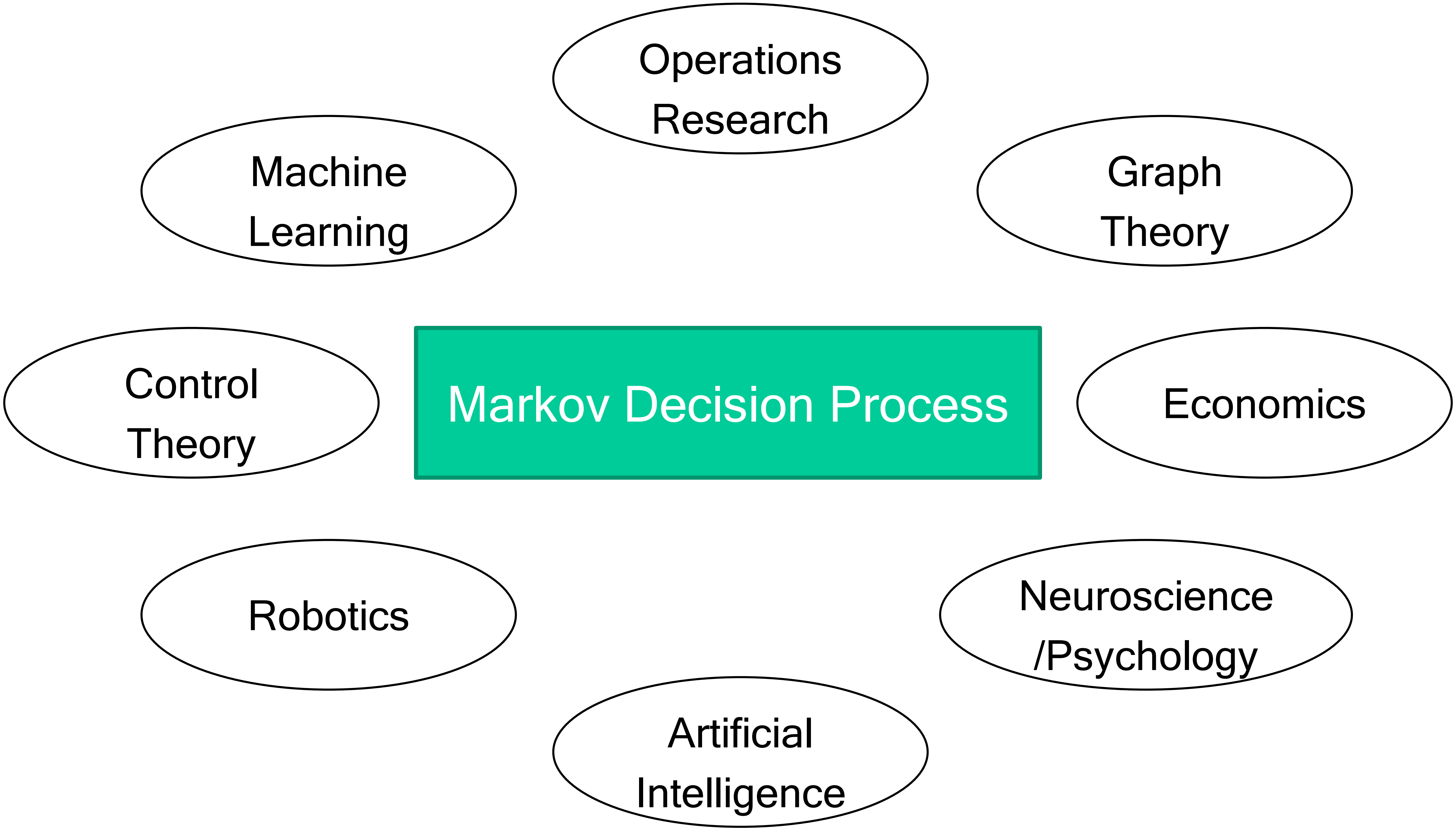
# Markov Decision Processes

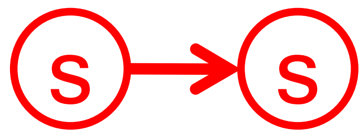
(Slides from Mausam)




