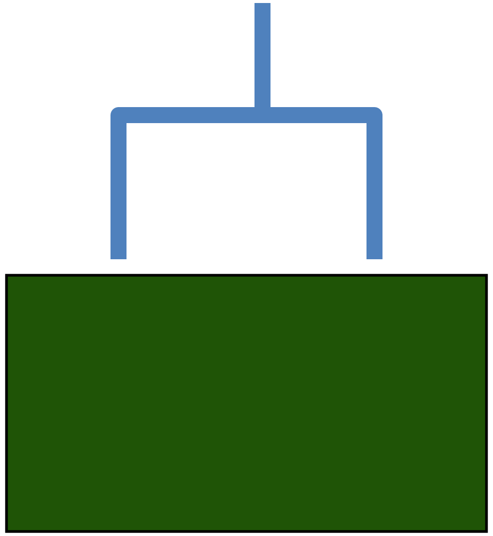
```
pickup(X): {Y: on(X,Y)}
    clear(X), inhand-nil, size(X)>2, size(X)<7→
            0.803 :¬on(X,Y)
            0.093 : no change
```



picking up middle-sized blocks usually works

Rules learned from data

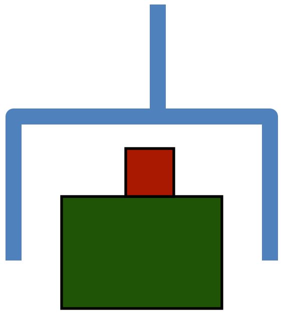
```
pickup(X):
    clear(X), inhand-nil, ¬size(X)<7 →
      0.906 : no change
```



it's impossible to pick up very big blocks

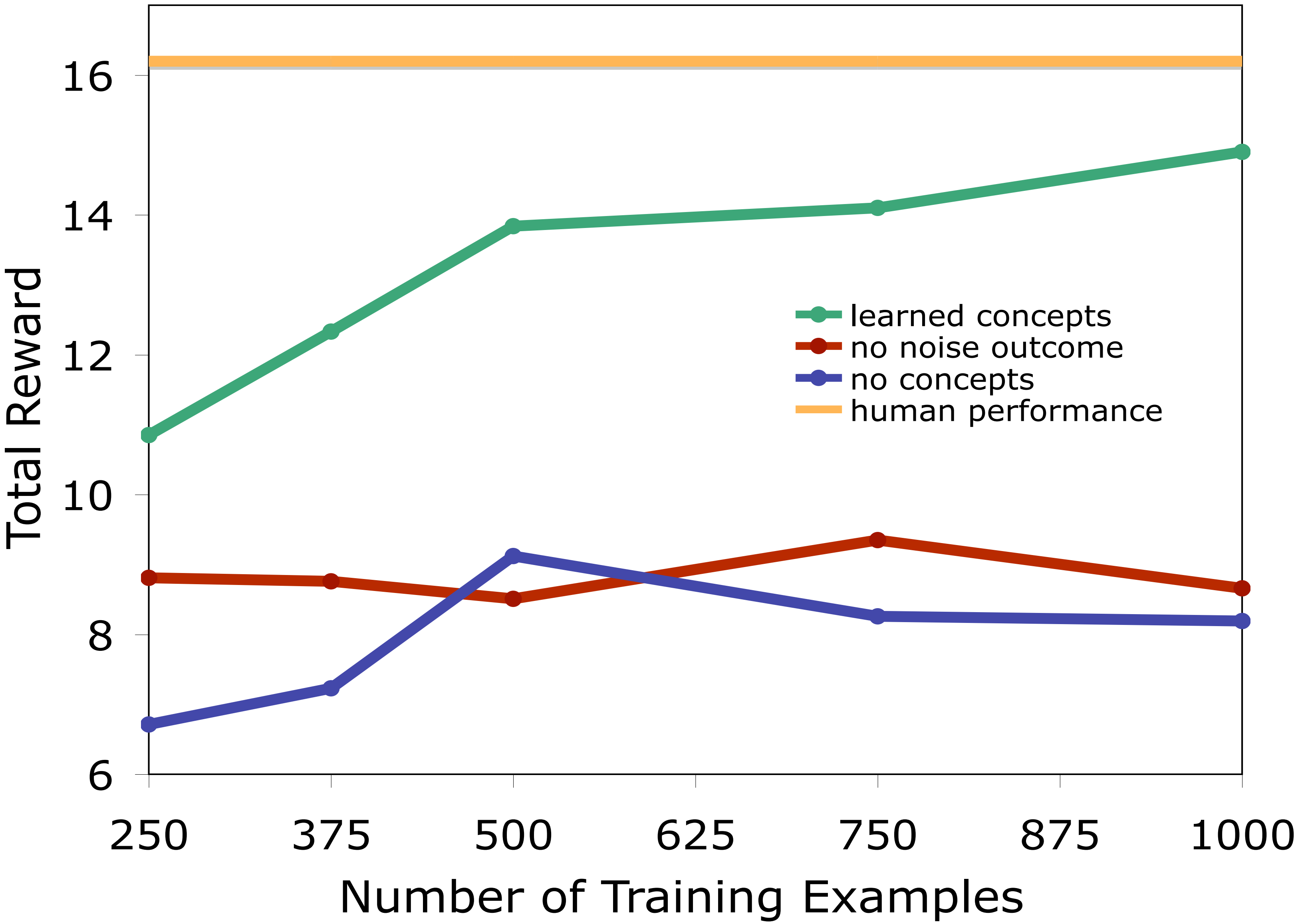
Rules learned from data

```
pickup(X): {T: table(T)}, {Y: on(X,Y), on(Y,T)}
      clear(X), inhand-nil, size(X)<2  →
            0.105 :¬on(X,Y)
            0.582 :¬on(Y,T)
            0.312 : no change
```



if a tiny block is on another block that is on the table, and we try to pick up the tiny block, we'll often pick up the other block as well, or fail

