6.882
Embodied Intelligence

Phillip Isola
Leslie Pack Kaelbling
Tomas Lozano-Perez

Logistics http://6.882.csail.mit.edu/

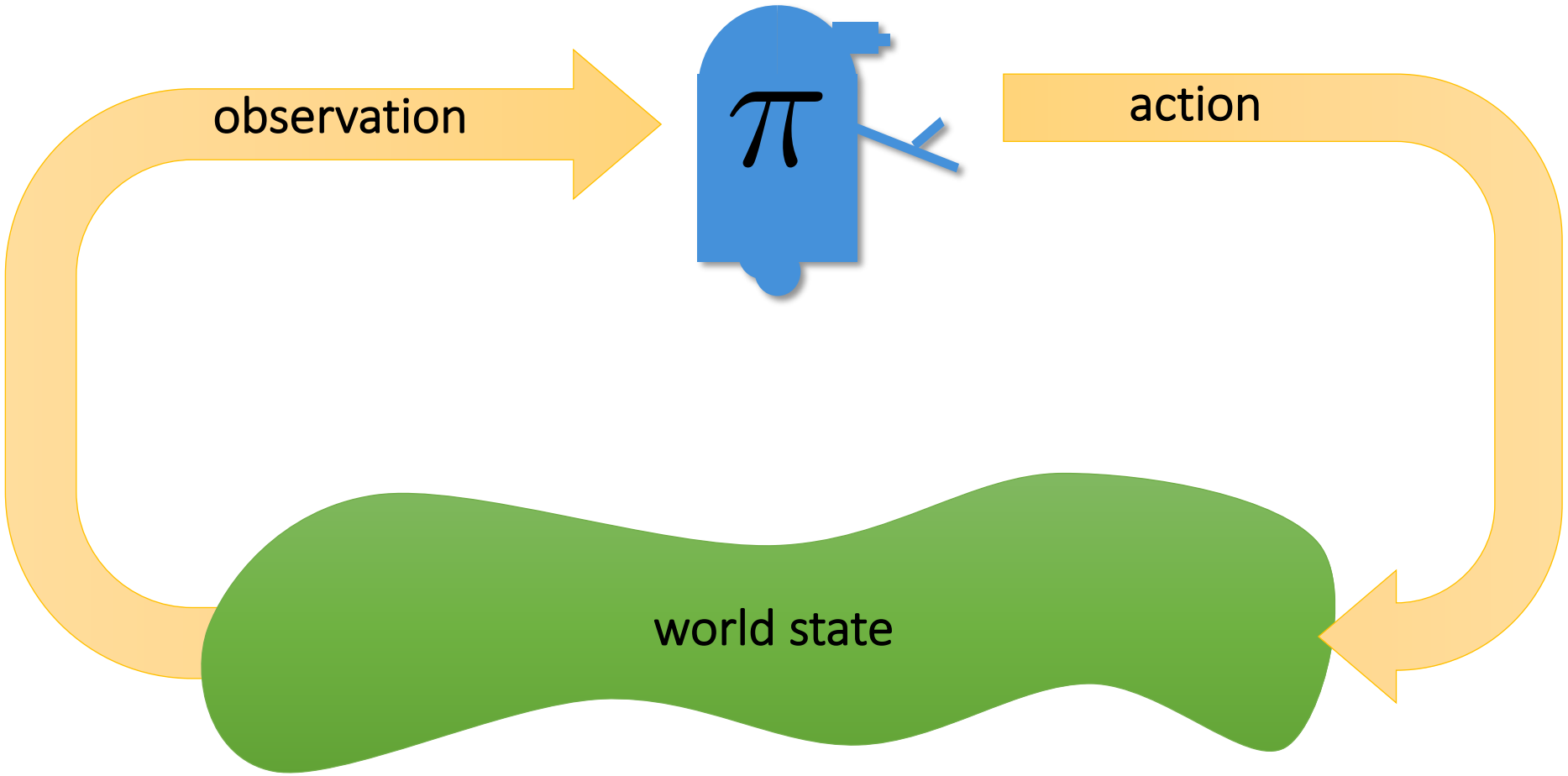
- This is the one and only lecture: we're all studying this together!
- You may only attend if officially enrolled
- Use Piazza for communication with instructors and fellow students
- Each subsequent class meeting
  - Papers to read; submit answers to questions before class
  - Students will briefly present papers and lead discussion
- Final project done singly or in pairs
- We assume you are comfortable with:
  - basic ML: hypothesis class, training error, testing error, overfitting, etc.
  - basic neural nets: feed-forward, CNN, RNN, gradient descent
  - basic path search: dynamic programming, heuristics, A*
  - basic MDPs: transition, reward, policy, value function

