Algorithmic Human-Robot Interaction

Task Planning I

CSCI 7000

Prof. Brad Hayes

Computer Science Department

University of Colorado Boulder

Papers for Thursday 1/31:

Need 1 PRO (10m) and 1 CON (5m) speaker each

E-mail Bradley.Hayes@Colorado.edu to sign up

Trajectories and Keyframes for Kinesthetic Teaching: A Human-Robot Interaction Perspective

Baris Akgun et al.

Planning human-aware motions using a sampling-based costmap planner

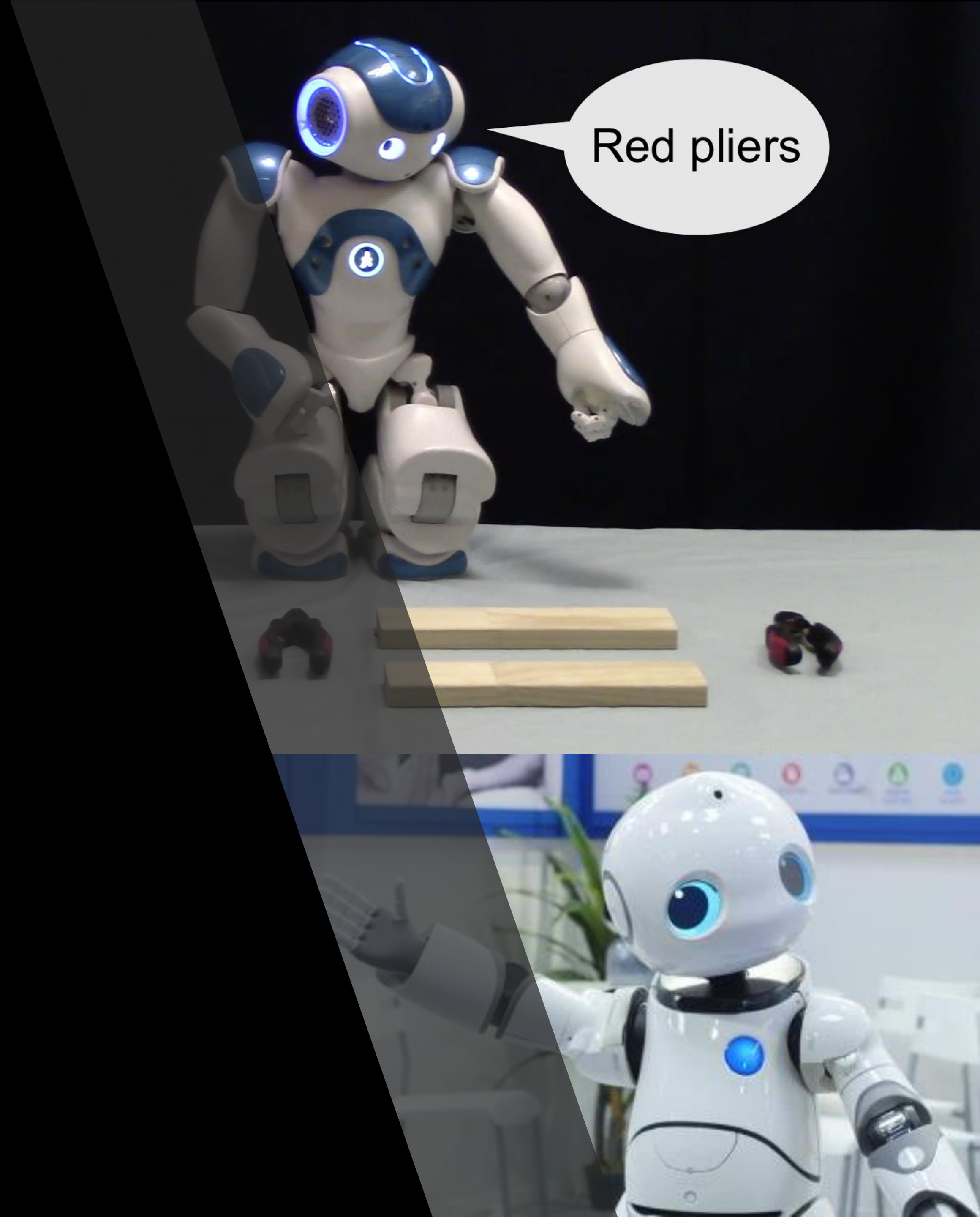
Jim Mainprice et al.

Interpreting human gesture and gaze

Producing meaningful non-verbal behaviors

Improving social interactions with non-verbal behavior

Topic: Non-verbal Cues During Collaboration

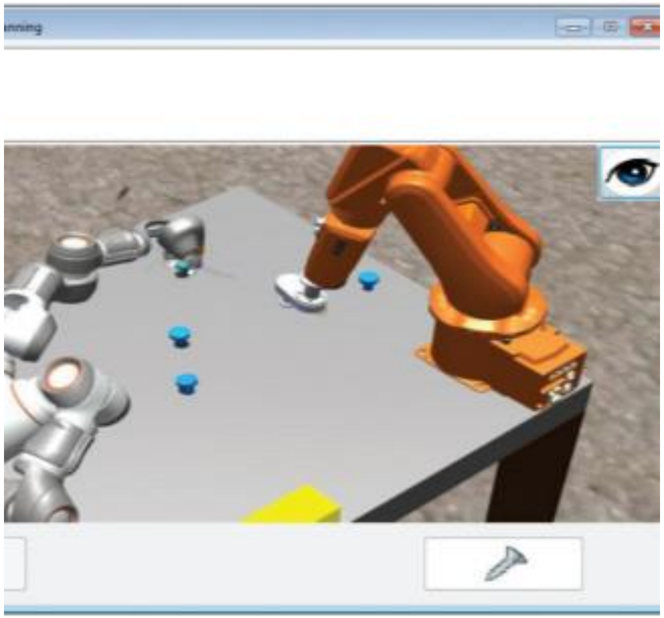


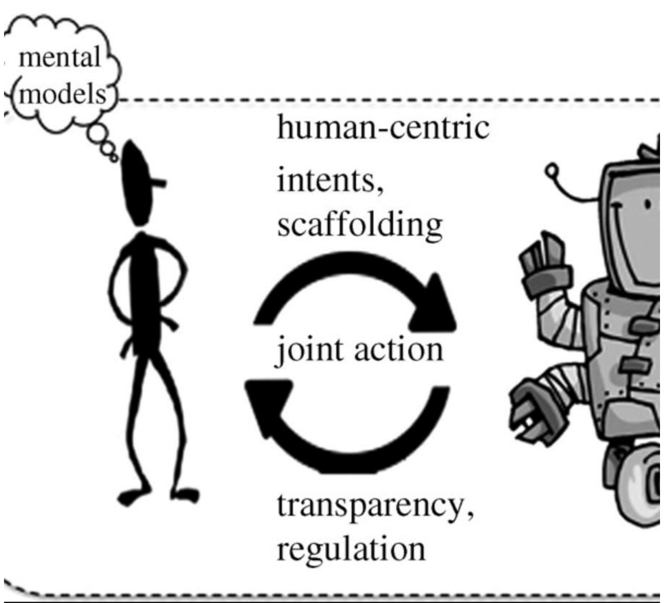
Topic:
Modeling Trust During Collaboration