Planning with learned rules



Learning a new operator

1. Determine which other objects may affect or be affected by this policy
    - Old approach (Pasula, Zettlemoyer, K)
    - Preliminary new approach (Xia, Wang, K)

2. Learn detailed relation on properties of all these objects that predicts when the policy will have its intended effect
    - Gaussian process method also supports planning (Wang, Garrett, K, Lozano-Perez)

Operator descriptions: sparse relational transition model

Planning is efficient given representation that is

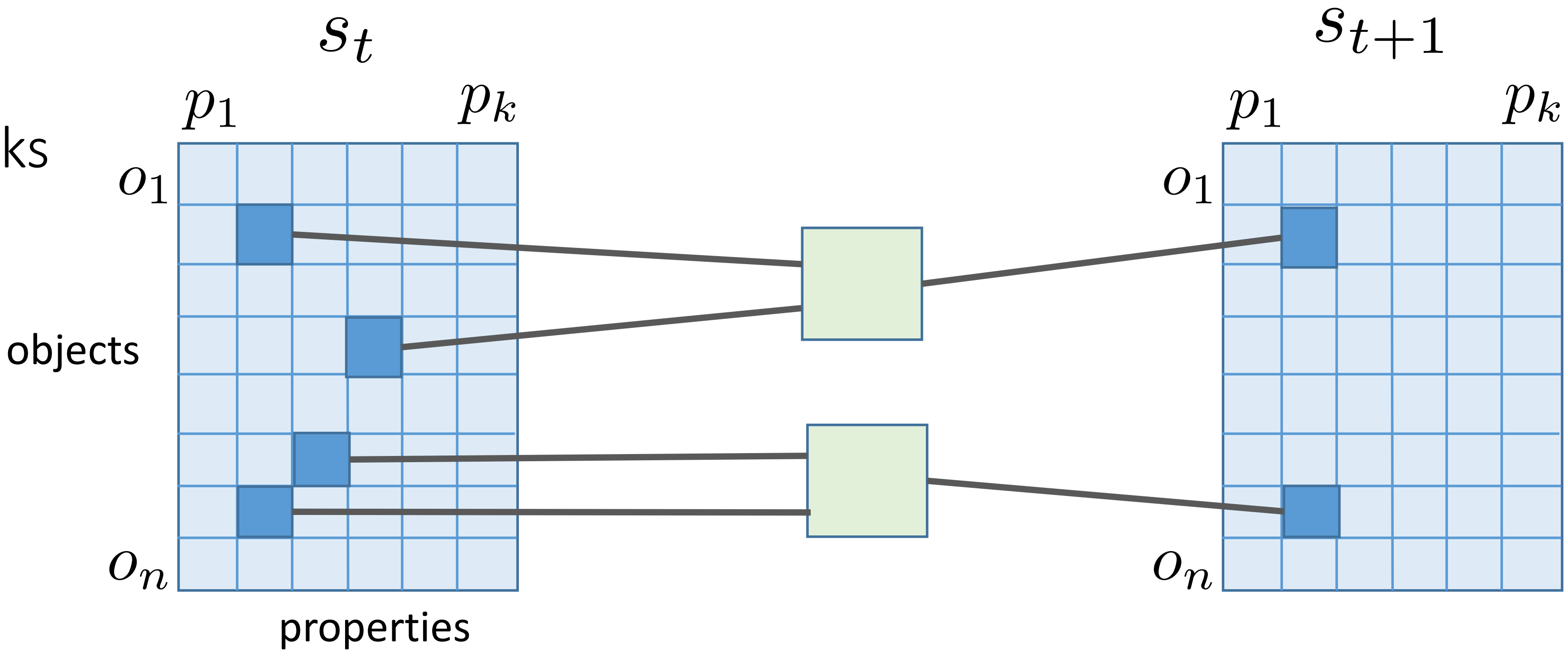
- factored, relational
- generalized over objects
- sparse and local

Operator descriptions represent transition model

Related models

- neural module networks
- graph networks
- attention models

$$\mathrm{Op}(A, B) : p_2(A) = v, p_4(B) = w, r_3(A, B) = c \rightarrow$$
$$p_2(A) = d$$



Xia, Wang, K; arXiv, 2018

Deictic references name related objects

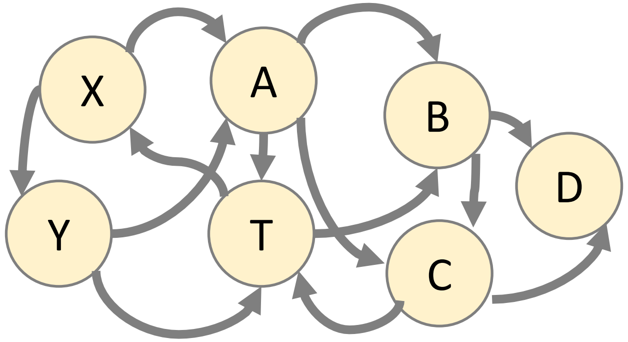
Rules automatically condition on and predict objects named in action: Push(Obj)