Goal: understand how to build **intelligent embodied agents**

- computational system that is connected to the physical world
- could be many kinds of systems, but we'll focus on robots
- variability in the domains / problems it will encounter: not a welding bot
- variability in the responses required by the system: not a Roomba
- relatively domain-independent tools:
  perception, control, planning, inference, learning
- a "catholic" approach:
  old-school, new-school, all fine
  evaluate methods on their technical merits

A robot is a transducer

$$\pi : (o, a)^* \rightarrow a$$



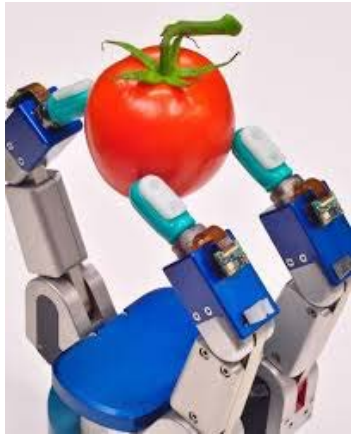
What is a world state?

Everything you would need to know to make the most reliable possible prediction about any aspect of the world

- configuration of the robot :
  - what are its joint angles?
  - position of base?
- configuration of objects in the world
  - what are the locations and shapes of objects?
  - what food is in them?
  - is it rotten?
  - …
- non-physical state
  - is the homeowner happy?
  - …

What is an observation?

- Internal perception (proprioception)
  - sensing of joint angles
  - force / torque information at joints
- External perception
  - camera image: 2D RGB (red-green-blue)
  - depth image: 2D depth
  - combined and interpreted as a point cloud
  - tactile contacts / pressures
  - tactile images
  - sound
  - smell
  - …

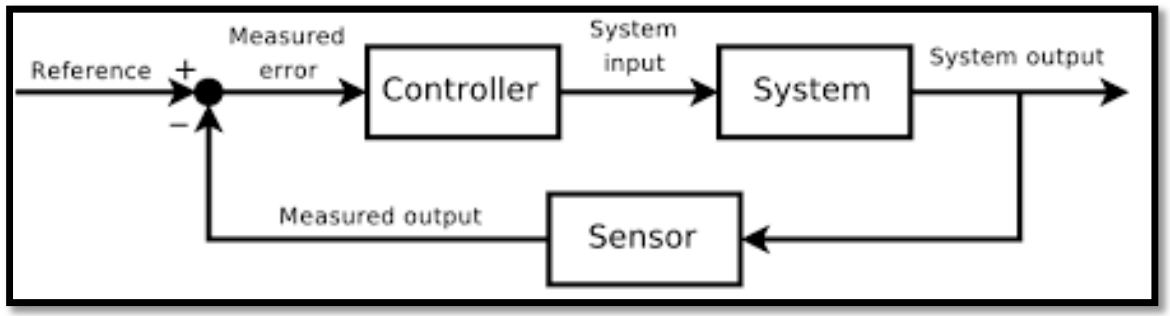


What is an action?