A robot that adapts its plans based on how much it thinks the human trusts it

A robot that adapts its collaboration based on how much it trusts the human to do certain tasks

Self-trust with respect to its ability to successfully execute tasks (trusted region of its policy)

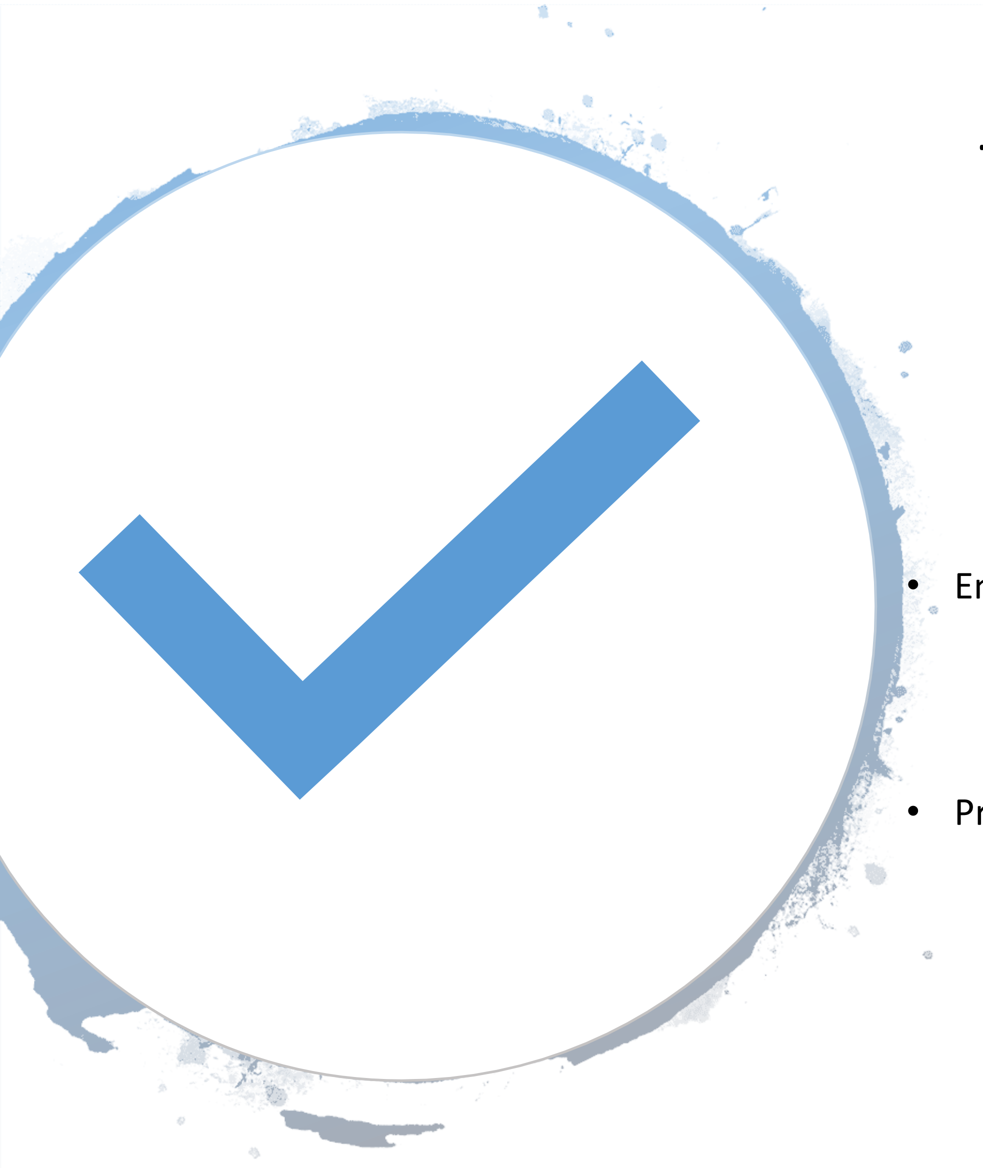




Topic: Synchronizing Mental Models of Tasks or Actions

Sharing tasks across human/robot teams

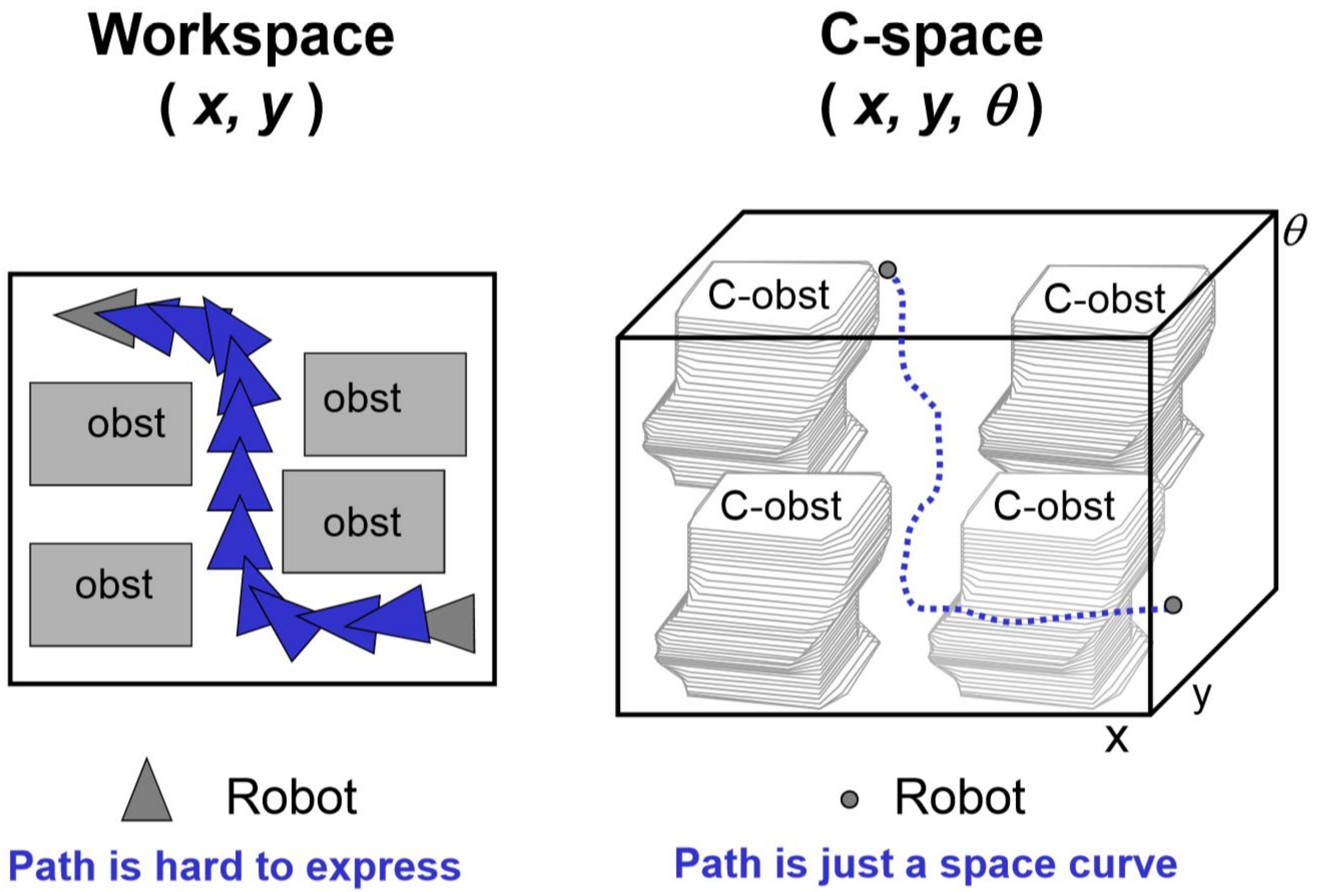
Assigning roles to teammates during multi-agent tasks



Today’s ideas will be posted on Moodle

- Enrollment is finalized: Skills survey will be posted tonight
  - Complete by Wednesday morning, should only take a few minutes
  - Will include fields for additional project ideas
- Project preferences survey will be posted Friday night
  - Will inform project brainstorming sessions next Tuesday

