# Combining State and Temporal Abstraction

George Konidaris
gdk@cs.duke.edu

DUKE
COMPUTER
SCIENCE

# Abstraction

# Abstraction

# Skill Hierarchies

Base control on *skills*.

- Component of behavior.
- Performs continuous, low-level control.
- *Temporal abstraction.*

Some evidence that humans organize their behavior this way.

# Skill Hierarchies

Base control on *skills*.

- Component of behavior.
- Performs continuous, low-level control.
- *Temporal abstraction*.

Some evidence that humans organize their behavior this way.

Development

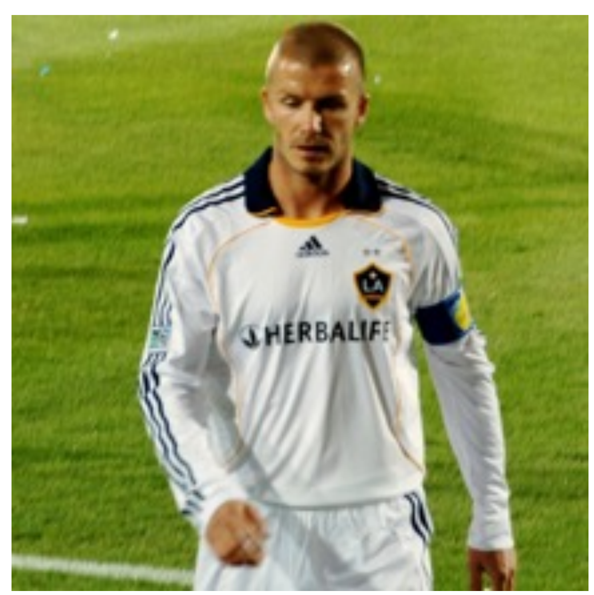
# Skill Hierarchies

Base control on *skills*.

- Component of behavior.
- Performs continuous, low-level control.
- *Temporal abstraction.*

Some evidence that humans organize their behavior this way.

Development



Specialization

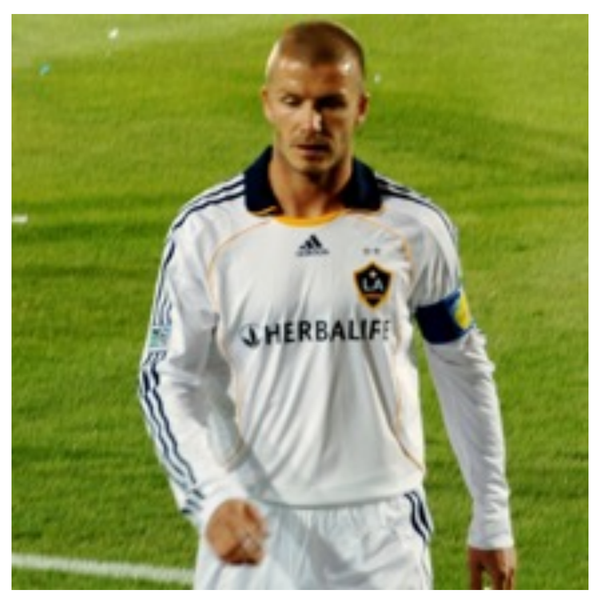
# Skill Hierarchies

Base control on *skills*.

- Component of behavior.
- Performs continuous, low-level control.
- *Temporal abstraction.*

Some evidence that humans organize their behavior this way.

Development



Specialization

Simplification

# Skill Hierarchies

***Behavior is modular and compositional.***

# Skill Hierarchies

***Behavior is modular and compositional.***

Skills are like *subroutines*.

```
def abs(x):
    if(x > 0):
      return x
    else:
      return -x
```

*[Wilkes, Wheeler and Gill, 1951]*

# Skill-Specific Abstractions

Skills should also be abstract.

- Many high-dimensional problems really are high-dimensional if you try to solve them monolithically
- Can split into subproblems, each of which support a solution using an abstraction.

*[IJCAI 2009]*

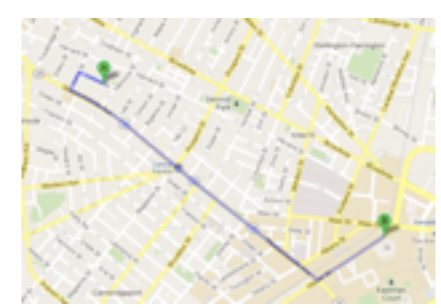
# Skill-Specific Abstractions

Skills should also be abstract.

- Many high-dimensional problems really are high-dimensional if you try to solve them monolithically
- Can split into subproblems, each of which support a solution using an abstraction.





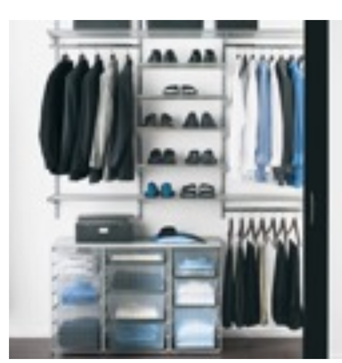


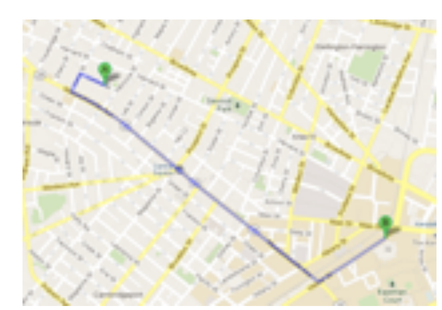
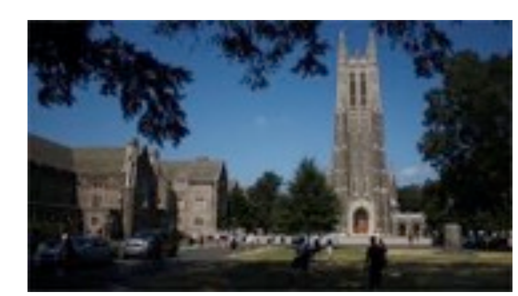

*[IJCAI 2009]*

# Skill-Specific Abstractions

Skills should also be abstract.

- Many high-dimensional problems really are high-dimensional if you try to solve them monolithically
- Can split into subproblems, each of which support a solution using an abstraction.

