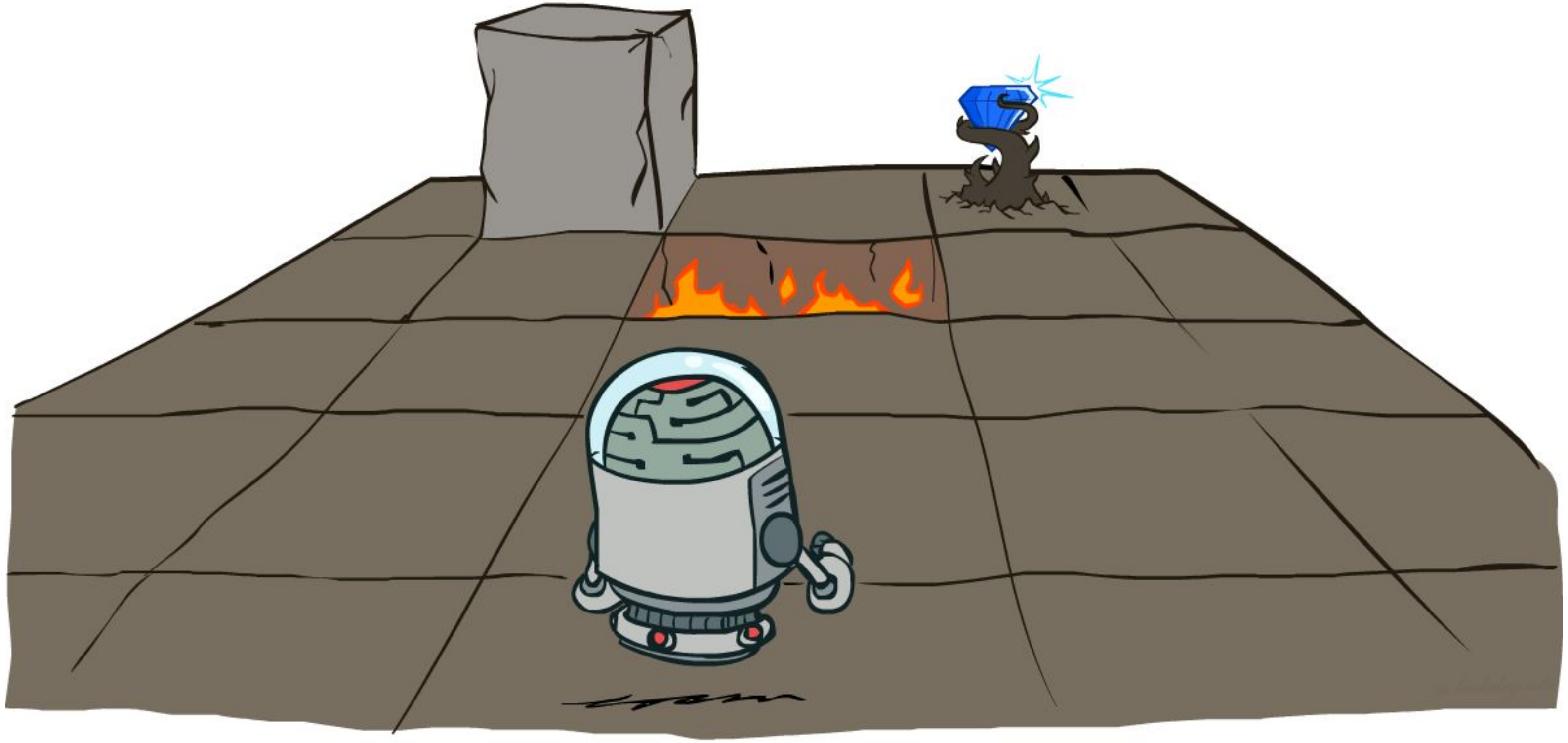
CS 188: Artificial Intelligence

Markov Decision Processes



Instructor: Stuart Russell and Dawn Song

University of California, Berkeley

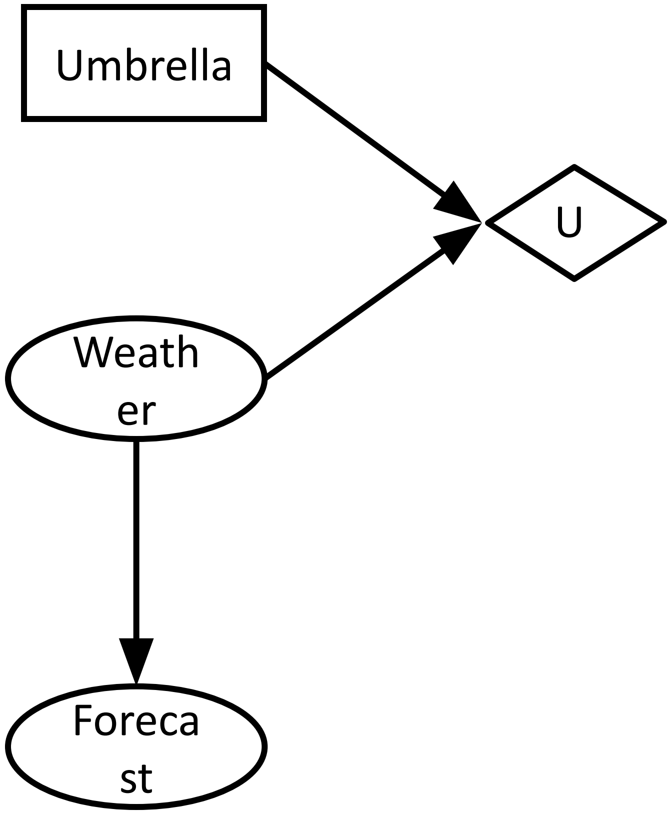
Recap: Decision Networks

- What is a decision network?