Levels of abstraction

- instantaneous motor torques
- execution of a parameterized **feedback controller**: e.g., linear joint-space motion
  - specify: target, velocity, acceleration profile
- move end-effector (hand) to 6D Cartesian pose (position and orientation)
  - requires solving **inverse kinematics**: what joint angles will do this?
- move robot base to relative 3D base pose
  - requires **odometry**: measurement of rotation/translation
- move head joints
- close gripper, close individual finger joints
- look at object
- pick up / put down object
- clean my house!

feedback controller



Designing an EIA

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

Need to know

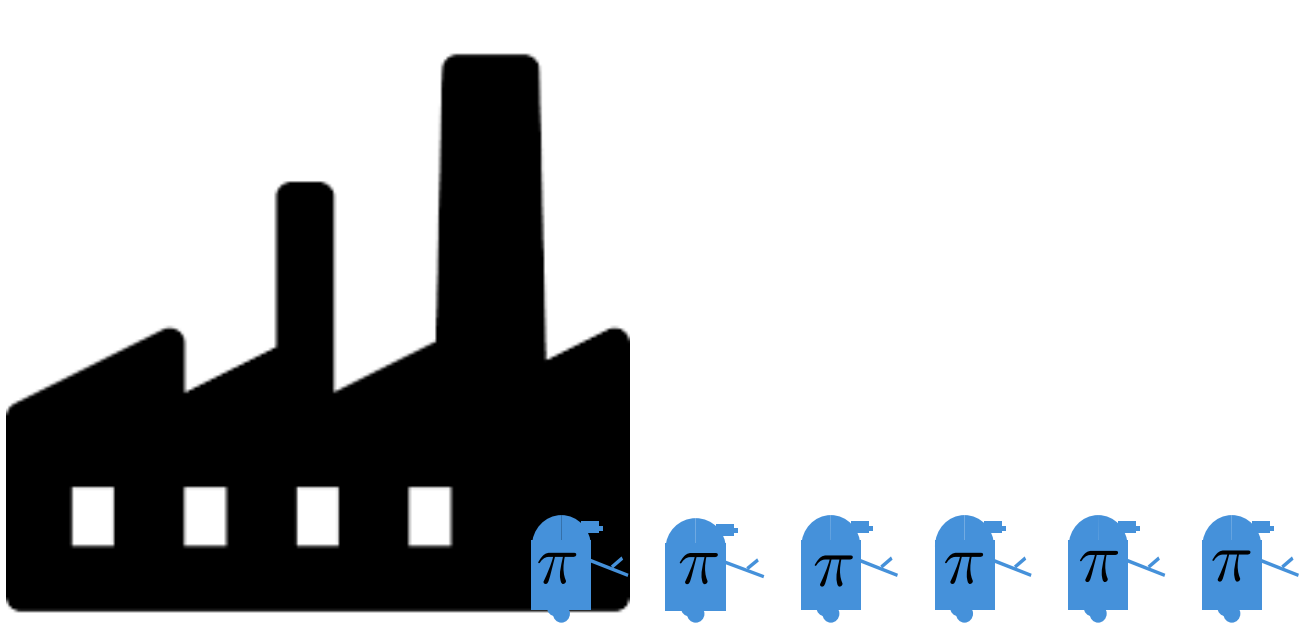
- a distribution over the environments that our agent is supposed to work in
- some measure of how well the agent is working in that environment
  - common choice: assign a scalar reward value at every time step
  - common choice: sum rewards over time (possibly with discounting)
- our position about optimization in the face of uncertainty
  - common choice: maximize expected (over environments) sum of rewards

