CMRoboBits:

Policy Learning and Multi-robot Coordination

Francisco Melo

Manuela Veloso

15-491, Fall 2008

http://www.andrew.cmu.edu/course/15-491

Computer Science Department

Carnegie Mellon

Outline

- **Planning Under Uncertainty**
- Learning
- Multi-robot Coordination
- Learning Coordination
- Conclusions

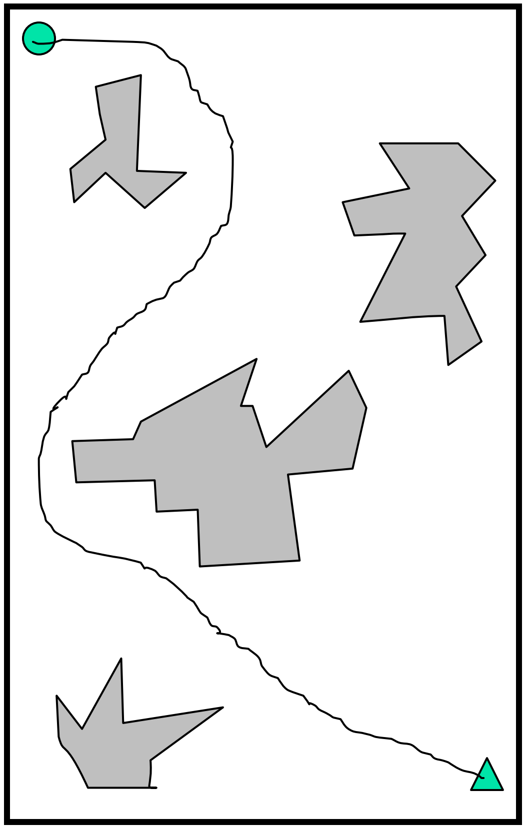
Planning Algorithm

Given:

- A goal;
- A description of available actions;

Output:

- A plan.



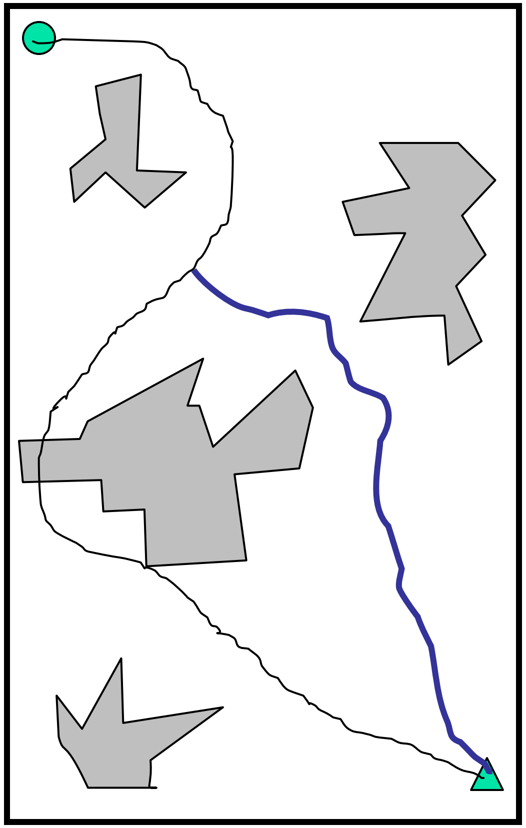
Planning and Execution

Given:

- A goal;
- A description of available actions;

Plan and execute:

- Follow a plan
- When needed – replan (e.g., ERRT)



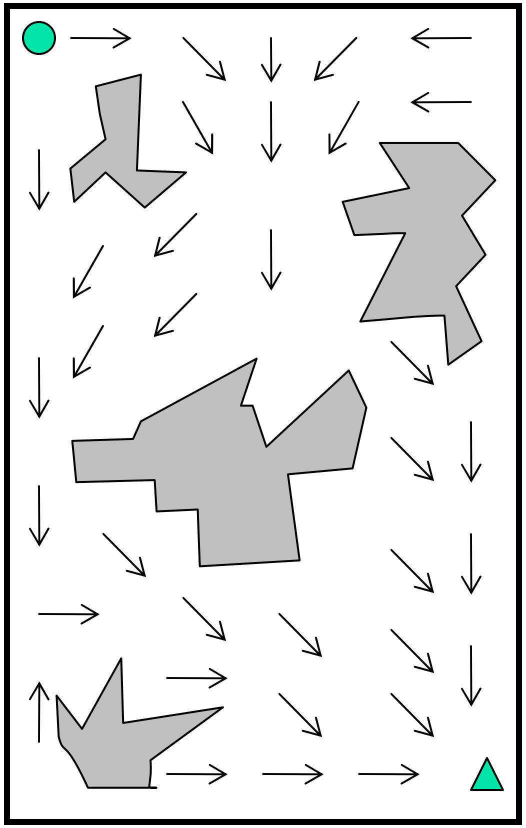
Planning Considering Uncertainty

Given:

- A goal;
- A description of available actions;

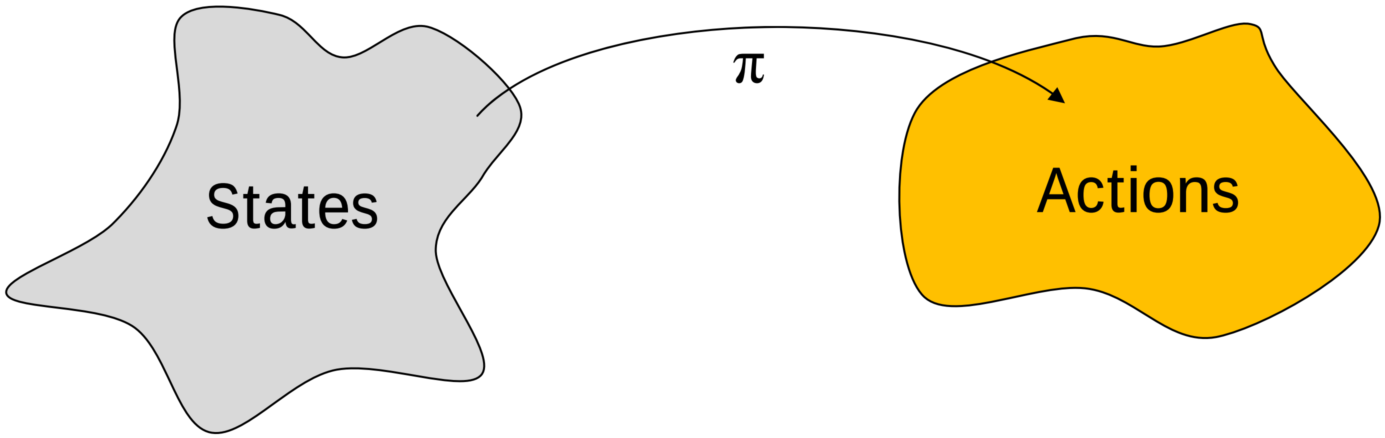
Output:

- A contingency plan (*an action to take at every state*) = a policy



Policy Definition

- A policy tells the robot what to do in every possible state;
- Reactive;
- Represented as a mapping:



Planning Using Policies

Advantages:

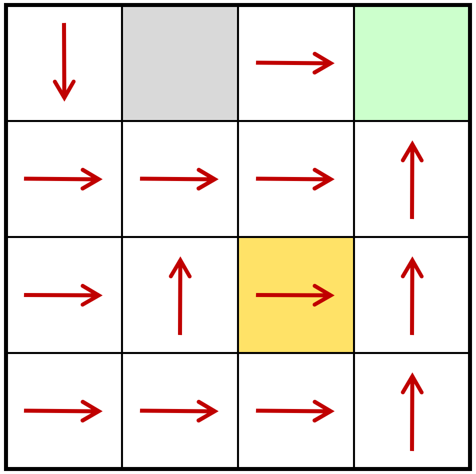
- Can handle uncertainty
- Simple and easy

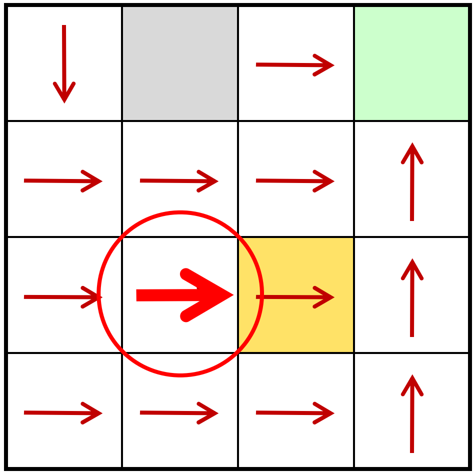
Disadvantages:

- Does not scale very well
- Same state → Same action

Evaluating a Policy

- A simple grid-world
- Which policy is better?
  - Policy 1
  - Policy 2

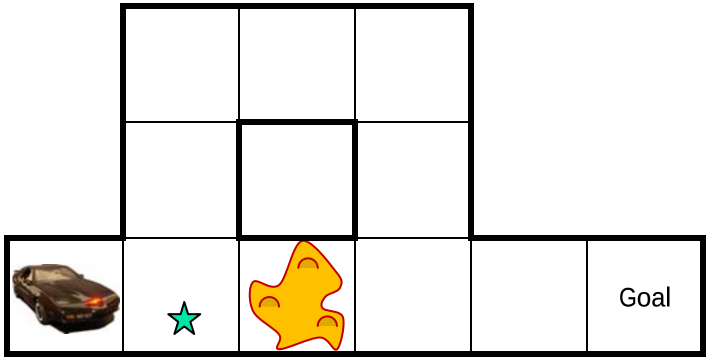




Answer: Policy 1.

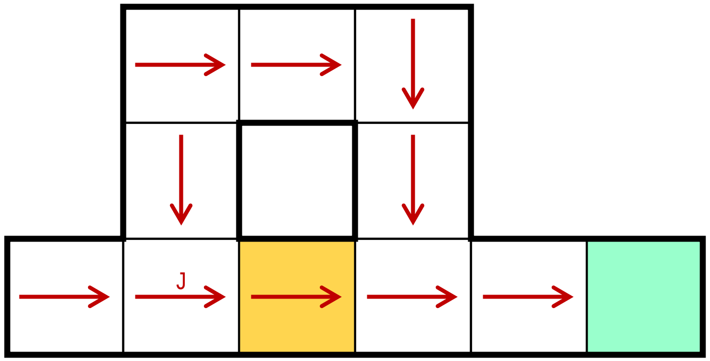
Evaluating a Policy II

- What about this case?
- Actions:
  - Up, Down, Left, Right, **Jump** (in the star-state)
  - Jump succeeds with probability *p*

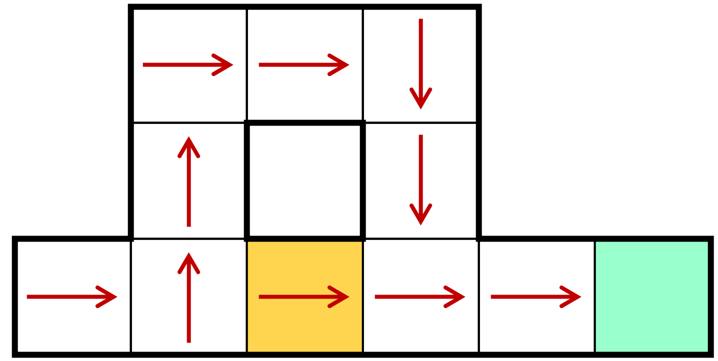


Evaluating a Policy III

- Which is the best policy?



Policy 1

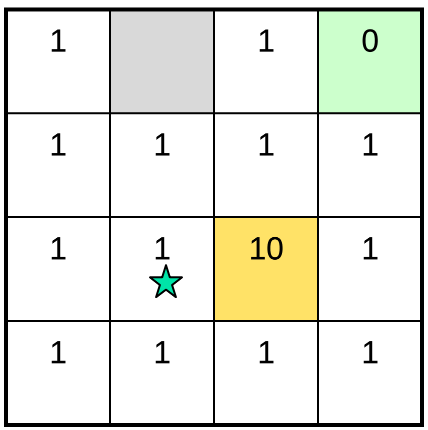


Policy 2

Answer: Depends on p and on how bad it is to fall into the lava.

Evaluating a Policy IV

- Costs in each transition or state;
- For the star-state:

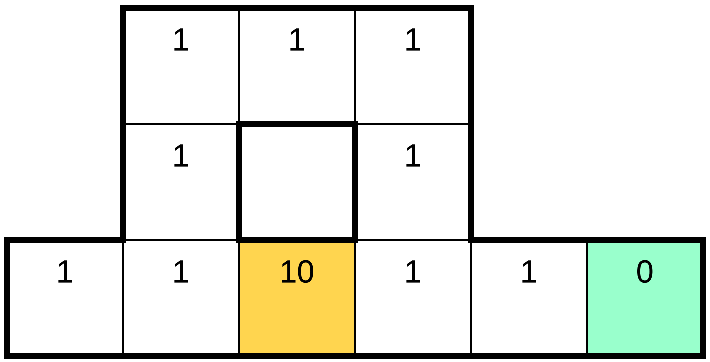


- Policy 1 (total cost): 4
- Policy 2 (total cost): 13

A policy is optimal if it minimizes total cost for all initial states

Evaluating a Policy V

- For the star-state, $p = 0.9$



  - Policy 1 (**expected** total cost): 4
  - Policy 2 (expected total cost): 8

- For the star-state, $p = 0.5$
  - Policy 1: 8
  - Policy 2: 8

Value of a Policy I

- An equivalent formulation:

0		0	1
0	0	0	0
0	0	-10	0
0	0	0	0

- Assign rewards to states;
- Neg. rewards = Penalties

Value of a policy at state *s* =

Total reward for starting at *s*

Value of a Policy II

- For a policy π,

“Probability” of moving *t* steps

$$V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t r(s_t) \mid s_0 = s$$

- What about uncertainty?

$$V^{\pi}(s) = \mathbf{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \mid s_0 = s\right]$$

SCS Carnegie Mellon

The Optimal Policy

- For the optimal policy, π^*,

$$V^*(s) \geq V^\pi(s)$$

- We can write

$$V^*(s) = \max_a \left[r(s) + \gamma \sum_{s'} \mathbf{P}_a(s,s') V^*(s') \right]$$

Probability of moving from *s* to *s'* with action *a*

The Value of an Action

- Invent Q-function: The value of action a in state s is:

$$Q^*(s,a) = r(s) + \gamma \sum_{s'} \mathbf{P}_a(s,s') V^*(s')$$

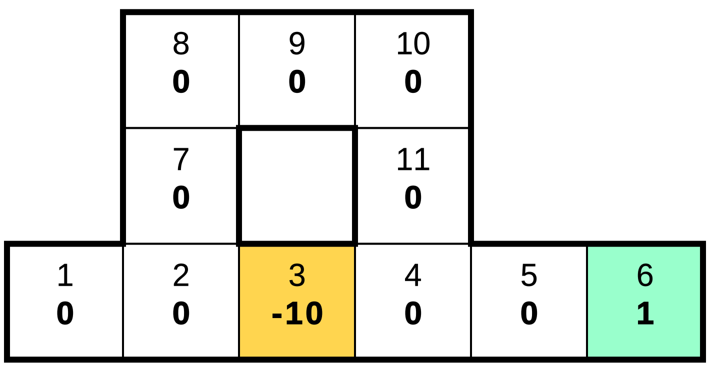
$$= r(s) + \gamma \sum_{s'} \mathbf{P}_a(s,s') \max_b Q^*(s',b)$$

- Gives a recipe to compute π^*:

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

Example of a *Q*–function:

- *Q*-function: Table with states as rows and actions as columns.



	☝	☟	☞	☜	✈
1	14.54	14.54	**15.3**	14.54	14.54
2	14.54	15.3	6.79	14.54	**16.11**
Lava	-3.21	-3.21	**7.15**	5.30	-3.21
4	16.29	17.15	**18.05**	6.79	17.15
5	18.05	18.05	**19**	17.15	18.05
Goal	**20**	**20**	**20**	19.05	**20**
7	13.97	**15.30**	14.54	14.54	14.54
8	13.97	14.54	**14.7**	13.97	13.97
9	14.7	14.7	**15.48**	13.97	14.7
10	15.48	**16.29**	15.48	14.7	15.48
11	15.48	**17.15**	16.29	16.29	16.29

How to Compute Q^*

- Dynamic Programming (DP) approach:
  - Start with some Q_0;
  - Repeat

$$Q_{k+1}(s,a) = r(s) + \gamma \sum_{s'} \mathbf{P}_a(s,s') \max_b Q_k(s',b)$$

until $\| Q_{k+1} - Q_k \|$ small.

Outline

- Planning Under Uncertainty
- **Learning**
- Multi-robot Coordination
- Learning Coordination
- Conclusions

DP Revisited

- Recall the DP iteration:

$$Q_{k+1}(s,a) = r(s) + \gamma \sum_{s'} \mathbf{P}_a(s,s') \max_b Q_k(s',b)$$

- Requires knowledge of **P** and *r*
- Iterates compute an expected value:

$$Q_{k+1}(s,a) = \mathbf{E}\left[r(s) + \gamma \max_b Q_k(s',b)\right]$$

Learning From Experience

- If **P** and *r* are unknown, expectation can be approximated by experience;
- Consider, for example, the deterministic case:

$$\mathbf{E}\left[r(s) + \gamma \max_b Q_k(s', b)\right] = r + \gamma \max_b Q_k(s', b)$$