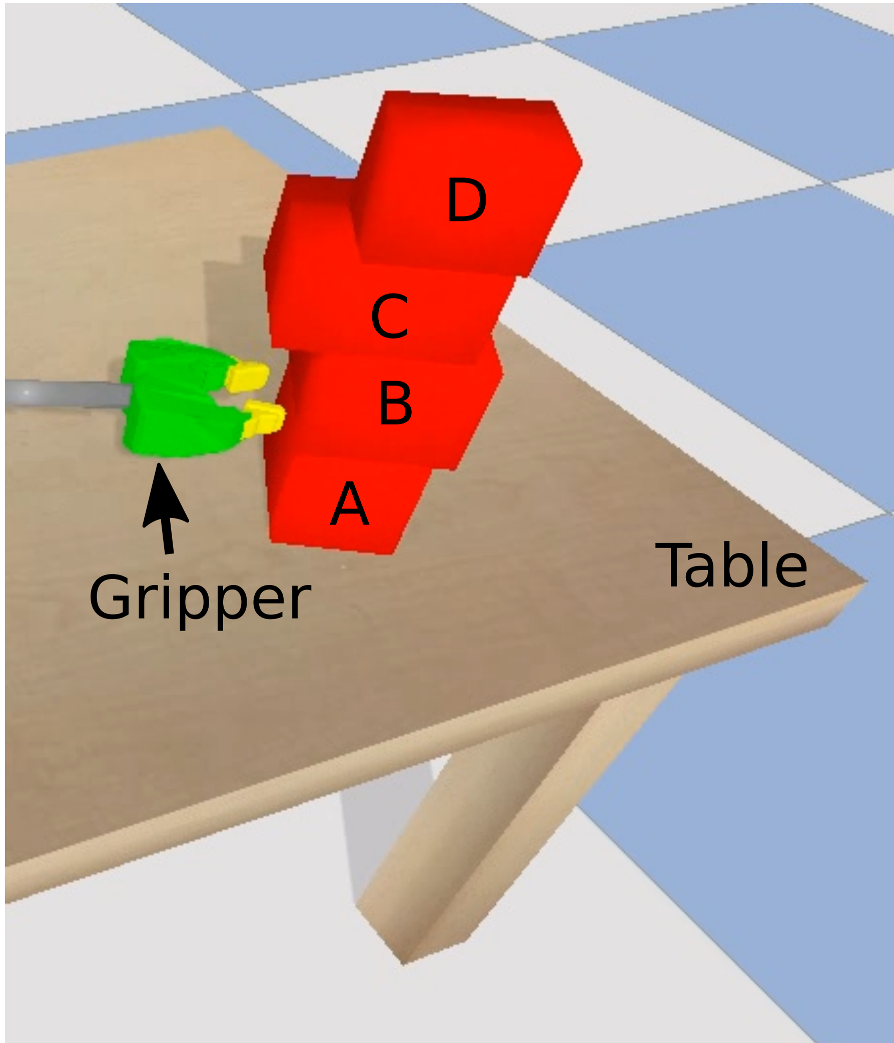
Refer to additional objects via **deictic references:**

- 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

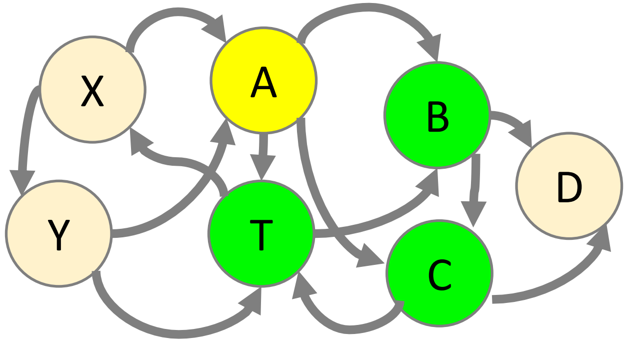
Rules automatically condition on and predict objects named in action: Push(Obj)

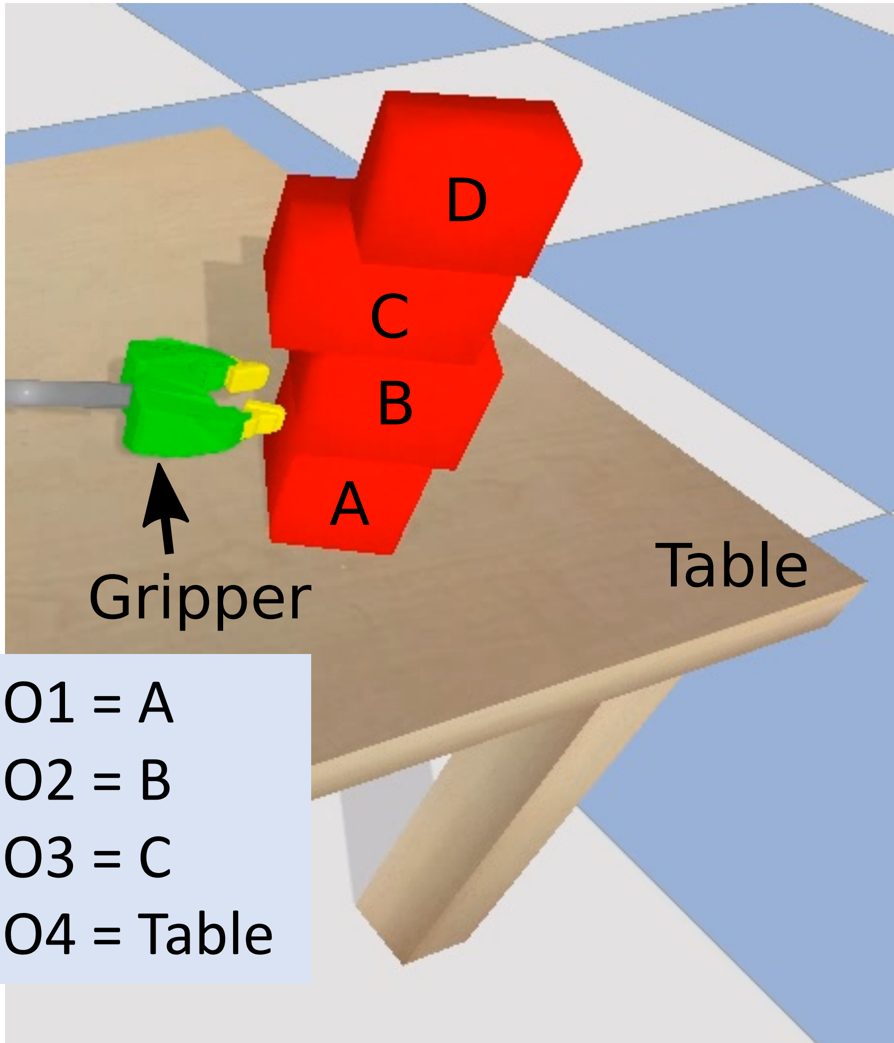
Refer to additional objects via **deictic references:**