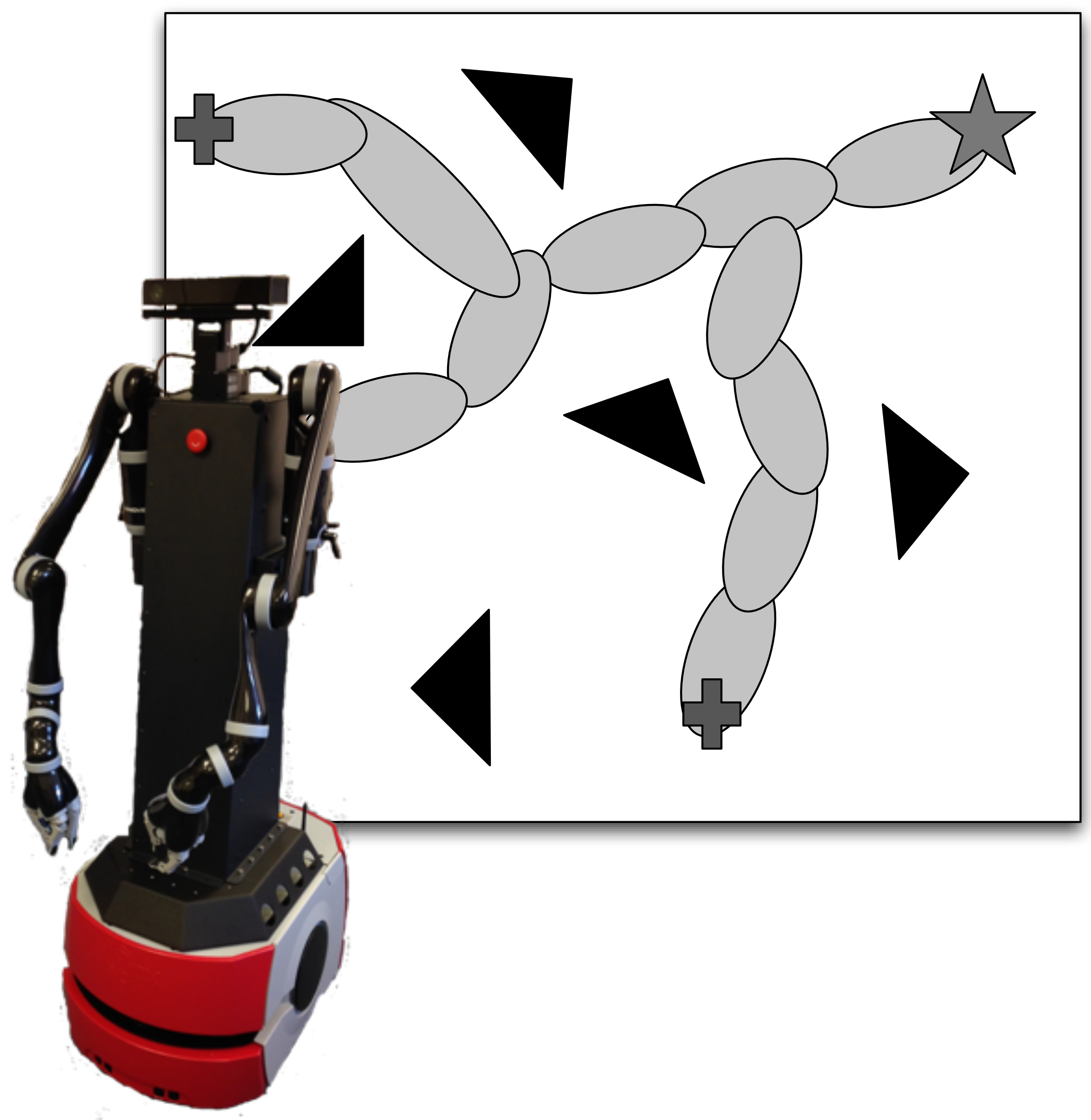



**Behavior is piecewise low-dimensional.**

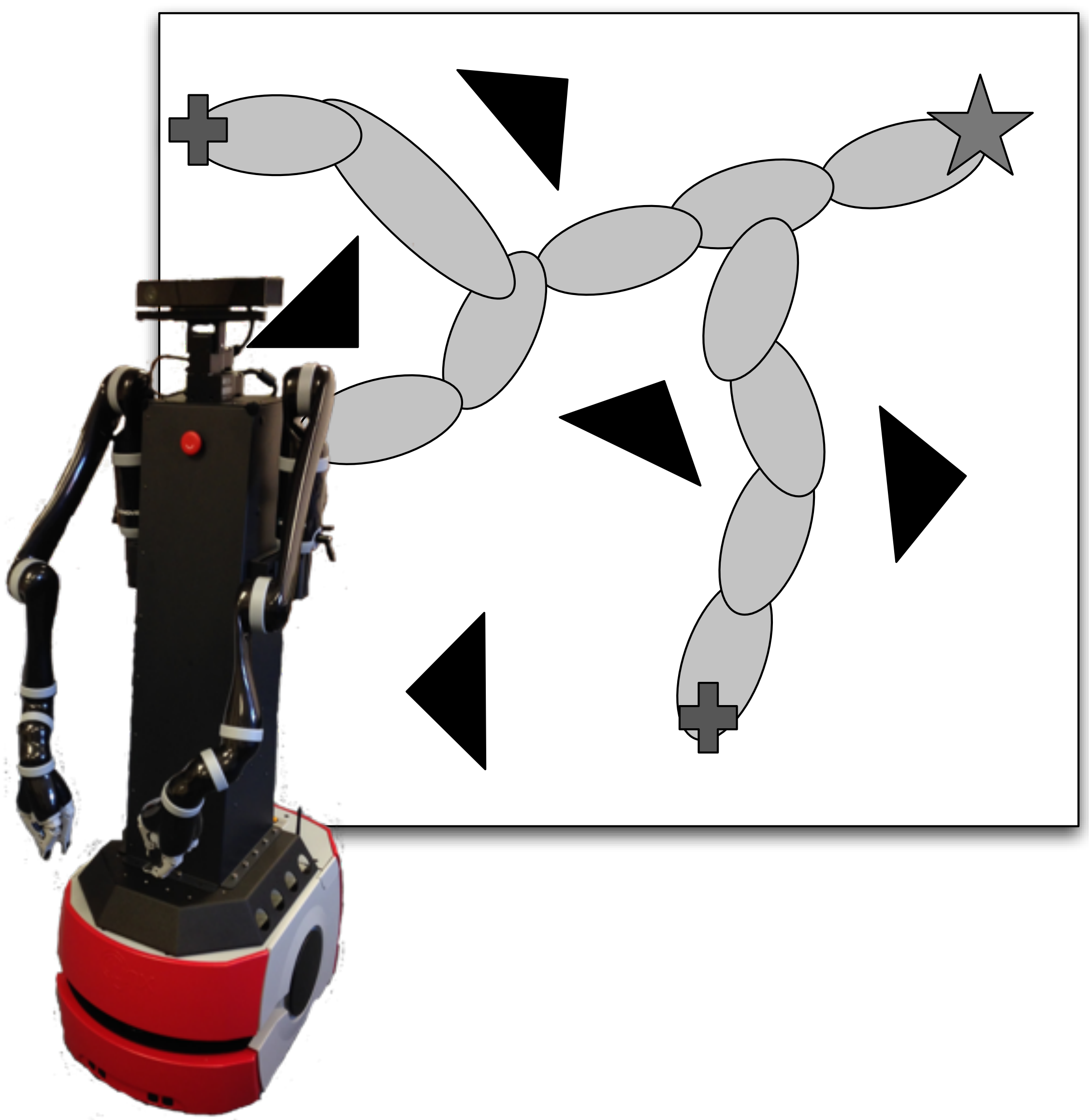
*[IJCAI 2009]*

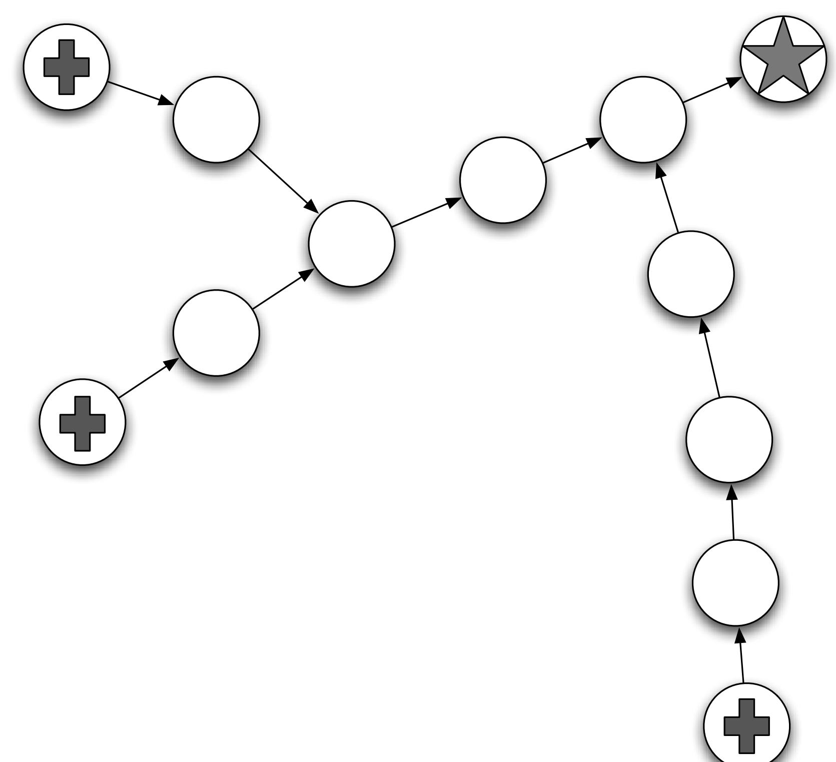
# Skill-Generated Abstractions

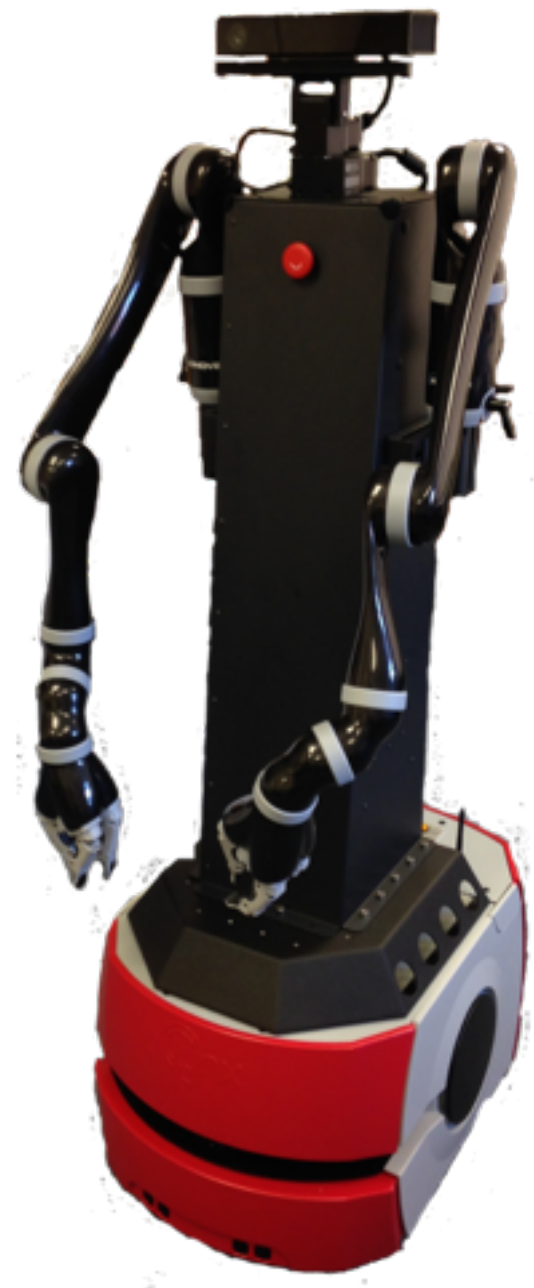


# Skill-Generated Abstractions

DUKE
COMPUTER
SCIENCE



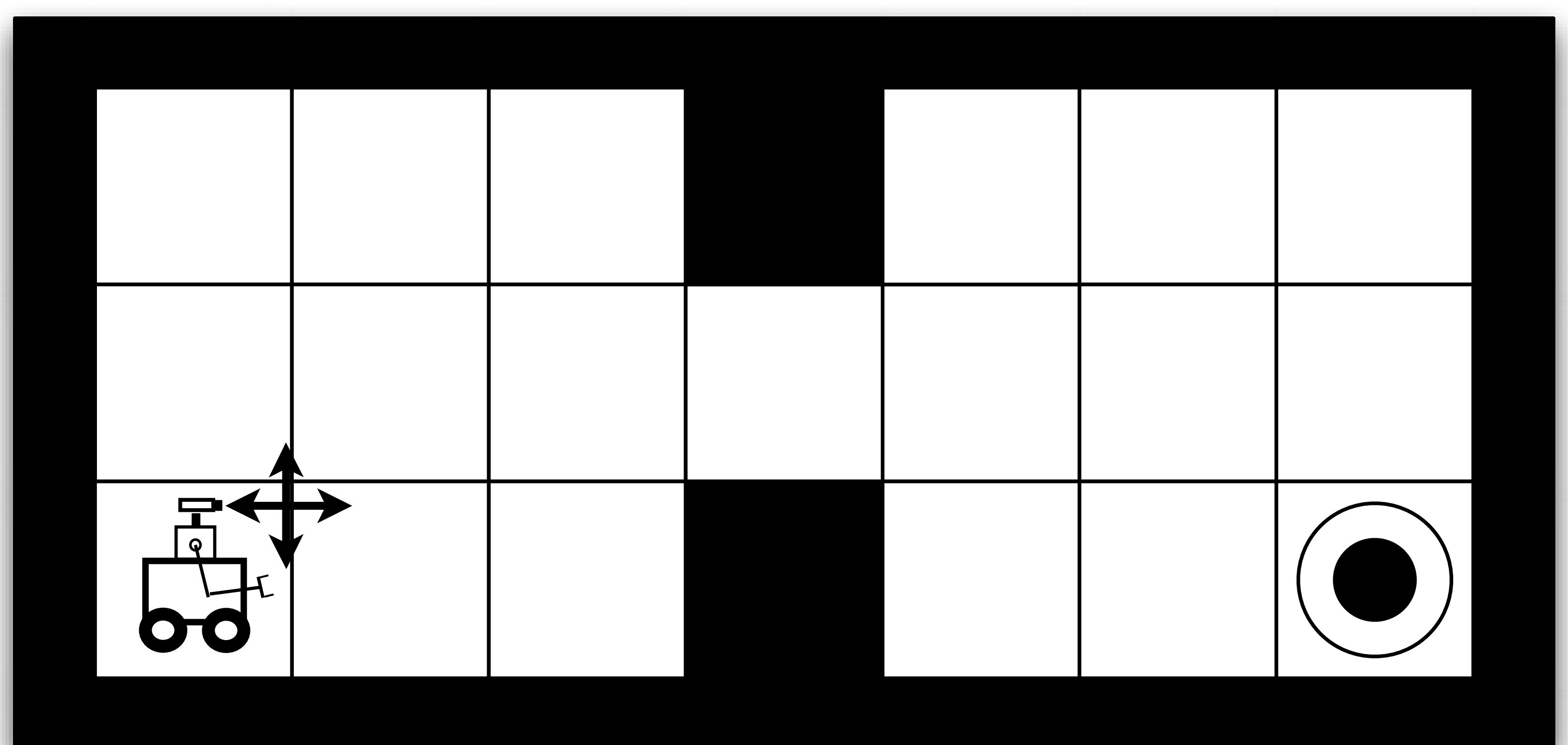




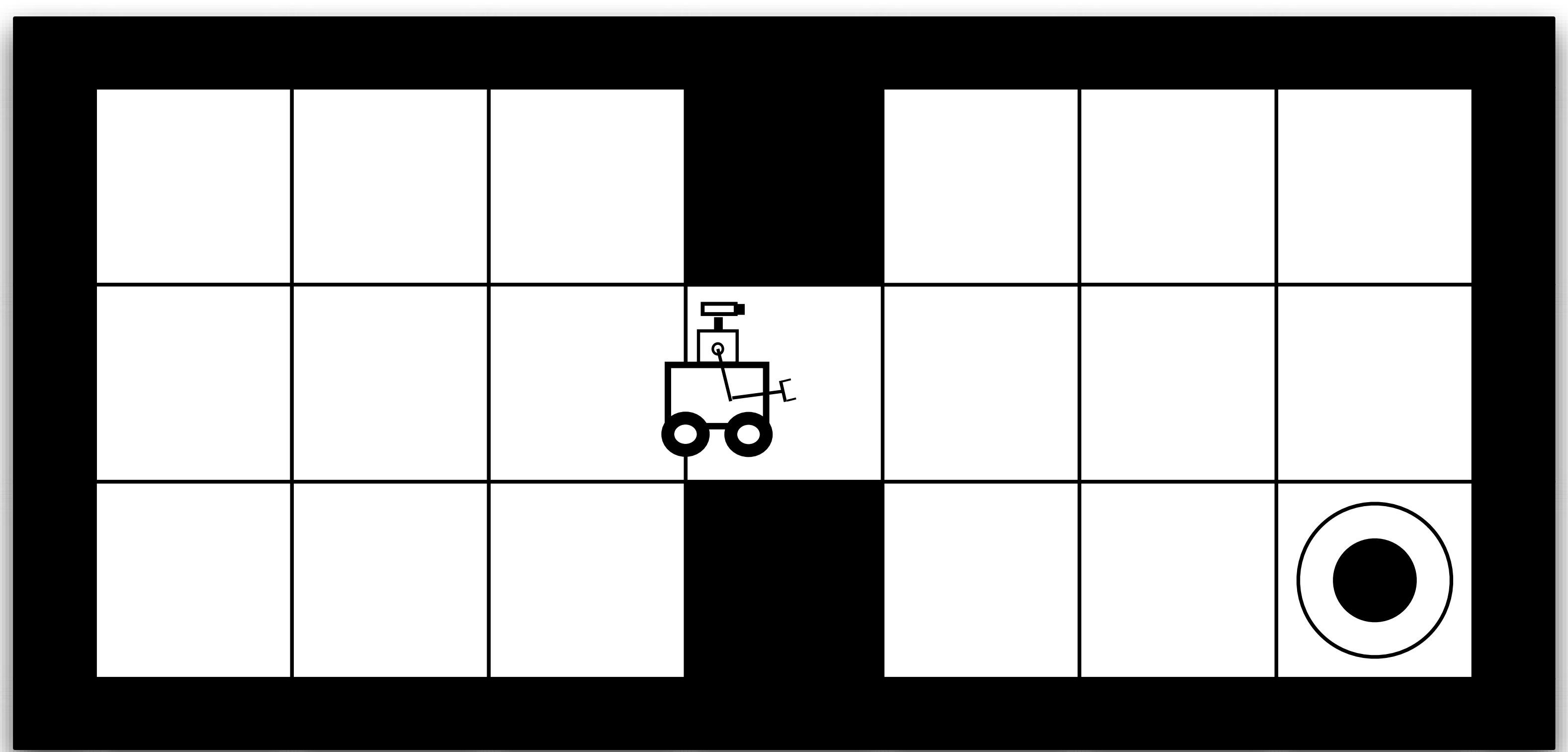
# Skills

# Skills

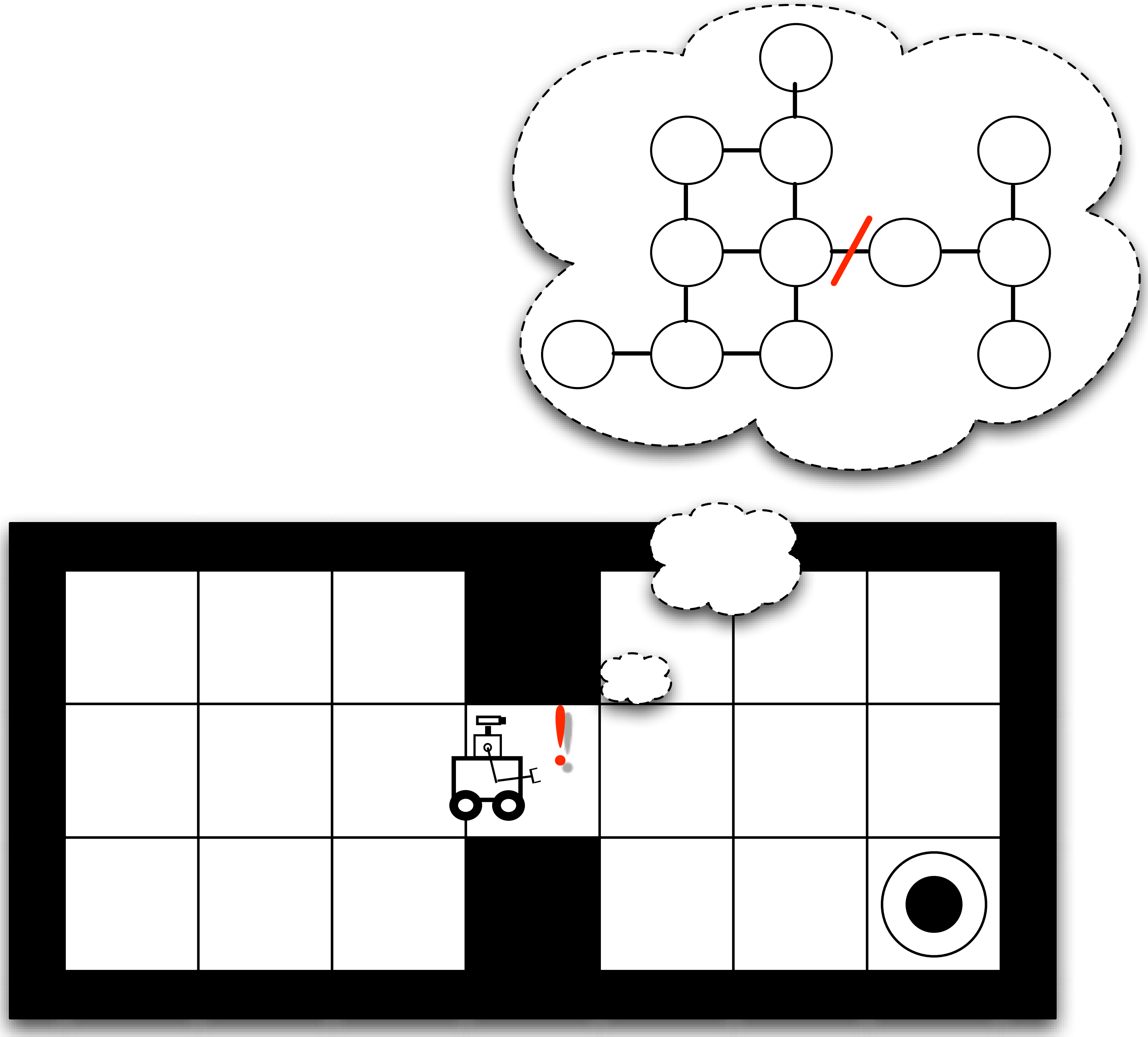




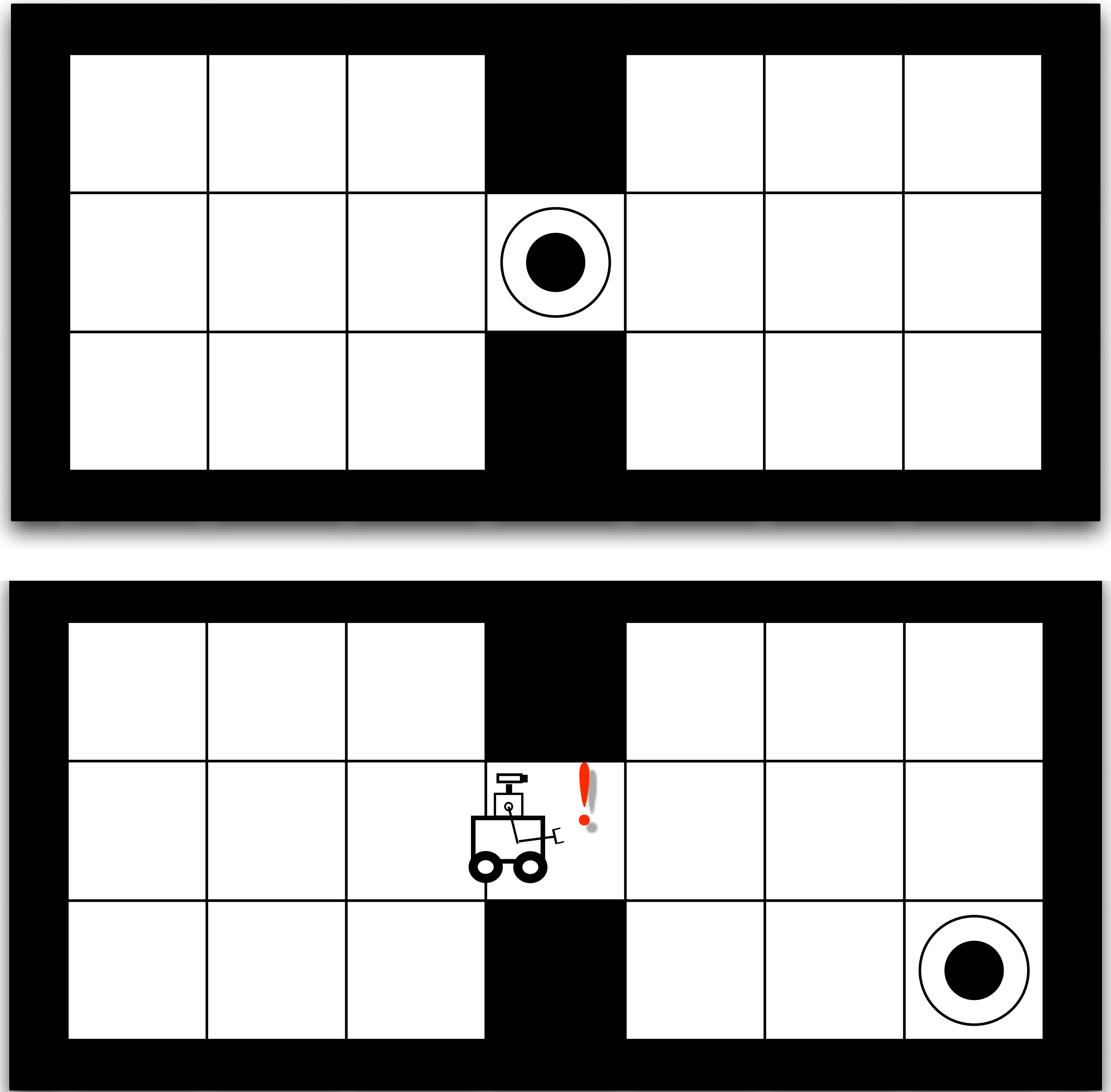
# Skills



# Skills

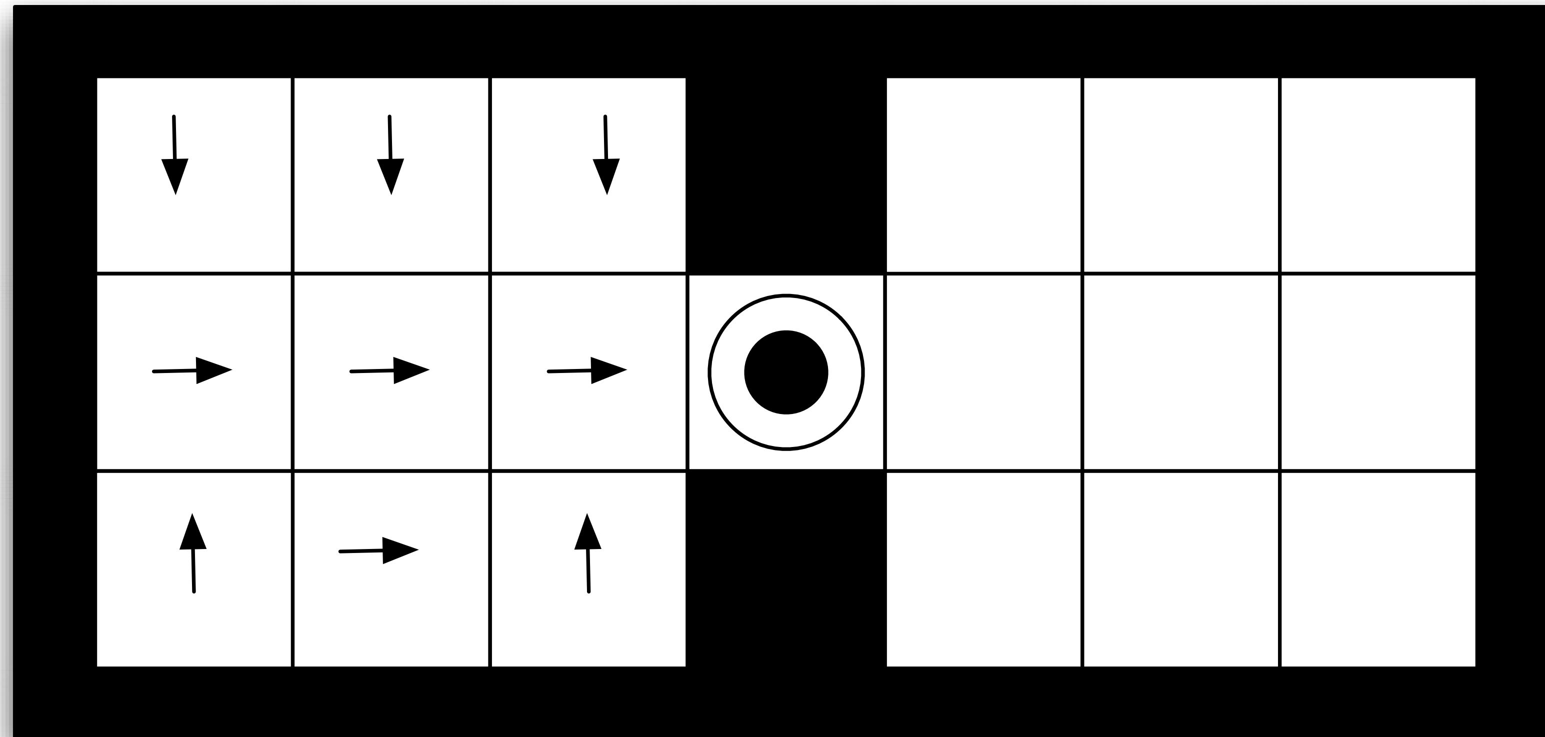


# Skills

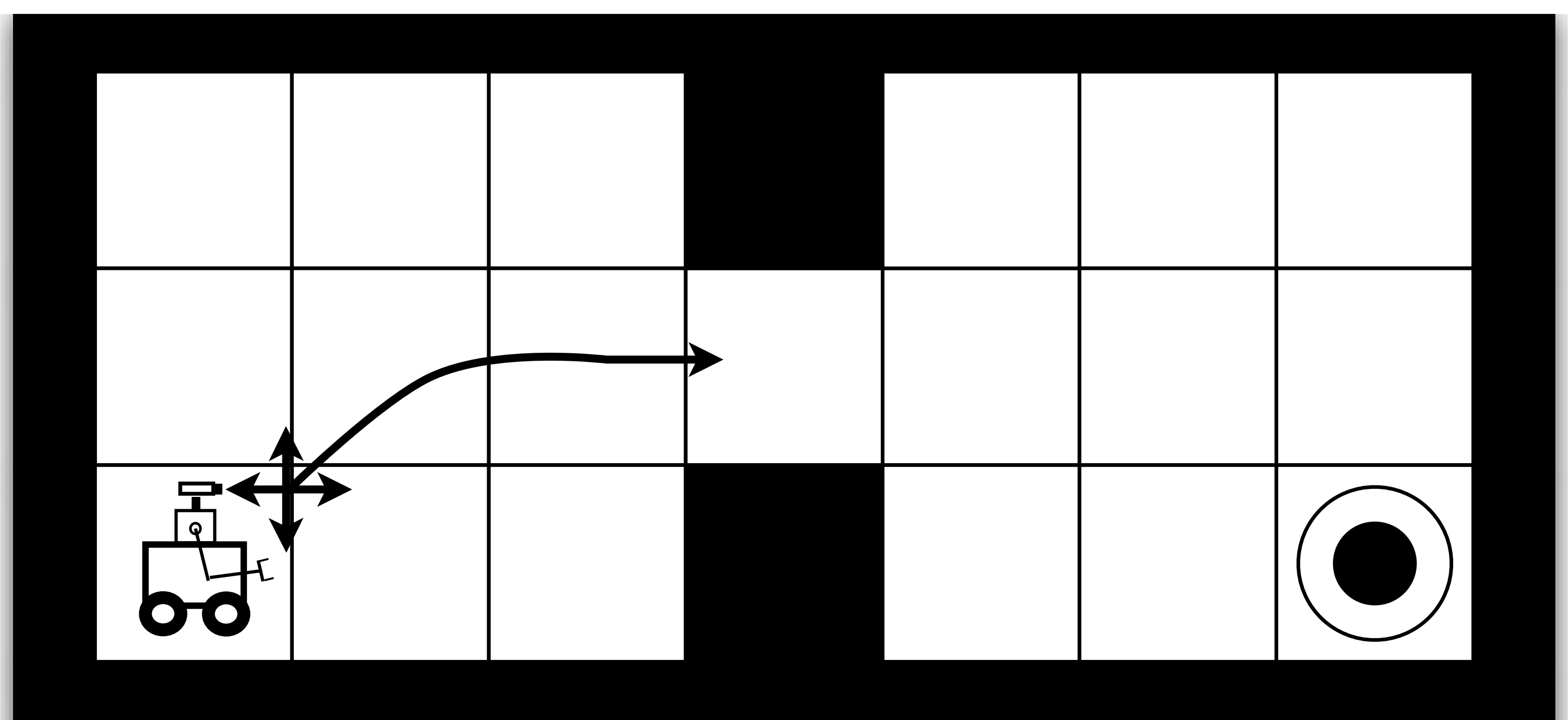


# Skills

Skill



Problem



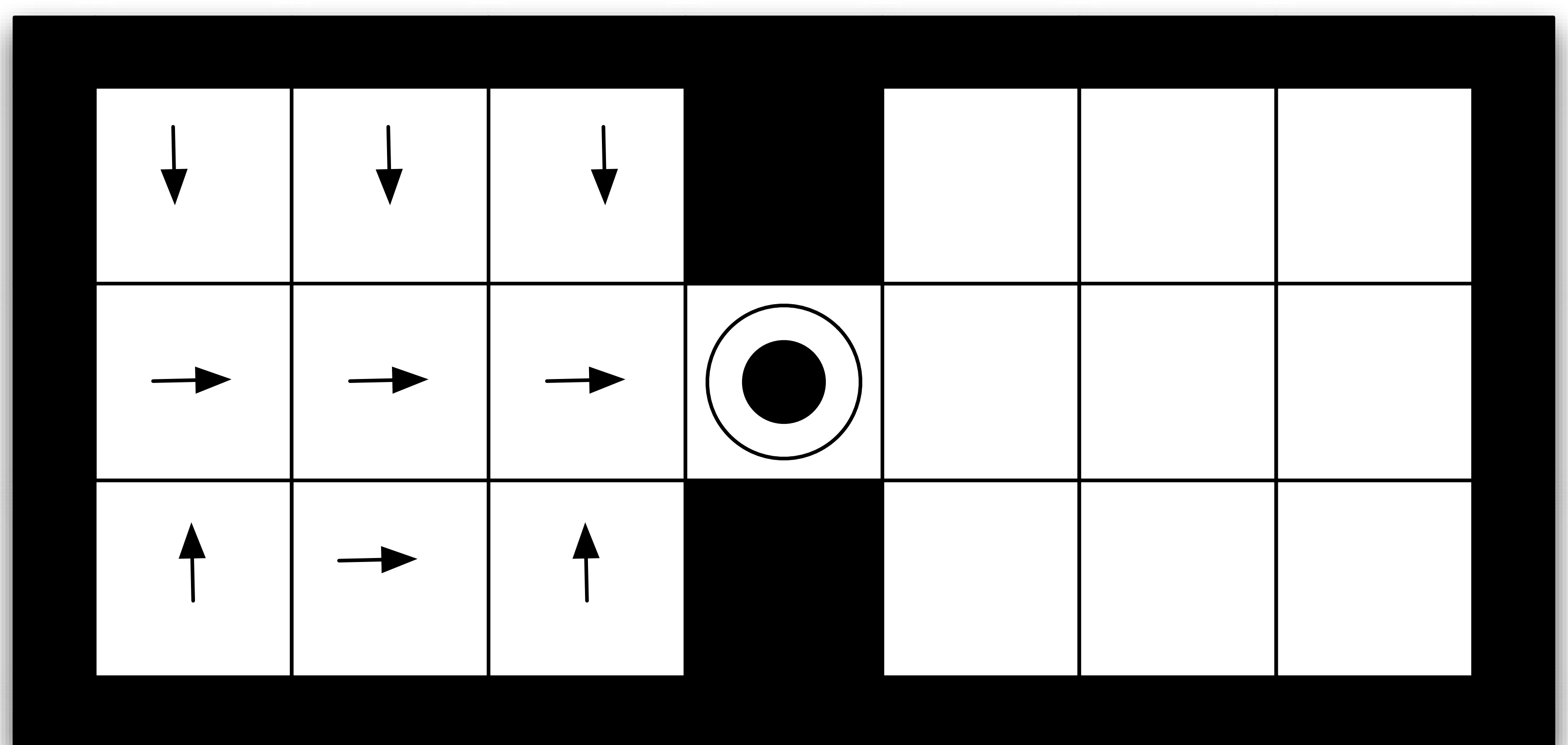
# The Options Framework

**Formal model of a skill.**

[Sutton, Precup and Singh 1999]

An option *o* is a policy unit:

- Initiation set
- Termination condition
- Option policy



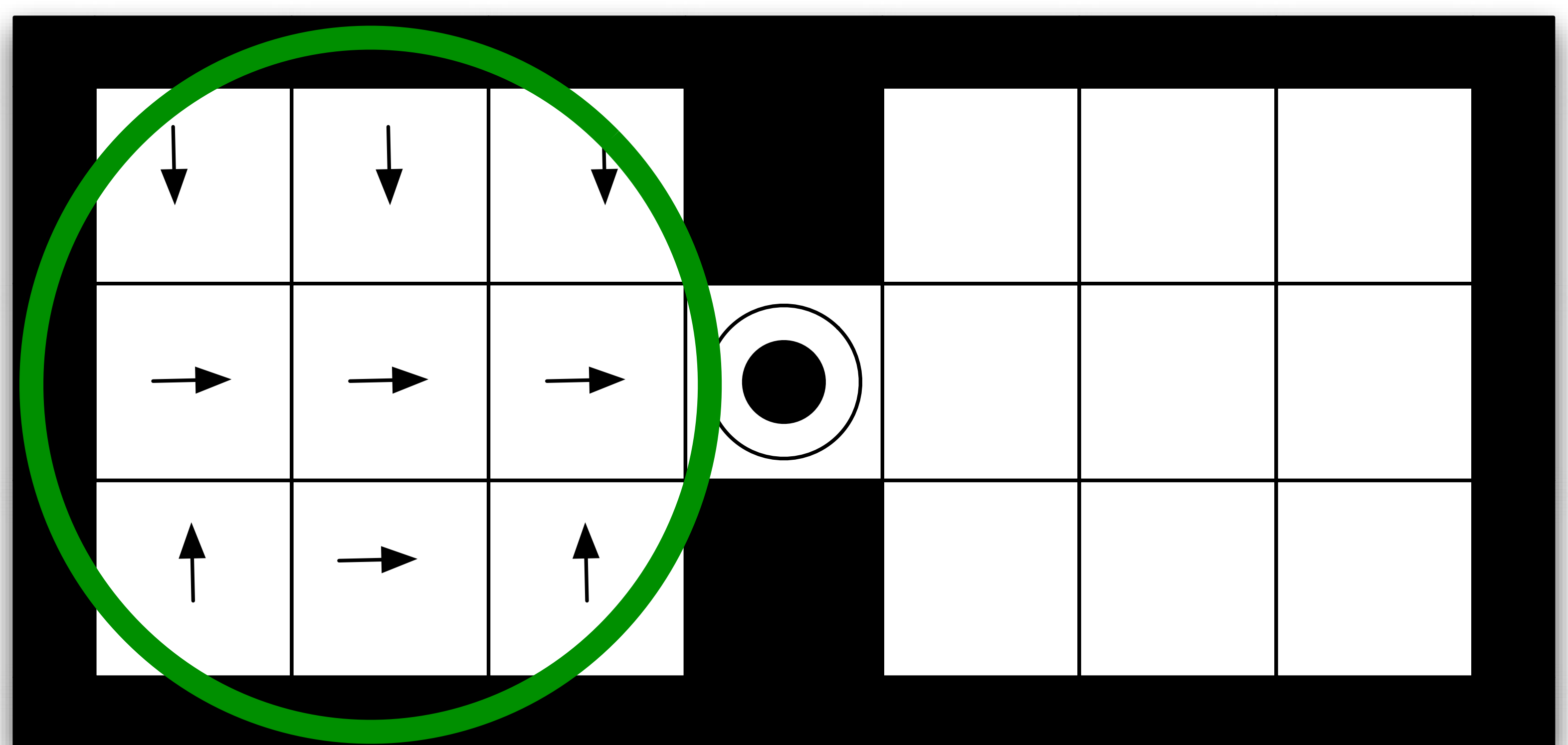
# The Options Framework

**Formal model of a skill.**

[Sutton, Precup and Singh 1999]

An option *o* is a policy unit:

- Initiation set
- Termination condition
- Option policy



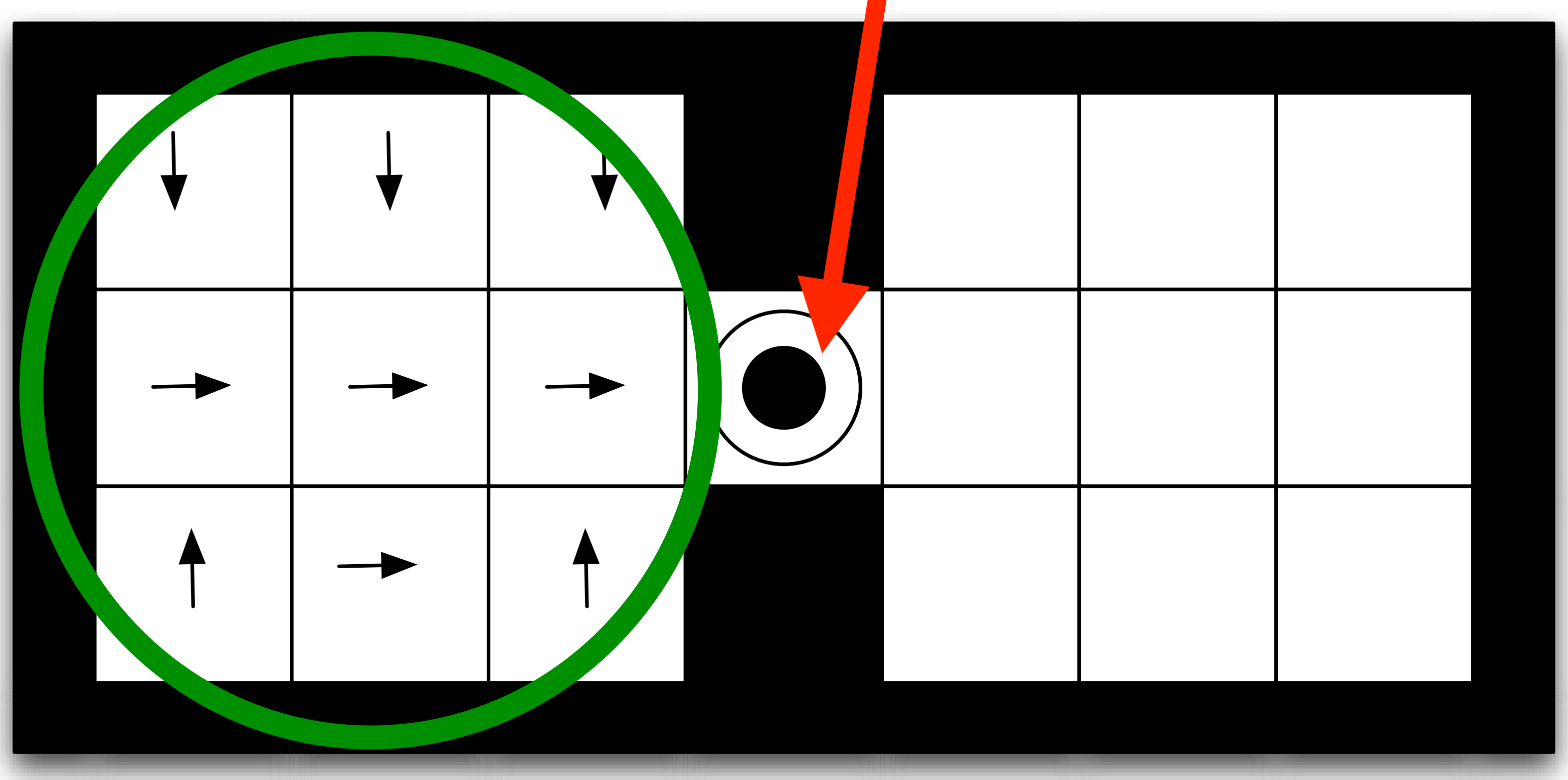
# The Options Framework

**Formal model of a skill.**

[Sutton, Precup and Singh 1999]