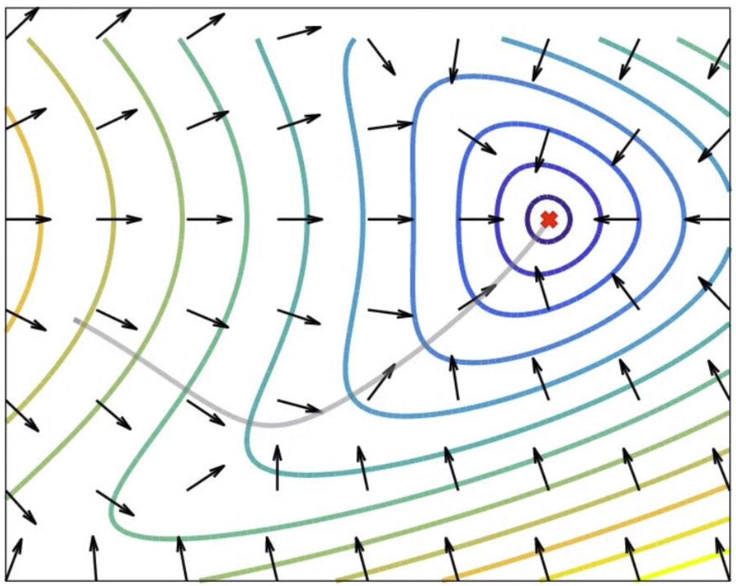
So Far… Motion Planning



Optimal Control

Optimal Control: Finding the best control policy for a desired goal

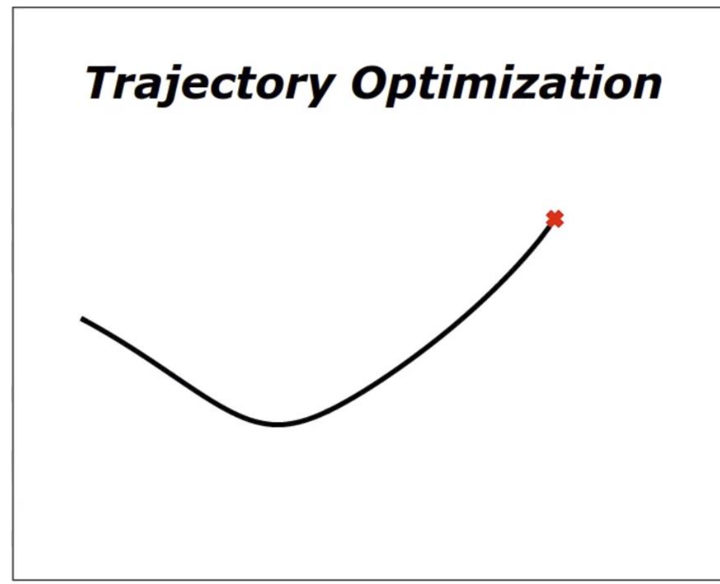
Closed-Loop Solutions



$$u = u(x)$$

"Global Method": Gives action at all states
Very expensive to compute

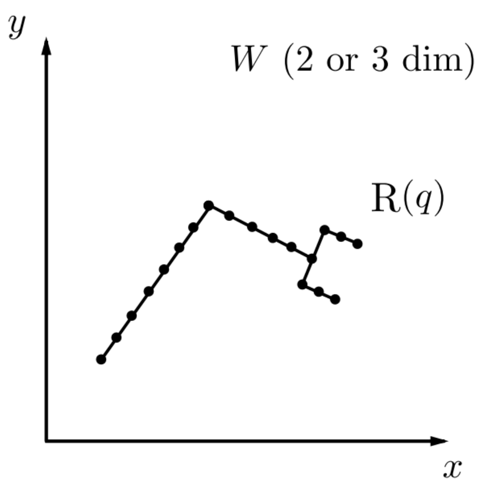
Open-Loop Solution

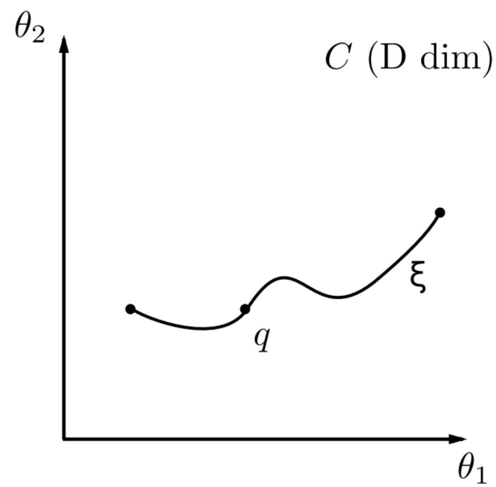


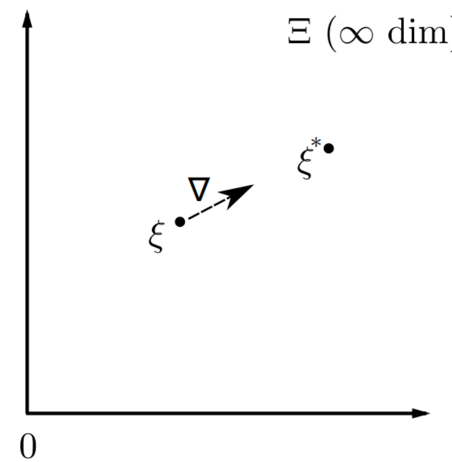
$$u = u(t)$$

"Local Method": Gives action at relevant states
Usable in high dimensions

Problem Specification: Spaces