- 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)

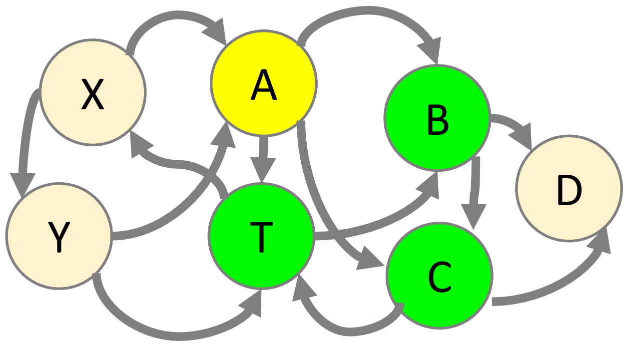


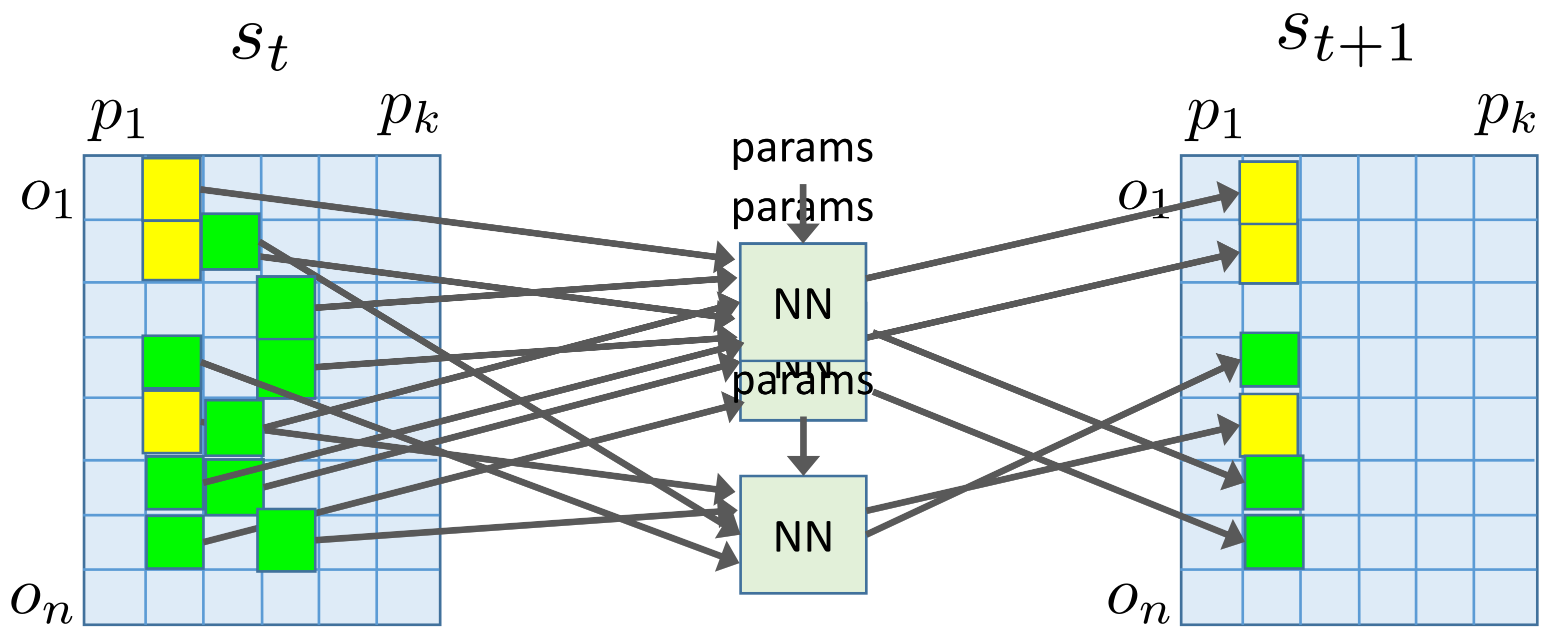


O1 = A
O2 = B
O3 = C
O4 = Table

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





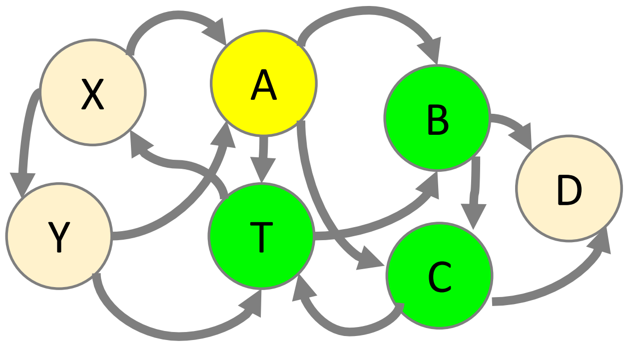
Rule learning: training data is (s, a, s')