Recap: Decision Networks

- Decision network = Bayes net + Actions + Utilities
  - ***Action nodes*** (rectangles, cannot have parents, will have value fixed by algorithm)
  - ***Utility nodes*** (diamond, depends on action and chance nodes)
- Decision network represents a decision problem, containing all the information needed to for the agent to decide

What types of decisions?

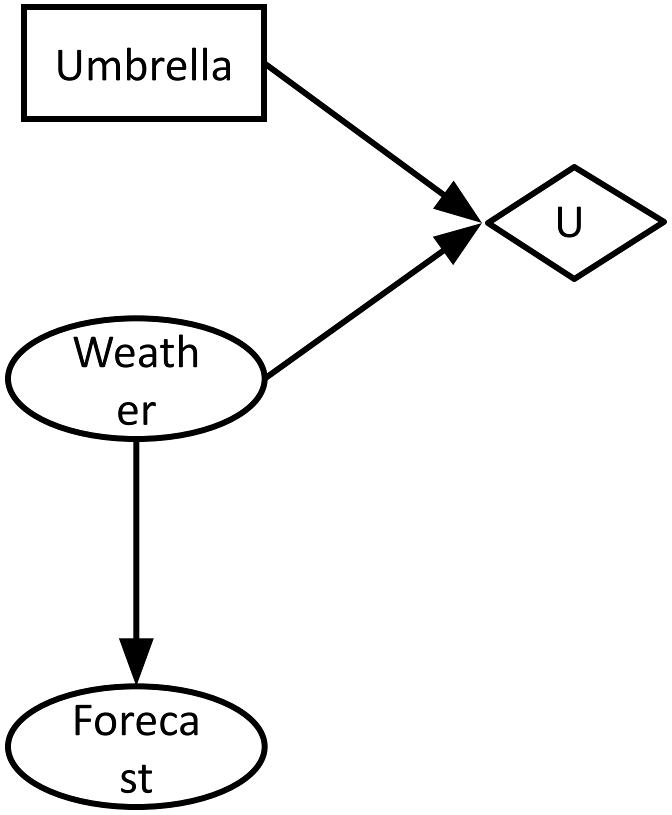


Recap: Decision Networks

- Decision network = Bayes net + Actions + Utilities

  - ***Action nodes*** (rectangles, cannot have parents, will have value fixed by algorithm)

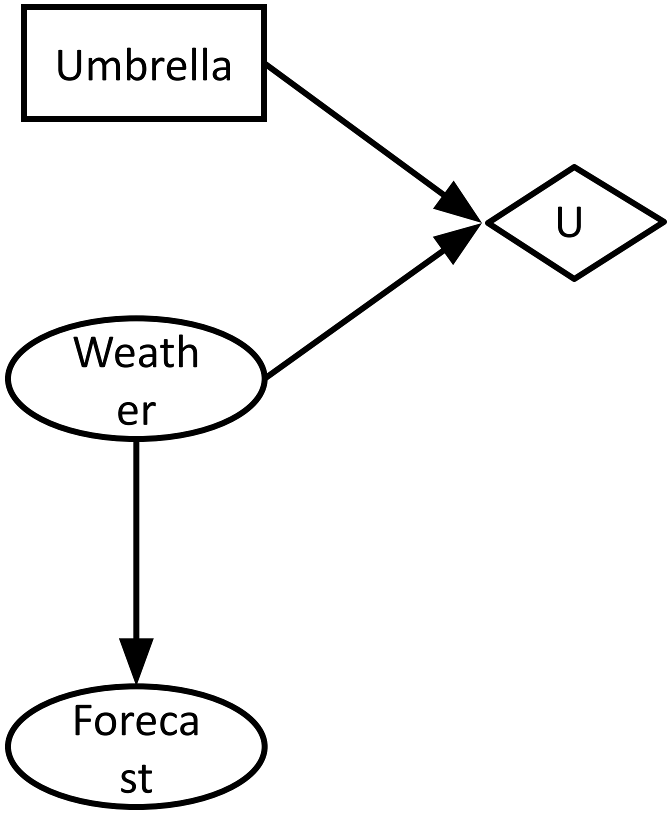
  - ***Utility nodes*** (diamond, depends on action and chance nodes)



- Decision network represents a decision problem, containing all the information needed to for the agent to decide
  - What action to take given evidence ***e***
    - Decision algorithm
      - Fix evidence ***e***
      - For each possible action a
        - Fix action node to a
        - Compute posterior $P(\mathbf{W}|\mathbf{e},a)$ for parents $\mathbf{W}$ of U
        - Compute expected utility $\sum_{\mathbf{w}} P(\mathbf{w}|\mathbf{e},a)\ U(a,\mathbf{w})$
      - Return action with highest expected utility

Recap: Decision Networks

- Decision network = Bayes net + Actions + Utilities
  - ***Action nodes*** (rectangles, cannot have parents, will have value fixed by algorithm)
  - ***Utility nodes*** (diamond, depends on action and chance nodes)
- Decision network represents a decision problem, containing all the information needed to for the agent to decide
  - What action to take given evidence ***e***
    - Decision algorithm
  - Value of information
    - $\text{VPI}(E_i \mid e) = [\sum_{e_i} P(e_i \mid e) \max_a \text{EU}(a \mid e_i, e)] - \max_a \text{EU}(a \mid e)$



