**Example: Grid World**

- A maze-like problem
  - The agent lives on a grid
  - Walls block the agent's path
- Non-deterministic 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 each step (can be negative)
  - Big rewards come at the end (good or bad)
- Goal: maximize sum of rewards

**Grid World Actions**

- Deterministic Grid World
- Stochastic Grid World

**Markov Decision Processes**

- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function $T(s, a, s')$
    - Probability that a from $s$ leads to $s'$, i.e., $P(s' | s, a)$
    - Also called the model or the dynamics
  - A reward function $R(s, a, s')$
    - $R(s, a, s') = -0.01$ if $s' = s_{42}$
    - $R(s, a, s') = -1.01$ if $s' = s_{42}$
    - Also called the model or the dynamics

$T$ is a Big Table!  
11 X 4 x 11 = 484 entries

For now, we give this as input to the agent

For now, we also give this to the agent
Markov Decision Processes

- An MDP is defined by:
  - A set of states \( s \) in \( S \)
  - A set of actions \( a \) in \( A \)
  - A transition function \( T(s, a, s') \)
  - Probability that a transition leads to \( s' \), i.e., \( P(s' | s, a) \)
  - A reward function \( R(s, a, s') \)

\[
\begin{align*}
R(s_{32}) &= -0.01 \\
R(s_{42}) &= -1.01 \\
R(s_{43}) &= 0.99
\end{align*}
\]

What is Markov about MDPs?

- "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:

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

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

Policies

- 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 \( \pi^*: S \rightarrow A \)
  - A policy \( \pi \) gives an action for each state.
  - An optimal policy is one that maximizes expected utility if followed.
  - An explicit policy defines a reflex agent.
  - Expectimax didn’t compute entire policies.
  - It computed the action for a single state only.

Optimal Policies

Example: Racing
Example: Racing
- A robot car wants to travel far, quickly
- Three states: Cool, Warm, Overheated
- Two actions: Slow, Fast
- Going faster gets double reward

Racing Search Tree

MDP Search Trees
- Each MDP state projects an expectmax-like search tree

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]

Discounting
- It’s reasonable to maximize the sum of rewards
- It’s also reasonable to prefer rewards now to rewards later
- One solution: values of rewards decay exponentially
Discounting

- How to discount?
  - Each time we descend a level, we multiply in the discount once

- Why discount?
  - Sooner rewards probably do have higher utility than later rewards
  - Also helps our algorithms converge

- Example: discount of 0.5
  - \( U([1,2,3]) = 1 \times 1 + 0.5 \times 2 + 0.25 \times 3 \)
  - \( U([1,2,3]) < U([3,2,1]) \)

Stationary Preferences

- Theorem: if we assume stationary preferences:
  - \( [a_1, a_2, \ldots] > [b_1, b_2, \ldots] \)
  - \( \gamma [a_1, a_2, \ldots] > \gamma [b_1, b_2, \ldots] \)

- Then: there are only two ways to define utilities
  - Additive utility: \( U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots \)
  - Discounted utility: \( U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \)

Quiz: Discounting

- Given: \[ \begin{array}{cccc}
  a & b & c & d \\
  10 & 1 & 1 & 1 \\
\end{array} \]
  - Actions: East, West, and Exit (only available in exit states a, e)
  - Transitions: deterministic

- Quiz 1: For \( \gamma = 1 \), what is the optimal policy?
  - 10 1

- Quiz 2: For \( \gamma = 0.1 \), what is the optimal policy?
  - 10 1

- Quiz 3: For which \( \gamma \) are West and East equally good when in state d?

Infinite Utilities?!}

- Problem: What if the game lasts forever? Do we get infinite rewards?

- Solutions:
  - Finite horizon: (similar to depth-limited search)
  - Terminate episodes after a fixed T steps (e.g., life)
  - Give nonstationary policies (\( \gamma \) depends on time left)
  - Discounting: use \( 0 < \gamma < 1 \)
  - \( U([r_0, r_1, r_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\text{max}}/(1 - \gamma) \)
  - Smaller \( \gamma \) means smaller "horizon" – shorter term focus
  - Absorbing state: guarantee that for every policy, a terminal state will eventually be reached (like "overheated" for racing)

Recap: Defining MDPs

- Markov decision processes:
  - Set of states \( S \)
  - Start state \( s_0 \)
  - Set of actions \( A \)
  - Transitions \( P(s'|s,a) \) (or \( T(s,a,s') \))
  - Rewards \( R(s,a,s') \) (and discount \( \gamma \))

- MDP quantities so far:
  - Policy = Choice of action for each state
  - Utility = sum of (discounted) rewards