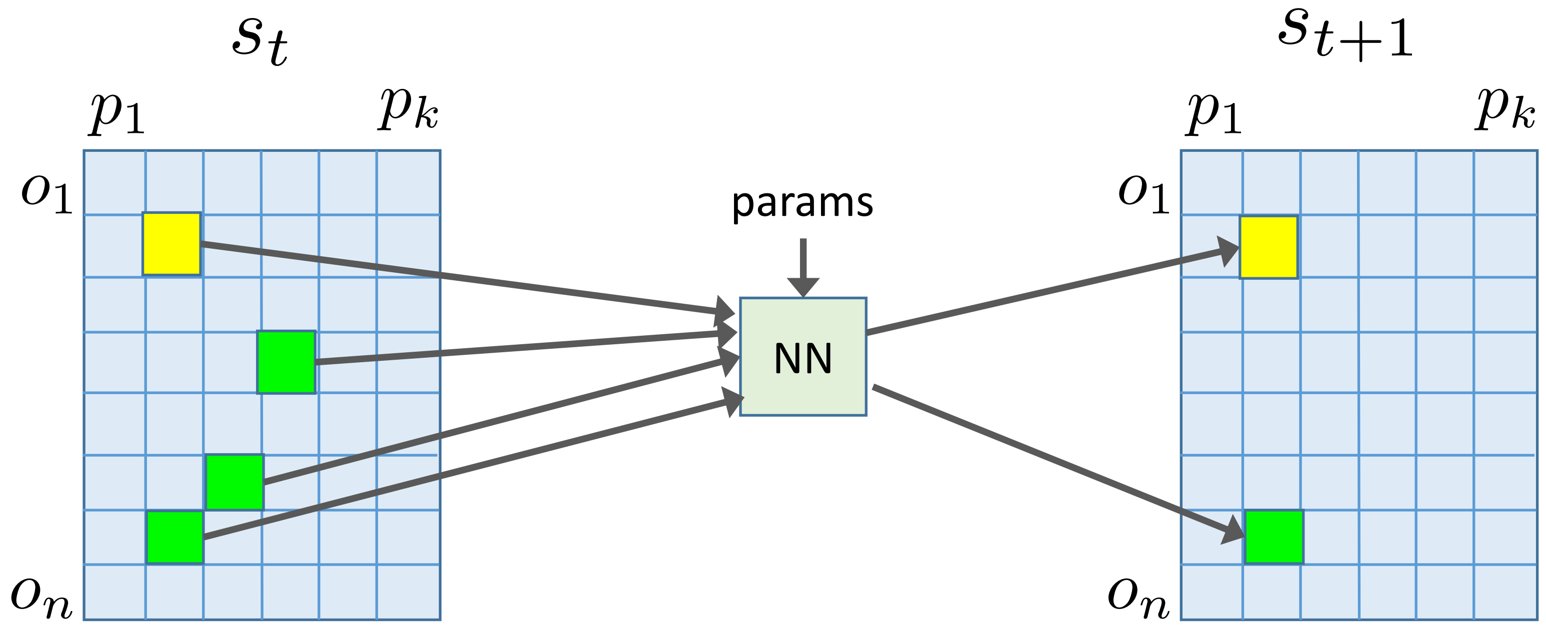
Outer loop over set of rules

- 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