An option *o* is a policy unit:

- Initiation set
- Termination condition
- Option policy



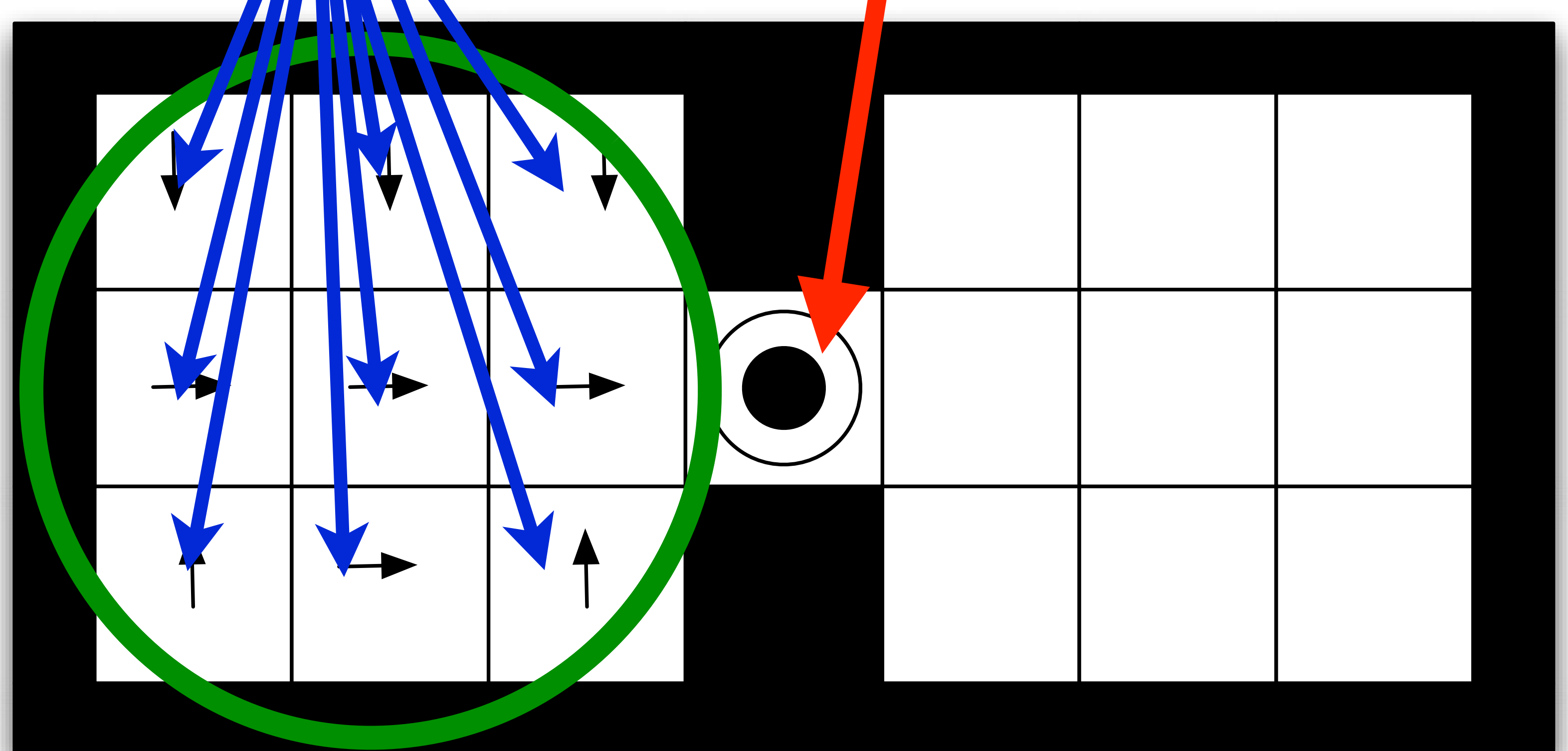
# The Options Framework

**Formal model of a skill.**

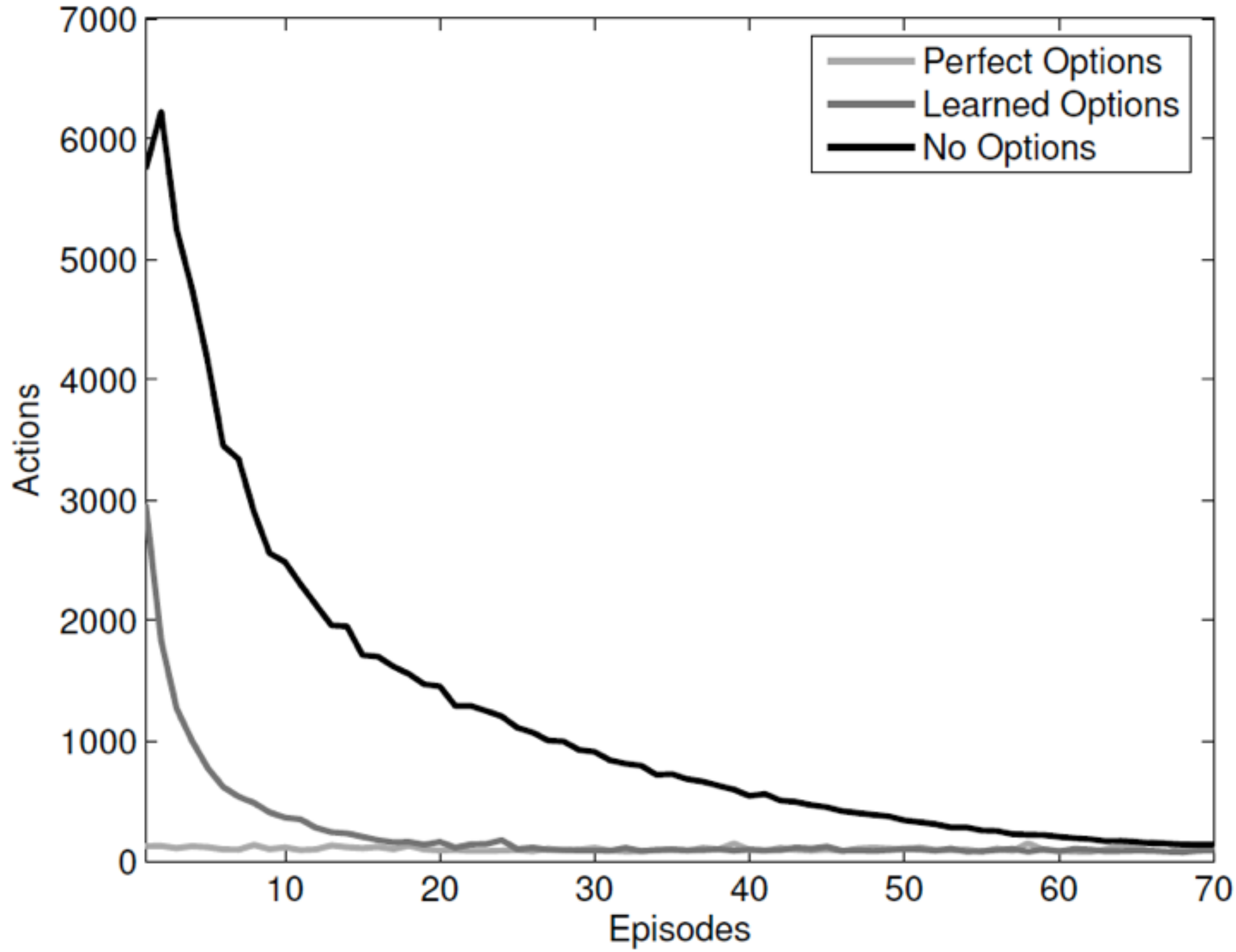
[Sutton, Precup and Singh 1999]

An option *o* is a policy unit:

- Initiation set
- Termination condition
- Option policy

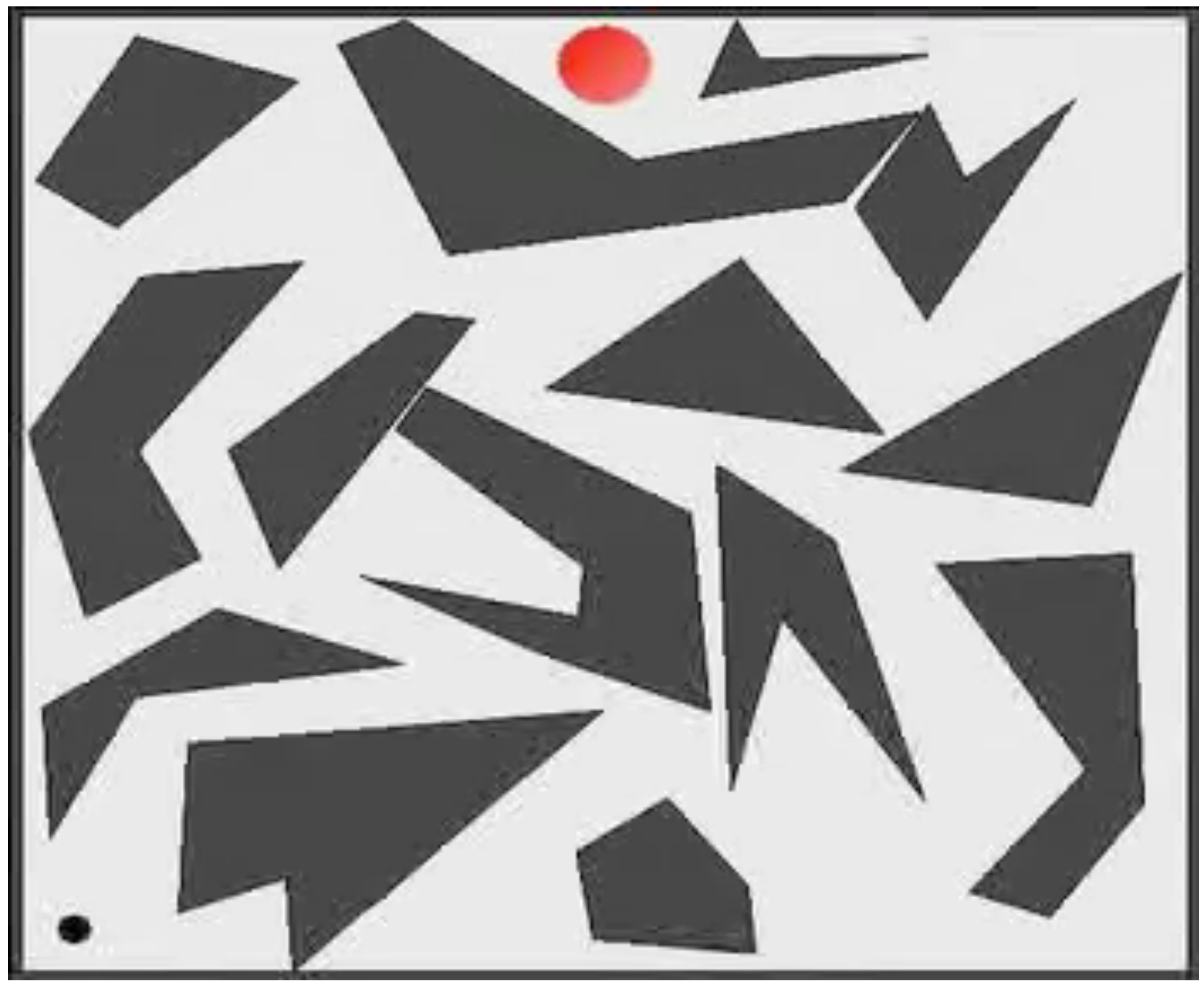


# Option Transfer



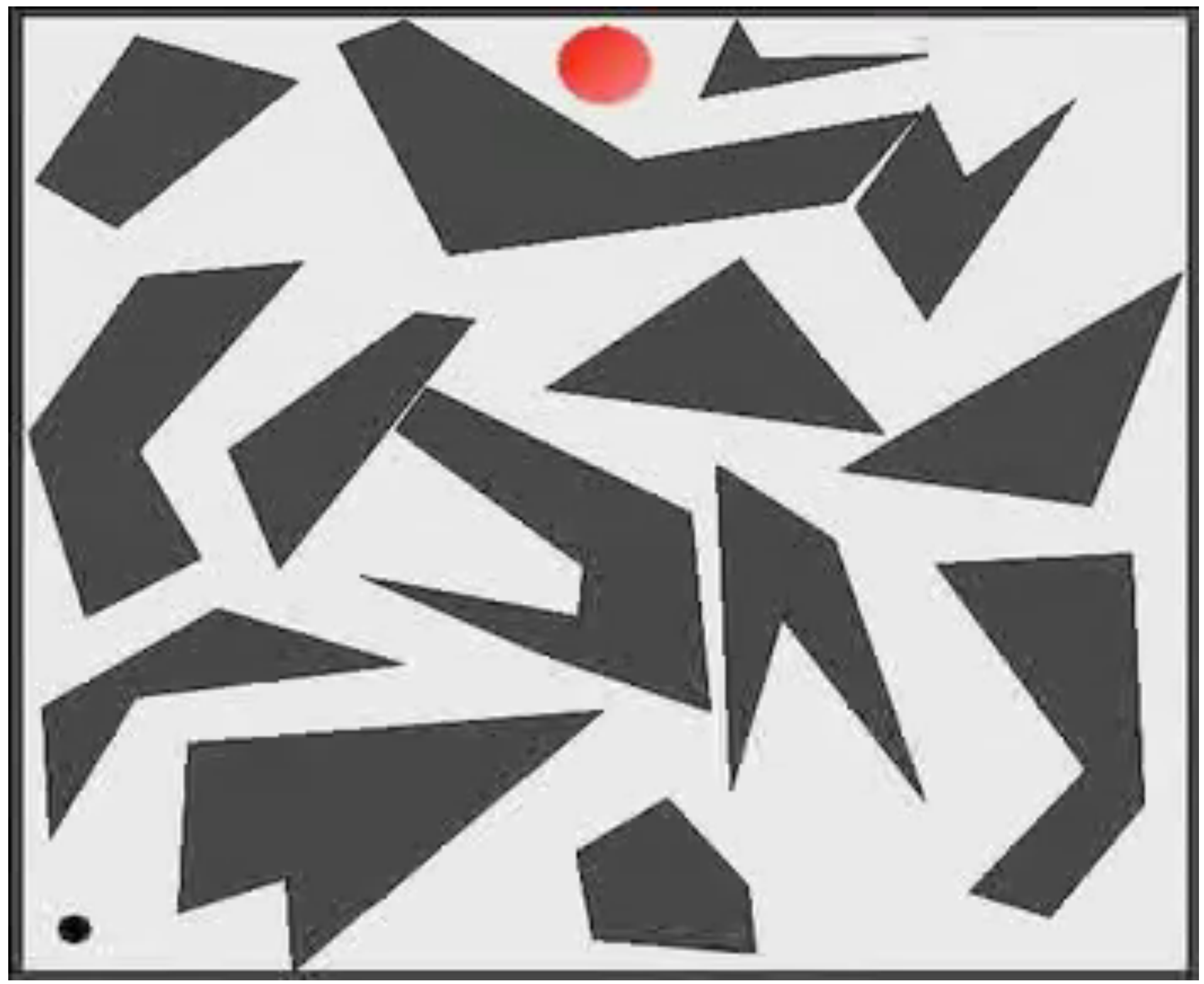

[IJCAI 2007]

# Skill Chaining



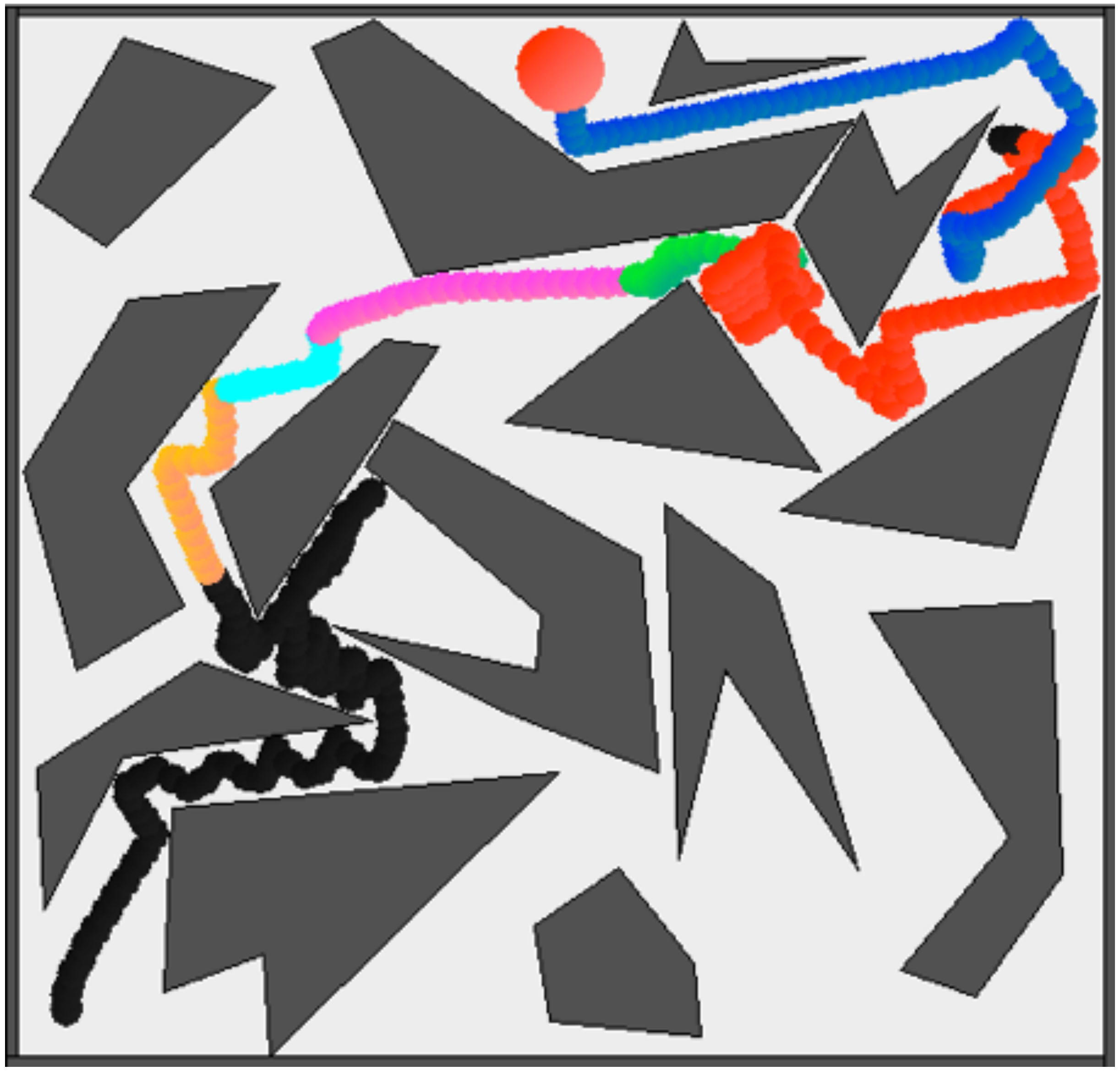
[NIPS 2009]

# Skill Chaining



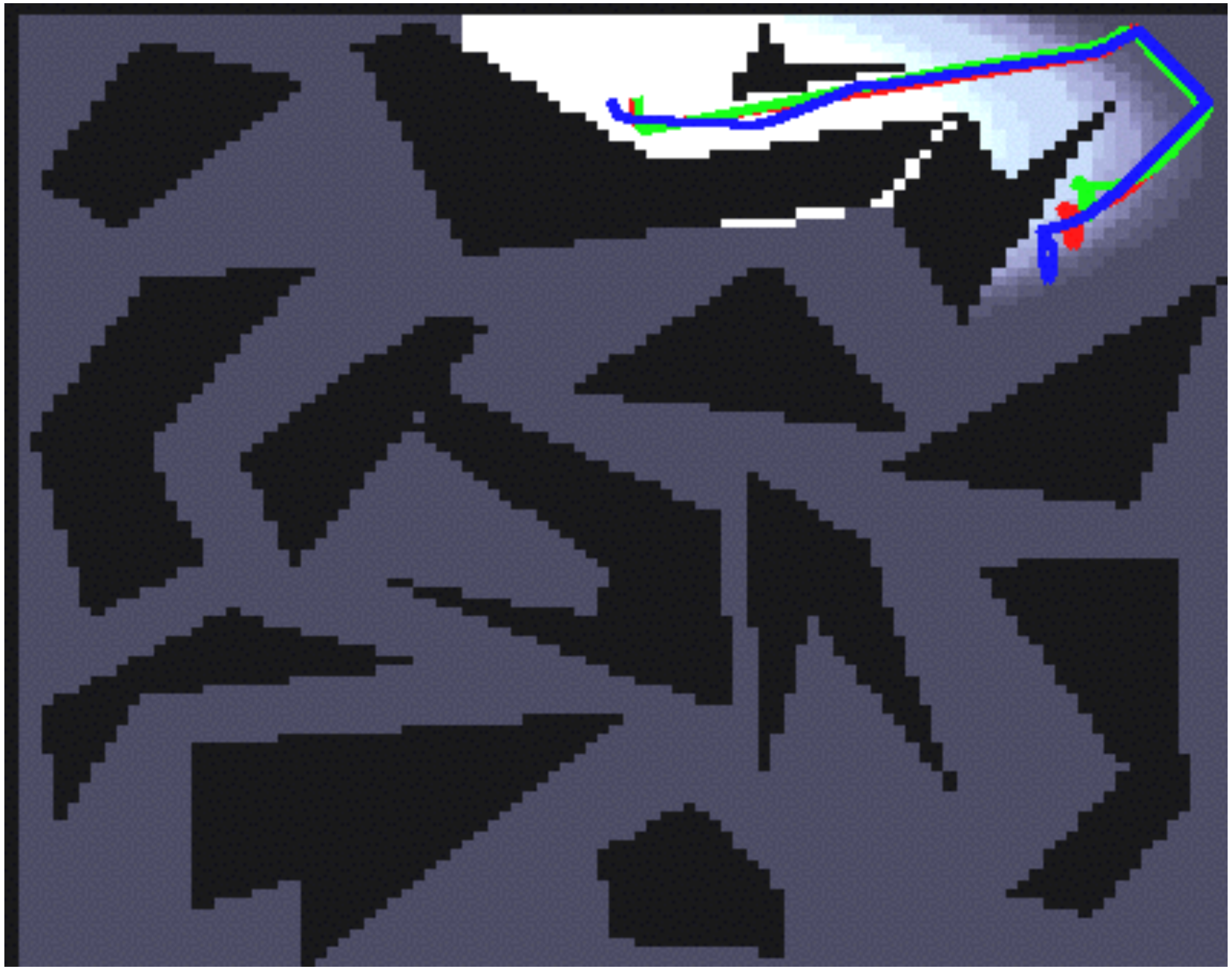
[NIPS 2009]

# Skill Chaining:



[NIPS 2009]

# Skill Chaining: Results



# Key Ideas

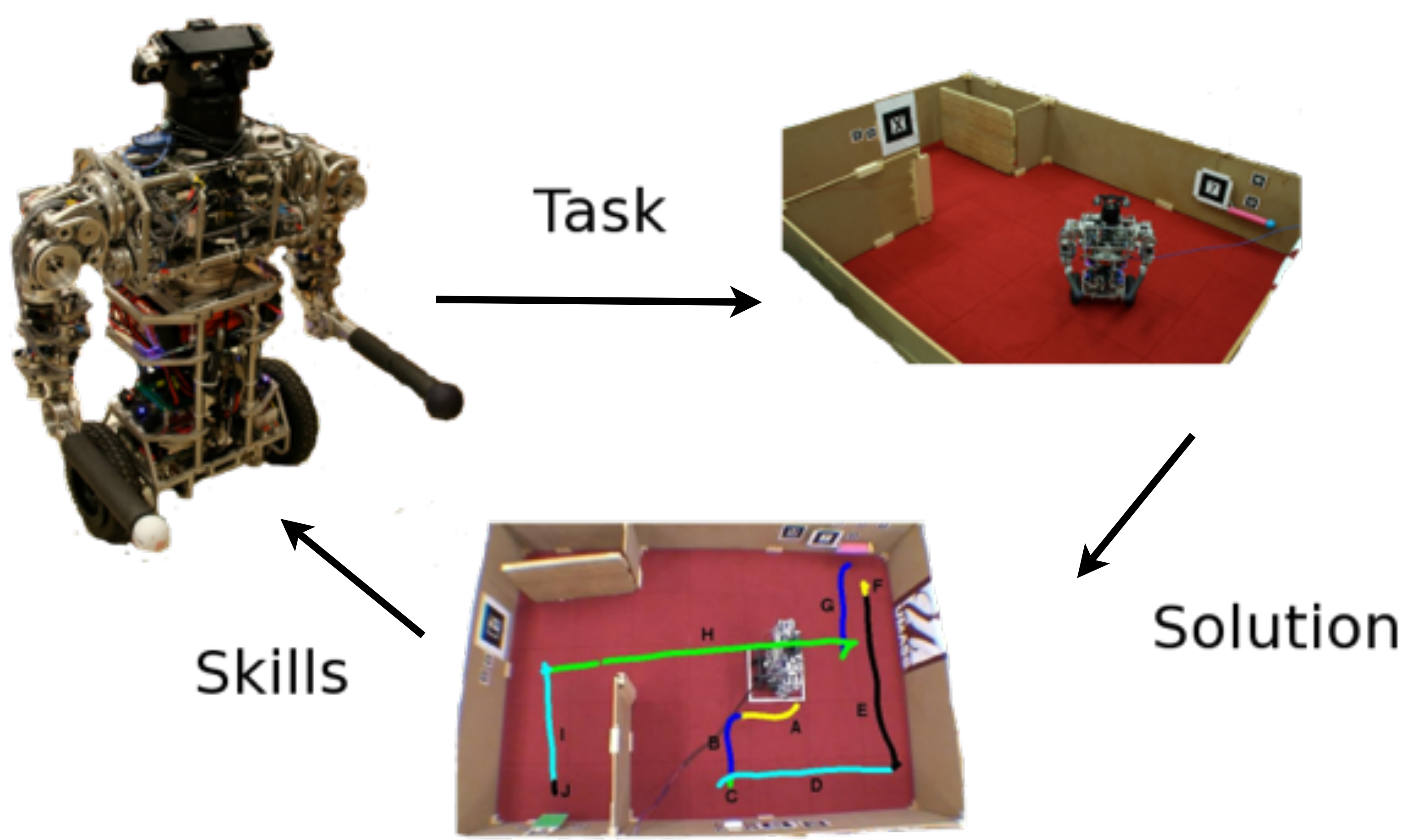
What should options do?