Decisions with unknown preferences

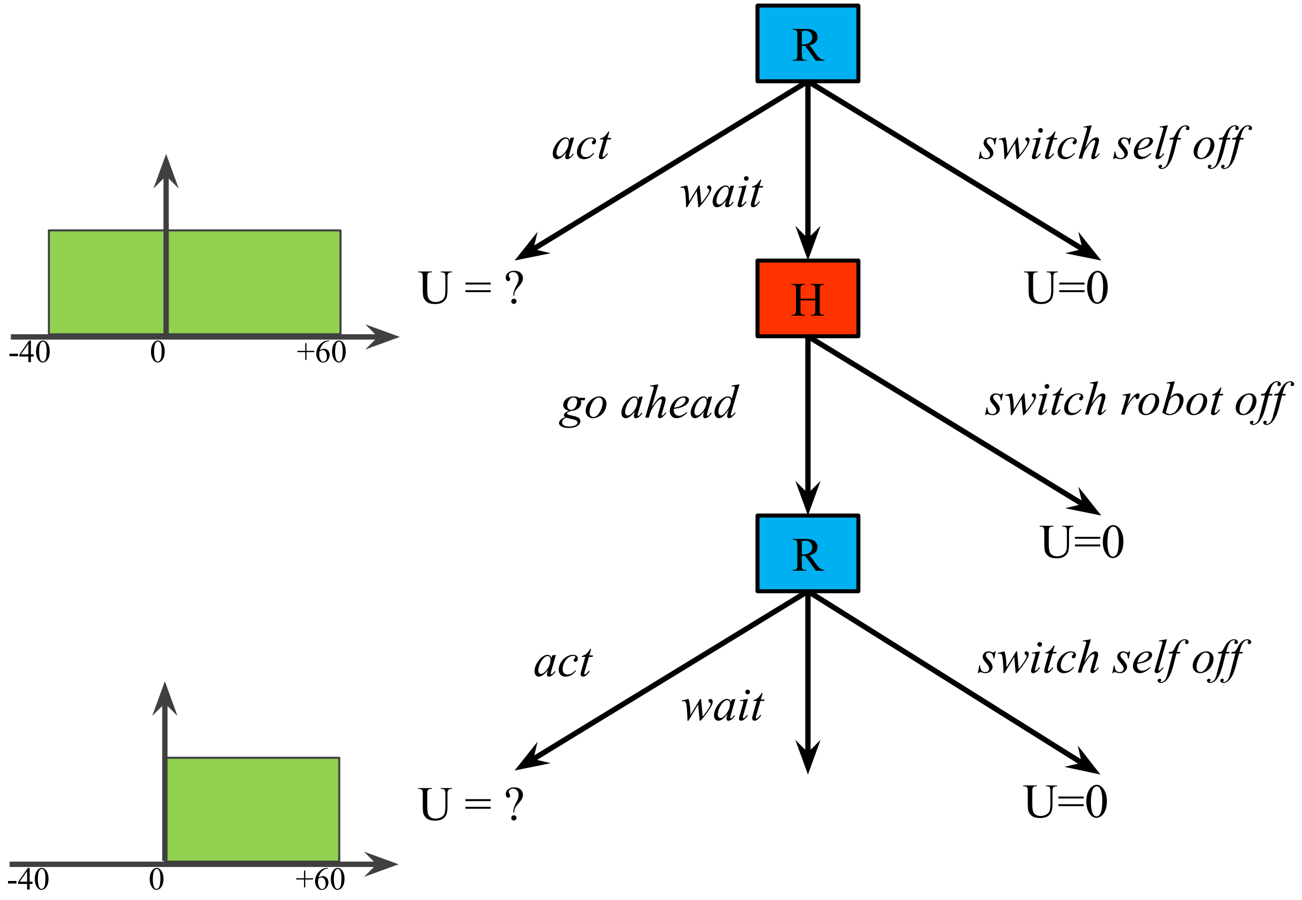
- In reality the assumption that we can write down our exact preferences for the machine to optimize is false
- A machine optimizing the wrong preferences causes problems

I’m sorry, Dave, I’m afraid I can’t do that

Decisions with unknown preferences

- In reality the assumption that we can write down our exact preferences for the machine to optimize is false
- A machine optimizing the wrong preferences causes problems
- A machine that is explicitly uncertain about the human’s preferences will defer to the human (e.g., allow itself to be switched off)

Off-switch problem (example)



EU(act) = +10

EU(wait) = (0.4 * 0) + (0.6 * 30) = +18

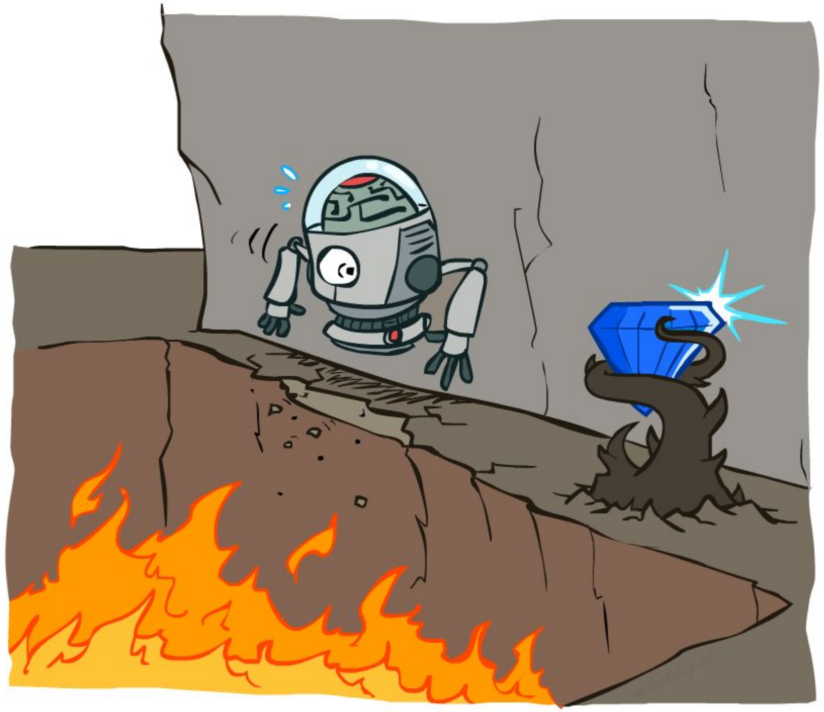
Off-switch problem (general proof)