Deterministic *Q*-learning

- We can replace

$$Q_{k+1}(s,a) = r(s) + \gamma \sum_{s'} \mathbf{P}_a(s,s') \max_b Q_k(s',b)$$

by

$$Q_{k+1}(s,a) = r + \gamma \max_b Q_k(s',b)$$

- Can be used from samples (s, a, r, s') to perform DP

Q-learning

- At iteration k, the sample error is:

$$Err_k = r + \gamma \max_b Q_k(s', b) - Q_k(s, a)$$

- We update Q_{k+1} proportionally to the error

$$Q_{k+1}(s, a) = Q_k(s, a) + \alpha \left[r + \gamma \max_b Q_k(s', b) - Q_k(s, a) \right]$$

Step-size

Remarks

- Q-learning is a reinforcement learning algorithm;
- The robot learns from a reinforcement signal (rewards/penalties);
- Requires exploration of the environment;

Reinforcement Learning

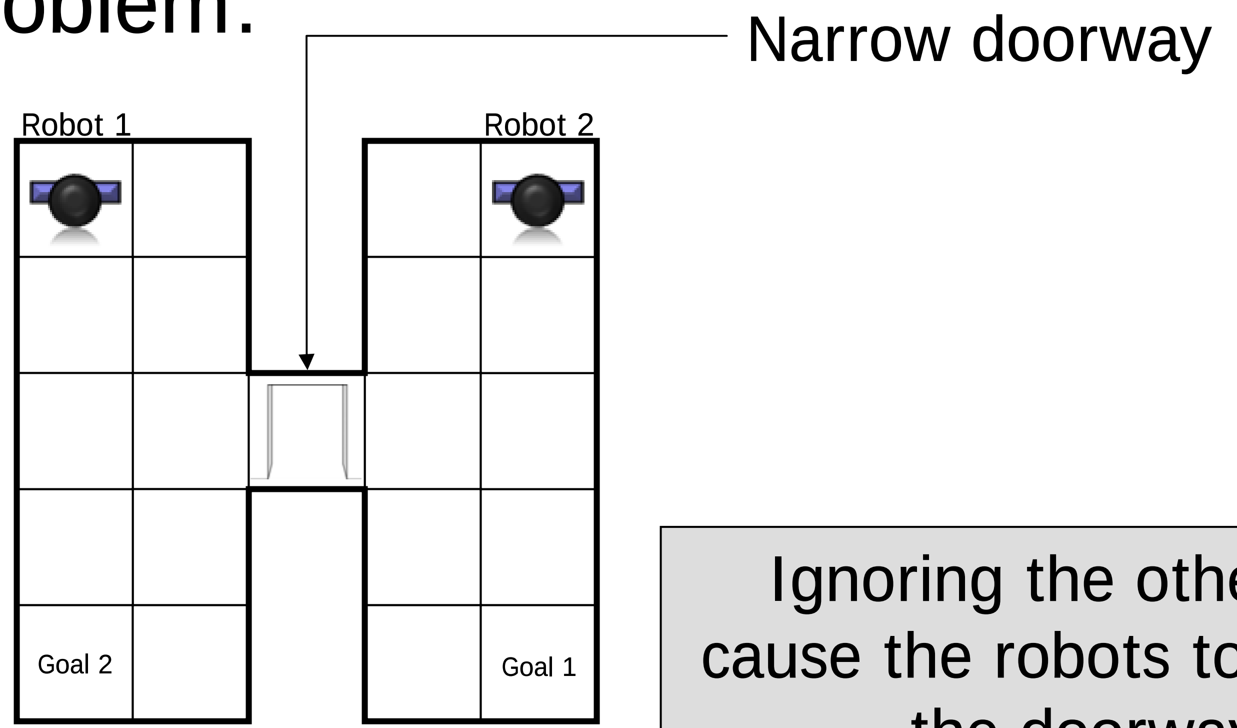
- Advantages
  - Requires no knowledge of the environment;
  - Can adapt to slowly changing environments;
- Disadvantages
  - Requires exploration (may be dangerous);
  - Does not handle rapidly changing domains;

Outline

- Planning Under Uncertainty
- Learning
- **Multi-robot Coordination**
- Learning Coordination
- Conclusions

What About Multiple Robots?

- Naïve approach: ignore others;
- Problem:



Ignoring the other may cause the robots to crash in the doorway.