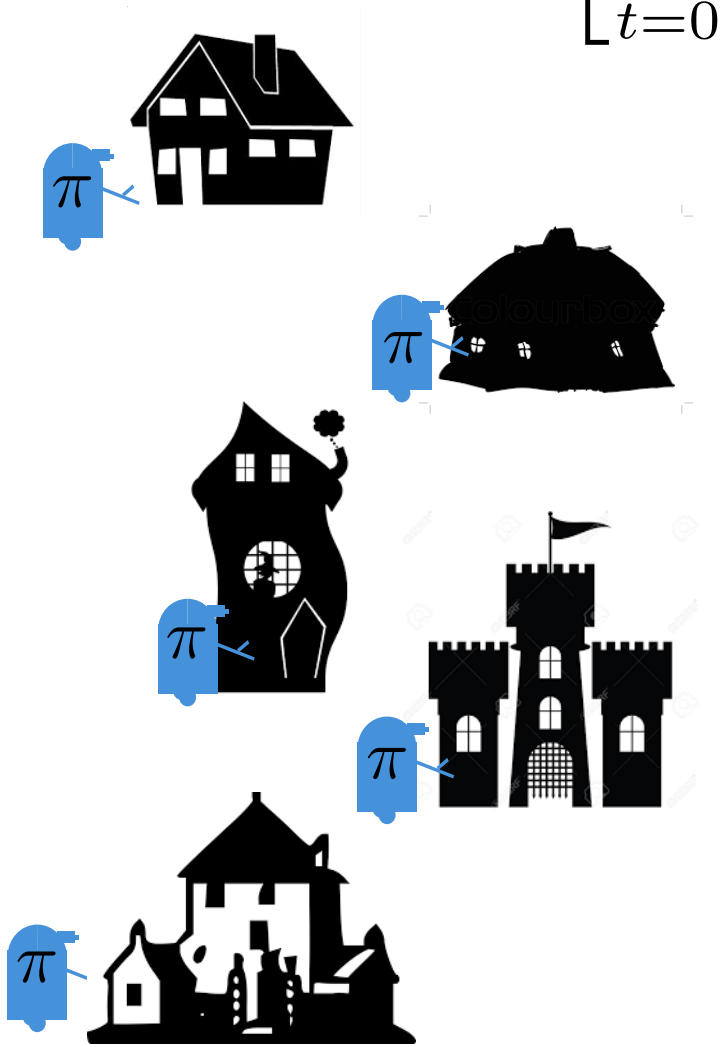
What is a reward and where does it come from?

- Goal of achievement:
  - r(s) = 1 when s satisfies the goal and 0 otherwise
  - episode typically terminates when r(s) = 1
- Goal of maintenance:
  - r(s) = 1 when s satisfies the goal and 0 otherwise
  - episode typically terminates when r(s) = 0
- Easy case:
  - r(s) is a “navigation function” or “value function” so it suffices to move to the neighboring s with highest r(s) value
- Dynamic goals:
  - let the goal g be encoded as part of the state, so that s = (g, w) where w is world state
  - r(s) = r((g, w)) = 1 when w satisfies g and 0 otherwise

Doing for our robots what nature did for us

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





Rationality

A **rational** agent selects actions that **maximize its expected future utility (reward)**

- not necessarily clairvoyant (able to predict the future accurately)
- not necessarily omniscient (knowing everything about the present)
- not necessarily successful
- not necessarily human-like

Property of the agent's behavior, **not its implementation**

Nature of solution depends on **properties of environment**

Deterministic: executing the same action in the same state always results in the same state

Observable: the agent's current observation contains complete information about the state

Episodic: agent's action choice doesn't effect the next state. More generally, we might think about the decision-making horizon.