*model the sequential decision making of a rational agent.*

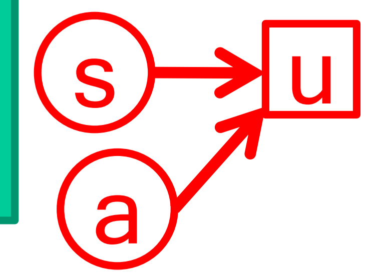
# A Statistician's view to MDPs

**Markov Chain**



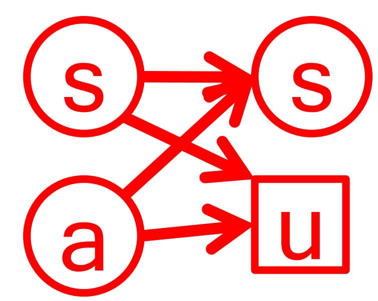

- sequential process
- models state transitions
- autonomous process

**One-step Decision Theory**




- one-step process
- models choice
- maximizes utility

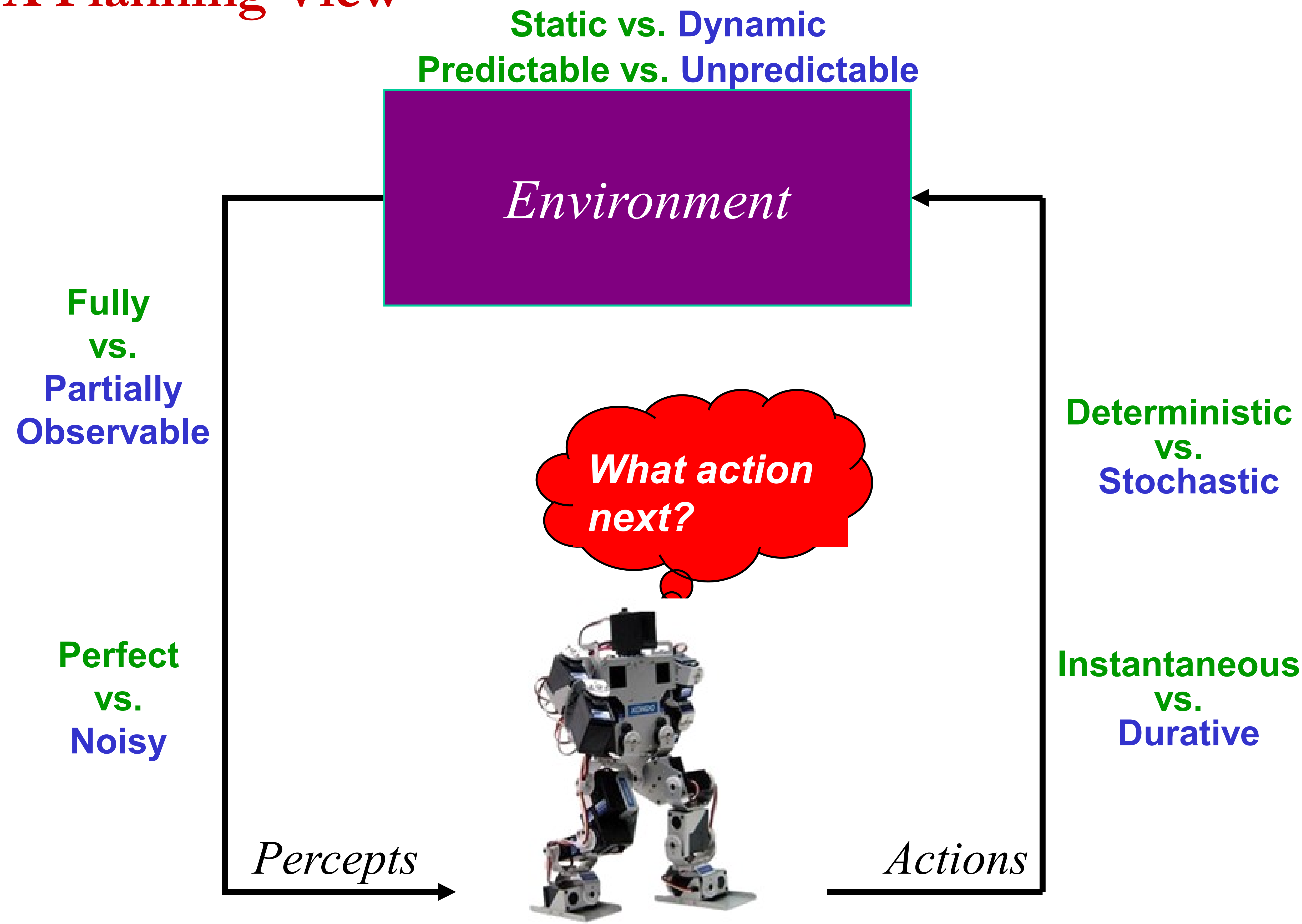
**Markov Decision Process**



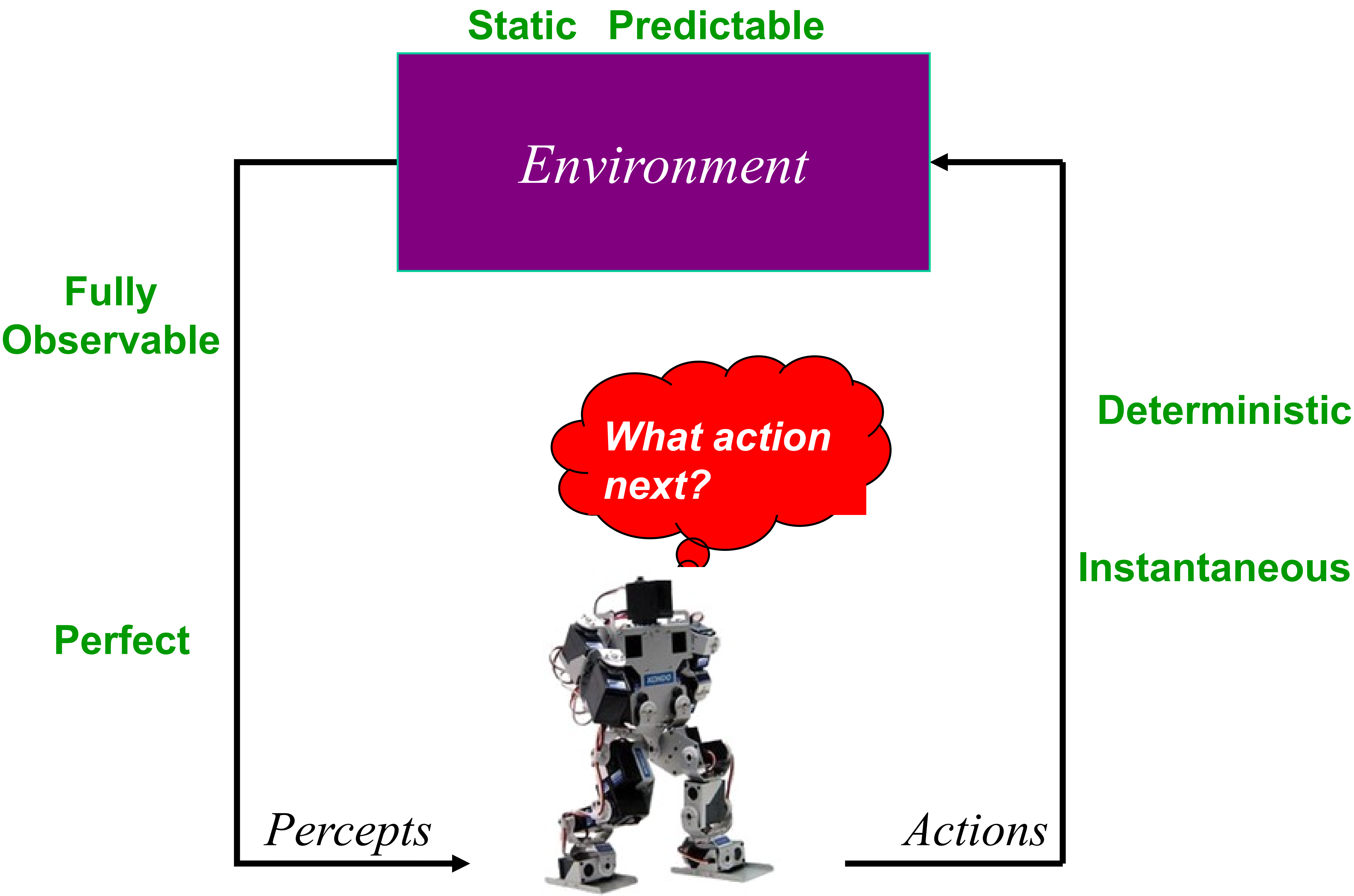

- Markov chain + choice
- Decision theory + sequentiality

- sequential process
- models state transitions
- models choice
- maximizes utility