Solway et al. [2014] (following Simsek and Barto [2009]):

- Agent faces distribution over future problems.
- Try to maximize performance averaged over distribution.
- Reasonable to use past problems as sample.

# Skill Acquisition

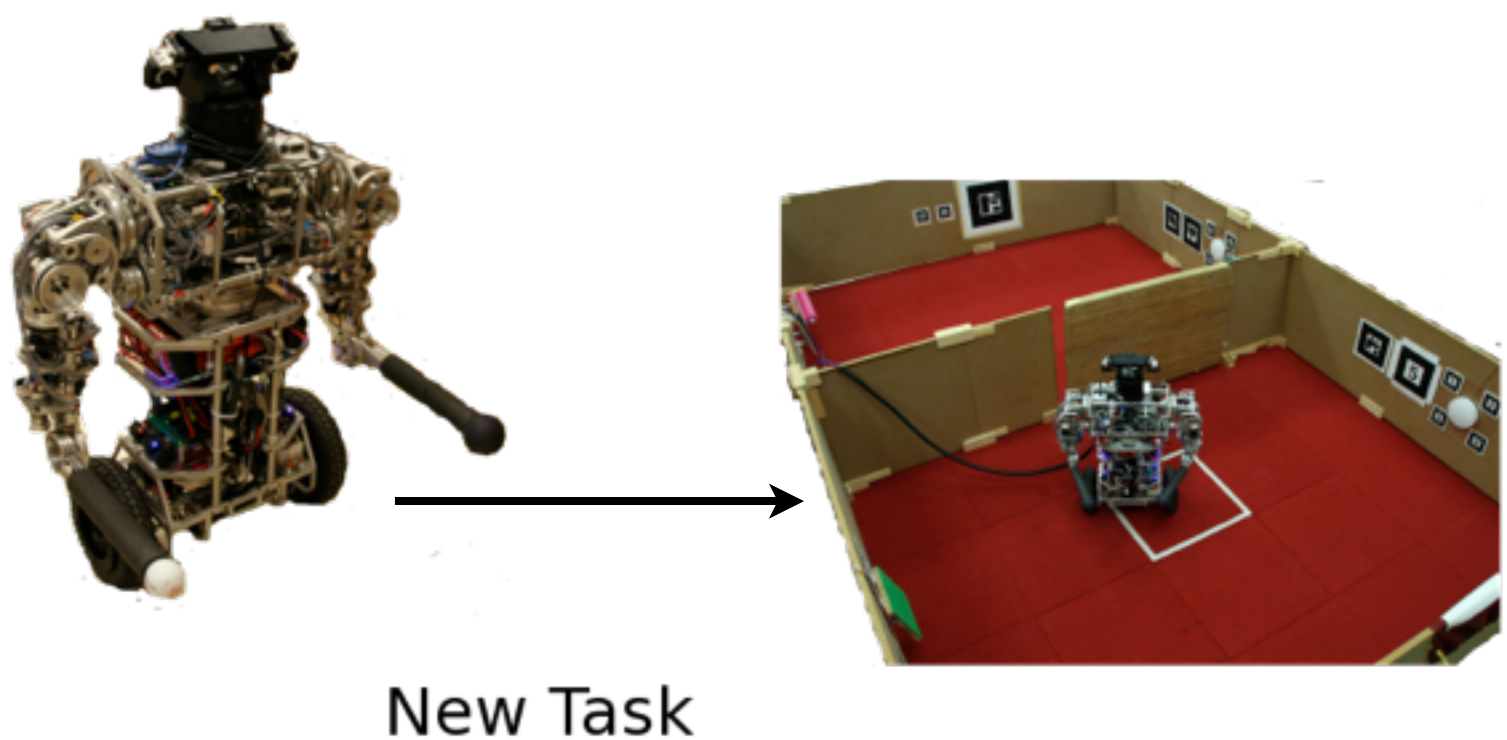
- A robot learning to solve a task
- Extracting skills from solution
- Deploying them in a new task




*[AAAI 2011]*

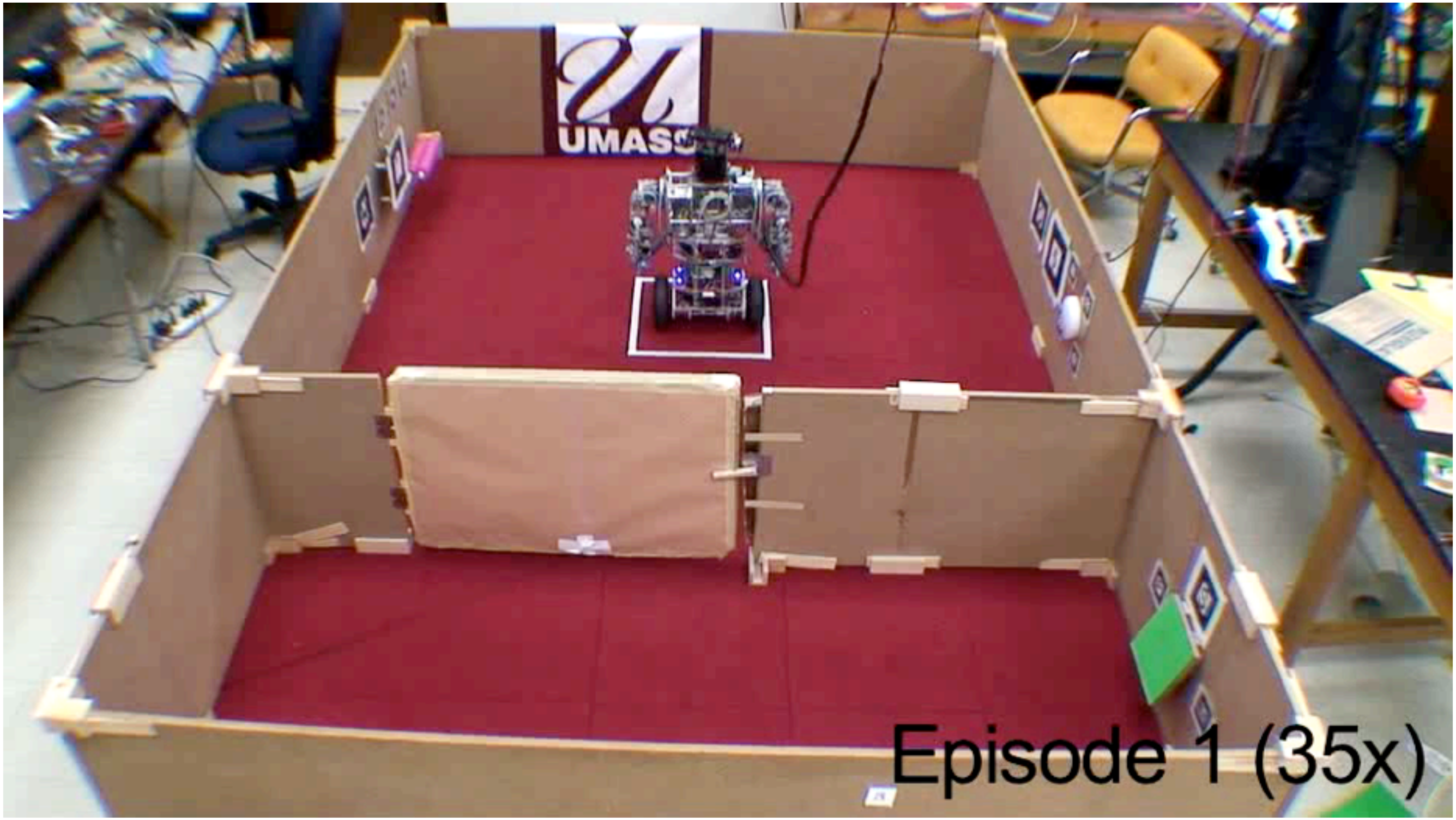
# Skill Acquisition

- A robot learning to solve a task
- Extracting skills from solution
- Deploying them in a new task