World Space W($\mathbb{R}$^3)	Configuration Space C($\#DoF$)	Trajectory Space $\Xi(\infty\ dim)$

Robot pose in World Space (set of points) → Single point in Configuration Space

Trajectory through Configuration Space (set of points) → Single point in Trajectory Space

Making Trajectory Optimization Useful

Need to provide a good choice for $\mathbf{U}[\xi]$.

CHOMP: Covariant Hamiltonian Optimization for Motion Planning

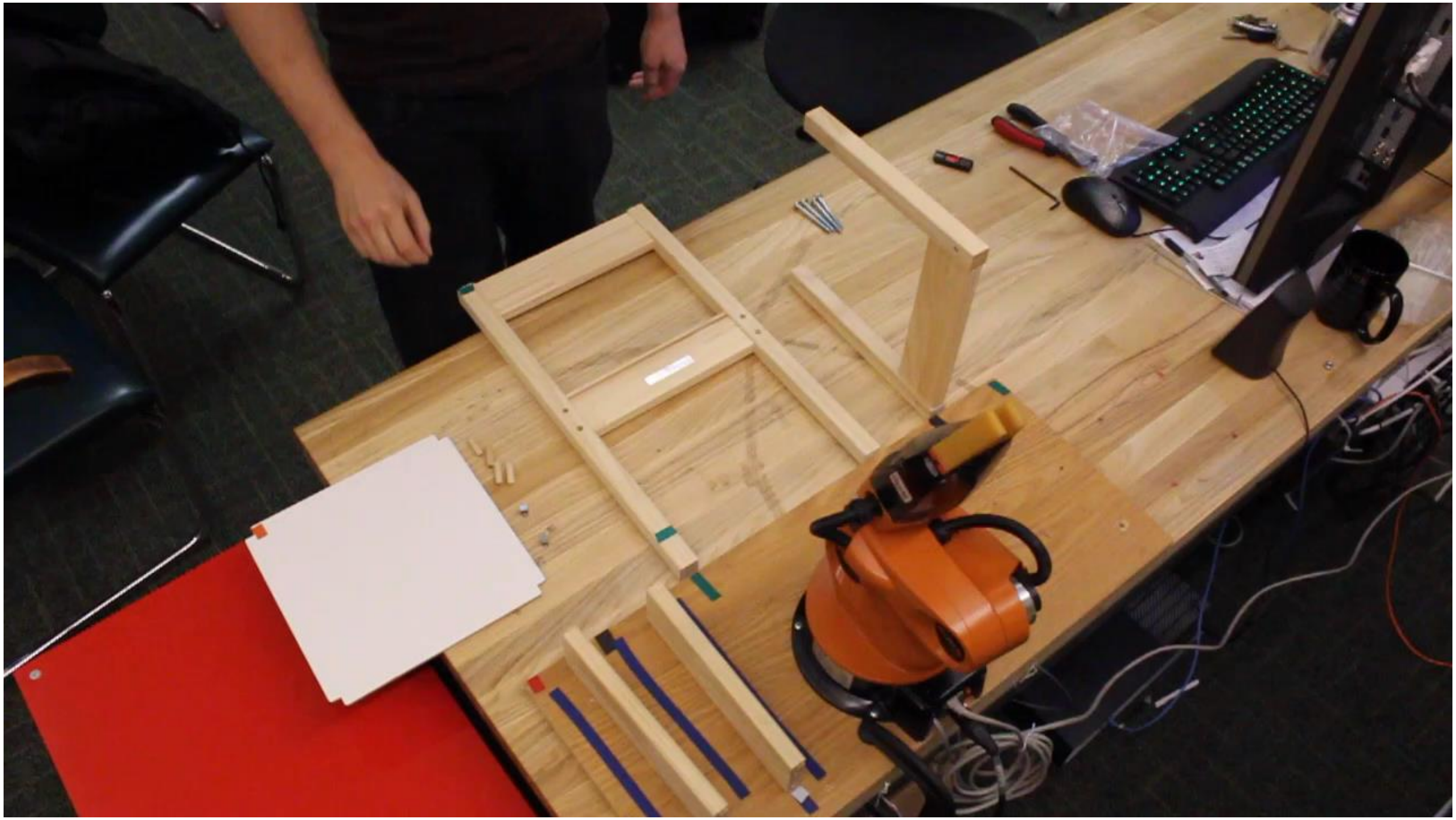
Uses a cost function $\boldsymbol{U[\xi] = U_{smooth}[\xi] + \lambda U_{obs}[\xi]}$