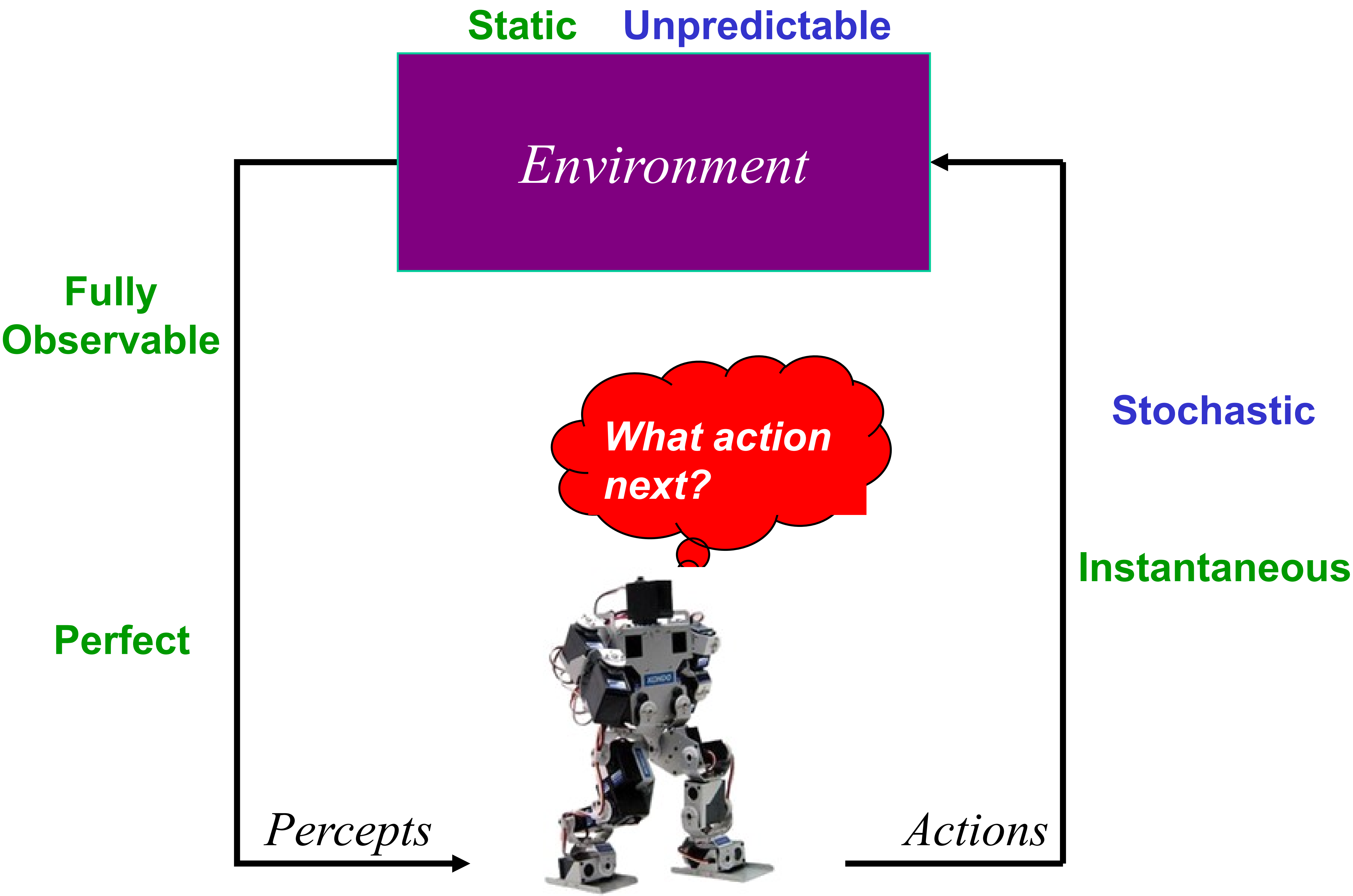
# A Planning View



# Classical Planning



# Stochastic Planning: MDPs



# Markov Decision Process (MDP)

- $\mathcal{S}$: A set of states
- $\mathcal{A}$: A set of actions
- $\mathcal{P}r(s'|s,a)$: transition model
- $\mathcal{C}(s,a,s')$: cost model
- $\mathcal{G}$: set of goals
- $s_0$: start state
- $\gamma$: discount factor
- $\mathcal{R}(s,a,s')$: reward model

factored

Factored MDP

absorbing/
non-absorbing

# Objective of an MDP

- Find a policy $\pi$: $\mathcal{S} \rightarrow \mathcal{A}$

- which optimizes
  - minimizes $\begin{pmatrix} \text{discounted} \\ \text{or} \\ \text{undiscount.} \end{pmatrix}$ expected cost to reach a goal
  - maximizes expected reward
  - maximizes expected (reward-cost)

- given a ____ horizon
  - finite
  - infinite
  - indefinite

- assuming full observability

# Role of Discount Factor ($\gamma$)

- Keep the total reward/total cost finite
  - useful for infinite horizon problems

- Intuition (economics):
  - Money today is worth more than money tomorrow.