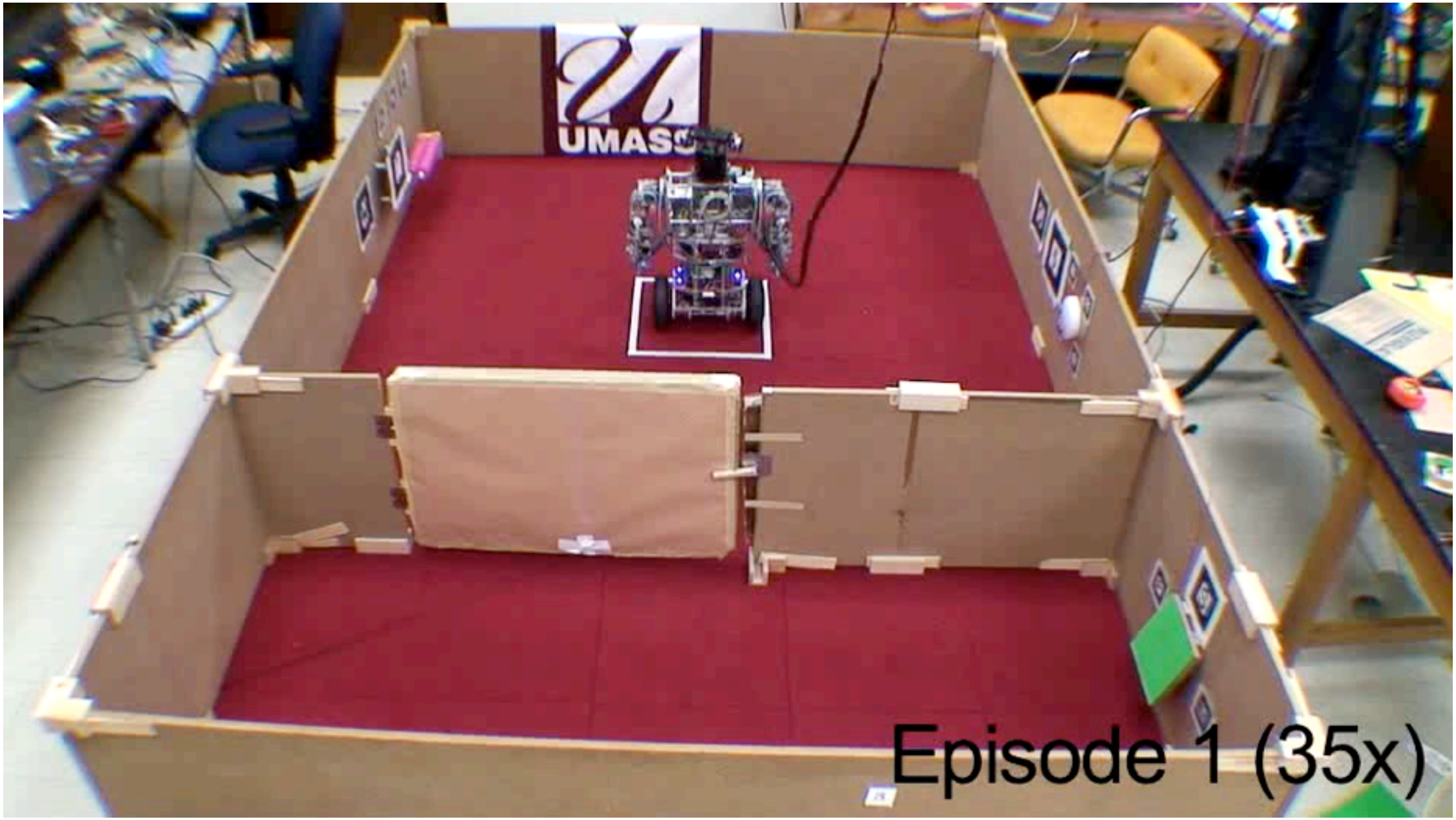



*[AAAI 2011]*

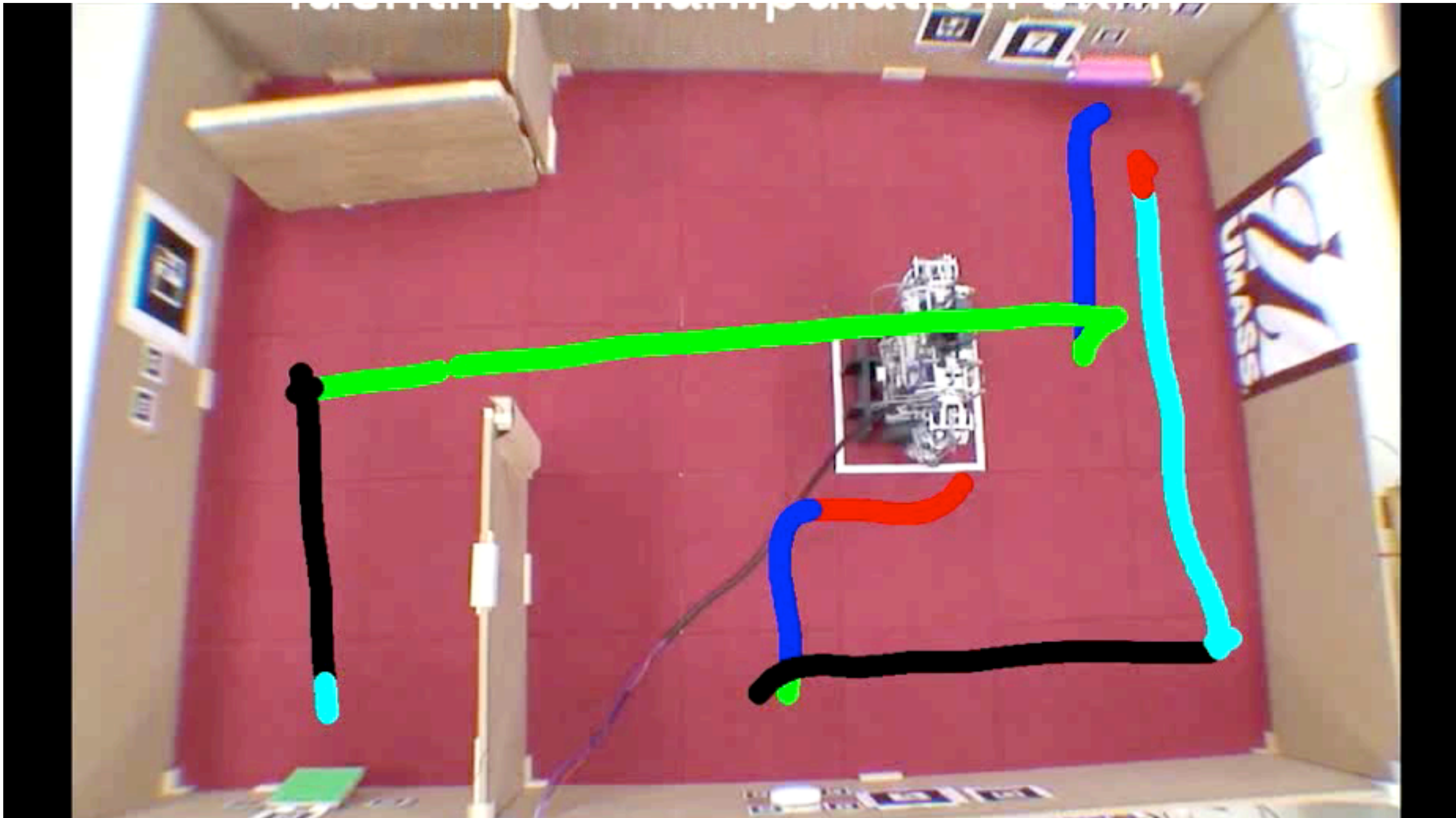
# Training Room



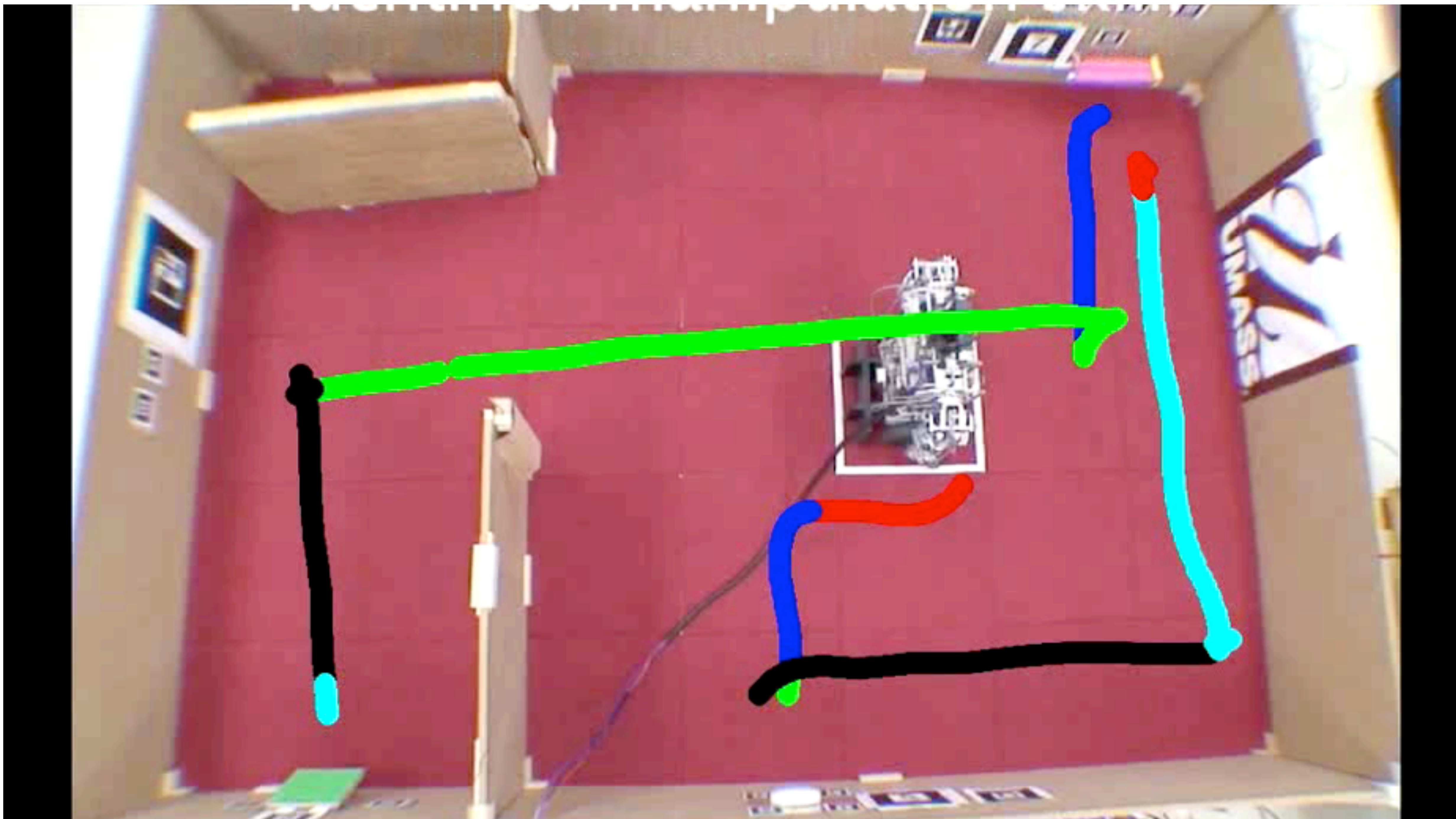
# Training Room



# Acquired Skills



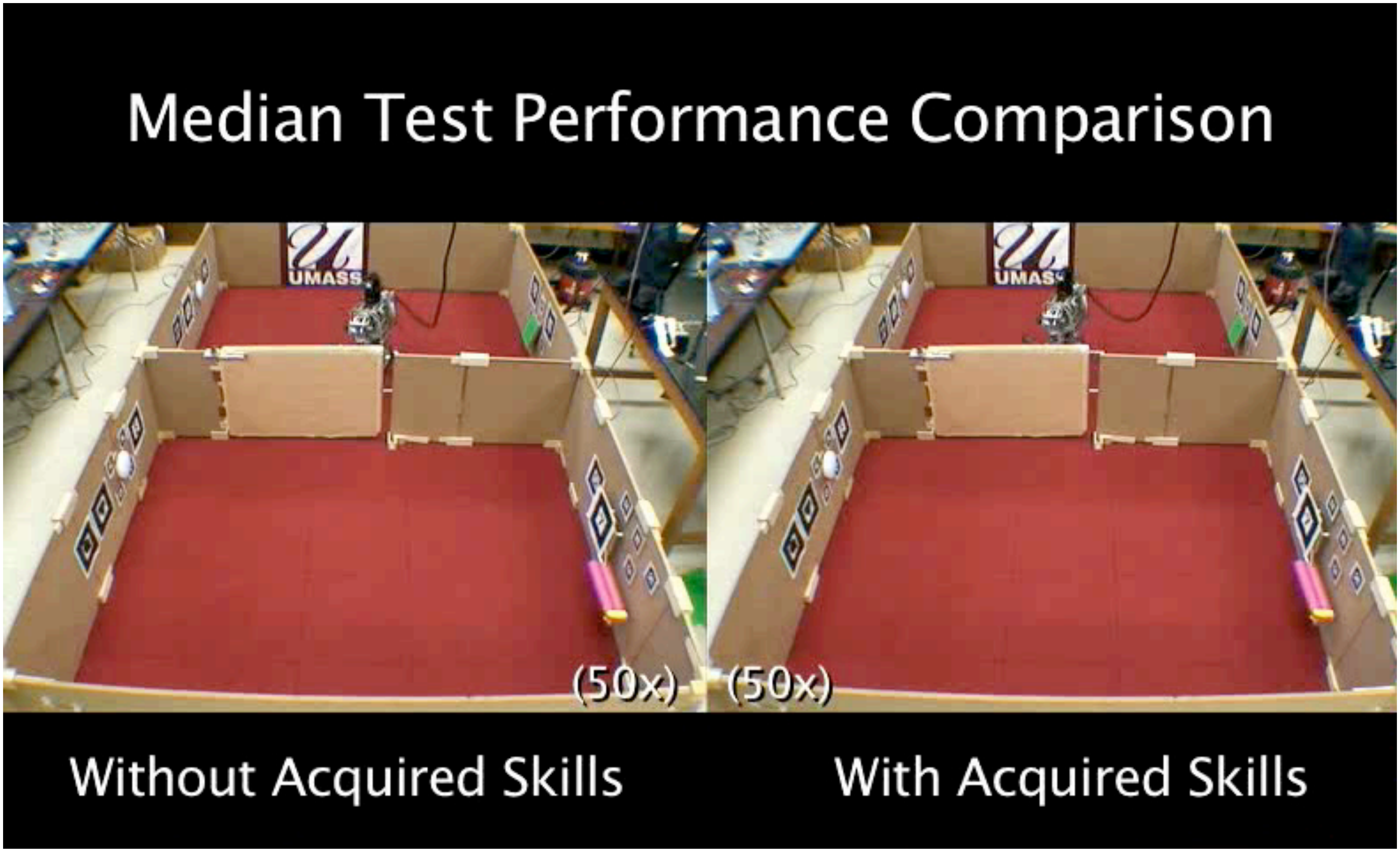
# Acquired Skills



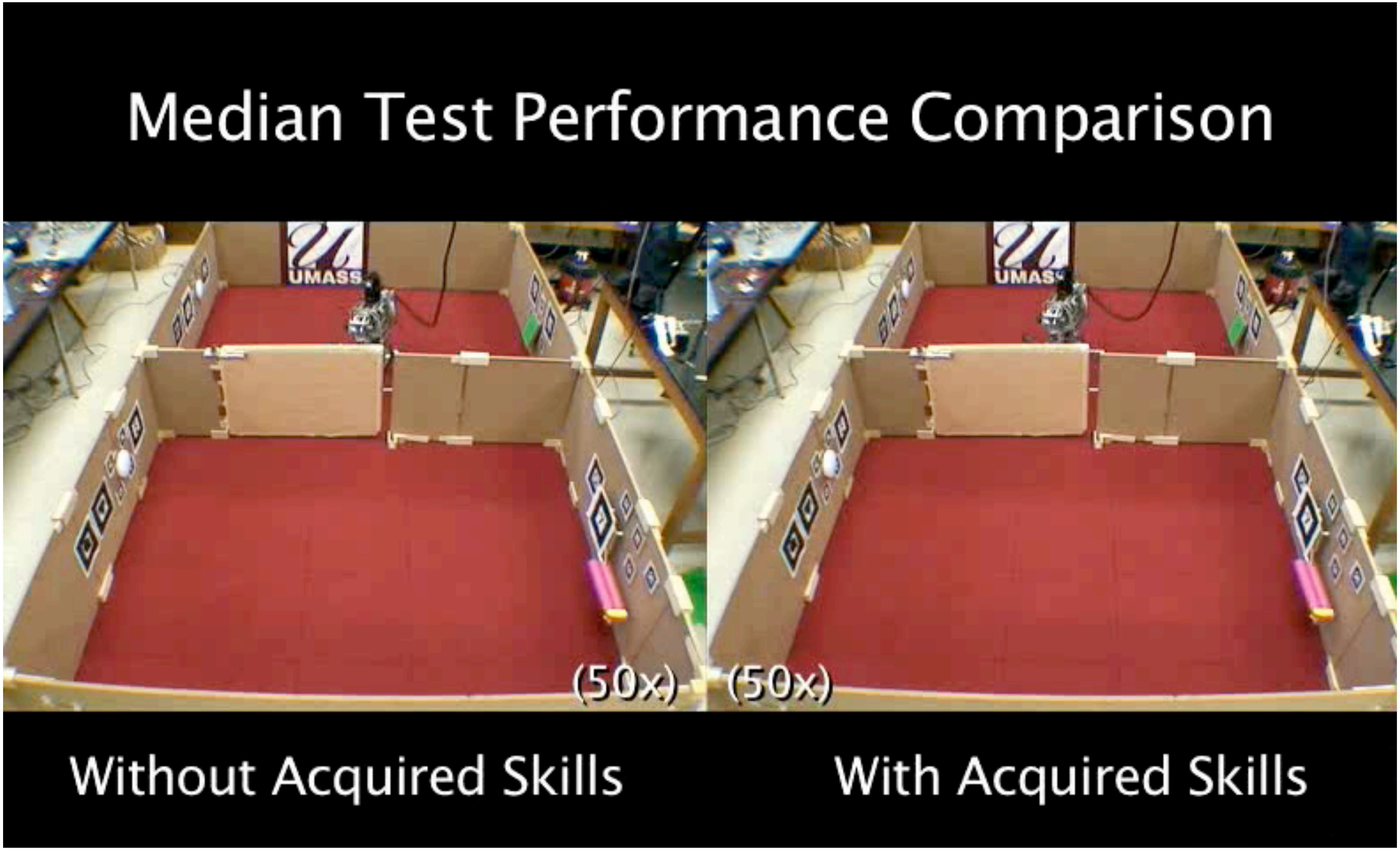
# The Test Room

# The Test Room

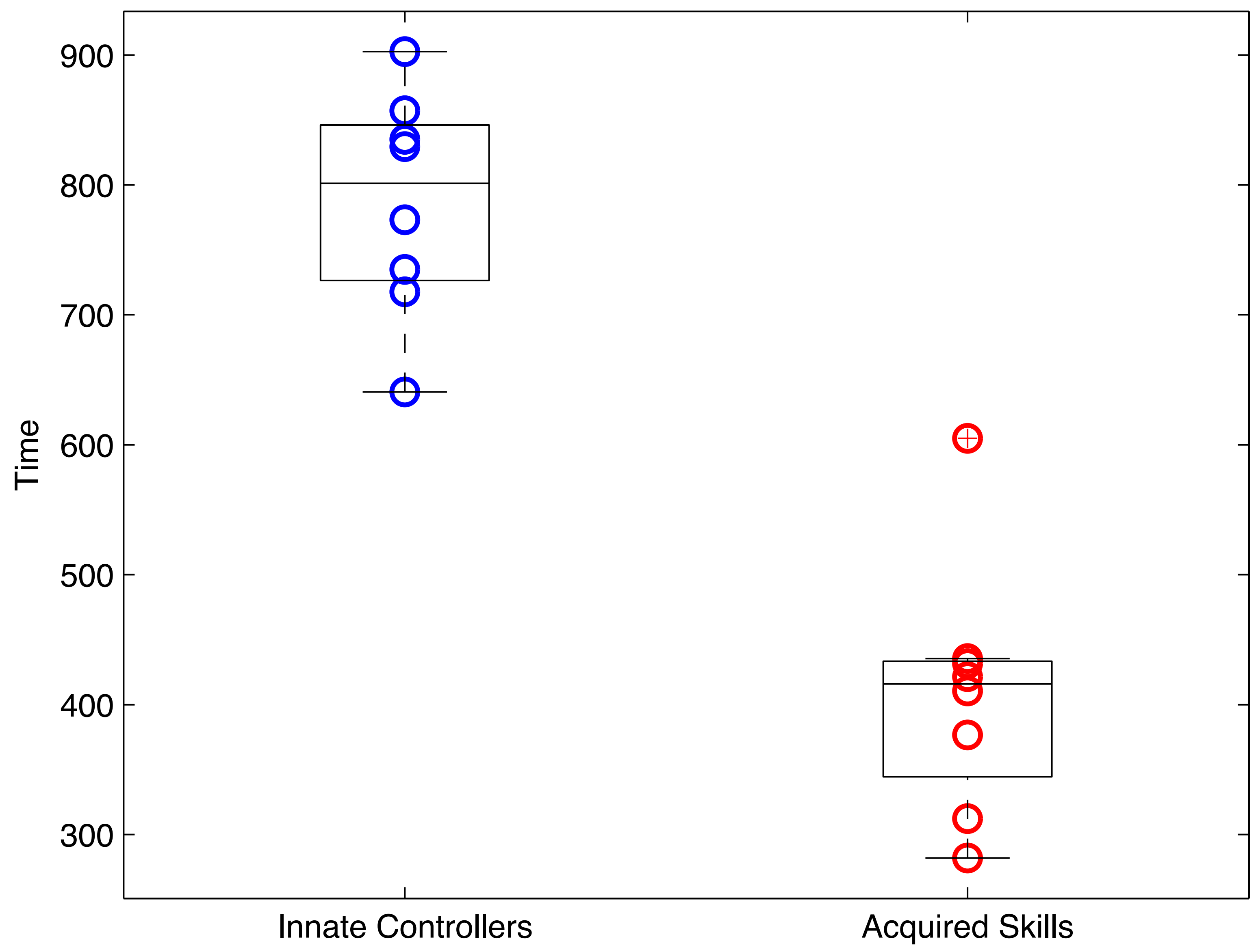
# The Test Room



# The Test Room



# The Test Room



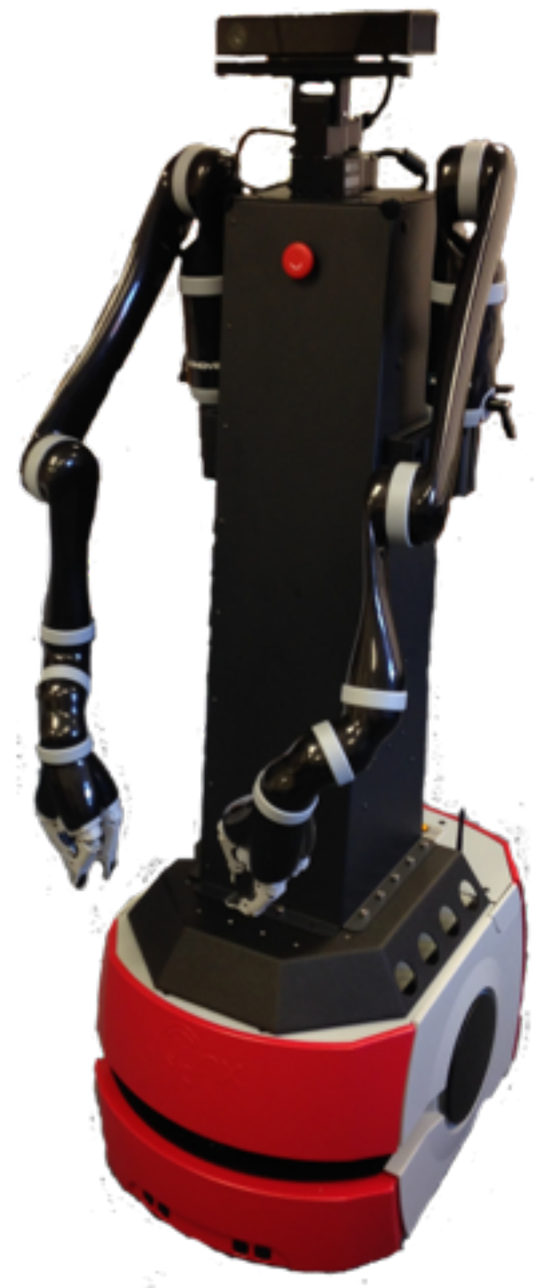

*[AAAI 2011]*

# Summary

Scaled skill acquisition to real robots:

- Skills extracted because they are useful
- Suitable for further learning (individually)
- Suitable for deployment in new problems

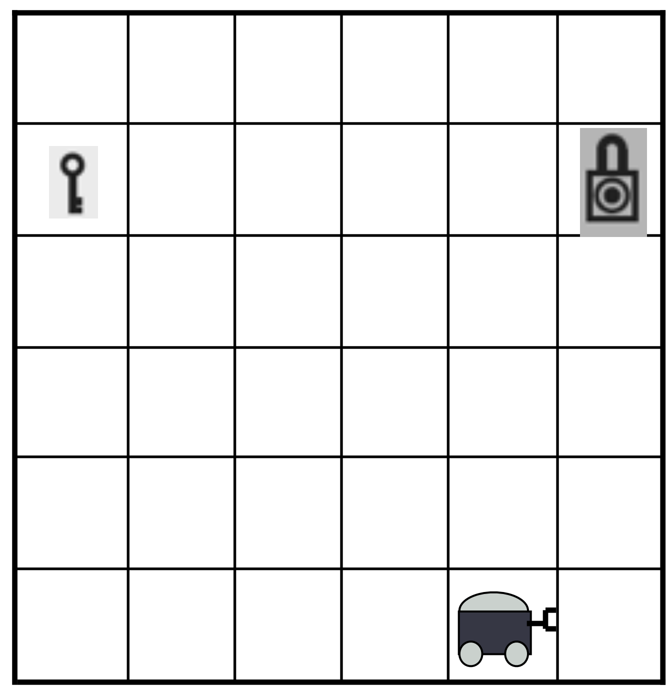
*Acquired skills can improve a robot's problem-solving abilities.*



# Skill-Generated Representations

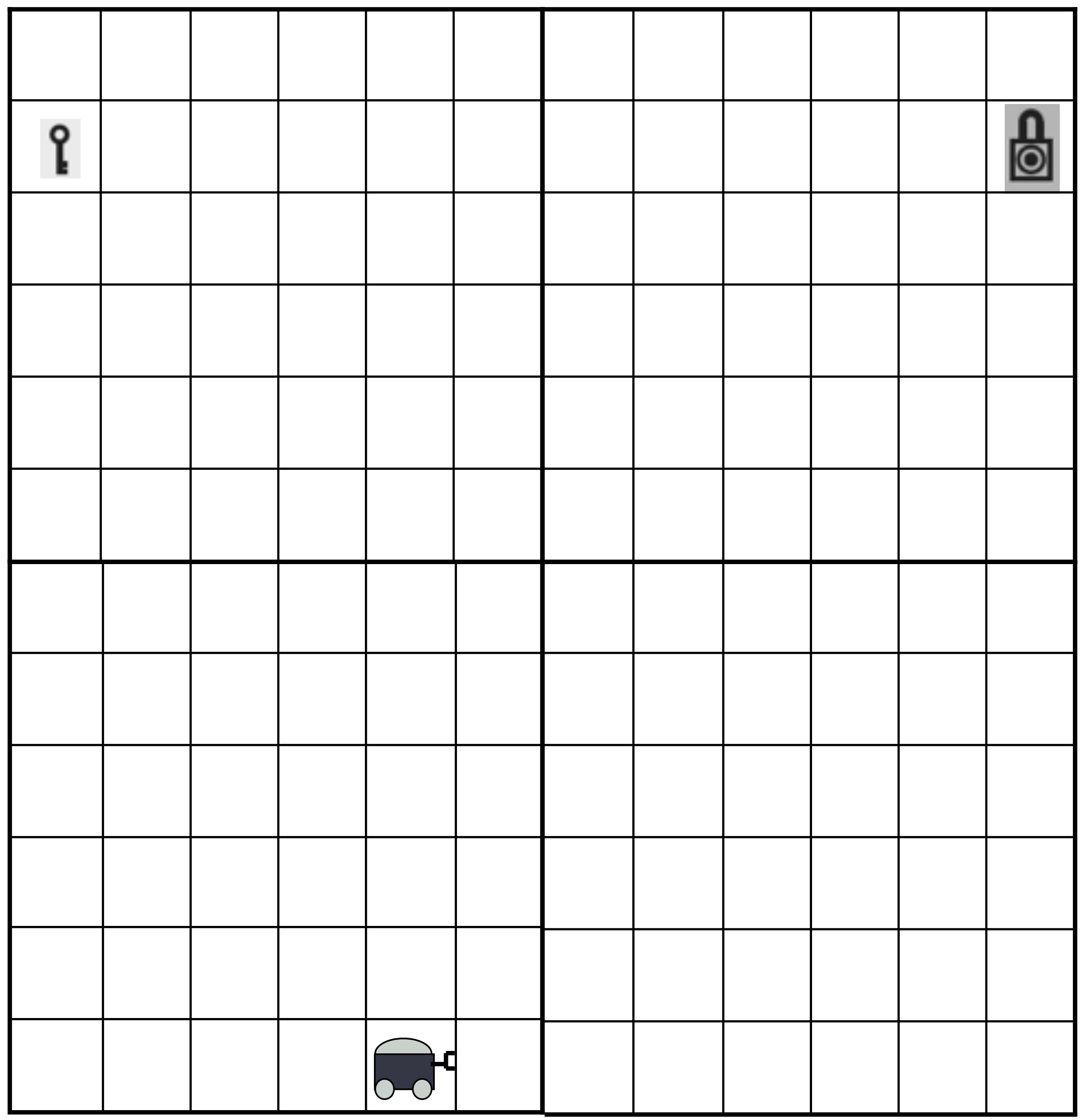
# Abstraction with Options

Problem difficulty shouldn't depend on low-level state space.



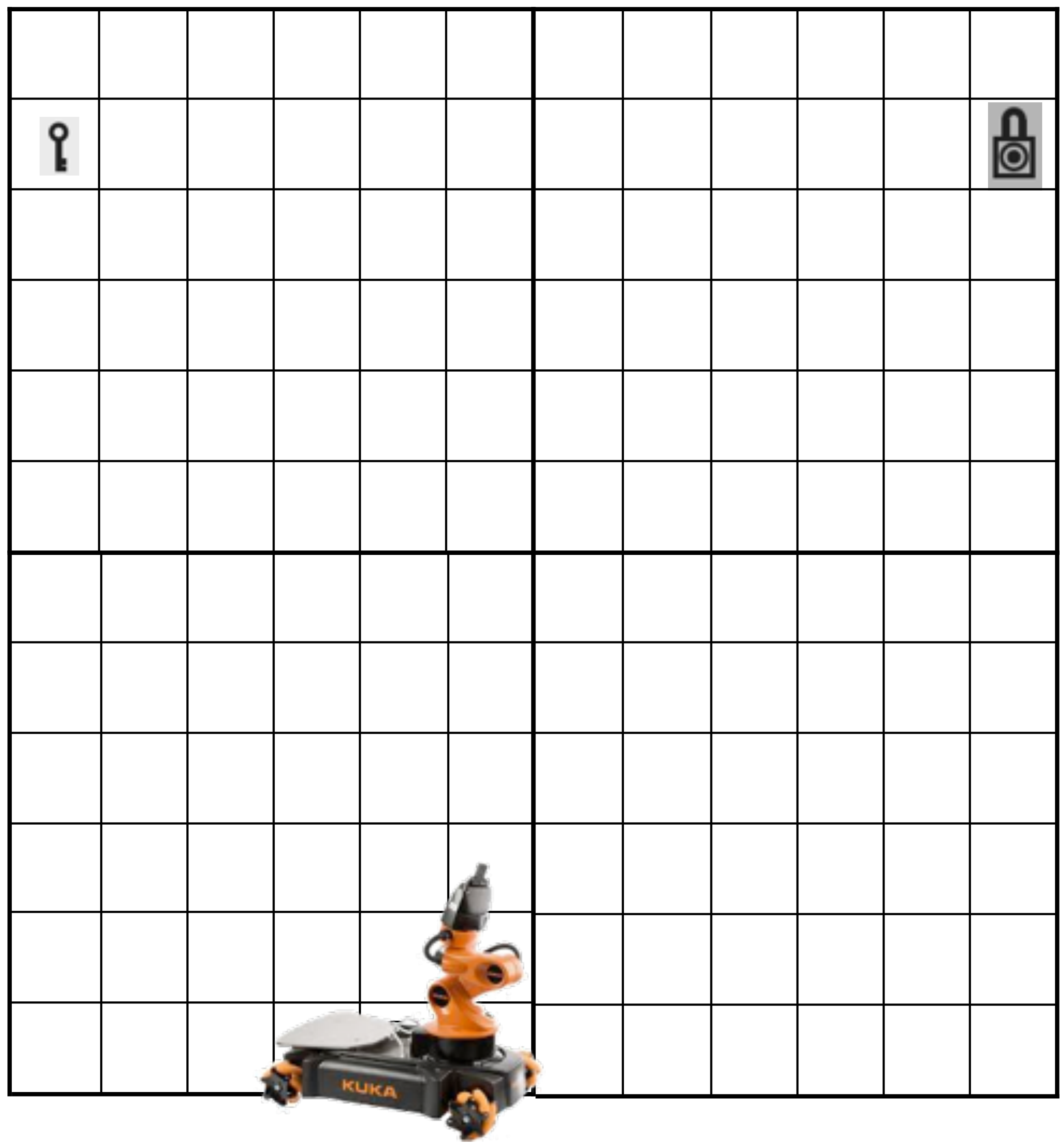
# Abstraction with Options

Problem difficulty shouldn't depend on low-level state space.



# Abstraction with Options

Problem difficulty shouldn't depend on low-level state space.



# Skills Cannot Be The Whole Story

**Representation Acquisition:**

- How should an agent's representations change as it acquires new skills?

# Skills Cannot Be The Whole Story



**Representation Acquisition:**

- How should an agent's representations change as it acquires new skills?

More precisely:

- Assume we have skills (SMDP).
- Can we *automatically derive* an appropriate *abstract* representation for planning with those skills?
- SMDP to more abstract MDP.

# Results

**The answer is yes!**
*We can write down the <u>right</u> abstract representation for planning using any set of skills.*

**<u>But</u>** the representation depends on properties of the skills.

[AAAI 2014]

# Key Idea

Formalize the fundamental question a representation needs to answer, and then *explicitly construct* it so that it can answer that question.

What is the fundamental question of probabilistic planning?

[AAAI 2014, IJCAI 2015]

# Key Idea

Formalize the fundamental question a representation needs to answer, and then *explicitly construct* it so that it can answer that question.

What is the fundamental question of probabilistic planning?

Given a state and a sequence of options $\{o_1, o_2, \ldots, o_n\}$:

- What is the probability of being able to execute it?
- What is the expected reward?

[AAAI 2014, IJCAI 2015]

# Symbols for Planning

A plan $p = \{o_1, ..., o_n\}$ from a state distribution $Z$ is a sequence of actions to be executed from a state drawn from $Z$.

Starting from the corridor ...

- GoToDoor
- TurnHandle
- PushDoorOpen
- EnterRoom ...

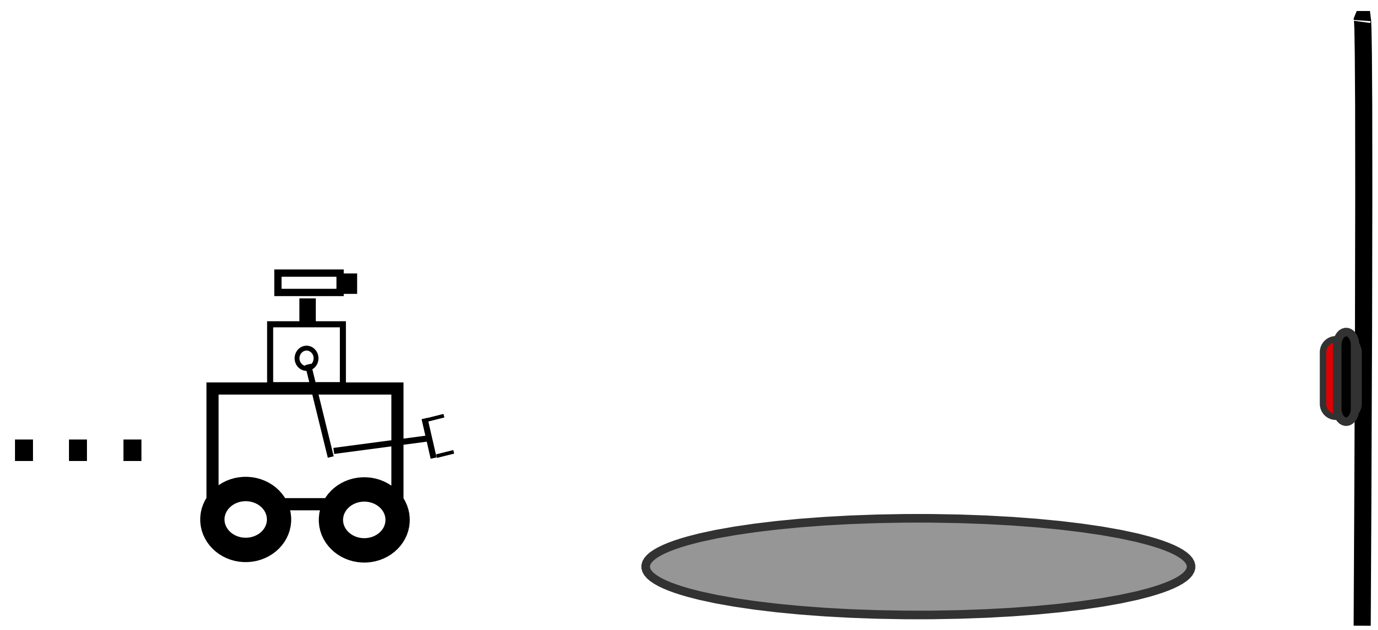
So:

- **Which distributions do we need to determine the feasibility of any plan $p$?**

# Symbols for Planning

We need **one distribution** and one operator per skill.

Initiation distribution:

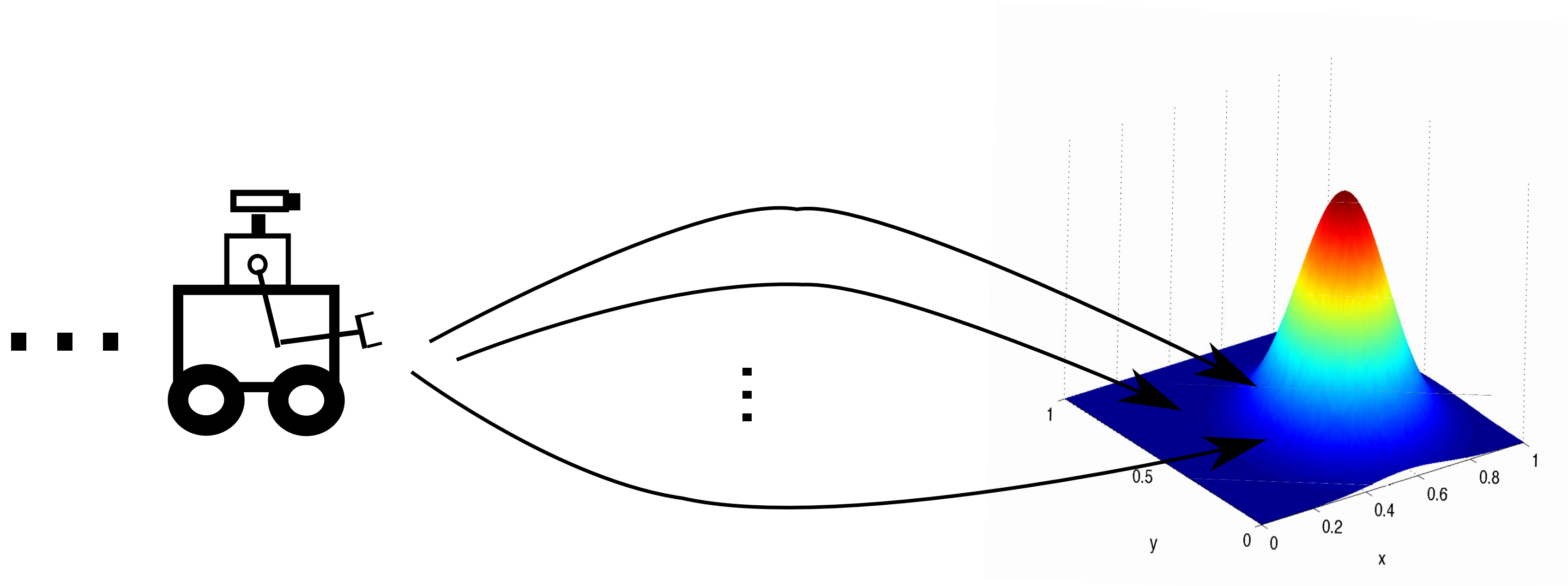


$$P(s \in I_o)$$

# Symbols for Planning

We need one symbol and **one operator** per skill.