Smoothness cost: $\boldsymbol{U_{smooth}[\xi] = \frac{1}{2}\int_0^T ||\xi'(t)||^2 \, dt}$

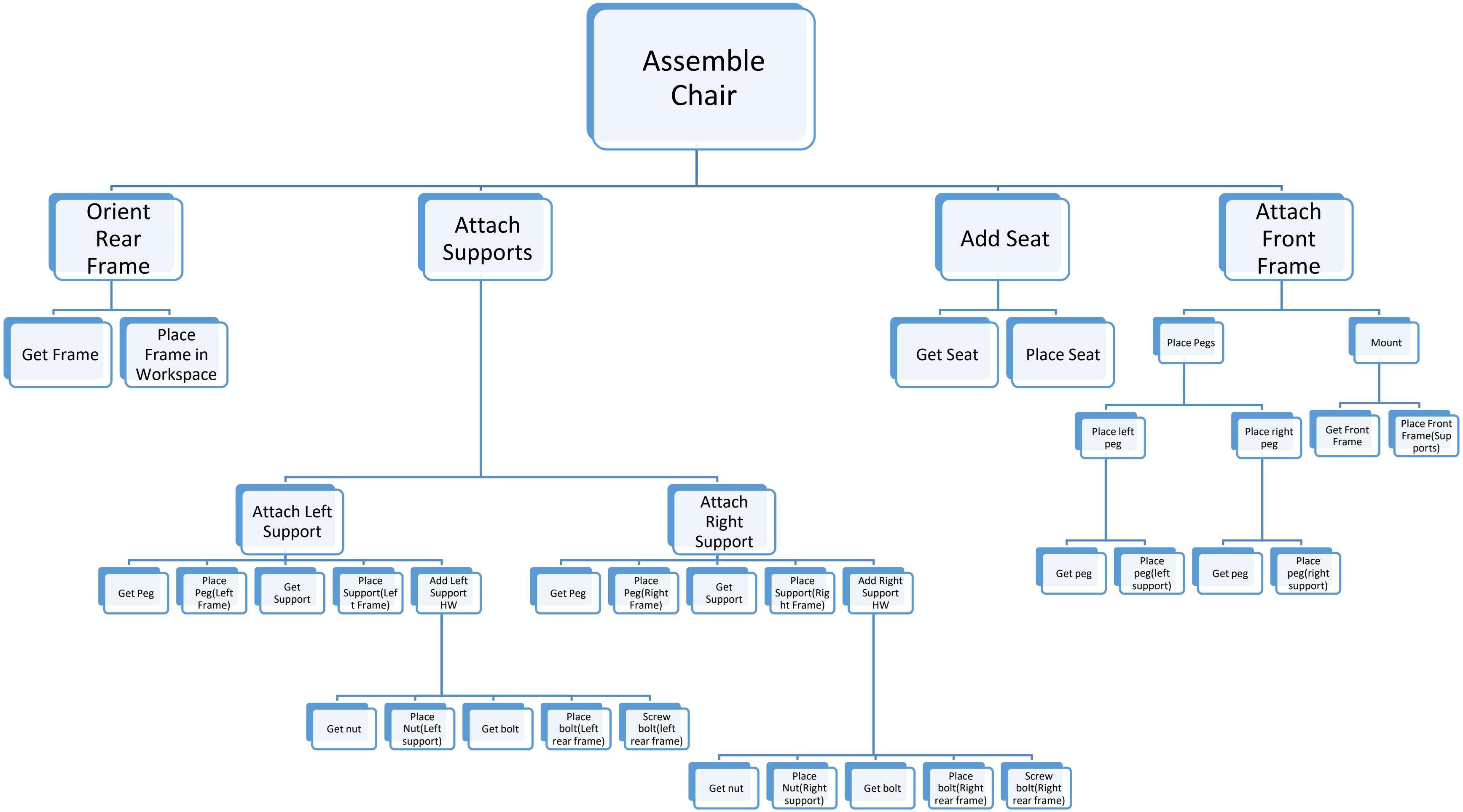
Obstacle cost: $\boldsymbol{U_{obs}[\xi] = \int_t \int_u c\left(\phi_u(\xi(t))\right) * \left\| \frac{d}{dt}\phi_u(\xi(t)) \right\| dudt}$

Cost function that computes distance to closest obstacle	Forward Kinematics function that computes location of robot body point u at time t in ξ	Norm of the velocity for body point u at time t in ξ

How do we generate motion plans for this?



How do we generate motion plans for this?



Artificial Intelligence vs. **Machine Learning** for Adversarial Environments

(a.k.a. How to play games and make it look like research)

- Search (AI) vs. Modeling (ML)
  - Can push complexity into different parts of the problem!
- Search techniques rely on informed heuristics to guide them efficiently to the goal
- Modeling techniques rely on troves of data to represent the statistical properties of the underlying solution.
- Not either/or, usually combination of both!

First Approach: Machine Learning

- Learn some function that takes state features as inputs and produces the next action as an output: $f: S \rightarrow A$

- Model updates with experience, rather than searching already-held knowledge
  - Small changes in training data can wildly change model behavior

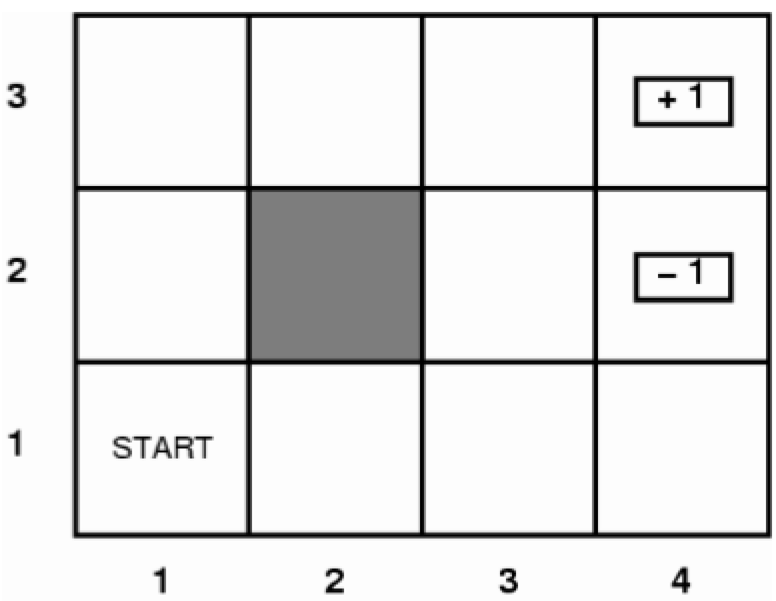
- Can be difficult to choose correct parameters for model, learning rate, etc. (without these, papers often irreproducible!)