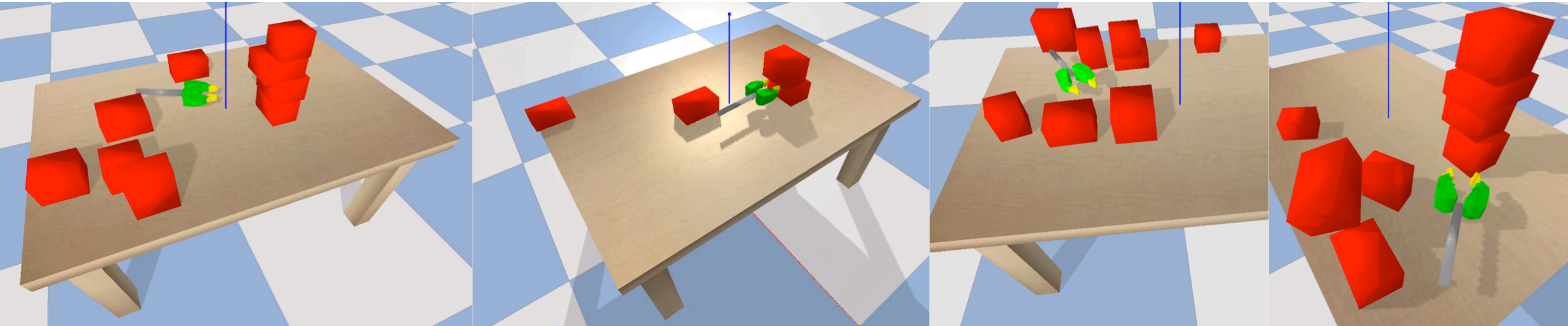




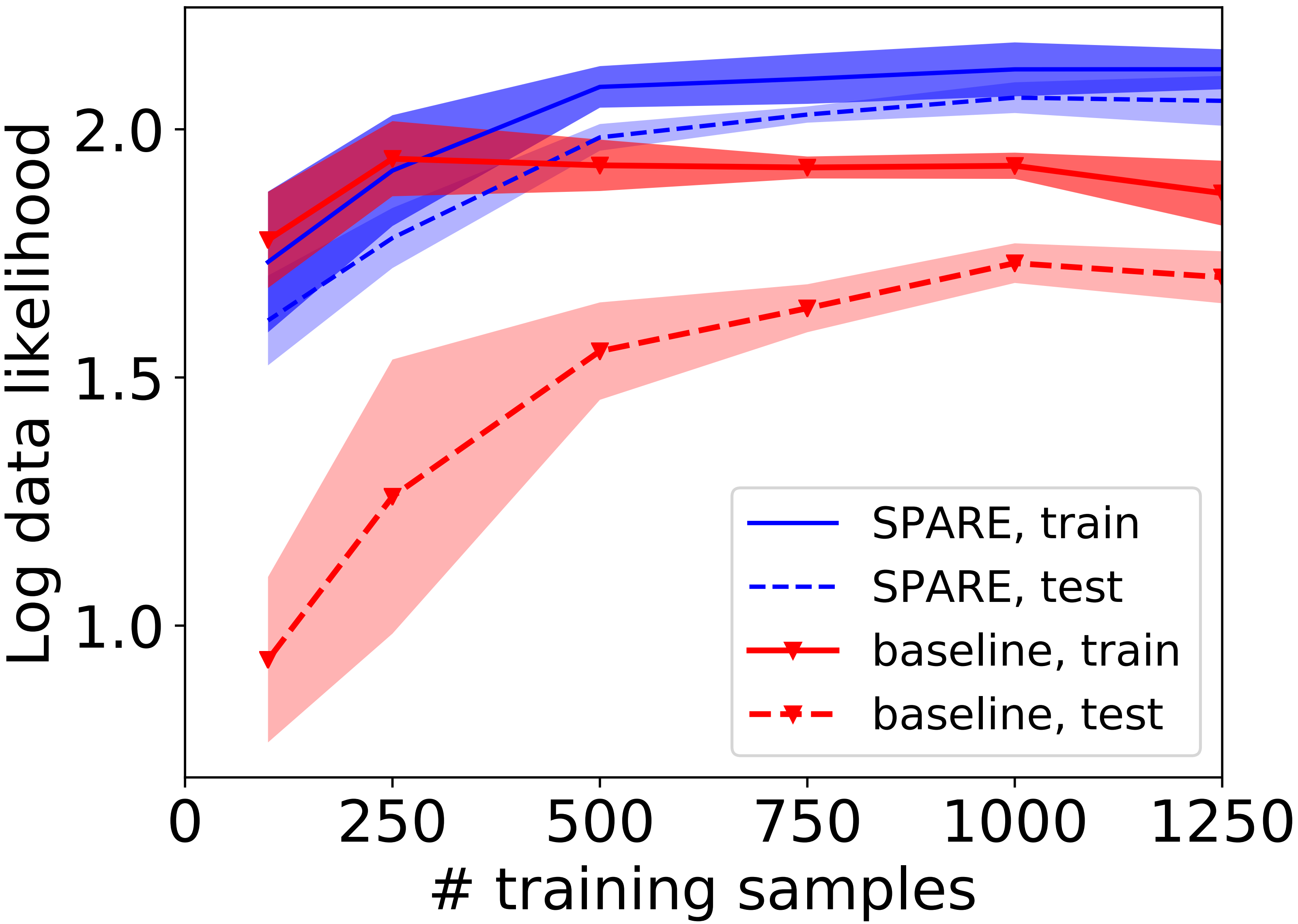
Preliminary results: pushing objects on crowded table