Image distribution:



**Definition** *Given a start distribution $Z(S)$ and an option $o$, we define the probabilistic image of $o$ from $Z(S)$ as:*

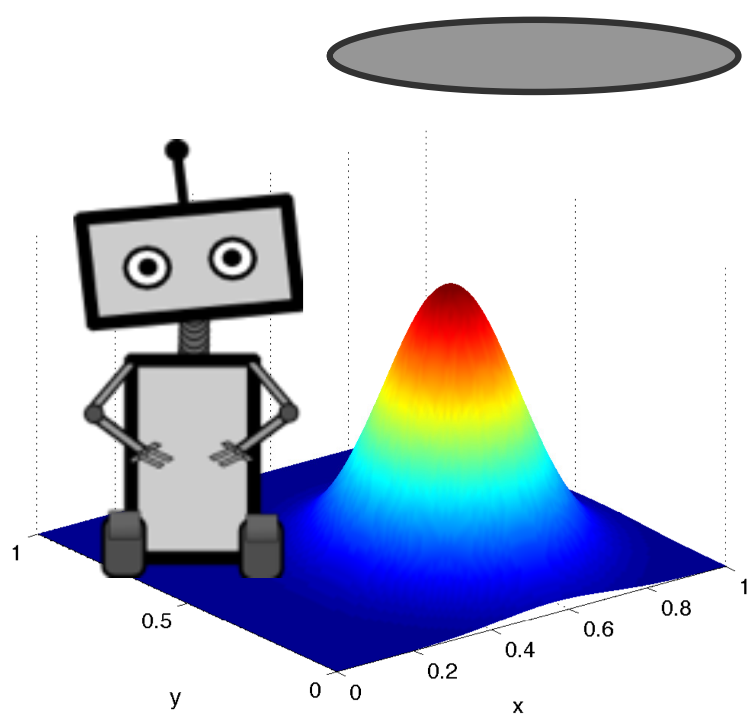
$$Im(o, Z) = \frac{\int_S P(s'|s, o) Z(s) P(I_o|s)\, \mathrm{d}s}{\int_S Z(s) P(I_o|s)\, \mathrm{d}s},$$

*where $P(s'|s, o) = \int P(s', \tau|s, o)\, \mathrm{d}\tau$, since we are not concerned with the time taken to execute $o$.*

# Probabilistic Planning

Must deal with *distributions over states* in the future.

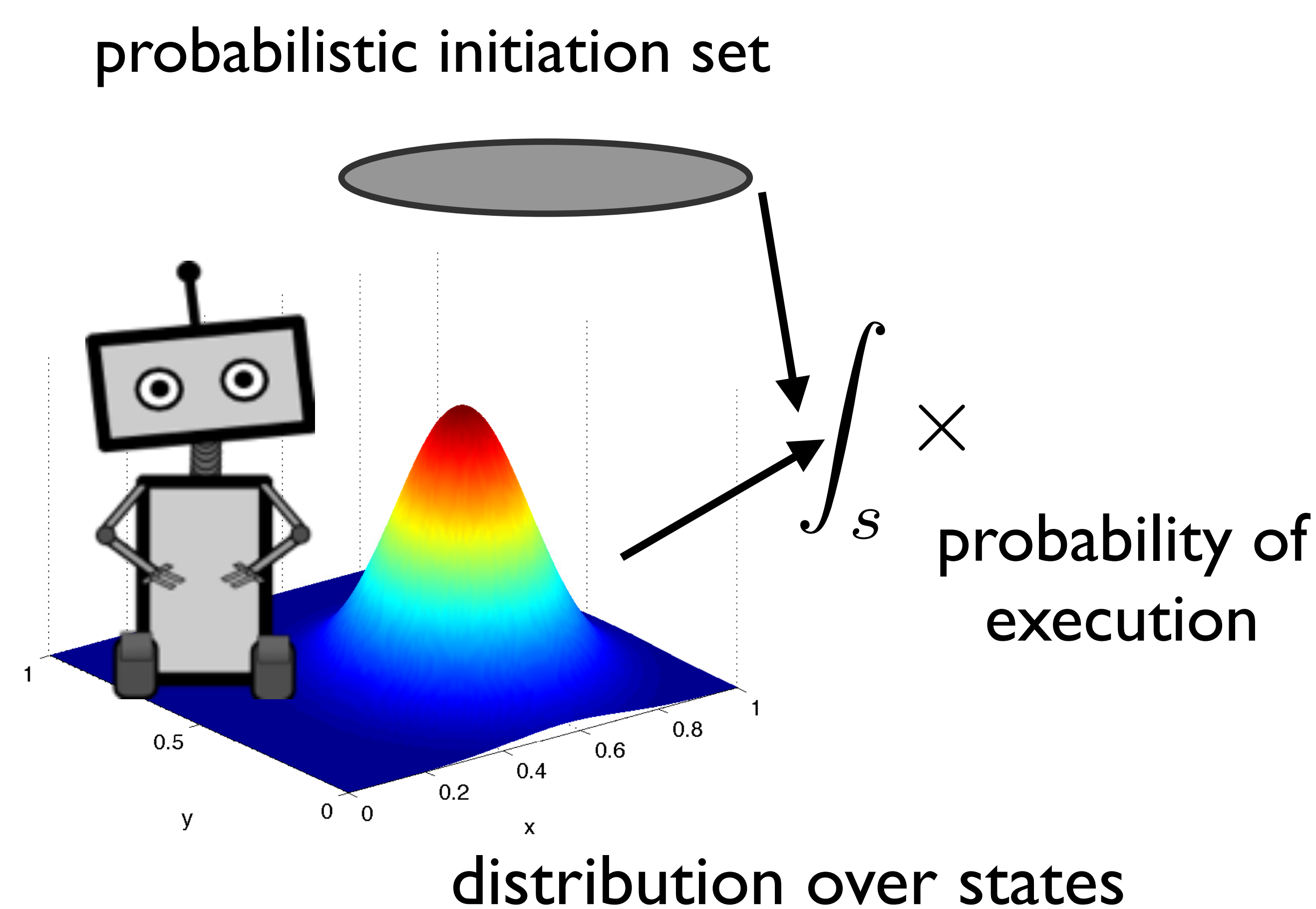
probabilistic initiation set




distribution over states

# Probabilistic Planning

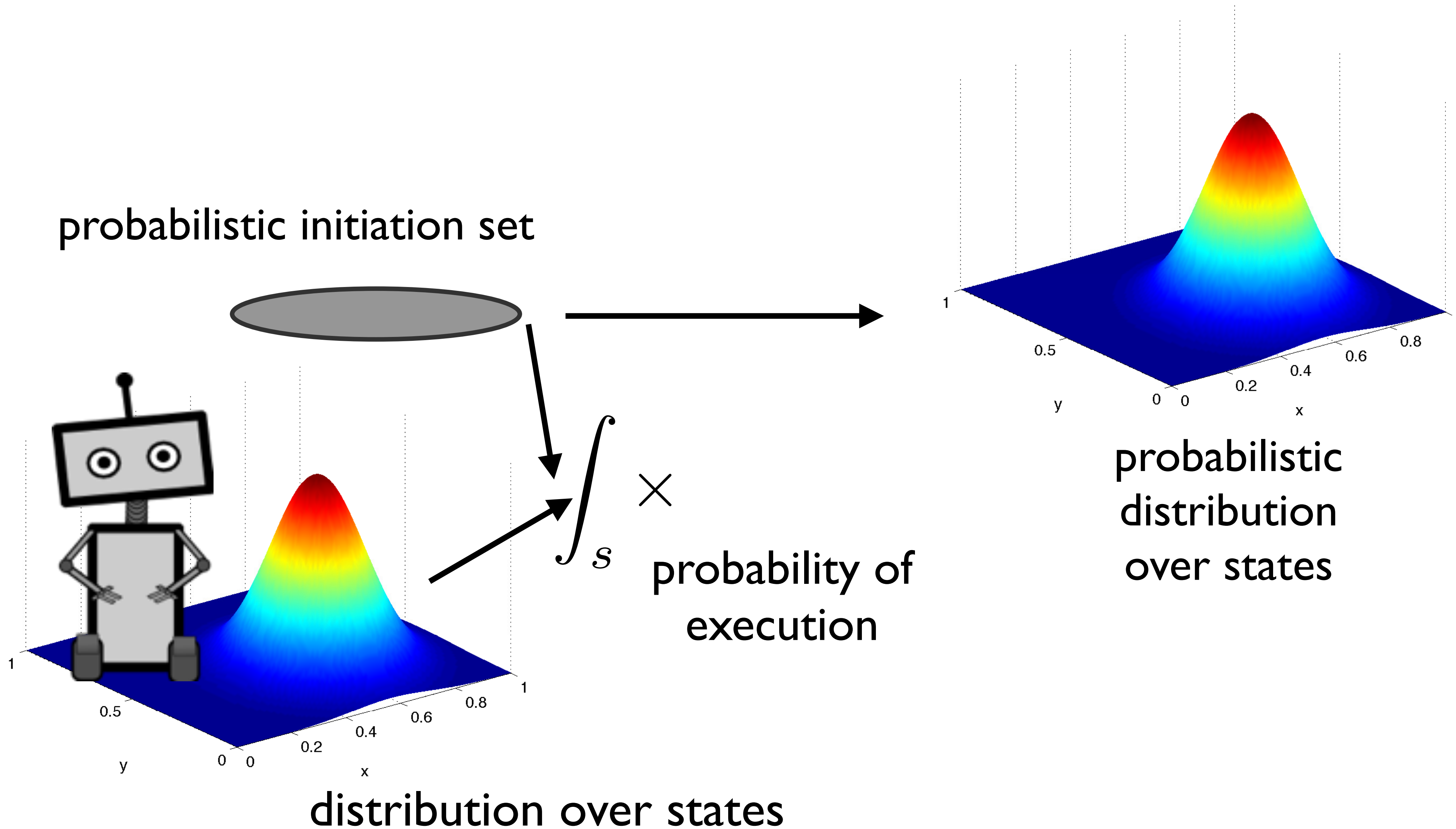
Must deal with *distributions over states* in the future.



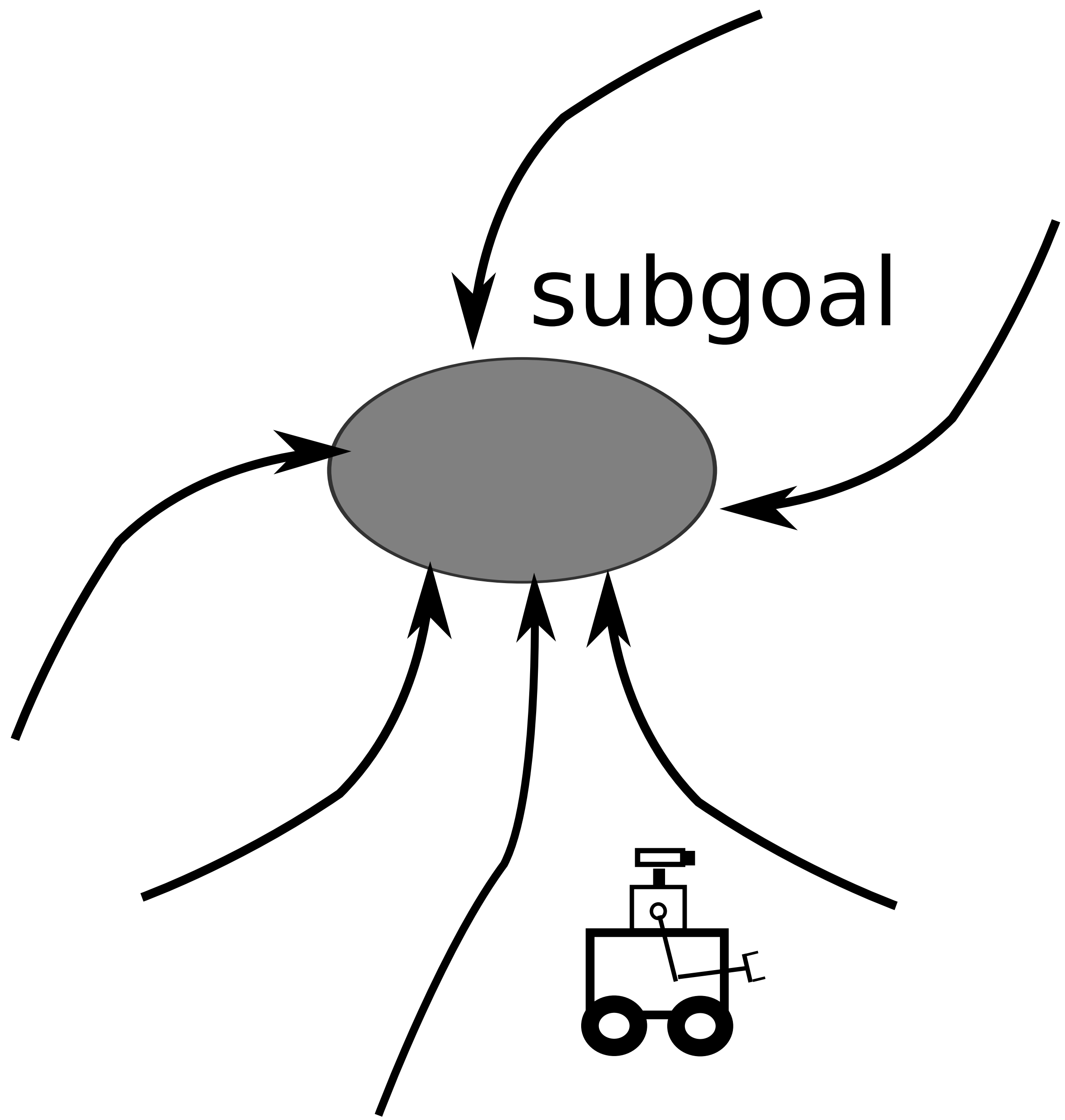
# Probabilistic Planning



Must deal with *distributions over states* in the future.



# Subgoal Options

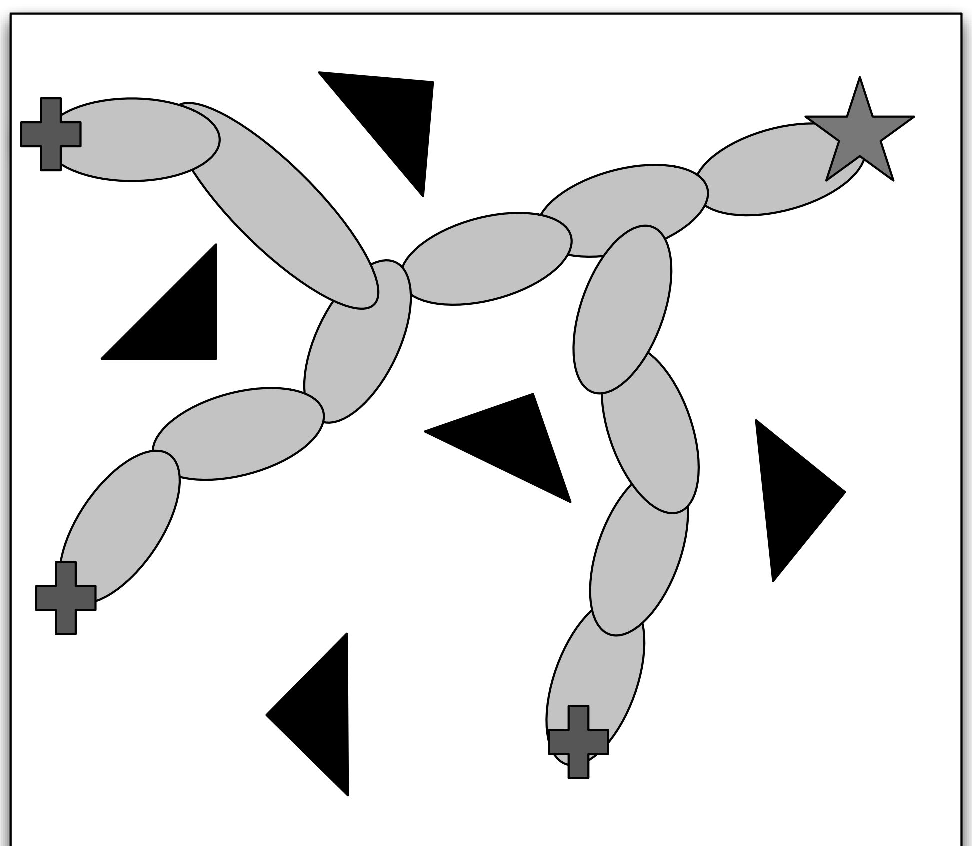



$$P(s'|o_i, s) = P(s'|o_i)$$

# Subgoal Options

Results in a *plan graph*.

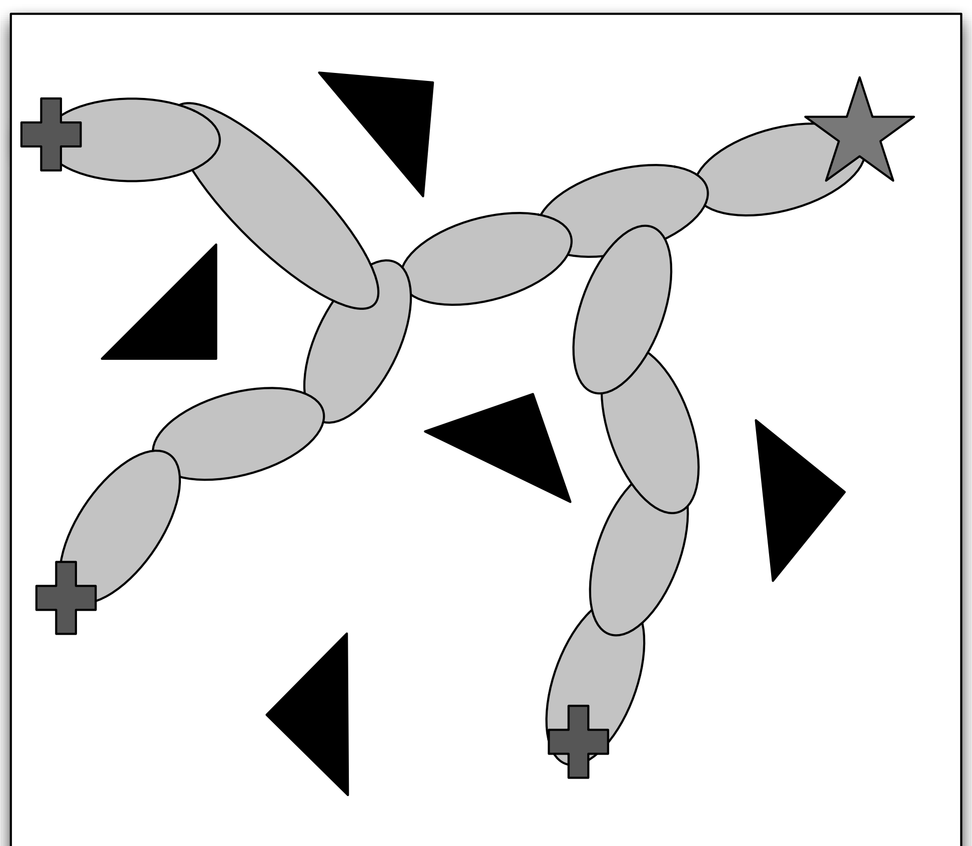
- Node for each option.
- Probability of moving from *i* to *j*

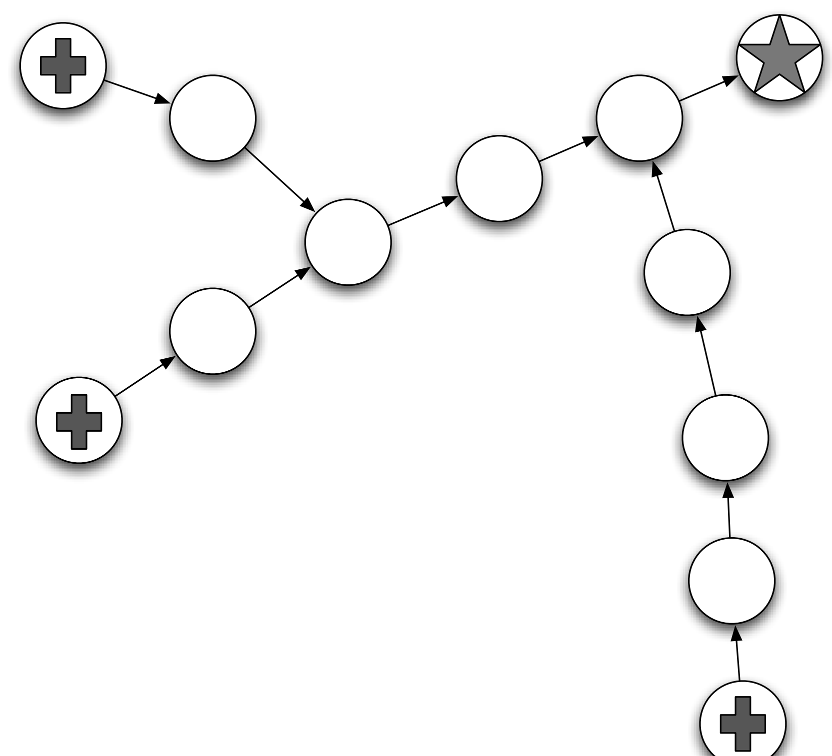


# Subgoal Options

Results in a *plan graph*.

- Node for each option.
- Probability of moving from *i* to *j*





# Abstract Subgoal Options



Abstract subgoal option:

- $s = [a, b]$
- $a$ (mask) is set to some subgoal distribution.
- $b$ remains unchanged.

$$[a, b, c, d, e, f, g, h]$$

$$\downarrow$$

$$[a, b, c, d, e, \overline{f', g', h'}]$$

# Abstract Subgoal Options

Abstract subgoal option:

- $s = [a, b]$
- $a$ (mask) is set to some subgoal distribution.
- $b$ remains unchanged.


$$[a, b, c, d, e, f, g, h]$$

$$\downarrow$$

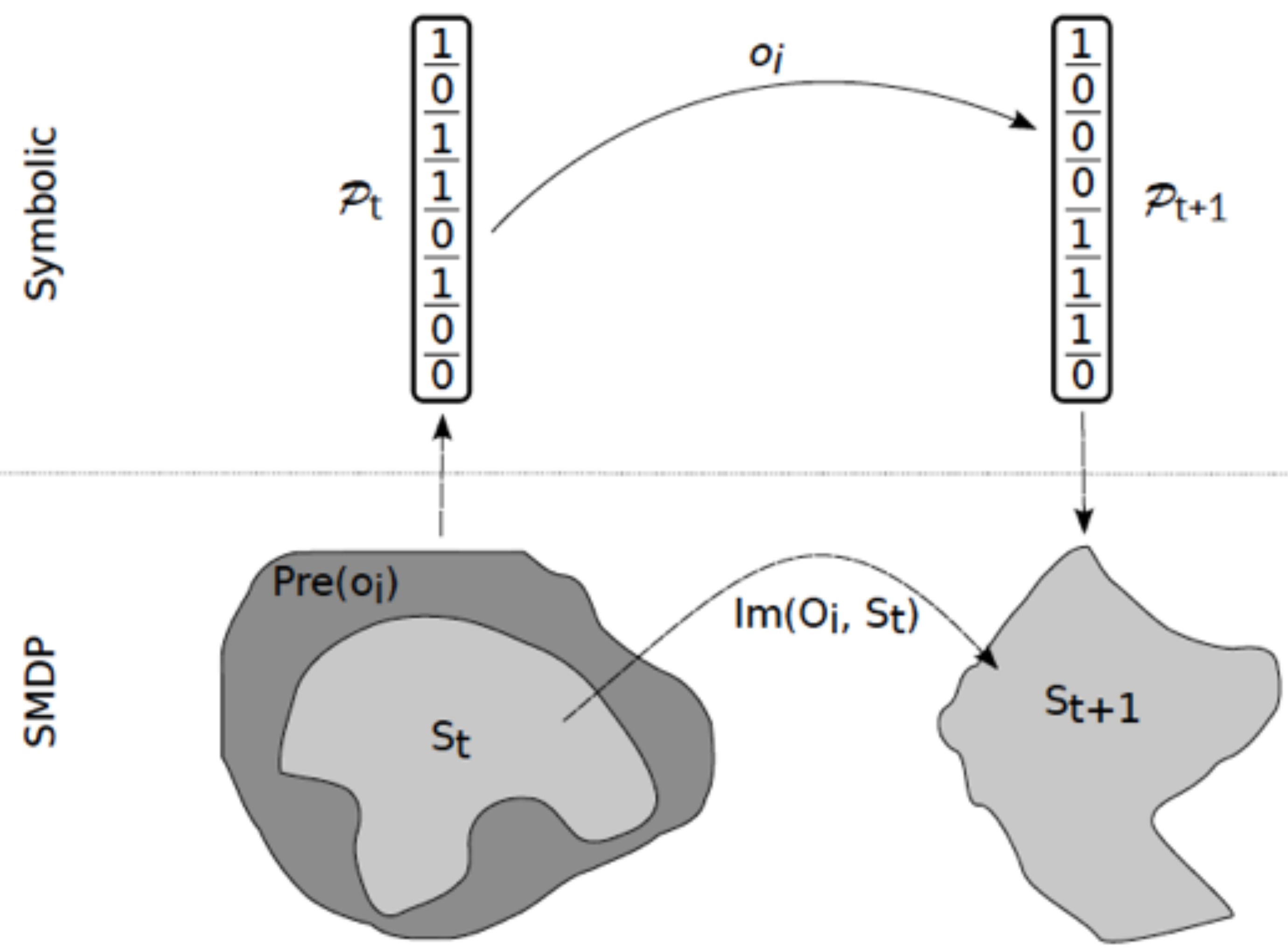
$$[a, b, c, d, e, \overline{f', g', h'}]$$

Factored MDP

# Abstract MDPs

Abstract subgoal options: can generate factored MDP

- Vocabulary of state factors + forward model
- Provably **sound** and **complete**
- Can discard grounding distributions once done



# What is a Symbol?

A (propositional) symbol is a *name* for a *set of low-level states*.