DP for Multiple Robots

- "Optimal" solution: Jointly model the whole team;
  - State: (s_1, s_2)
  - Action: (a_1, a_2)
  - Reward: Common to all robots

State of robot 2

State of robot 1

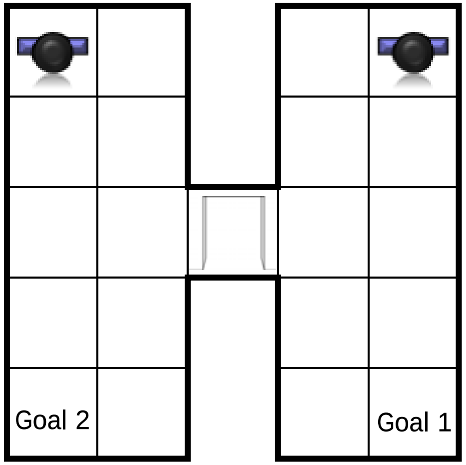
Action of robot 2

Action of robot 1

- Joint policy → Policy for both robots;

The Need for Coordination

- Joint Q^* enough for coordination?
  - Centralized decisions/communication;
  - Social conventions;
  - Other coordination mechanism;
- Example:
  - One of the robots must wait;
  - It is unimportant which one;
  - Both must agree which one;



Issues:

- State-action space grows exponentially with number of robots;
- Actions of one robot always depend on the other robots;
- Requires each robot to know where all robots are;

Outline

- Planning Under Uncertainty
- Learning
- Multi-robot Coordination
- **Learning Coordination**
- Conclusions

3 Simple Ideas I

- **Idea 1:** Use single robot policies when possible (decouple decisions)
  - When possible, “ignore” other agents in the environment
  - State-action space grows linearly with number of agents

3 Simple Ideas II

- **Idea 1 (cont.)**
  - Most of the time, actions of each robot no longer depend on the other robots;
  - Most of the time, each robot no longer needs to know where other robots are;

3 Simple Ideas III

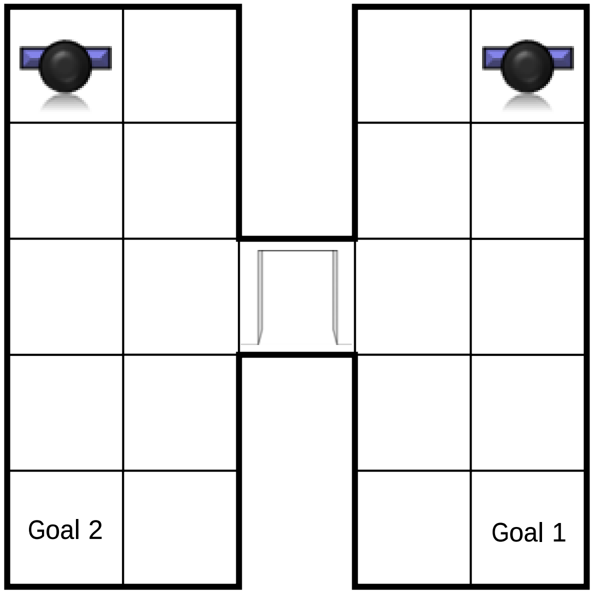
- **Idea 2:** Use communication to coordinate with other robots (when needed)
  - Handle miscoordinations;
  - Keep communication local (not global);
  - Consider local interactions;

3 Simple Ideas IV

- **Idea 3:** Learn when communication is necessary;
  - No pre-defined interaction;
  - Can be used in more general settings (non-cooperative/adversarial interactions);

Back to the Example

- Decouple robots decisions:
  - Each robot has to reach opposite corner;
  - Simultaneous doorway crossing still penalized;



Robots now coordinate only near the door using communication.

Outline

- Planning Under Uncertainty
- Learning
- Multi-robot Coordination
- Learning Coordination
- **Conclusions**

Conclusions I

- Policies provide “reactive behaviors” and handle uncertainty;
- To compute optimal policies, we can use
  - Dynamic Programming (model is known)
  - Reinforcement Learning (model is unknown)

Conclusions II

- Coordination is a fundamental issue in multi-robot domains
- Must be addressed explicitly:
  - Centralized decisions/communication;
  - Social conventions;
  - …
- Decoupled decisions and communication can diminish complexity of multi-robot domains

What to Take Home

- To handle uncertainty, robot relies on contingency plans – policies
- Compute optimal policies:
  - Using DP (model known)
  - Using RL (model unknown)
- Coordination is fundamental in multi-robot domains
- Decoupled decisions and communication can diminish complexity of multi-robot domains