Compare likelihood on held-out data

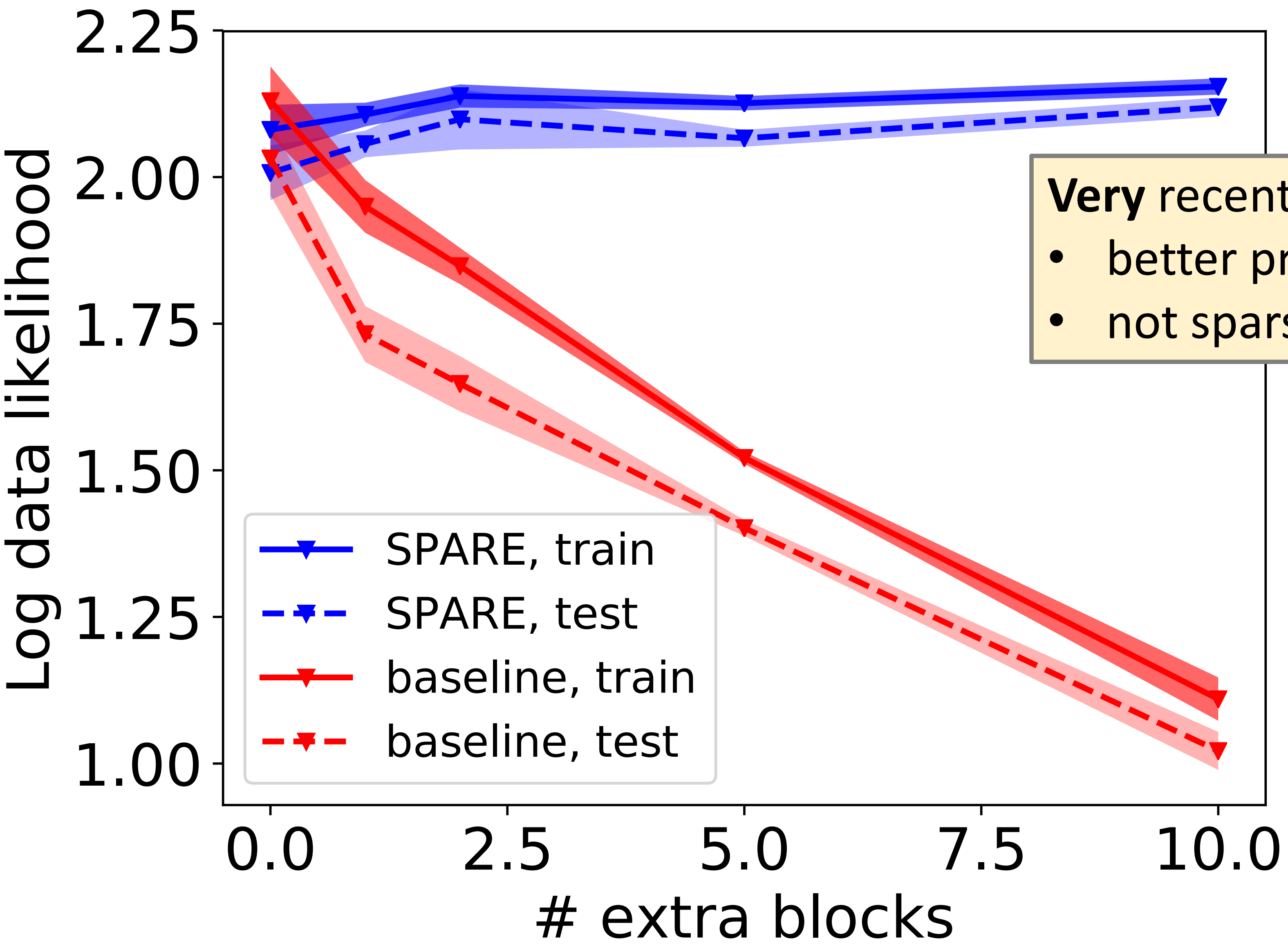
- Learned rule-based model
- Neural network trained on vector of attributes of all objects
  - Pushed object always first
  - Other objects sorted by distance from first



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



Sparse rules unaffected by clutter



Very recent results with graphNN

- better predictive likelihood
- not sparse for planning

Allen, Silver, now

Learning a new operator

1. Determine which other objects may affect or be affected by this policy
   - Old approach (Pasula, Zettlemoyer, K)
   - Preliminary new approach (Xia, Wang, K)

2. Learn detailed relation on properties of all these objects that predicts when the policy will have its intended effect
   - Gaussian process method also supports planning (Wang, Garrett, K, Lozano-Perez)

Operator description for planning: when will **skill** achieve **result**?

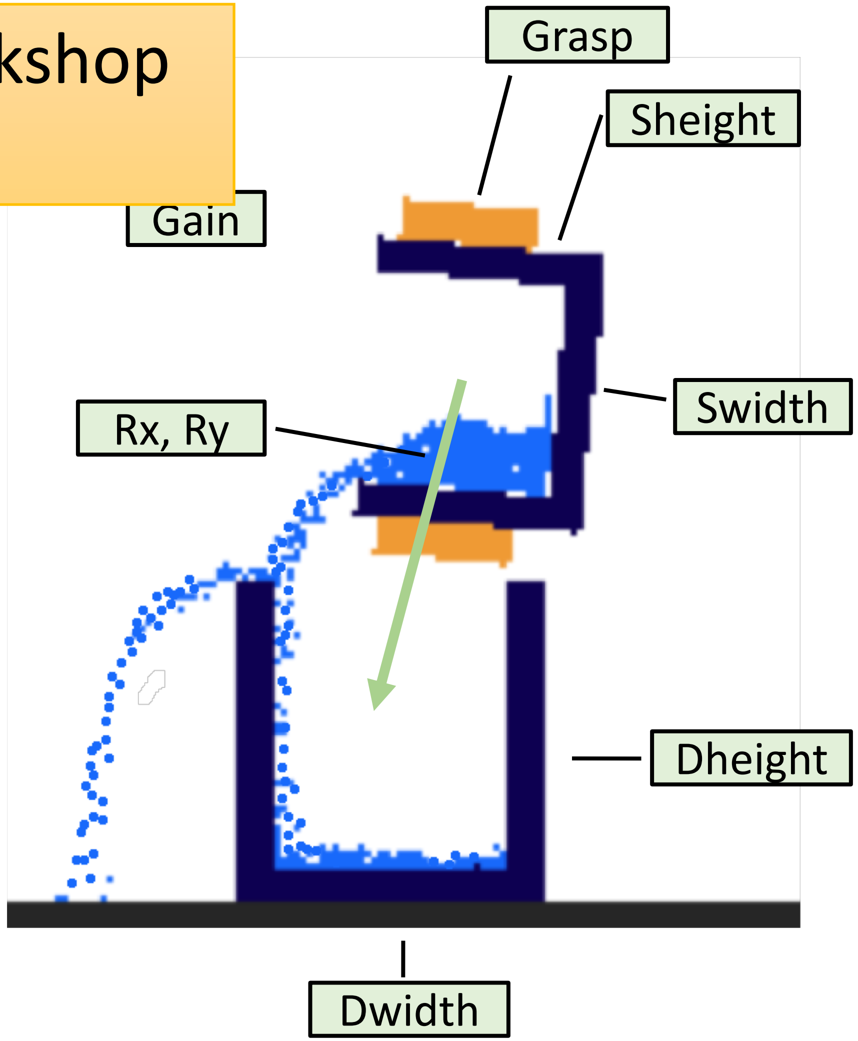
Result: Contains(Dest, Liquid)