**Definition** *A propositional symbol $\sigma_Z$ is the name associated with a test $\tau_Z$, and the corresponding set of states $Z = \{s \in S \mid \tau_Z(s) = 1\}$.*

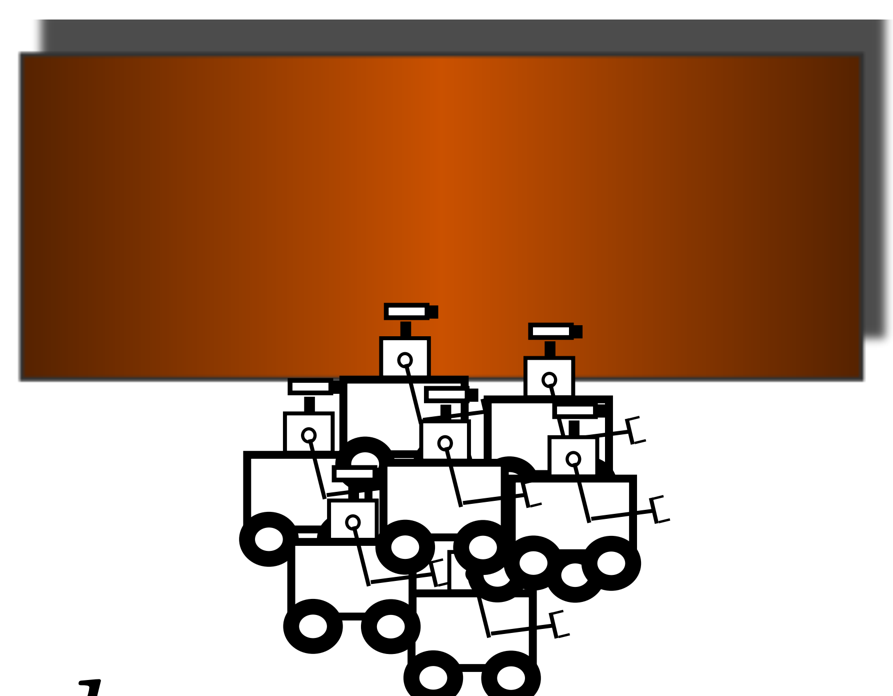
# What is a Symbol?



A (propositional) symbol is a *name* for a *set of low-level states*.

**Definition** *A propositional symbol* $\sigma_Z$ *is the name associated with a test* $\tau_Z$*, and the corresponding set of states* $Z = \{s \in S \mid \tau_Z(s) = 1\}$.

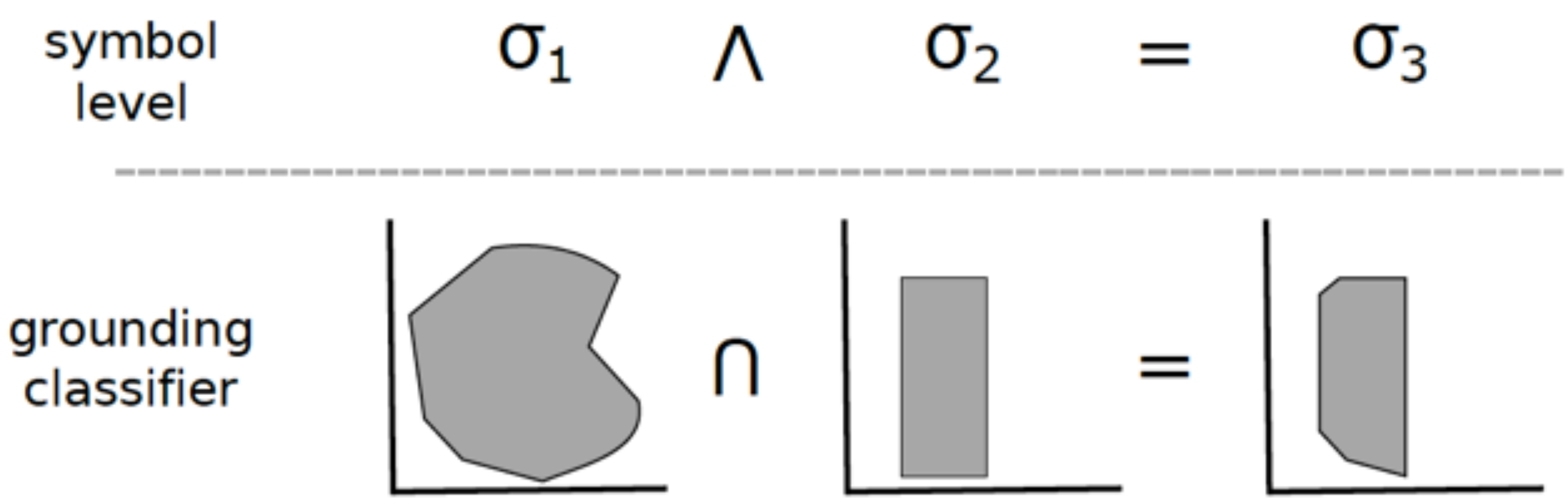
$$f(s) = \frac{1}{1 + e^{-\theta \cdot s}}$$



$$AtDesk$$

# Defining a Symbol

What do operations on our symbols mean?




(concrete boolean algebra)

# Probabilistic Symbols

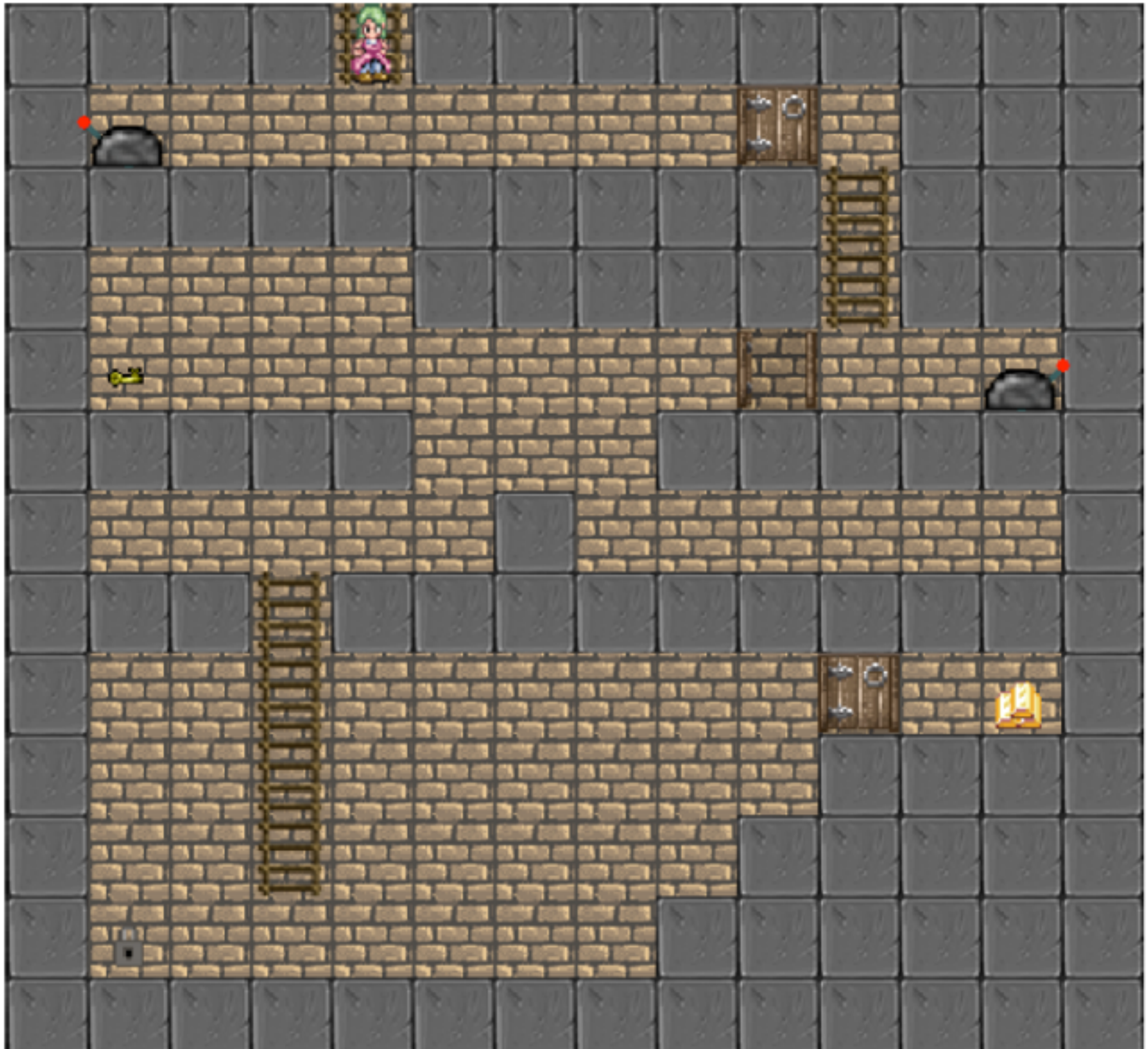
*Learning symbolic representations*

- Execute options and get some data

  $(s, o, s', r)$ $(s, I_o?)$

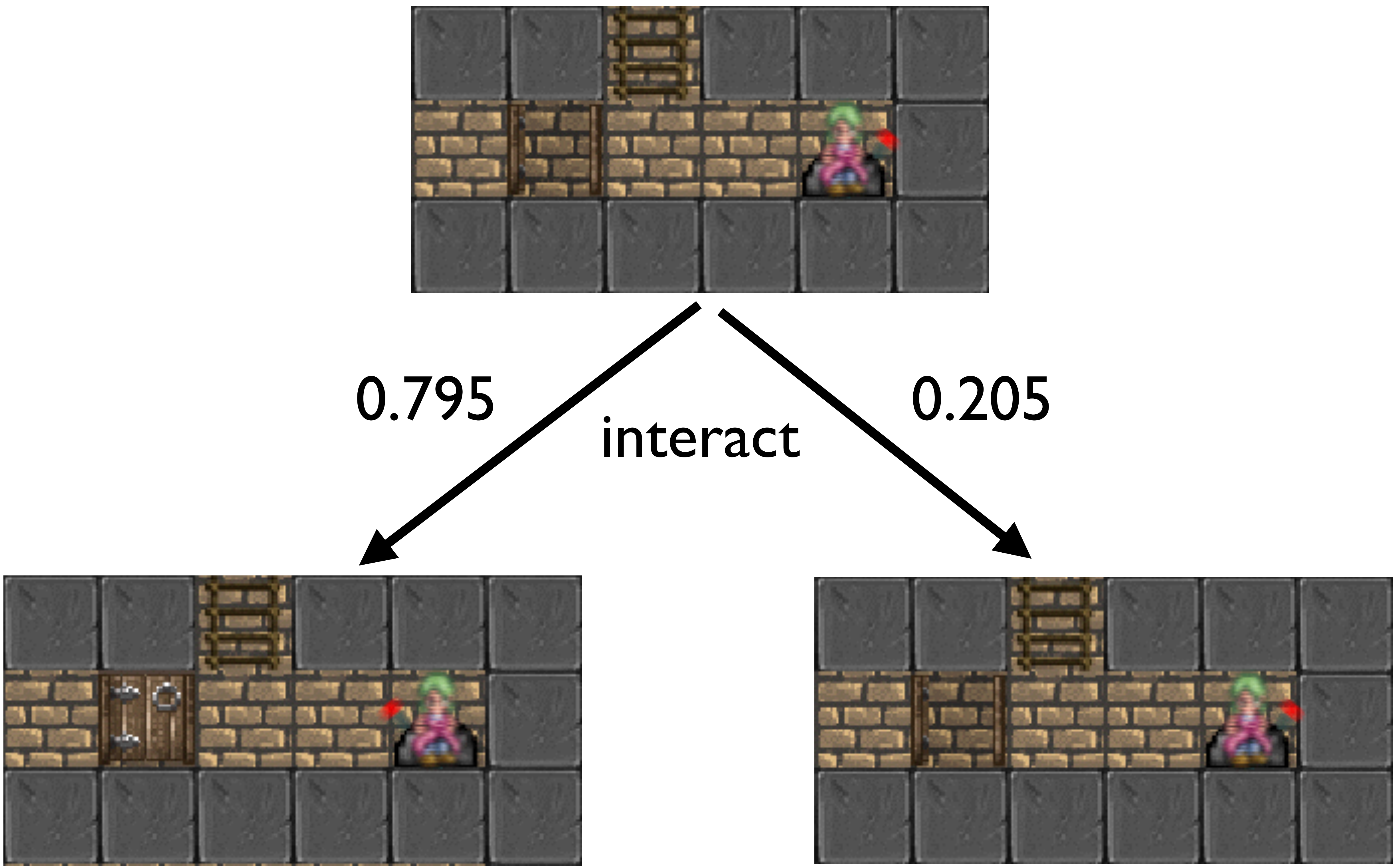
- For each option:
  - Partition into ~abstract subgoal options
  - For each partitioned option:
    - Probabilistic classifier for init distribution
    - Density estimator for image distribution
    - Regression for reward model

# Probabilistic Symbols



# Probabilistic Symbols



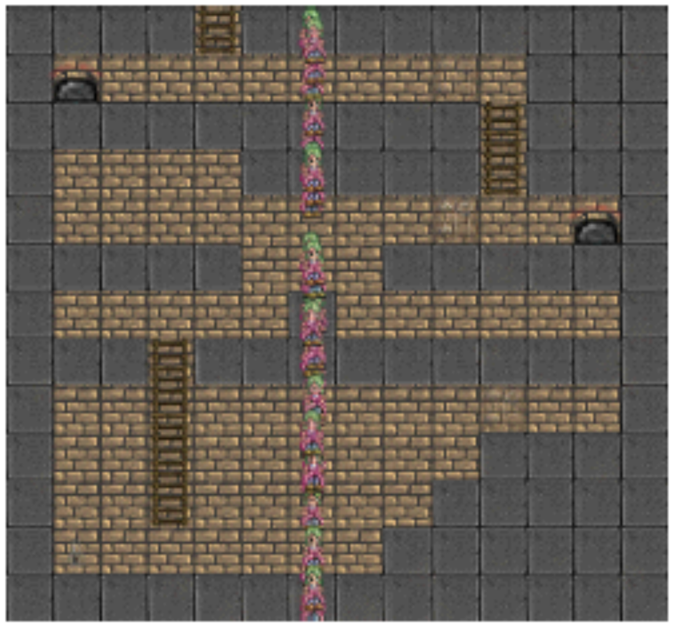


# PPDDL

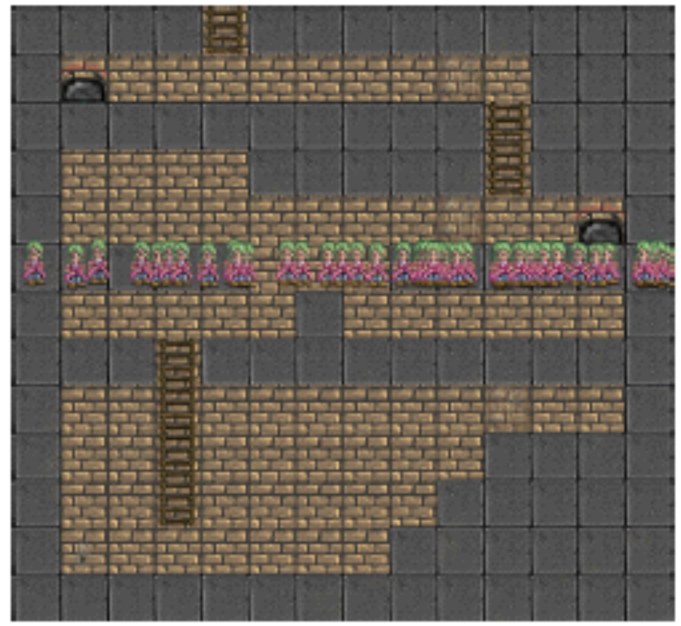
```
(:action interact_option
 :parameters ()
 :precondition (and (notfailed) (symbol14)
                    (symbol20) (symbol3))
 :effect (probabilistic
              0.7955 (and (symbol21) (symbol22)
                          (not (symbol3)))
              0.2045 (and (symbol4))
         )
)
```

*learned* PPDDL representation

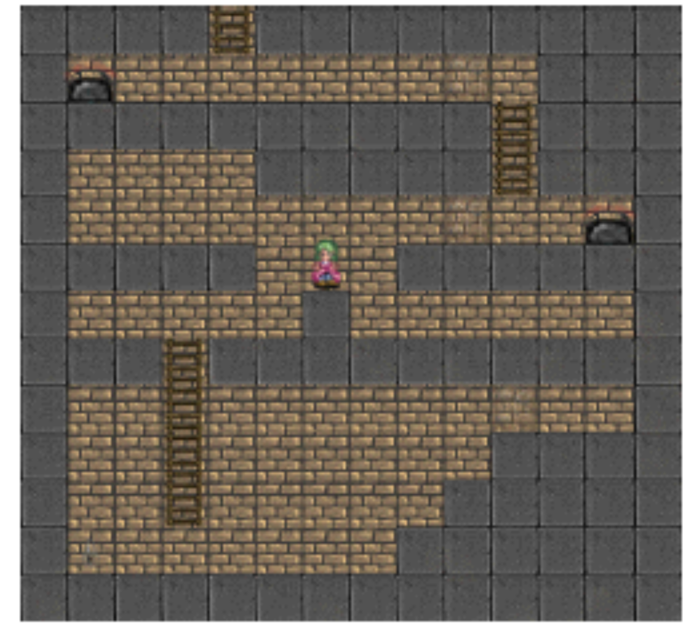
# Symbols



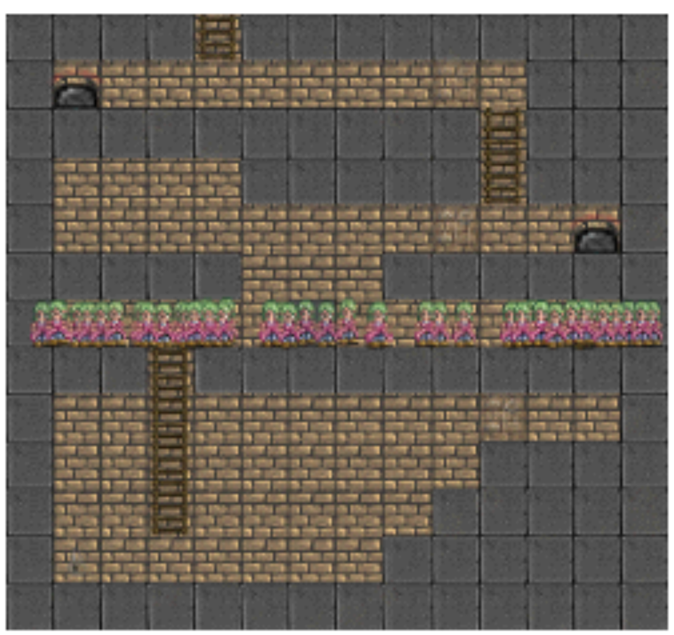
(b) symbol29



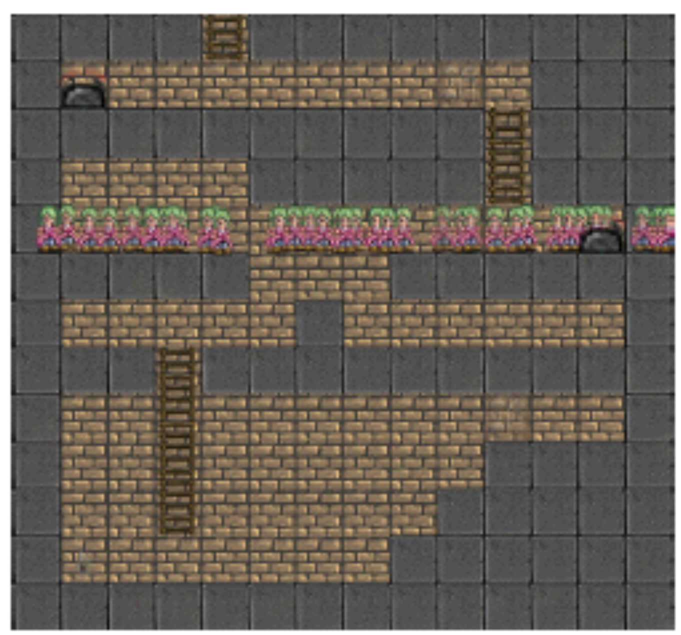
(c) symbol28



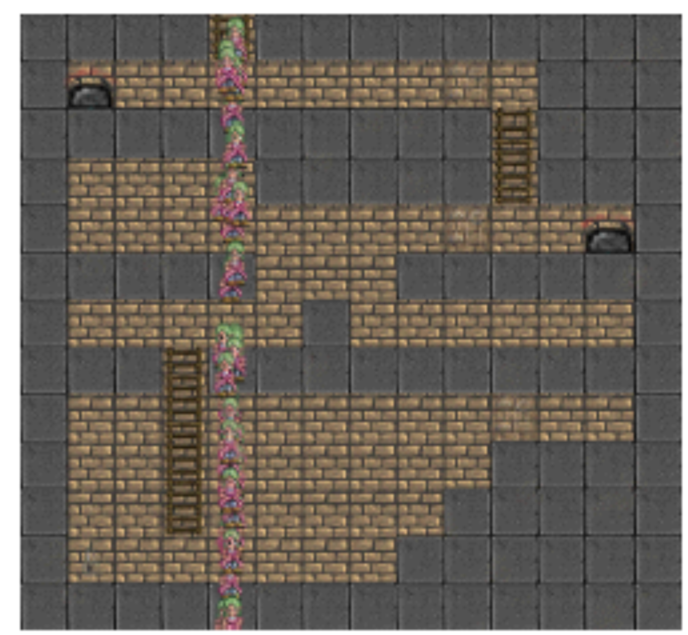
(d) symbol28 and symbol29



(e) symbol17



(f) symbol20



(g) symbol1

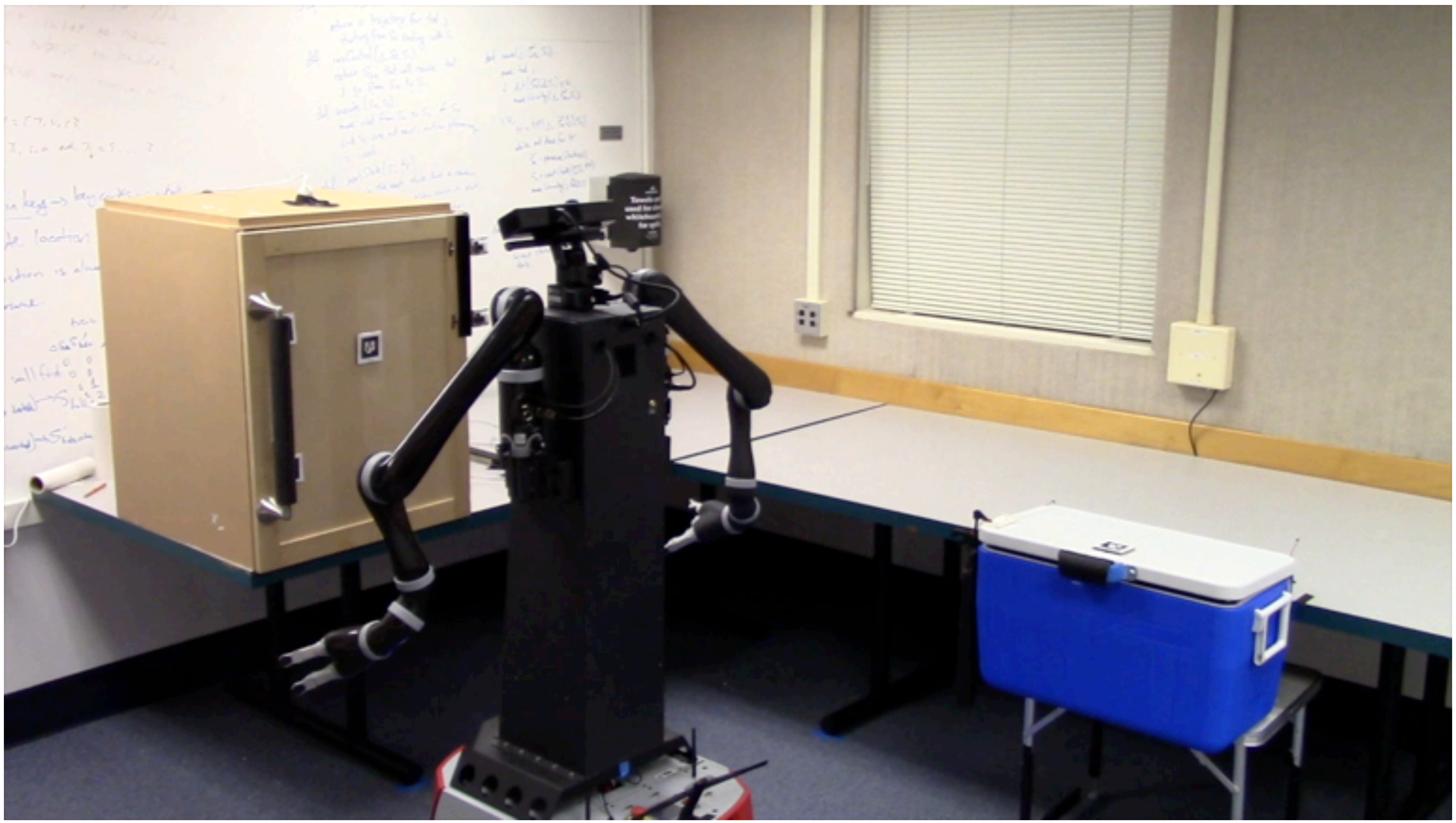
# Planning

| Goal | Min. Depth | Time (ms) |
| --- | --- | --- |
| Obtain Key | 14 | 35 |
| Obtain Treasure | 26 | 64 |
| Treasure & Home | 42 | 181 |