- $EU(act) = \int_{-\infty}^{+\infty} P(u) \cdot u \, du = \int_{-\infty}^{0} P(u) \cdot u \, du + \int_{0}^{+\infty} P(u) \cdot u \, du$
- $EU(wait) = \int_{-\infty}^{0} P(u) \cdot 0 \, du + \int_{0}^{+\infty} P(u) \cdot u \, du$
- Obviously $\int_{-\infty}^{0} P(u) \cdot u \, du \leq \int_{-\infty}^{0} P(u) \cdot 0 \, du$
- Hence $EU(act) \leq EU(wait)$

Sequential decisions under uncertainty

So far, decision problem is one-shot --- concerning only one action

Sequential decision problem: agent's utility depends on a sequence of actions



Markov Decision Process (MDP)

- Environment history: $[s_0, a_0, s_1, a_1, \ldots, s_t]$
- "Markov" generally means that given the present state, the future and the past are independent
- For Markov decision processes, "Markov" means action outcomes depend only on the current state

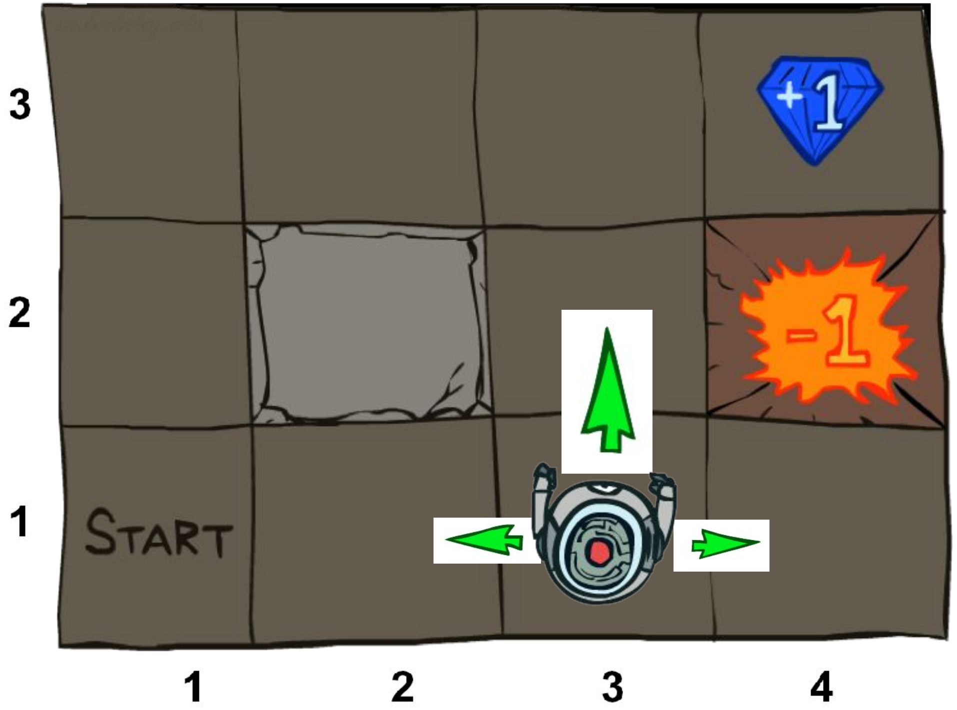
$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots S_0 = s_0)$$
$$=$$
$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

- This is just like search, where the successor function could only depend on the current state (not the history)

Andrey Markov
(1856-1922)

Markov Decision Process (MDP)

- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition model $T(s, a, s')$
    - Probability that a from s leads to s', i.e., $P(s' \mid s, a)$
  - A reward function $R(s, a, s')$ for each transition
  - A start state
  - Possibly a terminal state (or absorbing state)
  - Utility function which is additive (discounted) rewards

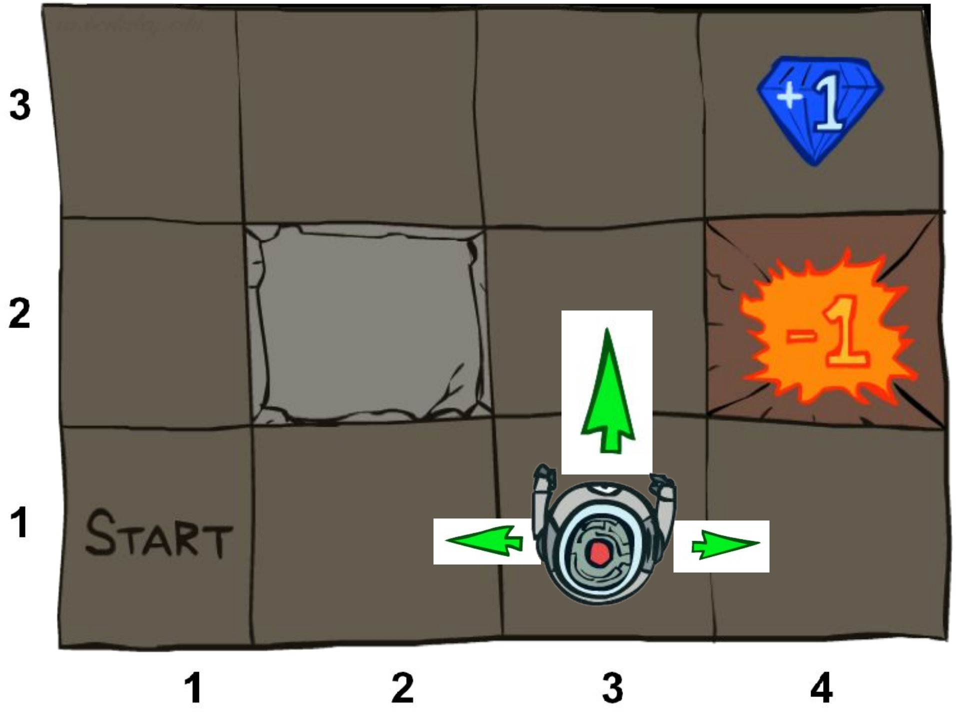


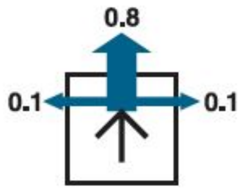
- MDPs are fully observable but probabilistic search problems