- Can be difficult to obtain sufficient experience/training data

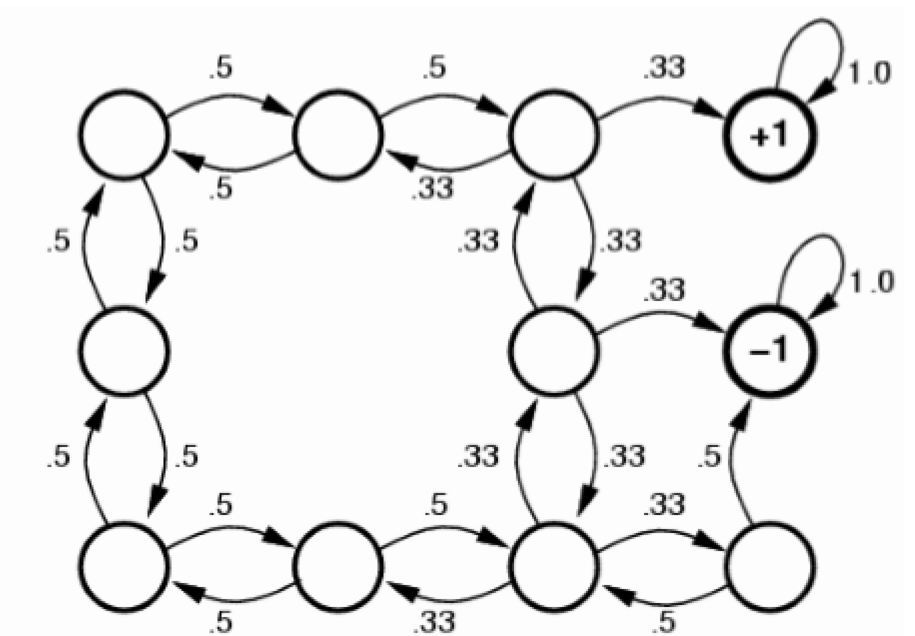
Solving Problems with Reinforcement Learning

Example world:

- Reward only defined at terminal states
- Equal transition probabilities between neighboring states



State Transitions

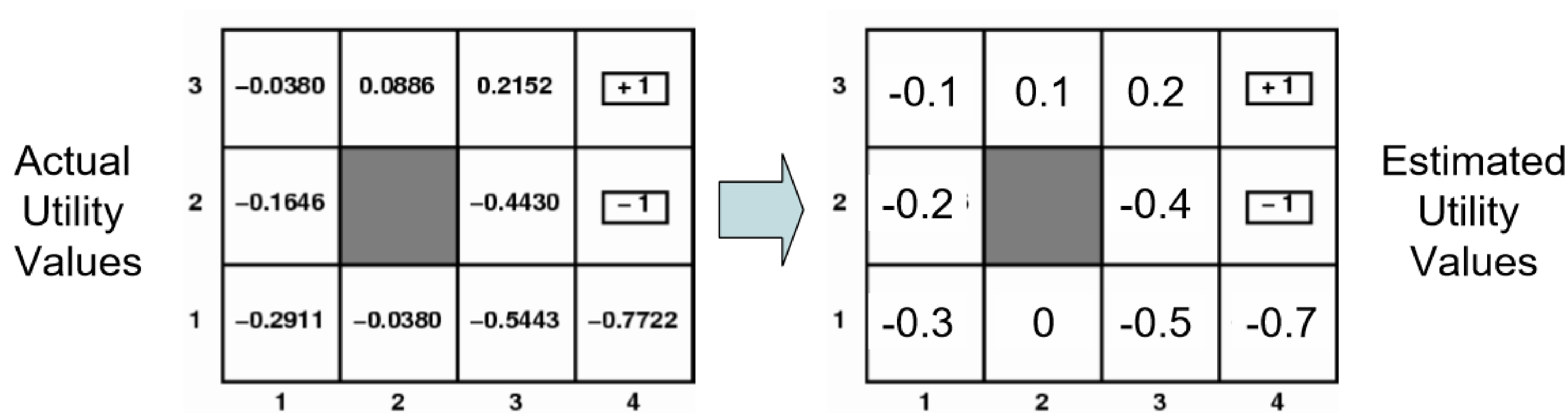


Solving Problems with Reinforcement Learning

Given a set of training sequences that end in a terminal state (with a reward)…

(1,1) → (1,2) → (1,3) → (2,3) → (3,3) → (4,3) → **+1**
(1,1) → (2,1) → (3,1) → (3,2) → (4,2) → **-1**
(1,1) → (2,1) → (3,1) → (3,2) → (3,3) → (4,3) → **+1**
(1,1) → (2,1) → (3,1) → (4,1) → (3,1) → (3,2) → (4,2) → **-1**

Determine the expected utility **U(s)** associated with each non-terminal state **s**



Learning an Action-Value Function (Q-Learning)

- Q-value is the expected utility of taking a given action in a given state:

  Q(s,a) = Value of action ‘a’ in state ‘s’

- For a state s, choose action ‘a’ that maximizes Q(s,a)
- Q-values do not require an environment model:
  The transition function isn’t necessary, since it will be learned from experience:

$$Q(s,a) = (1-\alpha) * Q(s,a) + \alpha(\ r + \ \gamma\ * \max_{a'} Q(s',a'))$$

Reliance on existing vs. new Knowledge

Current understanding

reward

Discount factor

Best we think we can do in the future from here

Solving Problems with Reinforcement Learning

(1,1) → (1,2) → (1,3) → (2,3) → (3,3) → (4,3) → **+1**
(1,1) → (2,1) → (3,1) → (3,2) → (4,2) → **-1**
(1,1) → (2,1) → (3,1) → (3,2) → (3,3) → (4,3) → **+1**
(1,1) → (2,1) → (3,1) → (4,1) → (3,1) → (3,2) → (4,2) → **-1**

Determine the expected utility **U(s)** associated with each non-terminal state **s**

Actual Utility Values

3	−0.0380	0.0886	0.2152	+1
2	−0.1646		−0.4430	−1
1	−0.2911	−0.0380	−0.5443	−0.7722
	1	2	3	4

