- Markov Decision Process:

  - Chapter 1 covers "Sequential Uncertainty" with "rewards"
  - Here we extend to "Sequential decisioning"
    - ↪ We will use Inventory example
      - ↪ What is the right amount of inventory to order?
        - Over-ordering has a hold cost
        - Under-ordering has a (higher) stock-out cost.
          - ↪ Empty shelves worse than overflowing shelves
      - ↪ Action influences future states which influences future actions.
      - ↪ Time + Sequential uncertainty, cyclic interplay, and randomness complicate this problem

- Markov Decision Process for Discrete time, Countable States (MDP)

  - ↪ Countable States S, with terminal states $T \subseteq S$ and countable actions A specified for each state
  - ↪ Environment generates random pairs of states $S_t \in S$ and rewards $R_t \in D$ (countable subset of $\mathbb{R}$), alternating with actions $A_t \in A$ for time steps $t = 0, 1, 2, \ldots$
    - ↪ Markov process which terminates at time T

$$S_0 A_0 R_1 S_1 A_1 R_2 S_2 \ldots$$

Time Ticks

  - ↪ Stationary transitions independent of t:

$$P[(R_{t+1}, S_{t+1}) \mid (S_t, A_t)]$$

↳ Integrate out rewards to get state transition probability function

↳ Expected reward on Slide 5

↳ Reward transition

↳ You can express expected reward given current state alone by summing up over all next steps and their associated rewards.

* Review Slide 5 for theory

- **Policy:** Function of actions given states $(\mathcal{N} \times \mathcal{A} \to [0, 1])$ ← Carry This!

$$\pi(s, a) = P[A_t = a \mid S_t = s]$$

↑ Assuming policy is stationary and Markovian

↳ For non-stationary, make time index a part of the state (pair of value and time)

↳ Requires discrete times

- **Deterministic Policy:** Each state has a single action

$$\pi_D : \mathcal{N} \to \mathcal{A}$$

↳ To implement Policy class, implement act() method

↳ Should return a probability distribution of actions that we are trying to control

[MDP, Policy] :: MRP

$$P_R^\pi(s, r, s') = \sum_{a \in A} \pi(s, a) P_R(s, a, r, s')$$

↳ "π-implied MRP" ↳ MDP

↳ Review Slide 7 for all π-implied values

- Abstract Class Markov Decision Process

  ↳ Implement ① actions : Set of actions (empty iterable if no actions)

  ③ Step

- Apply-policy produces a markov reward process given a policy

  ↑ In MDP class

  ↳ Defines a reward process by Specifying transition-reward method

  ↳ We will be iteratively optimizing policy later in this lecture, but policy is fixed at each time point.

- Finite Markov Decision Process : Finite sets of states and actions

$$N \to (A \to (S \times \mathbb{D} \to [0, 1]))$$

StateActionMapping = Mapping [S, Optional [ActionMapping]]

ActionMapping = Mapping [A, StateReward [S]]

↑ Pairs of states and finite distribution of rewards

- Finite Policy : Dictionary mapping State to distribution of actions

  * What is a constructor ??

- **Example :** Inventory MDP

  α := On-hand inventory
  β := On-order inventory

  p := Stockout cost
  c := Shelf Capacity

  $S = \{(\alpha, \beta) : 0 \leq \alpha + \beta \leq c\}$

  $A((\alpha, \beta)) = \{\theta : 0 \leq \theta \leq c - (\alpha + \beta)\}$

  - Send email asking about paper linking inventory example to options pricing with closed-form solution of inventory example

  $f(\cdot)$ = PMF of Demand, $F(\cdot)$:= CMF of Demand

  ↳ Finish reviewing equations on slide 13.

State → Action → TIME TICK → Reward → State → ...

- Because MDP + Policy = MRP, we describe value function of π-implied MRP

$$V^{\pi}(s) = R^{\pi}(s) + \gamma \cdot \sum_{s' \in N} P^{\pi}(s, s') \cdot V^{\pi}(s')$$

↳ Bellman Equation

* Write all these things on our own to check understanding

- Action-Value Function (for policy π) $Q^{\pi} : N \times A \to \mathbb{R}$

↳ Equation on slide 15

$$V^{\pi}(s) = \sum_{a \in A} \pi(s, a) \cdot Q^{\pi}(s, a)$$

$V^{\pi}(s)$: State Value Function

$Q^{\pi}(s, a)$: Action-Value Function

P : probability of States (defined by environment)

π : Probability of actions (defined by policy)

MDP Prediction process : Predict values of accumulated rewards for a fixed policy

Optimal State-Value Function V^*

$$V^*(s) = \max_{\pi \in \Pi} V^{\pi}(s) \quad \text{for all } s \in N$$

$$V^*(s) = \max_{a \in A} Q^*(s, a)$$

↳ Reward comes after the action

$$Q^{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in N} P(s, a, s') \cdot V^*(s')$$

$R(s, a)$: Expected reward

$P(s, a, s')$: Weight from probability of State transition

$V^*(s')$: Optimal value from future State

- Bellman Optimality Equations do not solve Control

↳ Unlike Bellman Policy Equations, these are nonlinear

↳ Provide foundations for DP / RL algos for control

Optimal Policy $\pi^* \in \Pi$ if $V^{\pi^*}(s) \geq V^{\pi}(s)$ for all s

↳ See theorem on slide 23

↳ There can be more than one optimal policy.

$\pi^*_D = \underset{a \in A}{\text{argmax}}\ Q^*(s, a)$ for all $s \in N$

↑ Optimal (deterministic) policy : one action per state

↳ This is a candidate for optimal policy

↳ See textbook for proof on slide 29

↳ Proof by contradiction

- Qualitative Reality of MDP
  - Tabular Algorithm for State Space that are not too large (finite space)
  - In real world, state space is large / continuous / infinite
    - ↳ Curse of Dimensionality
    - ↳ Curse of Modeling
  - Use Dimension - Reduction techniques
  - * In reality, approximate state spaces at fixed points and interpolate between them.
  - Policy gradient great at handling large State Spaces.
  - In real world, you see partially observable MDPs.
    - ↳ Next lecture : this and DP algorithms.