[Demo – gridworld manual intro (L8D1)]

Example: Grid World

- A maze-like problem
  - The agent lives in a grid
  - Walls block the agent's path
- Noisy movement: actions do not always go as planned
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put
- The agent receives rewards each time step
  - Small "living" reward r each step (can be negative)
  - Big rewards come at the end (good or bad)
- Goal: maximize sum of rewards



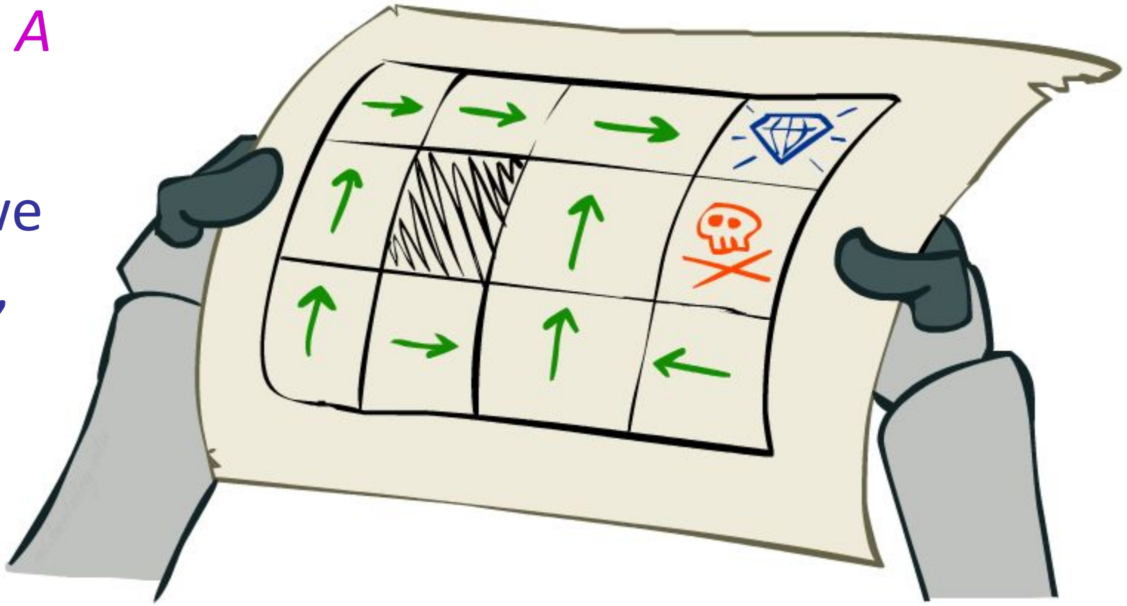


Policies

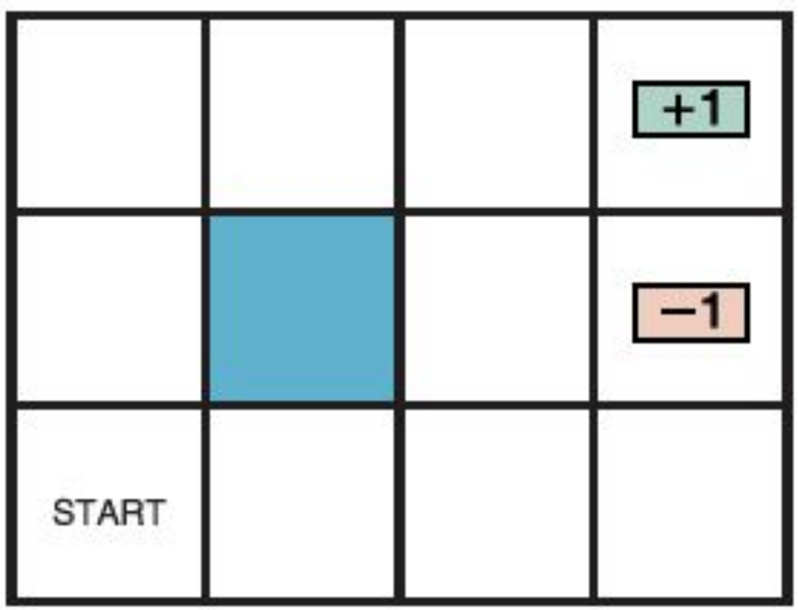
- A policy π gives an action for each state, π: *S* → *A*

- In deterministic single-agent search problems, we wanted an optimal ***plan***, or sequence of actions, from start to a goal

- For MDPs, we want an optimal ***policy*** π*: *S* → *A*
  - An optimal policy maximizes expected utility
  - An explicit policy defines a reflex agent