- Total reward: $r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots$
- Total cost: $c_1 + \gamma c_2 + \gamma^2 c_3 + \ldots$

# Examples of MDPs

- Goal-directed, Indefinite Horizon, Cost Minimization MDP
  - $<\mathcal{S}, \mathcal{A}, \mathcal{P}r, \mathcal{C}, \mathcal{G}, s_0>$
  - Most often studied in planning, graph theory communities
- Infinite Horizon, Discounted Reward Maximization MDP
  - $<\mathcal{S}, \mathcal{A}, \mathcal{P}r, \mathcal{R}, \gamma>$
  - Most often studied in machine learning, economics, operations research communities

**most popular**

- Goal-directed, Finite Horizon, Prob. Maximization MDP
  - $<\mathcal{S}, \mathcal{A}, \mathcal{P}r, \mathcal{G}, s_0, T>$
  - Also studied in planning community
- Oversubscription Planning: Non absorbing goals, Reward Max. MDP
  - $<\mathcal{S}, \mathcal{A}, \mathcal{P}r, \mathcal{G}, \mathcal{R}, s_0>$
  - Relatively recent model

# Bellman Equations for $MDP_2$

- <$\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$r, $\mathcal{R}$, $s_{0,}$ $\gamma$>
- Define **V*(s)** {optimal **value**} as the **maximum** expected **discounted reward** from this state.
- V* should satisfy the following equation:

$$\boxed{V^*(s) \;\; = \;\; \max_{a \in Ap(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}r(s'|s,a) \left[ \mathcal{R}(s,a,s') + \gamma V^*(s') \right]}$$

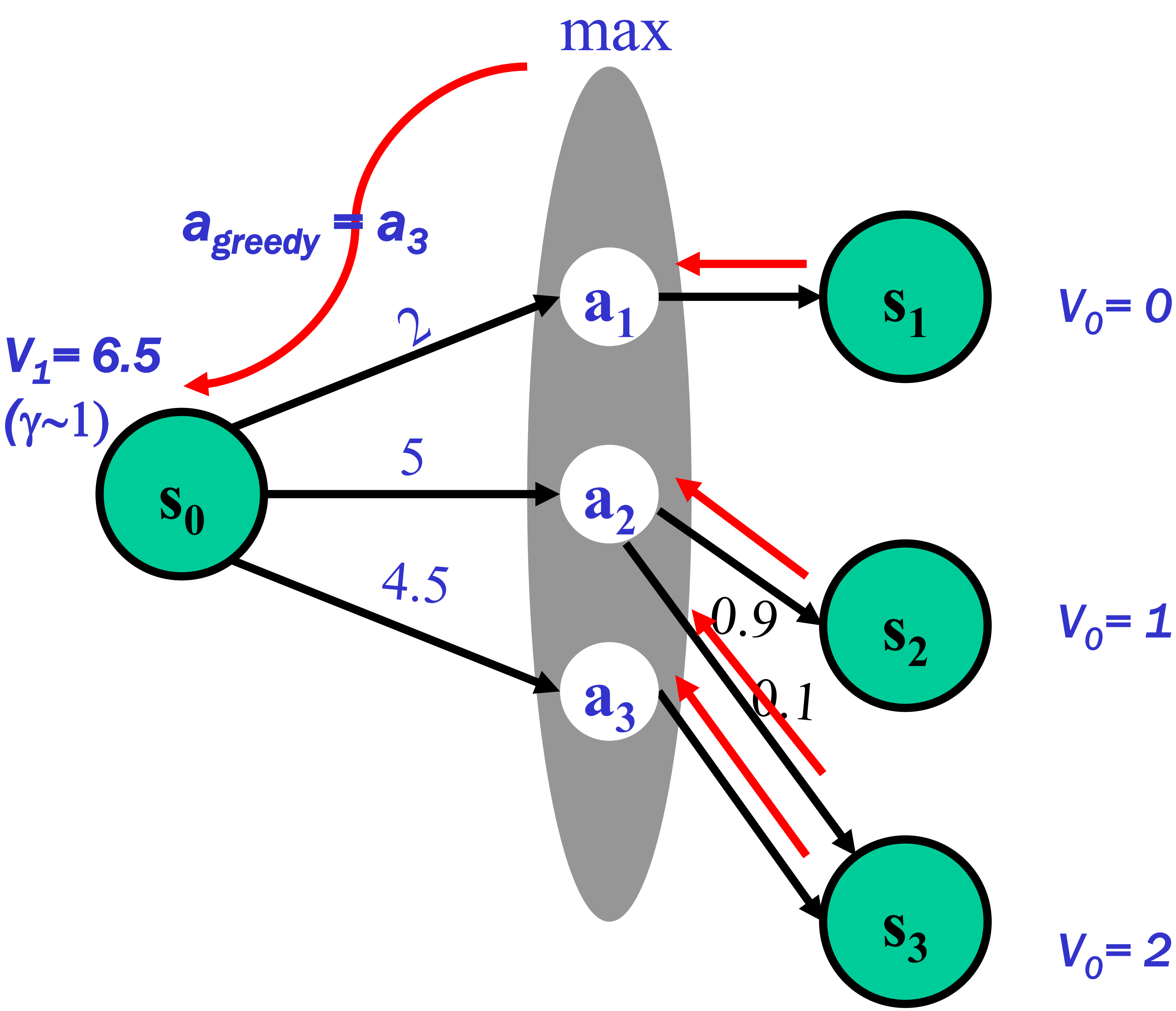
# Bellman Backup ($MDP_2$)