… using mGPT (Bonet and Geffner, 2005)

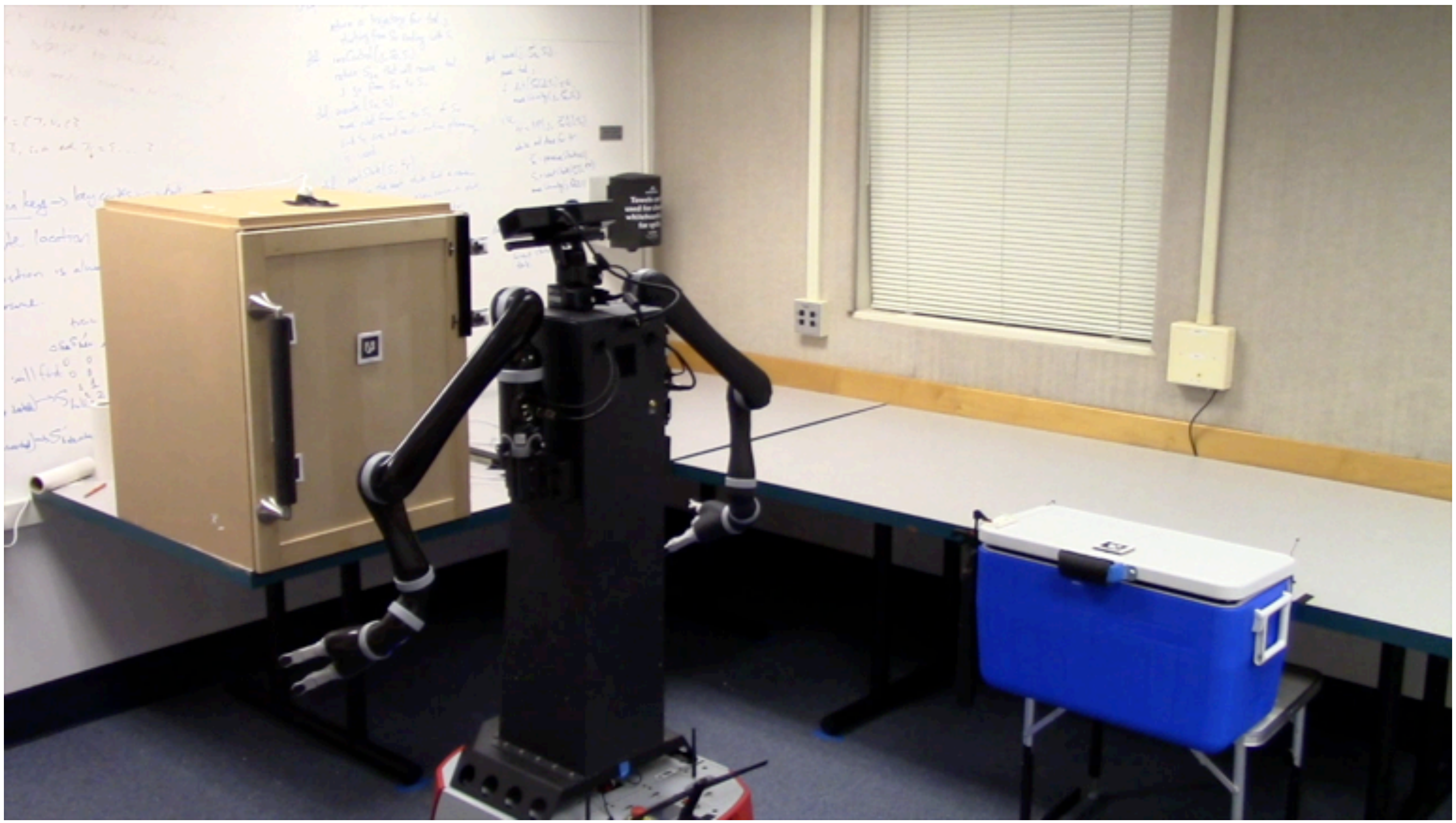
# Robots

DUKE
COMPUTER
SCIENCE



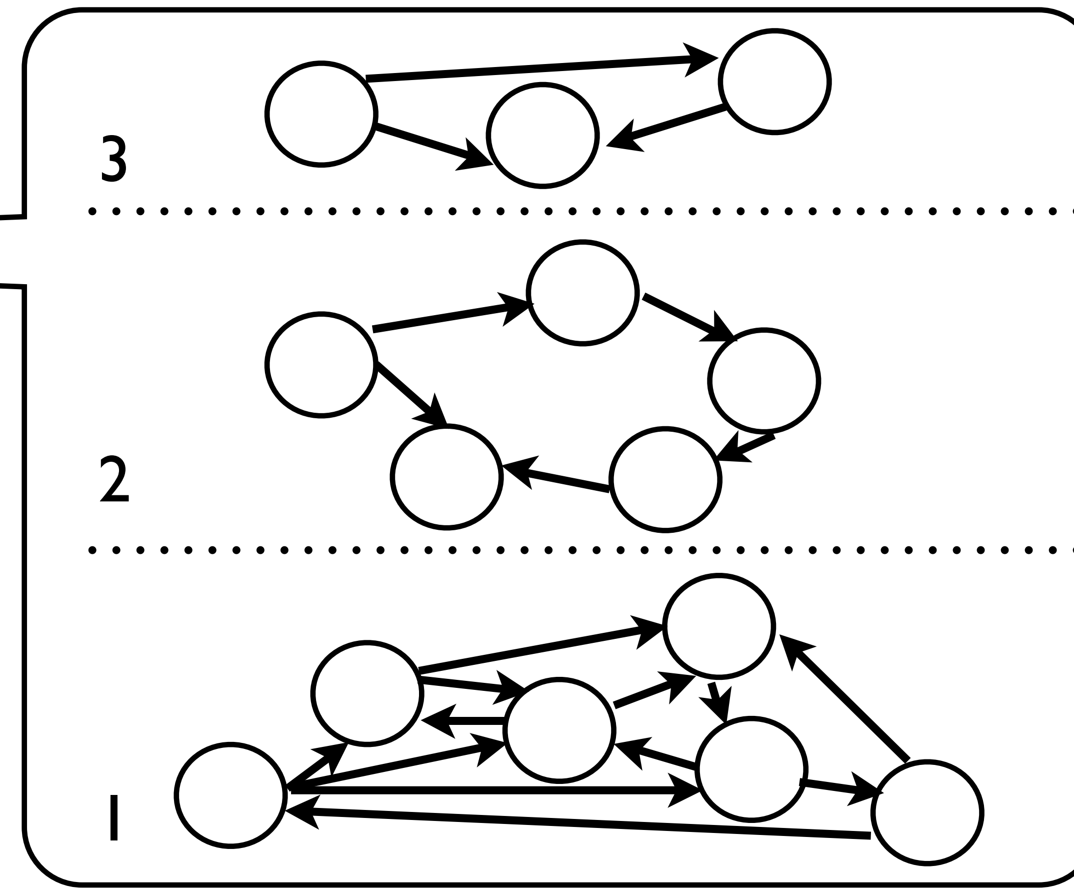
# Robots





# True Abstraction Hierarchies

Base MDP: $M_0 = \{S_0, A_0, R_0, P_0\}$
Successive MDPs: $M_i = \{S_i, A_i, R_i, P_i\}$



# True Abstraction Hierarchies

Basic assumption of hierarchical RL:

- $A_j$ is a set of options over $M_{j-1}$

$$M_j = \{S_j, A_j, R_j, P_j\}$$

options over

$$M_{j-1} = \{S_{j-1}, A_{j-1}, R_{j-1}, P_{j-1}\}$$

# True Abstraction Hierarchies

Basic assumption of hierarchical RL:

- $A_j$ is a set of options over $M_{j-1}$

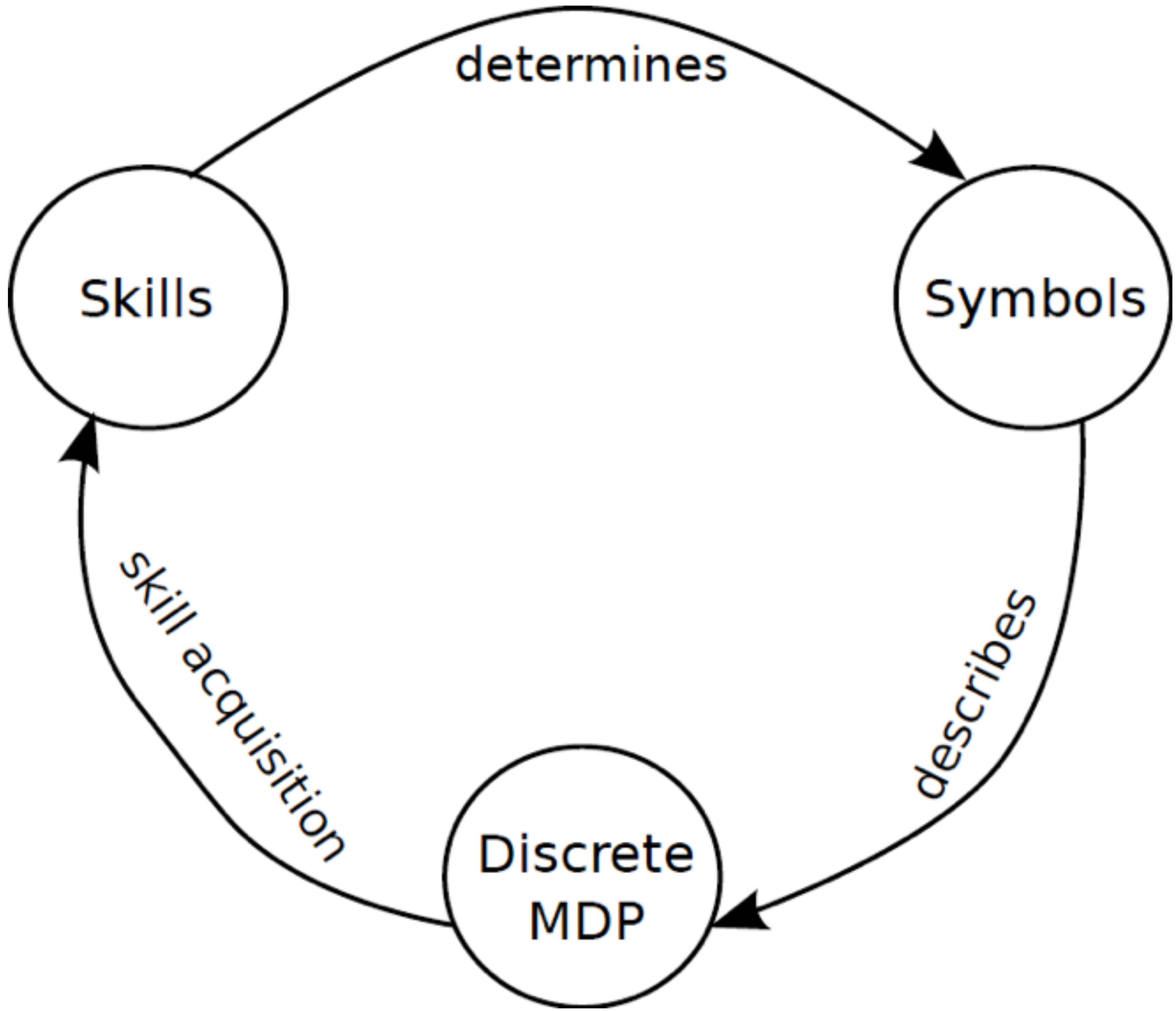
$$M_j = \{S_j, A_j, R_j, P_j\}$$

options over

$$M_{j-1} = \{S_{j-1}, A_{j-1}, R_{j-1}, P_{j-1}\}$$

Now we know what $S_j, R_j, P_j$ ***must*** be.

# The Skill-Symbol Loop



# The Skill-Symbol Loop

*Skills:*

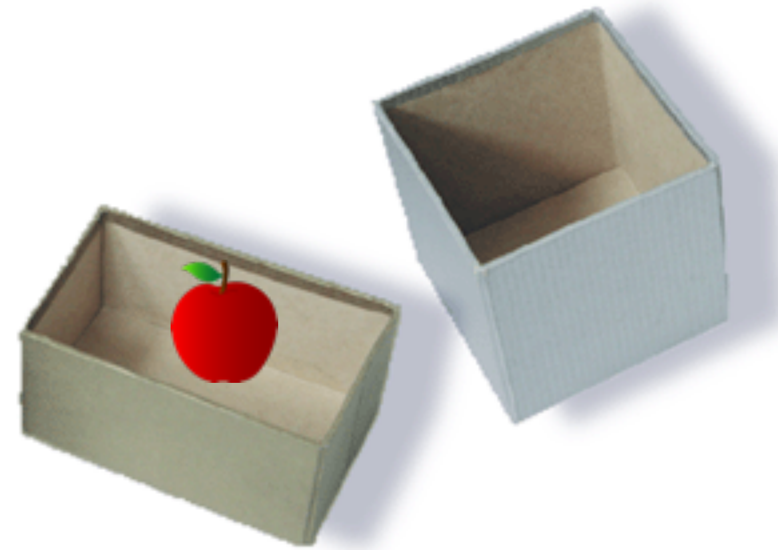
Pregrasp

Grasp

Lift

Move Arm to Above Box

Release



*Factors:*

Above-Box-Apple

Pregrasped

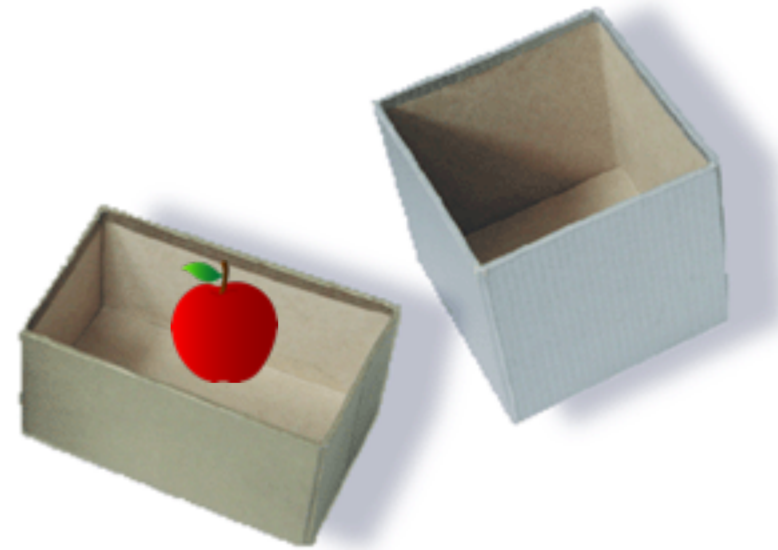
Grasped Apple

Apple-in-Air

Arm above B1, B2

Apple in B1, B2

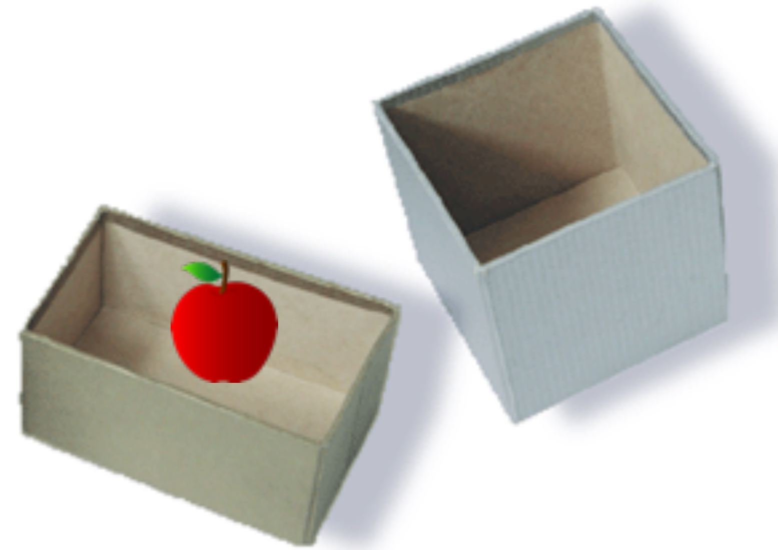
# The Skill-Symbol Loop



*New Skills:*
Grab-Apple
Move Arm to Above Box
Drop-Apple

*Factors:*
Grasped/Lifted Apple
Arm above B1, B2
Apple in B1, B2

# The Skill-Symbol Loop



*New Skills:*

MoveAppleTo

*Factors:*

Apple in B1, B2

# SSL

Succession of MDPs:

$$M_i = \{S_i, A_i, R_i, P_i\}$$

As we go up in the hierarchy:

- Symbols more general (refer to broader distributions)
- Eventually reach "basic" problem description.
- Robot details wash out.

***No choice*** other than the skill discovery algorithm.

# Planning

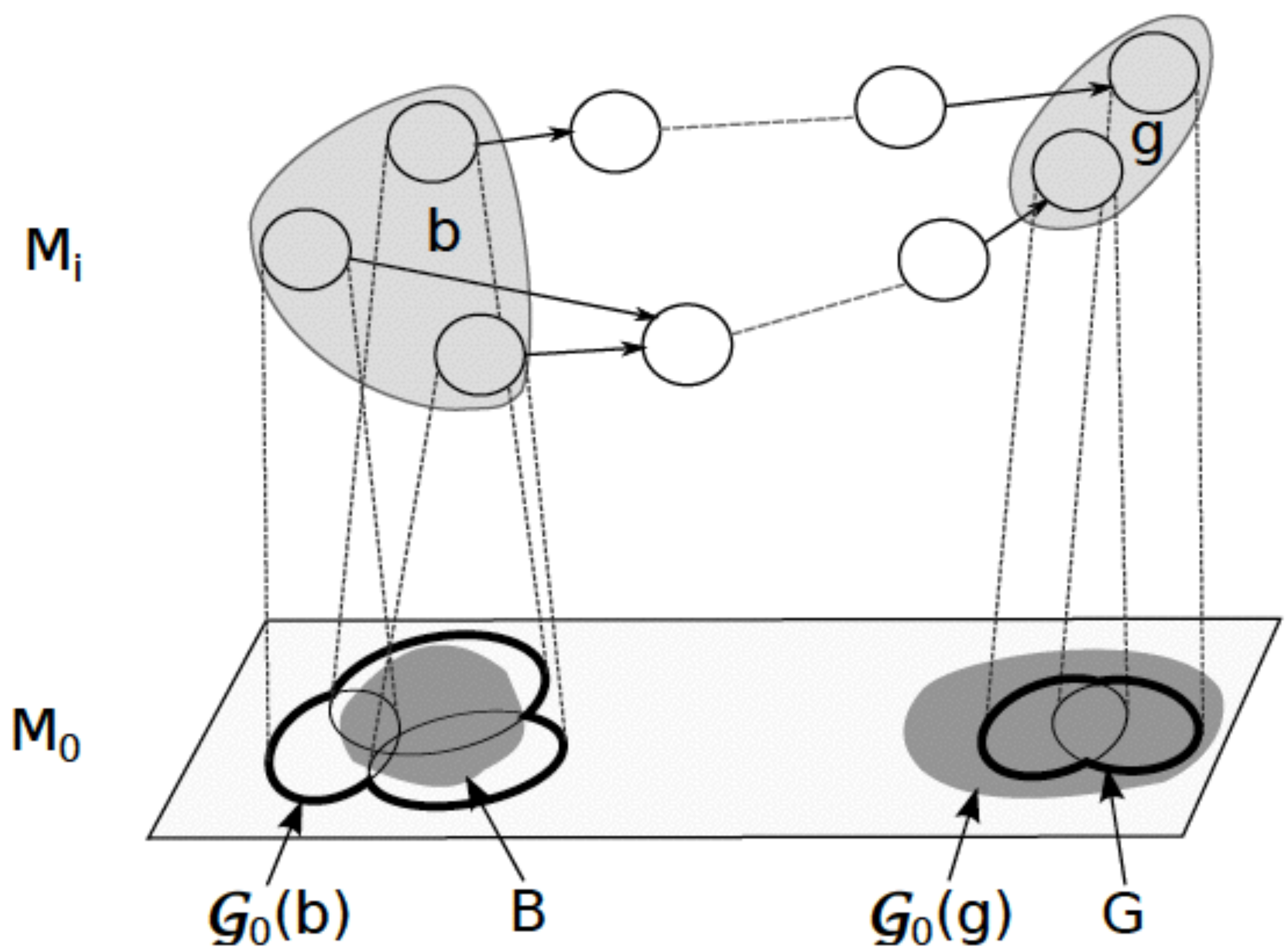
A solution at *any* level $i$ is a solution to $M_0$.

Consequently, for a **given start and goal set**, we need to find highest $i$ (smallest problem) to plan at.

# Planning

A solution at *any* level *i* is a solution to $M_0$.

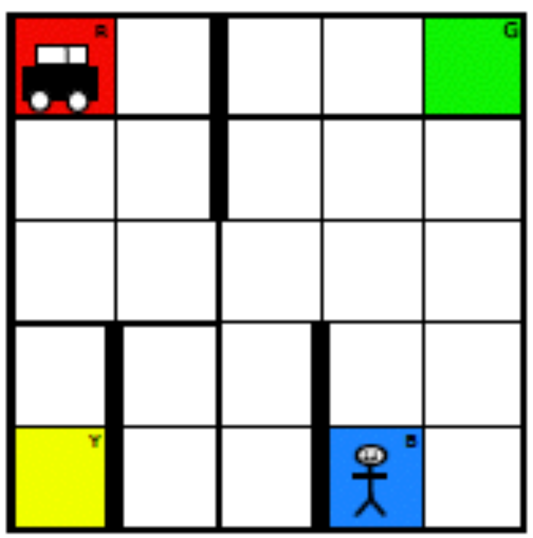
Consequently, for a **given start and goal set**, we need to find highest *i* (smallest problem) to plan at.

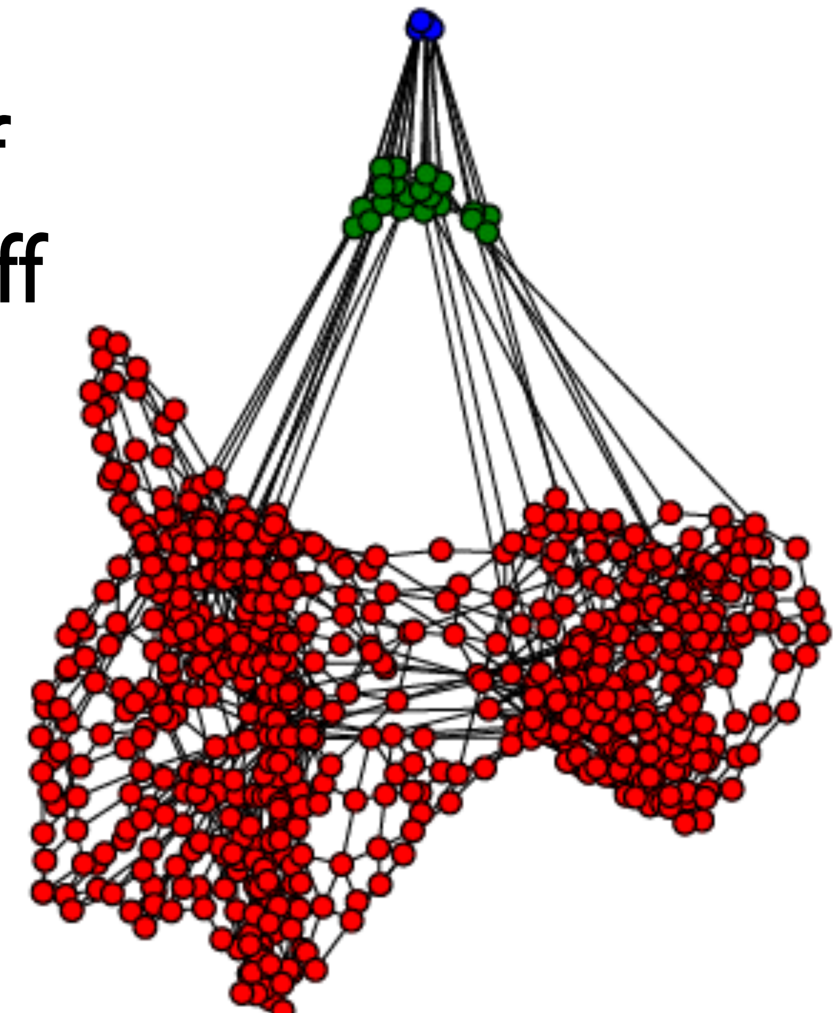


# Taxi

Options:

1. up, down, left, right, pick up, drop off
2. drive to each depot, pick up, drop off
3. passenger-to-depot





[IJCAI 2016]

| | | Hierarchical Planning | | | | |
|---|---|---|---|---|---|---|
| Query | Level | Matching | Planning | Total | Base + Options | Base MDP |
| 1 | 2 | <1 | <1 | **<1** | 770.42 | 1423.36 |
| 2 | 1 | <1 | 10.55 | **11.1** | 1010.85 | 1767.45 |
| 3 | 0 | 12.36 | 1330.38 | 1342.74 | **1174.35** | **1314.94** |

# Summary

Close link between symbolic representation and skills

- Environment + goal + skills *specify* symbolic representation we need.

- That representation is learnable.

*Skills determine the symbols you need to create plans with them.*

*We can combine skills and high-level representations to achieve true abstraction hierarchies.*

# Thank you!

Questions?