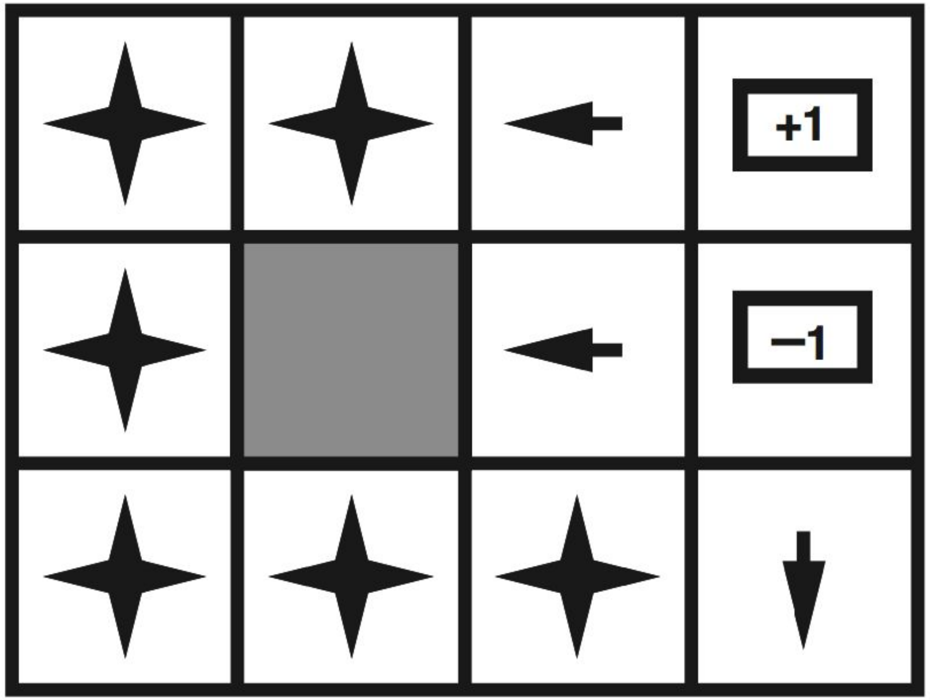


Optimal policy for r>0



r > 0

Optimal policy for r>0



r > 0

Sample Optimal Policies

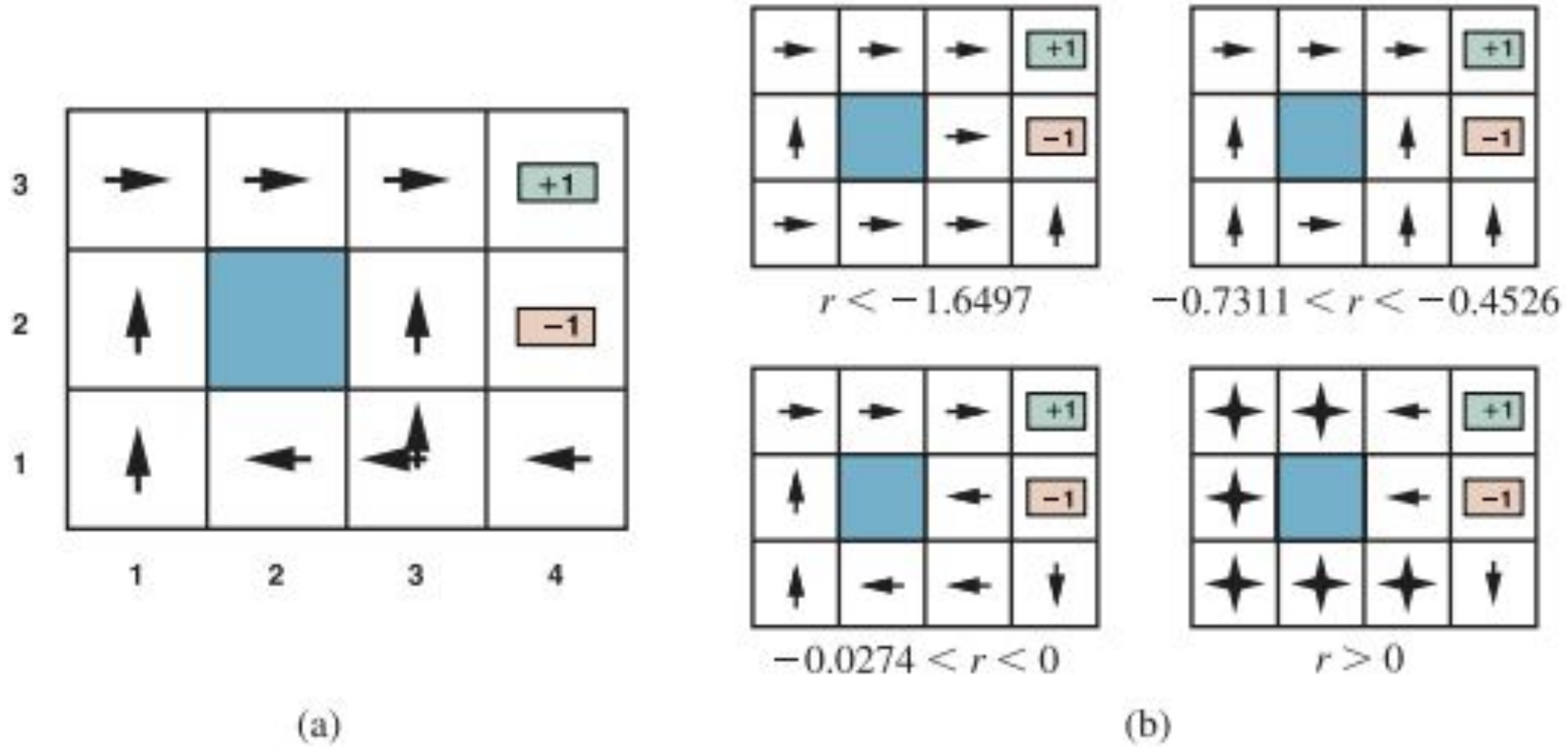
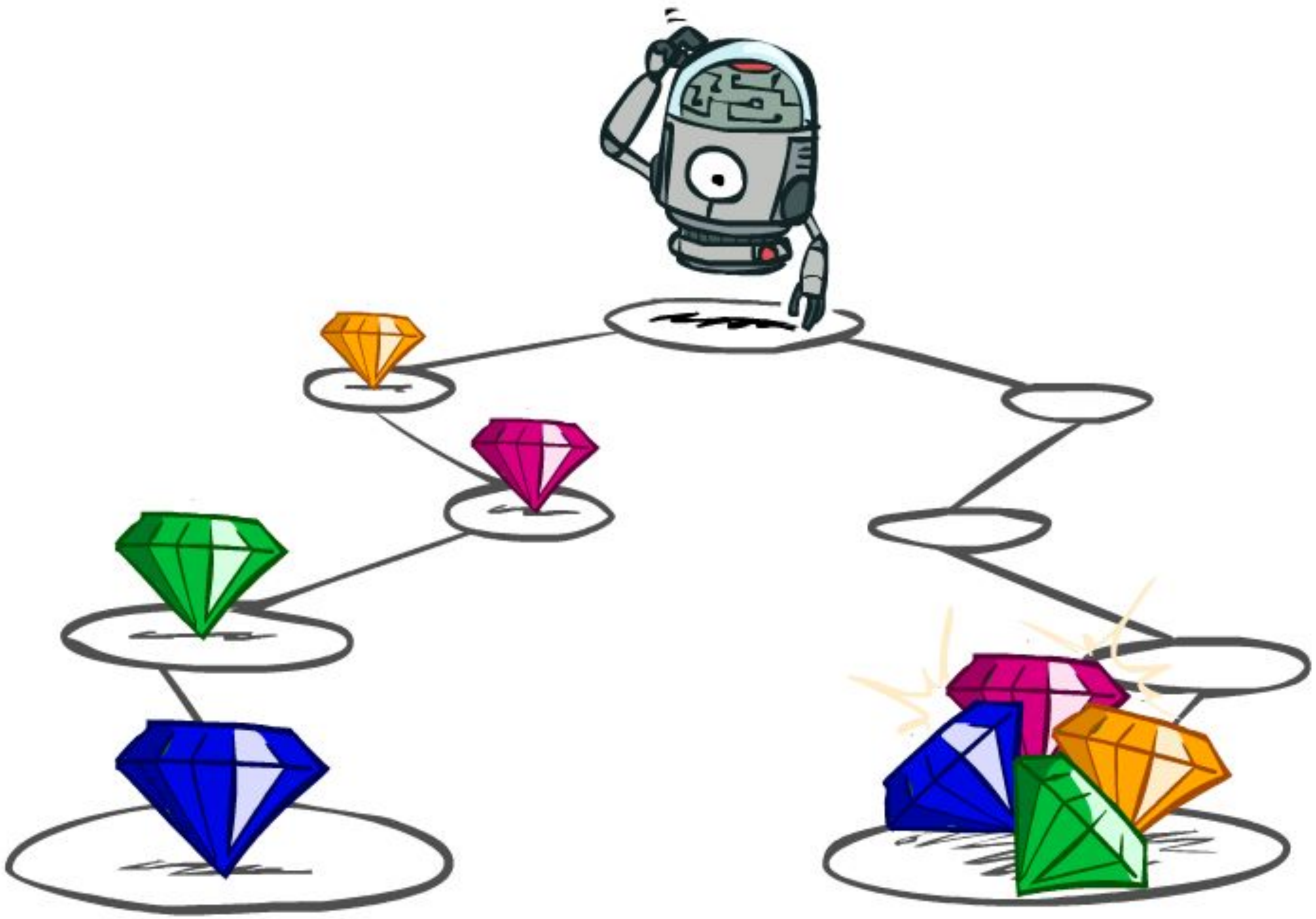