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)**

Infer2Control Workshop
Saturday 3:30PM



Wang, Garrett, K, Lozano-Perez; IROS 2018

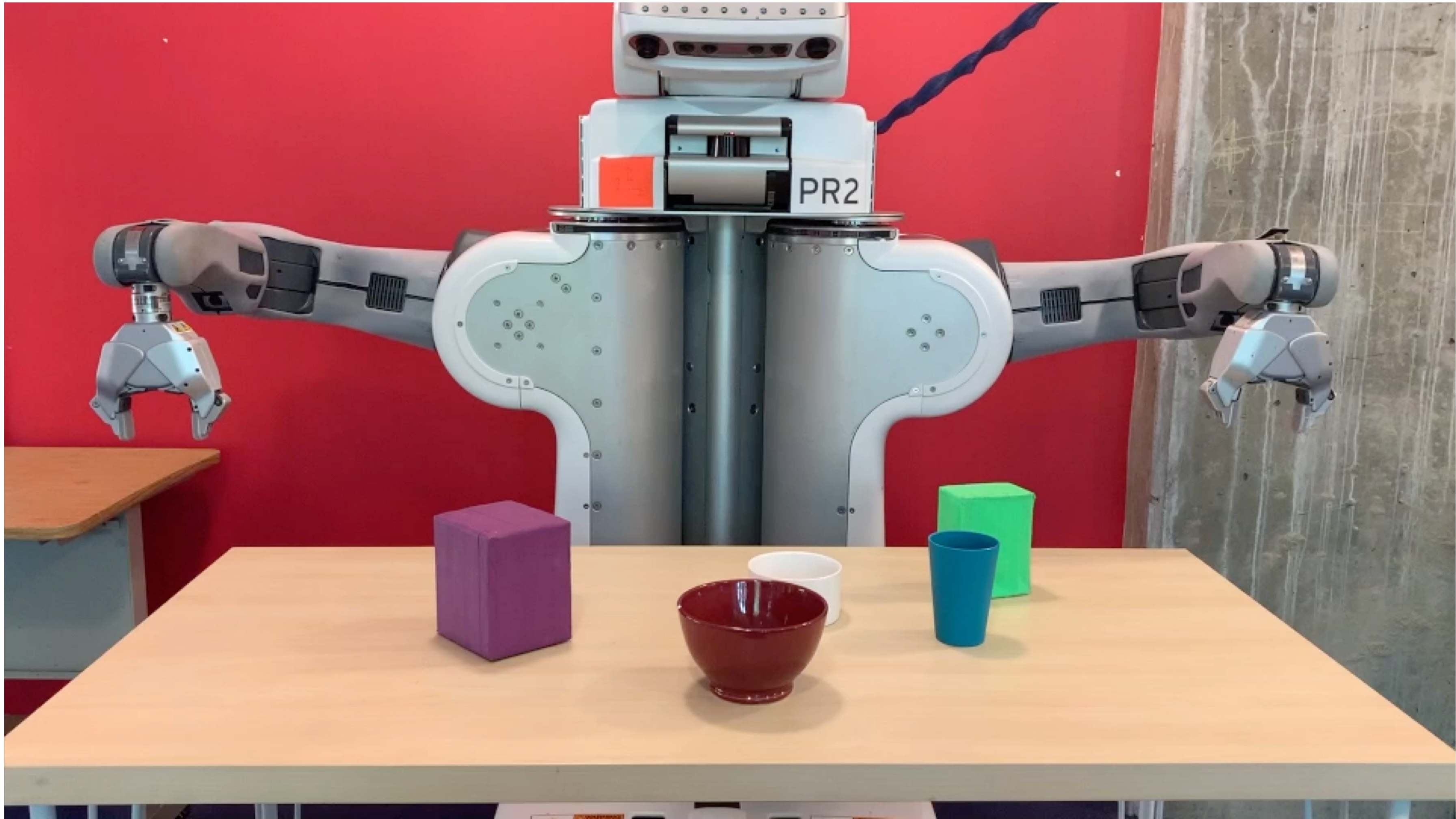
Preliminary results on robot (learning constraint only)

Substantial variability in

- starting arrangement
- goal

Given pick/place operators

Learned pour and push



Lots to do

- Quantify "epistemic" uncertainty in learned models
- Combine with active learning approach
- Extend to partial observability

Infer2Control Workshop
Saturday 3:30 PM

RL in POMDP Workshop
Saturday 9:05 AM

Thanks. And out-takes to watch during questions