- Given an estimate of V* function (say $V_n$)
- Backup $V_n$ function at state s
  - calculate a new estimate ($V_{n+1}$) :

$$
\begin{aligned}
Q_{n+1}(s,a) &= \sum_{s' \in \mathcal{S}} Pr(s'|s,a)\left[\mathcal{R}(s,a,s') + \gamma V_n(s')\right] \\
V_{n+1}(s) &= \max_{a \in Ap(s)} \left[Q_{n+1}(s,a)\right]
\end{aligned}
$$

- $Q_{n+1}(s,a)$ : value/cost of the strategy:
  - execute action a in s, execute $\pi_n$ subsequently
  - $\pi_n = \text{argmax}_{a \in Ap(s)} Q_n(s,a)$

# Bellman Backup




$$Q_1(s,a_1) = 2 + 0\,\gamma$$

$$Q_1(s,a_2) = 5 + \gamma\, 0.9 \times 1 + \gamma\, 0.1 \times 2$$

$$Q_1(s,a_3) = 4.5 + 2\,\gamma$$

# Value iteration [Bellman'57]

- assign an arbitrary assignment of $V_0$ to each state.

- repeat
  - for all states s
    - compute $V_{n+1}(s)$ by Bellman backup at s.

- until $\max_s |V_{n+1}(s) - V_n(s)| < \varepsilon$

Iteration n+1

Residual(s)

ε-convergence

# Comments

- Decision-theoretic Algorithm
- Dynamic Programming
- Fixed Point Computation
- Probabilistic version of Bellman-Ford Algorithm
  - for shortest path computation
  - $MDP_1$ : Stochastic Shortest Path Problem

- Time Complexity
  - one iteration: $O(|\mathcal{S}|^2|\mathcal{A}|)$
  - number of iterations: poly($|\mathcal{S}|$, $|\mathcal{A}|$, $1/(1-\gamma)$)
- Space Complexity: $O(|\mathcal{S}|)$
- Factored MDPs
  - exponential space, exponential time

# Convergence Properties

- $V_n \rightarrow V^*$ in the limit as $n \rightarrow \infty$
- $\varepsilon$-convergence: $V_n$ function is within $\varepsilon$ of $V^*$
- Optimality: current policy is within $2\varepsilon\gamma/(1-\gamma)$ of optimal

- Monotonicity
  - $V_0 \leq_p V^* \Rightarrow V_n \leq_p V^*$ ($V_n$ monotonic from below)
  - $V_0 \geq_p V^* \Rightarrow V_n \geq_p V^*$ ($V_n$ monotonic from above)
  - otherwise $V_n$ non-monotonic

# Policy Computation

$$\pi^*(s) = \operatorname*{argmax}_{a \in Ap(s)} Q^*(s,a)$$
$$= \operatorname*{argmax}_{a \in Ap(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}r(s'|s,a) \left[\mathcal{R}(s,a,s') + \gamma V^*(s')\right]$$

# Policy Evaluation

$$V_\pi(s) = \sum_{s' \in \mathcal{S}} \mathcal{P}r(s'|s,\pi(s)) \left[\mathcal{R}(s,\pi(s),s') + \gamma V_\pi(s')\right]$$

A system of linear equations in $|\mathcal{S}|$ variables.