Estimated Utility Values

3	-0.1	0.1	0.2	+1
2	-0.2		-0.4	−1
1	-0.3	0	-0.5	-0.7
	1	2	3	4

$$Q(s,a) = (1-\alpha) * Q(s,a) + \alpha(r + \gamma * \max_{a'} Q(s',a'))$$

Types of Reinforcement Learning:

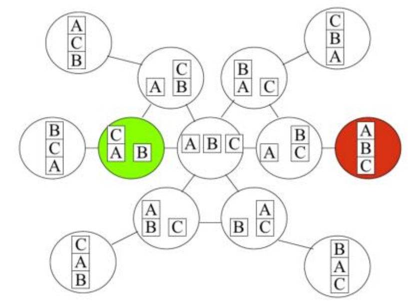
Utility Learning

- Learn a utility function that maps states to utilities and select an action by maximizing the expected value
- Needs a model of the environment (needs transition function)
- Predictive

Action-Value Learning

- Learn an action-value function that gives the expected utility of taking a given action in a given state
- No need for an environment model
- Don’t know where actions lead, so cannot look ahead

Task Planning: General Idea



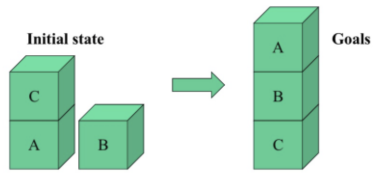
Task Planning is a **model-based** approach to autonomous behavior

Classical Planning:

Deterministic, Full information

Robotics Planning:

Non-deterministic, Partial information



Planning Outline

- Introduction to Planning and Problem Solving
- Classical Planning: Deterministic actions and complete information
- Probabilistic Models: Markov Decision Processes (MDPs), and Partially Observable MDPs (POMDPs)
- Handling Uncertainty

– Reference: A concise introduction to models and methods for automated planning, H. Geffner and B. Bonet, Morgan & Claypool, 6/2013.

– Other references: Automated planning: theory and practice, M. Ghallab, D. Nau, P. Traverso. Morgan Kaufmann, 2004, and Artificial intelligence: A modern approach. 3rd Edition, S. Russell and P. Norvig, Prentice Hall, 2009.

Planning: High Level

- Thinking before acting
- Determining how to achieve a given goal

Three approaches to the control problem (what to do next):

1. Programming-based: Specify control by hand
2. Learning-based: Learn control from experience

Inverse Reinforcement Learning (MaxEnt IRL)	Reinforcement Learning (SARSA, REINFORCE)

3. Model-based (Planning): Derive control from a domain model

$$\textit{Problem} \Longrightarrow \boxed{\textit{Solver}} \Longrightarrow \textit{Solution}$$

Planners, Models, and Solvers

Solvers for a given model type (e.g., SAT, MDP) are general

(Not tailored to specific instances)

Primary challenge is to **scale solvers up**

Methodology is **empirical**: benchmarks and competitions

Constraint Satisfaction (SAT)

SAT is the problem of determining whether there is a truth assignment that satisfies a set of clauses:

$$x \vee \neg y \vee z \vee \neg w \vee \ldots$$

Problem is NP-Complete, which in practice means worst-case behavior of SAT algorithms is exponential in number of variables ($2^{100} = 10^{30}$)

Yet current SAT solvers manage to solve problems with **thousands of variables and clauses**, and used widely (circuit design, verification, planning, etc)

Fully Observable, Deterministic: Classical AI Planning

- finite and discrete state space S
- a **known initial state** $s_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **deterministic transition function** $s' = f(a, s)$ for $a \in A(s)$
- positive **action costs** $c(a, s)$

A **solution** or **plan** is a sequence of applicable actions $\pi = a_0, \ldots, a_n$ that maps s_0 into S_G; i.e., there are states $s_0, \ldots, s_{n+1}$ s.t. $s_{i+1} = f(a_i, s_i)$ and $a_i \in A(s_i)$ for $i = 0, \ldots, n$, and $s_{n+1} \in S_G$.

The plan is **optimal** if it minimizes the **sum of action costs** $\sum_{i=0,n} c(a_i, s_i)$. If costs are all 1, plan cost is plan **length**

Different **models** obtained by relaxing assumptions in **bold** . . .

Uncertainty but No Feedback: Conformant Planning

- finite and discrete state space S
- a **set of possible initial state** $S_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **non-deterministic** transition function $F(a, s) \subseteq S$ for $a \in A(s)$
- uniform action costs $c(a, s)$

A **solution** is still an **action sequence** but must achieve the goal for **any possible initial state and transition**

More complex than **classical planning**, verifying that a plan is **conformant** intractable in the worst case; but special case of **planning with partial observability**

Planning with Markov Decision Processes

MDPs are **fully observable, probabilistic** state models:

- a state space S
- initial state $s_0 \in S$
- a set $G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each state $s \in S$
- **transition probabilities** $P_a(s'|s)$ for $s \in S$ and $a \in A(s)$
- action costs $c(a, s) > 0$

– **Solutions** are **functions (policies)** mapping states into actions

– **Optimal** solutions minimize **expected cost** to goal

What about uncertainty?

Adding Uncertainty to MDPs

- Traditional MDPs are defined with:
  - States – {(0,0), (0,1), …}
  - Actions – {move_north, …}
  - Rewards – R(S,A,S’) -> Reward
  - Transition Probabilities – P(State | State,Action)

- But what if I don’t know what state I’m in?

  - Sometimes state variables can be ***latent***
    (e.g., “is the campus wi-fi working?”,
    “is my teammate in a bad mood today?”)

  - Rather than directly measuring them, we receive ***observations***
    (e.g., “nobody’s staring into their laptops”,
    “my teammate just punched Sawyer in its tablet-face”)

Adding Uncertainty to MDPs

- Traditional MDPs are defined with:
  - States – {(0,0), (0,1), …}
  - Actions – {move_north, …}
  - Rewards – R(S,A,S') -> Reward
  - Transition Probabilities – P(State | State,Action)

- But what if I don't know what state I'm in?