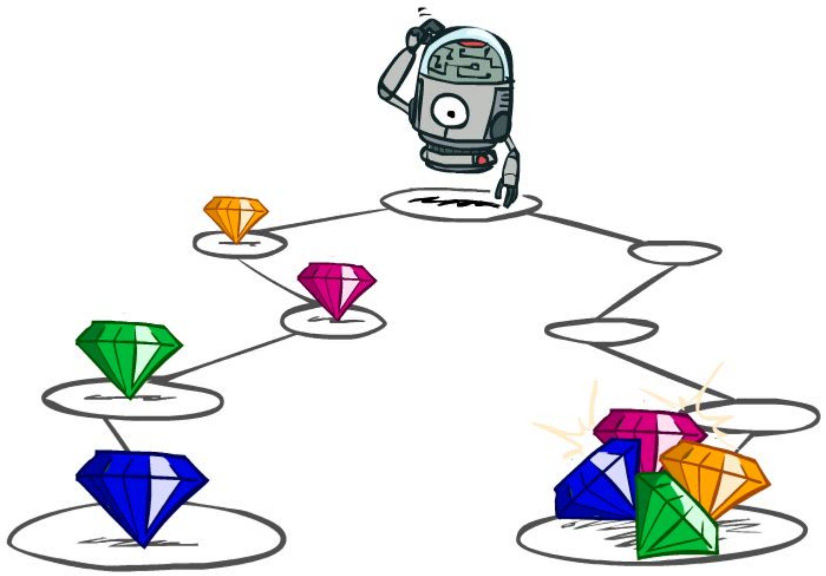
Figure 17.2 (a) The optimal policies for the stochastic environment with $r = -0.04$ for transitions between nonterminal states. There are two policies because in state (3,1) both *Left* and *Up* are optimal. (b) Optimal policies for four different ranges of r.

Utilities of Sequences



Utilities of Sequences

- What preferences should an agent have over reward sequences?
- More or less? [1, 2, 2] or [2, 3, 4]
- Now or later? [0, 0, 1] or [1, 0, 0]



Stationary Preferences

- Theorem: if we assume ***stationary preferences***:

  $[s_0, a_0, s_1, a_1, s_2, \ldots] > [s'_0, a'_0, s'_1, a'_1, s'_2, \ldots]$, $s_0 = s'_0$, $a_0 = a'_0$, and $s_1 = s'_1$