# Changing the Search Space

- Value Iteration
  - Search in value space
  - Compute the resulting policy

- Policy Iteration
  - Search in policy space
  - Compute the resulting value

# Policy iteration [Howard'60]

- assign an arbitrary assignment of $\pi_0$ to each state.
- repeat
  - Policy Evaluation: compute $V_{n+1}$: the evaluation of $\pi_n$
  - Policy Improvement: for all states s
    - compute $\pi_{n+1}(s)$: $\text{argmax}_{a \in Ap(s)} Q_{n+1}(s,a)$
- until $\pi_{n+1} = \pi_n$

costly: $O(n^3)$

**Modified Policy Iteration** → **approximate by value iteration using fixed policy**

## Advantage

- searching in a finite (policy) space as opposed to uncountably infinite (value) space ⇒ convergence faster.
- all other properties follow!

## Modified Policy iteration

- assign an arbitrary assignment of $\pi_0$ to each state.

- repeat
  - Policy Evaluation: compute $V_{n+1}$ the *approx.* evaluation of $\pi_n$
  - Policy Improvement: for all states s
    - compute $\pi_{n+1}(s)$: $\mathrm{argmax}_{a \in Ap(s)} Q_{n+1}(s,a)$
- until $\pi_{n+1} = \pi_n$

## Advantage

- probably the most competitive synchronous dynamic programming algorithm.

# Asynchronous Value Iteration

- States may be backed up in any order
  - instead of an iteration by iteration
- As long as all states backed up infinitely often
  - Asynchronous Value Iteration converges to optimal

# Asynch VI: Prioritized Sweeping

- Why backup a state if values of successors same?
- Prefer backing a state
  - whose successors had most change

- Priority Queue of (state, expected change in value)
- Backup in the order of priority
- After backing a state update priority queue
  - for all predecessors

# Reinforcement Learning

# Reinforcement Learning

- Still have an MDP
  - Still looking for policy $\pi$
- New twist: don't know $\mathcal{P}$r and/or $\mathcal{R}$
  - i.e. don't know which states are good
  - and what actions do
- Must actually try out actions to learn

# Model based methods

- Visit different states, perform different actions
- Estimate $\mathcal{P}$r and $\mathcal{R}$

- Once model built, do planning using V.I. or other methods

- Con: require _huge_ amounts of data

# Model free methods

- Directly learn Q*(s,a) values

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}r(s'|s,a)\left[\mathcal{R}(s,a,s') + \gamma V^*(s')\right]$$

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}r(s'|s,a)\left[\mathcal{R}(s,a,s') + \gamma max_{a'} Q^*(s',a')\right]$$

- sample = $\mathcal{R}(s,a,s') + \gamma \max_{a'} Q_n(s',a')$
- Nudge the old estimate towards the new sample
- $Q_{n+1}(s,a) \leftarrow (1-\alpha)Q_n(s,a) + \alpha[\text{sample}]$

# Properties

- Converges to optimal if
  - If you explore enough
  - If you make learning rate ($\alpha$) small enough
  - But not decrease it too quickly

  - $\sum_i \alpha(s,a,i) = \infty$
  - $\sum_i \alpha^2(s,a,i) < \infty$

  where i is the number of visits to (s,a)

# Model based vs. Model Free RL

- Model based
  - estimate $O(|\mathcal{S}|^2|\mathcal{A}|)$ parameters
  - requires relatively larger data for learning
  - can make use of background knowledge easily
- Model free
  - estimate $O(|\mathcal{S}||\mathcal{A}|)$ parameters
  - requires relatively less data for learning