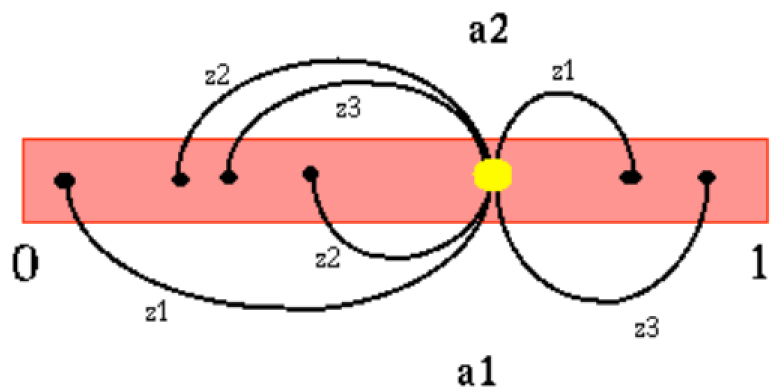
  - Rather than maintaining a "current state", maintain a **belief distribution**
    - A distribution over states indicating the probability that I think I'm in each one

  - By taking an **action** in a **state**, I receive **observations** that tell me about the state I just entered

Partially Observable MDPs (POMDPs)

- Traditional MDPs are defined with:
  - States – S = {(0,0), (0,1), …}
  - Actions – A = {move_north, …}
  - Rewards – R(s,a,s') -> Reward
  - Transition Probabilities – T(s,a,s') -> P(s' | s, a)
- Now we have to add:
  - Observation set – $\mathrm{O} = \{o_1, o_2, \ldots\}$
  - Observation prob. – $\Omega = P(o_1, \ldots, o_i \mid S)$
- Also have to augment:
  - Current State (now *belief*) – B = [0.1, 0.6, 0.2, 0.1, …]
  - Policy *(no longer $S \rightarrow A$)* – $\pi: B \rightarrow A$

POMDP: Trivial Example



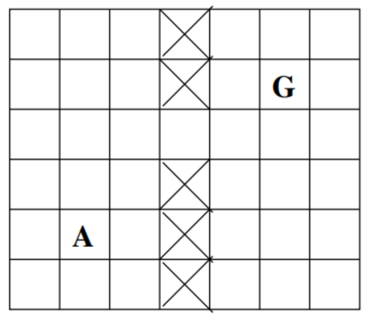
Two states: {0,1}   Two Actions: {a1, a2}

Three Observations: {z1, z2, z3}

- A dot's position in the red bar indicates our belief over these states.
  (Yellow is current belief)
- B = [p, 1-p] indicates $p\%$ chance of being in State 0, and $1 - p\%$ chance of being in State 1.
- Executing a1 and observing z3 tells us that we're very likely to be in State 1
- Executing a1 and observing z1 tells us that we're very likely to be in State 0

Identifying Types of Planning Problems

Agent **A** must reach **G**, moving one cell at a time in **known** map



- If actions deterministic and initial location known, planning problem is **Classical**
- If actions non-deterministic and location observable, it's an **MDP** or **FOND**
- If actions non-deterministic and location partially obs, **POMDP** or **Contingent**

Different combinations of uncertainty and feedback: diff problems, diff models

Planner is generic solver for instances of a particular model

Classical planners, MDP Planners, POMDP planners, . . .

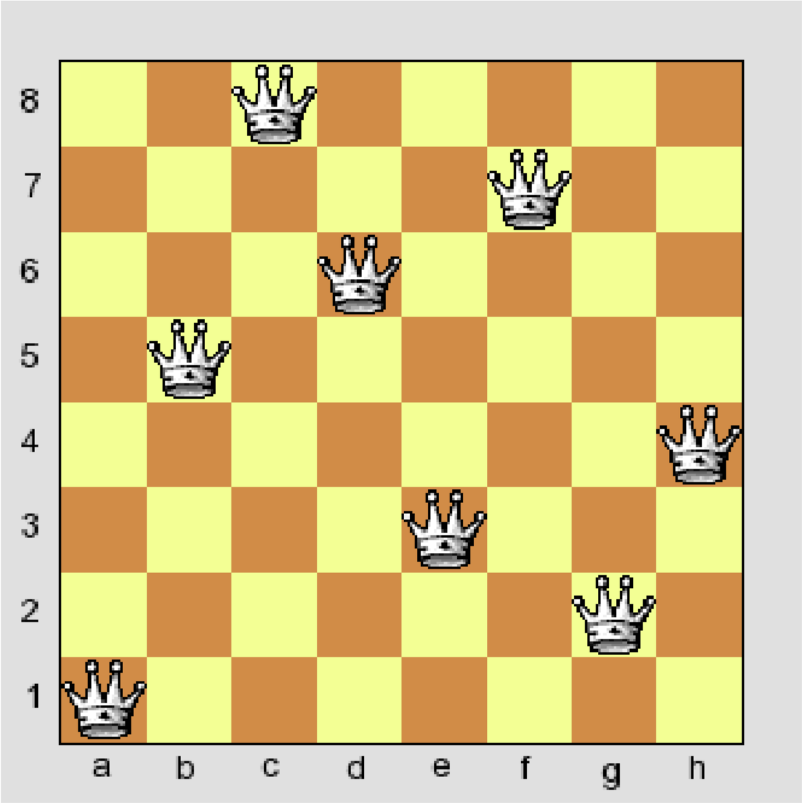
STRIPS: Language for Classical Planning

- A **problem** in Strips is a tuple $P = \langle F, O, I, G \rangle$:
  - ▷ F stands for set of all **atoms** (boolean vars)
  - ▷ O stands for set of all **operators** (actions)
  - ▷ $I \subseteq F$ stands for **initial situation**
  - ▷ $G \subseteq F$ stands for **goal situation**

- Operators $o \in O$ **represented** by
  - ▷ the **Add** list $Add(o) \subseteq F$
  - ▷ the **Delete** list $Del(o) \subseteq F$
  - ▷ the **Precondition** list $Pre(o) \subseteq F$

- Pickup(X)
  - P: grip(∅) $\wedge$ clear(X) $\wedge$ ontable(X)
  - A: grip(X)
  - D: onTable(X) $\wedge$ grip(∅)

Problem Formulation Matters!
The 8 Queens Problem



- Formulation #1:
  - Place a queen on any open square
  - Repeat until all queens are placed
  - State space of $\frac{64!}{56!} = 1.78 * 10^{14}$
- Formulation #2:
  - Place a queen on any row 1 square
  - Place a queen on any row 2 square
  - State space of $8^8 = 1.68 * 10^7$
- Formulation #3:
  - Place a queen on any row 1 square
  - Place a queen on any row 2 square not sharing a column…
  - State space of $8! = 40{,}320$

Planning across length scales