Static: world state does not change while agent is selecting its next action

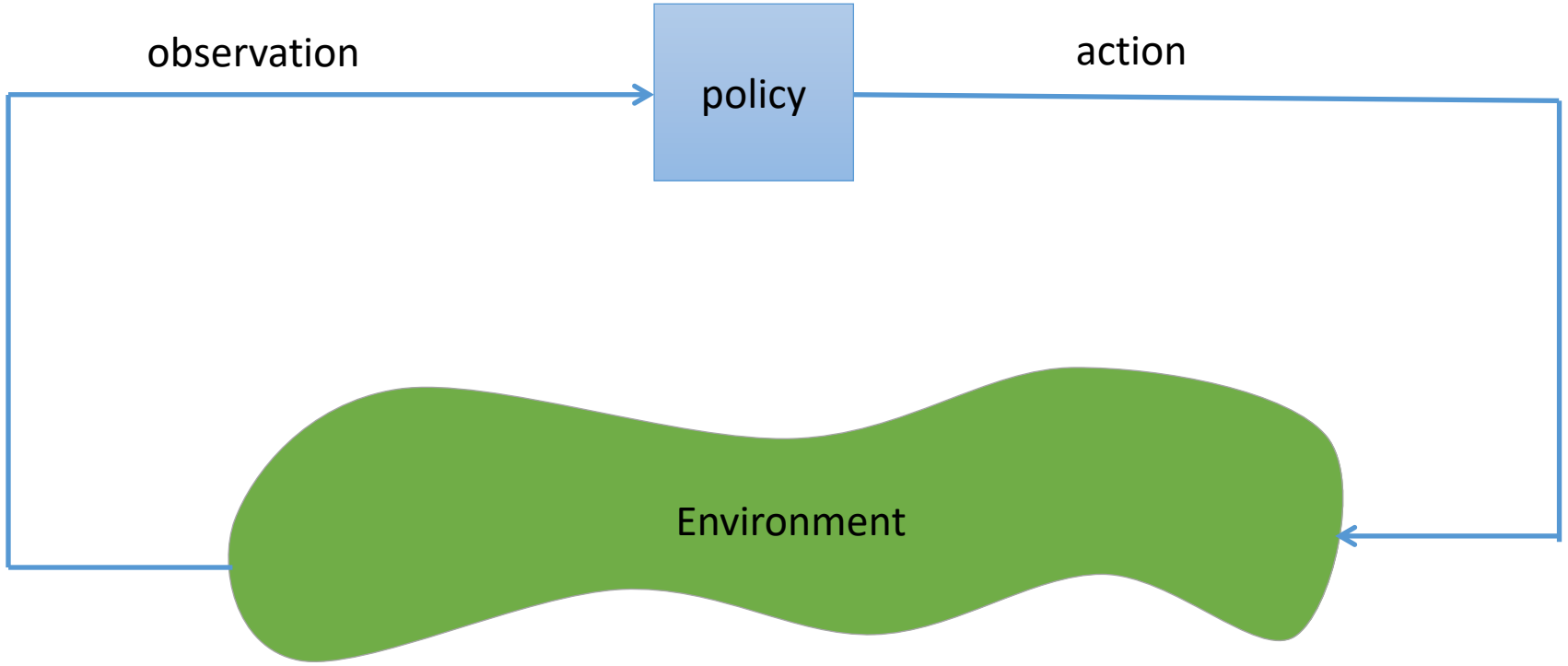
Discrete: states, actions, and observations drawn from a discrete set

Single-agent: the rest of the environment can be modeled as deterministic or stochastic, but does not depend on the actions selected by another agent