# Exploration vs. Exploitation

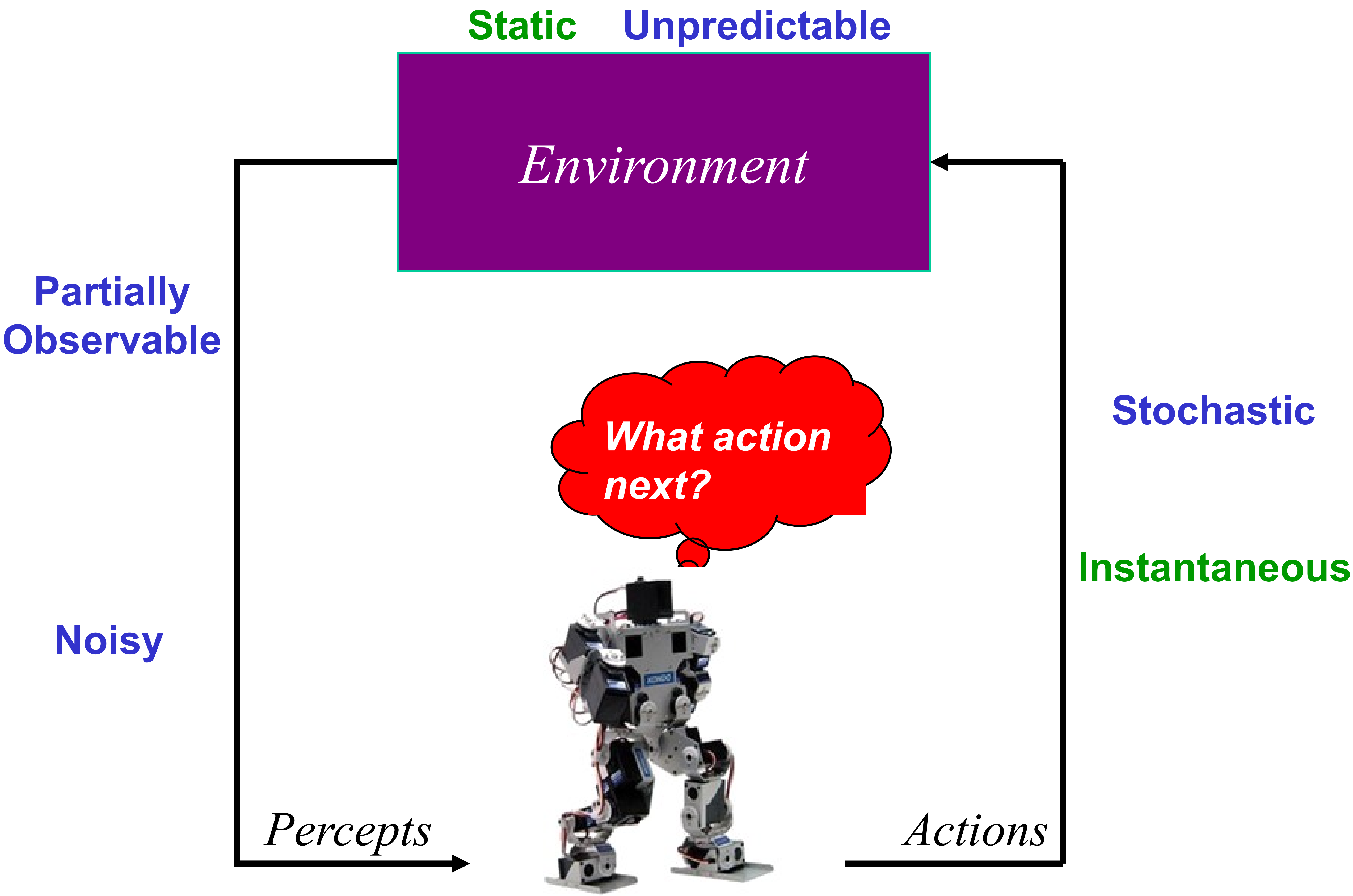
- Exploration: choose actions that visit new states in order to obtain more data for better learning.
- Exploitation: choose actions that maximize the reward given current learnt model.
- $\varepsilon$-greedy
  - Each time step flip a coin
  - With prob $\varepsilon$, take an action randomly
  - With prob 1-$\varepsilon$ take the current greedy action
- Lower $\varepsilon$ over time
  - increase exploitation as more learning has happened

# Q-learning

- Problems
  - Too many states to visit during learning
  - Q(s,a) is still a BIG table
- We want to generalize from small set of training examples
- Techniques
  - Value function approximators
  - Policy approximators
  - Hierarchical Reinforcement Learning

# Partially Observable Markov Decision Processes

# Partially Observable MDPs



# POMDPs

- In POMDPs we apply the very same idea as in MDPs.

- Since the state is not observable,

  the agent has to make its decisions based on the belief state which is a posterior distribution over states.

- Let $b$ be the belief of the agent about the current state
- POMDPs compute a value function over belief space:

$$V_T(b) \quad = \quad \max_a \left[ r(b,a) + \gamma \int V_{T-1}(b') p(b' \mid b, a) \, db' \right]$$

## POMDPs

- Each belief is a probability distribution,
  - value fn is a function of an entire probability distribution.
- Problematic, since probability distributions are continuous.
- Also, we have to deal with huge complexity of belief spaces.

- For finite worlds with finite state, action, and observation spaces and finite horizons,
  - we can represent the value functions by piecewise linear functions.

# Applications

- Robotic control
  - helicopter maneuvering, autonomous vehicles
  - Mars rover - path planning, oversubscription planning
  - elevator planning
- Game playing - backgammon, tetris, checkers
- Neuroscience
- Computational Finance, Sequential Auctions
- Assisting elderly in simple tasks
- Spoken dialog management
- Communication Networks – switching, routing, flow control
- War planning, evacuation planning