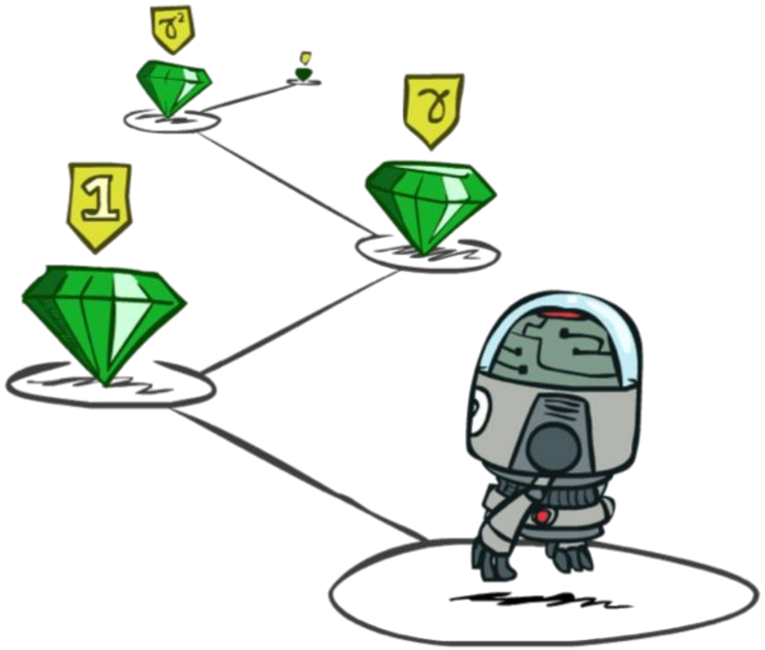
  $\Leftrightarrow$ $[s_1, a_1, s_2, \ldots]$ $[s'_1, a'_1, s'_2, \ldots]$

  then there is only one way to define utilities:

  - ***Additive discounted utility***:

$$U_h([s_0, a_0, s_1, a_1, s_2, \ldots]) = R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \cdots$$

where $\gamma \in [0,1]$ is the ***discount factor***



Discounting

Worth r now

Worth γr next step

Worth $\gamma^2 r$ in two steps

- Discounting with conveniently solves the problem of infinite reward streams!
  - Geometric series: $1 + \gamma + \gamma^2 + \ldots = 1/(1 - \gamma)$
  - Assume rewards bounded by $\pm R_{max}$
  - Then $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$ is bounded by $\pm R_{max}/(1 - \gamma)$
- (Another solution: environment contains a ***terminal state***; ***and*** agent reaches it with probability 1)

Quiz: Discounting

- Given:

  - Actions: East, West, and Exit (only available in exit states a, e)
  - Transitions: deterministic

- Quiz 1: For $\gamma = 1$, what is the optimal policy?

- Quiz 2: For $\gamma = 0.1$, what is the optimal policy?

- Quiz 3: For which γ are West and East equally good when in state d?

The utility of a policy

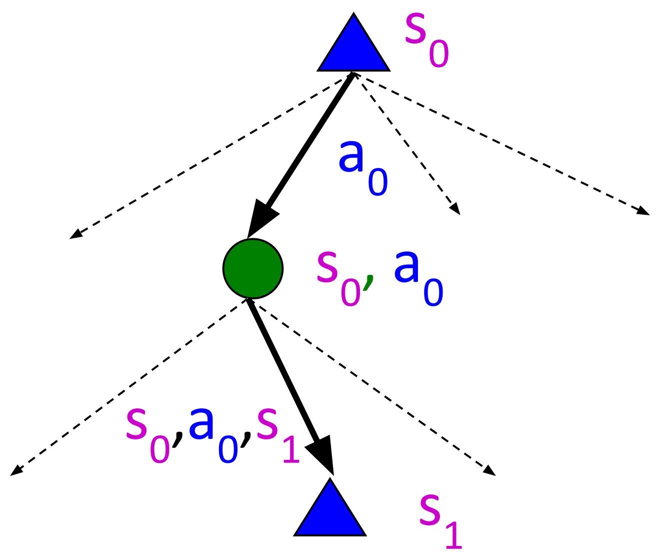
- Executing a policy π from any state s_0 generates a sequence

  $s_0, \pi(s_0), s_1, \pi(s_1), s_2, \ldots$
- This corresponds to a sequence of rewards

  $R(s_0, \pi(s_0), s_1), R(s_1,\pi(s_1), s_2), \ldots$
- This reward sequence happens with probability

  $P(s_1 \mid s_0, \pi(s_0)) \times P(s_2 \mid s_1, \pi(s_1)) \times \ldots$
- The value (expected utility) of π in s_0 is written $U^\pi(s_0)$
  - It's the sum over all possible state sequences of (discounted sum of rewards) x (probability of state sequence)

$U^\pi(s_0) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})\right]$



Optimal Quantities

- The optimal policy:

  $\pi^*(s)$ = optimal action from state s

  Gives highest $U^\pi(s)$ for any π

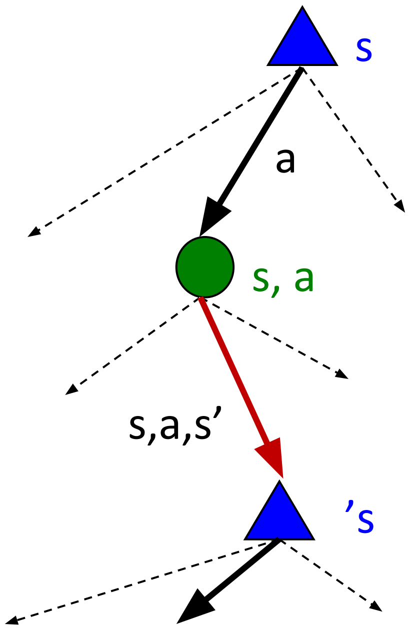
- The value (utility) of a state s:

  $U^*(s) = U^{\pi^*}(s)$ = expected utility starting in s and acting optimally

- The value (utility) of a q-state (s,a):

  $Q^*(s,a)$ = expected utility of taking action a in state s and (thereafter) acting optimally

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



s is a *state*

(s, a) is a *q-state*

(s,a,s') is a *transition*

Bellman equations (Shapley, 1953)

- The value/utility of a state is
  - The expected reward for the next transition plus the discounted value/utility of the next state, assuming the agent chooses the optimal action

- Hence we have a recursive definition of value (Bellman equation):

$$U(s) = \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma U(s')]$$

- Similarly, Bellman equation for Q-functions

$$\begin{aligned} Q(s,a) &= \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma U(s')] \\ &= \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')] \end{aligned}$$