Keep in mind the **distribution over problems** the agent will have to face

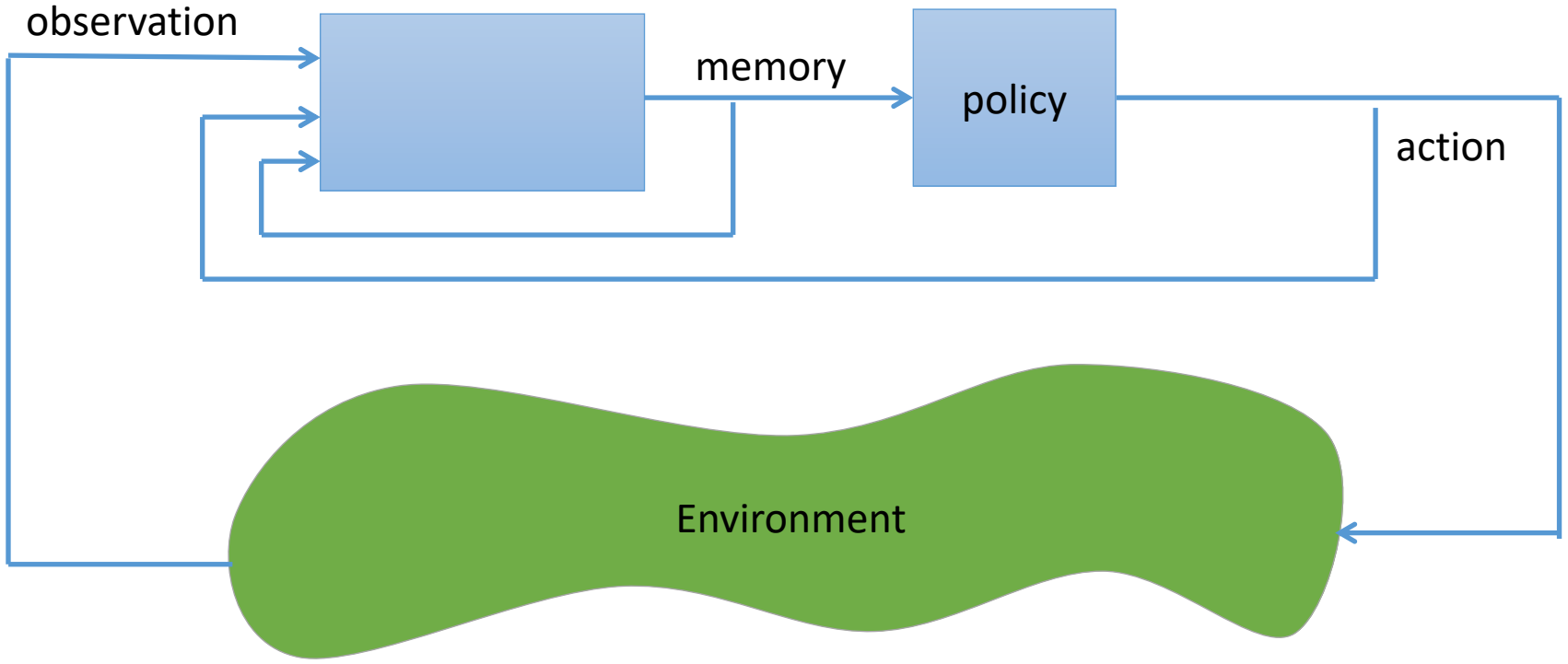
Policy with no memory

Functional program, feed-forward NN, etc.



Policy with memory

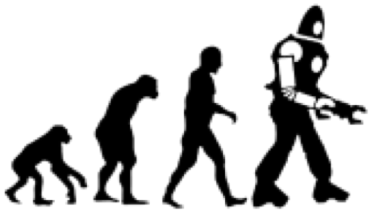
First box could be called: perception, learning, estimation, etc.



Designing the robot factory: four views

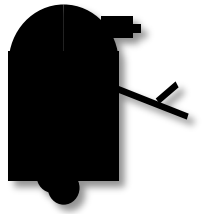
1. Reverse-engineer human

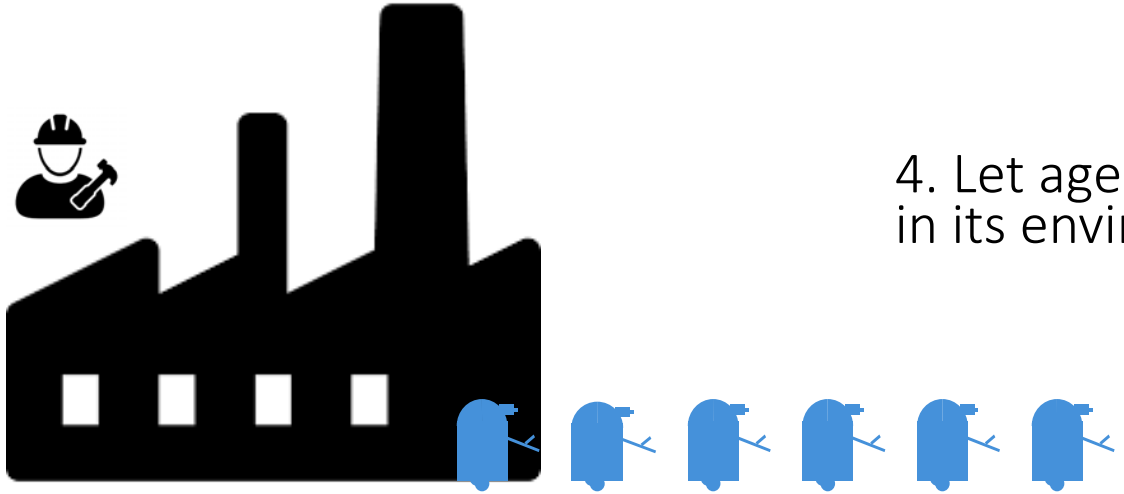
2. Recapitulate evolution



3. Solve decision theory problem in factory

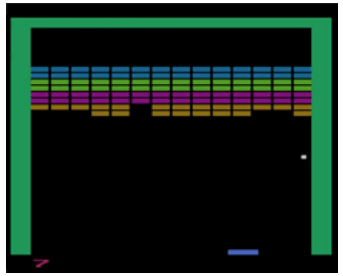
4. Let agent learn completely from scratch in its environment

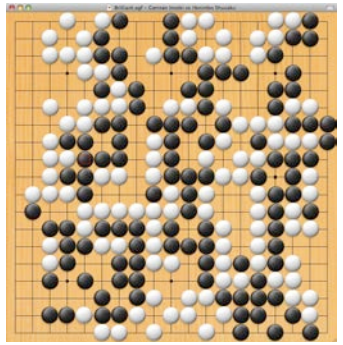




Some ways to define a policy (to compute a given s)

1. **Direct**: program, table-lookup, neural network : pi(s)
2. **Value-based**:
   - given Q(s, a): expected long-term value of taking action a in state s
   - pi(s) = argmax_a Q(s, a)
3. **Search-based**:
   - given T(s, a): distribution over state resulting from taking a in s
   - given R(s, a): expected reward for taking a in state s
   - pi(s) = root of optimal path (deterministic domain)
     or tree (stochastic domain) starting from s, given T and R
4. All of the above, but with observation history, or memory, instead of s



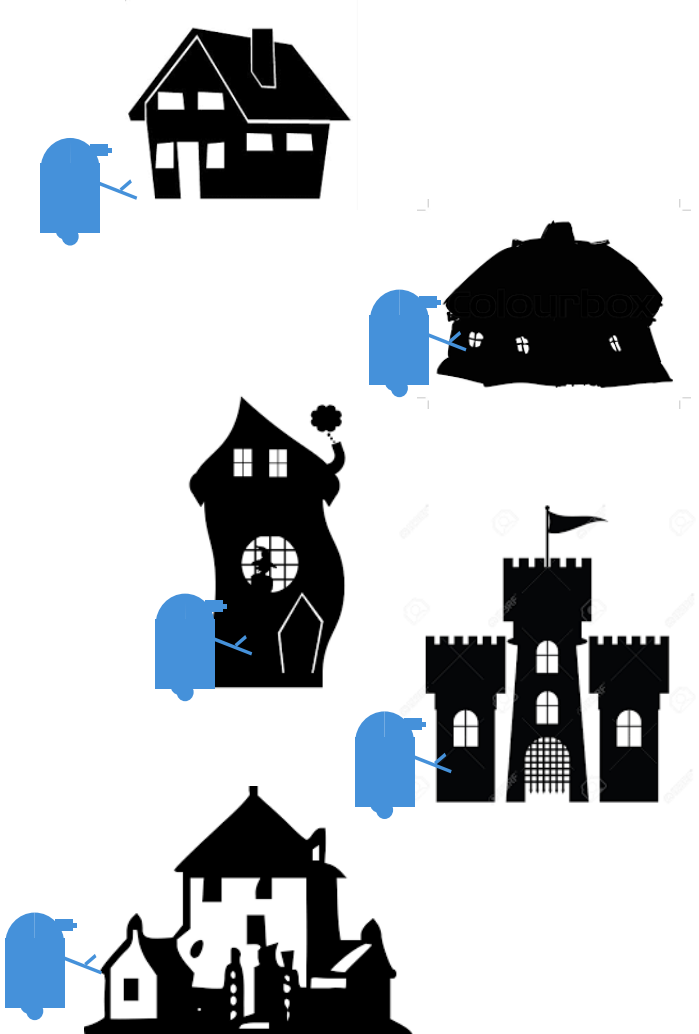


Many interesting and useful combinations possible!

Learning in the wild

Individual robots learn how to operate effectively in their particular environment

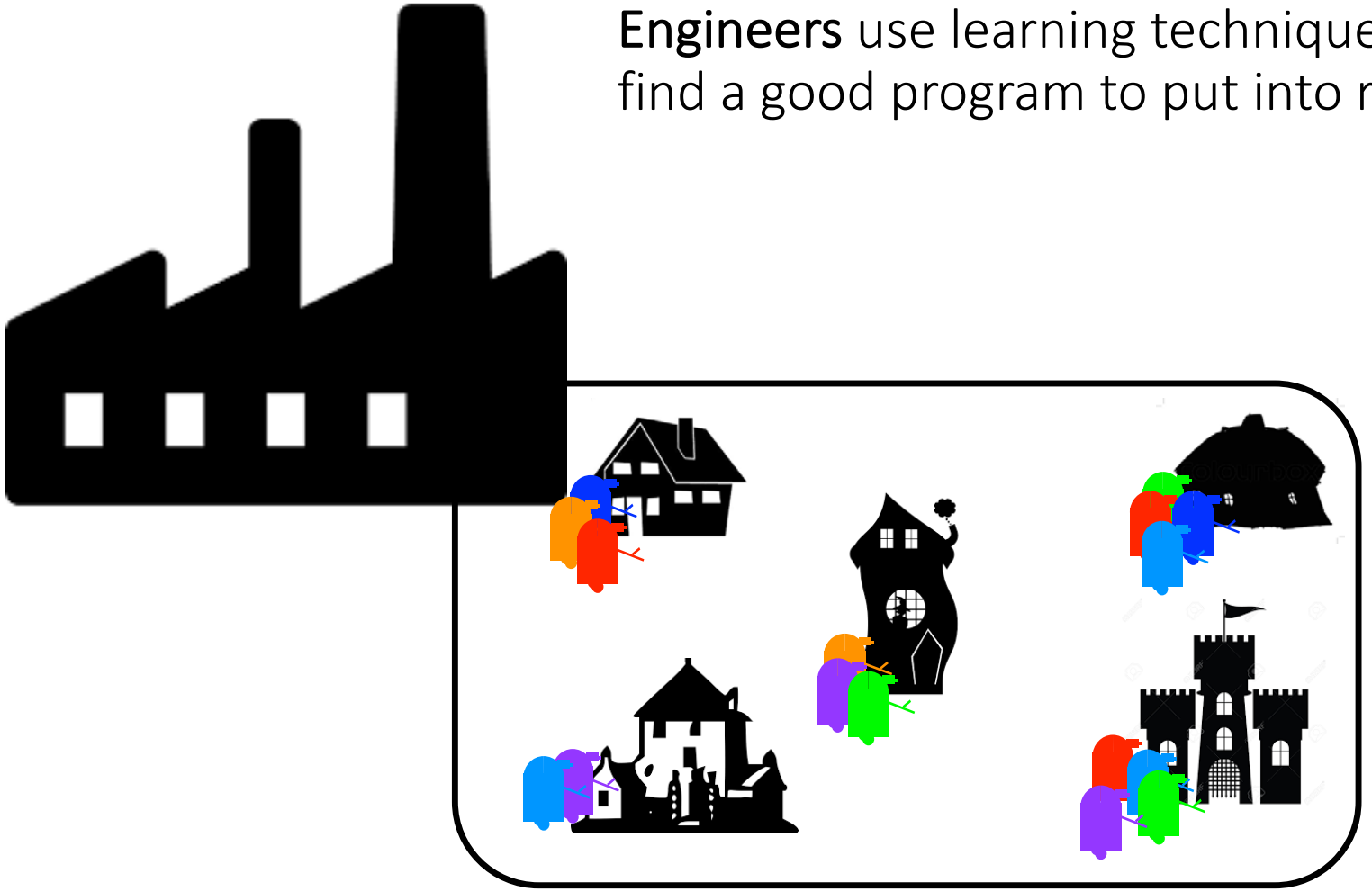
Engineers put learning algorithms into robots



Learning in the factory

Individual robots execute fixed program

Engineers use learning techniques in variety of environments to find a good program to put into robots



Meta-learning: Learning in the factory to learn in the wild

Individual robots execute learning program

Engineers use learning techniques in variety of environments to find a good learning program to put into robots

Course outline by week

1. Classic papers
2. The problem of long-term decision-making
3. The problem of continuous action spaces
4. The problem situation awareness
5. The problem of partial observability
6. The problem of learning for long horizons
7. The problem of very little data
8. The problem of non-stationarity
9. The problem of adversaries

10, 11. Case studies

12, 14. Project proposals and presentations