Module 5
Introduction to
Markov Decision Processes

CS 886 Sequential Decision Making and Reinforcement Learning

University of Waterloo

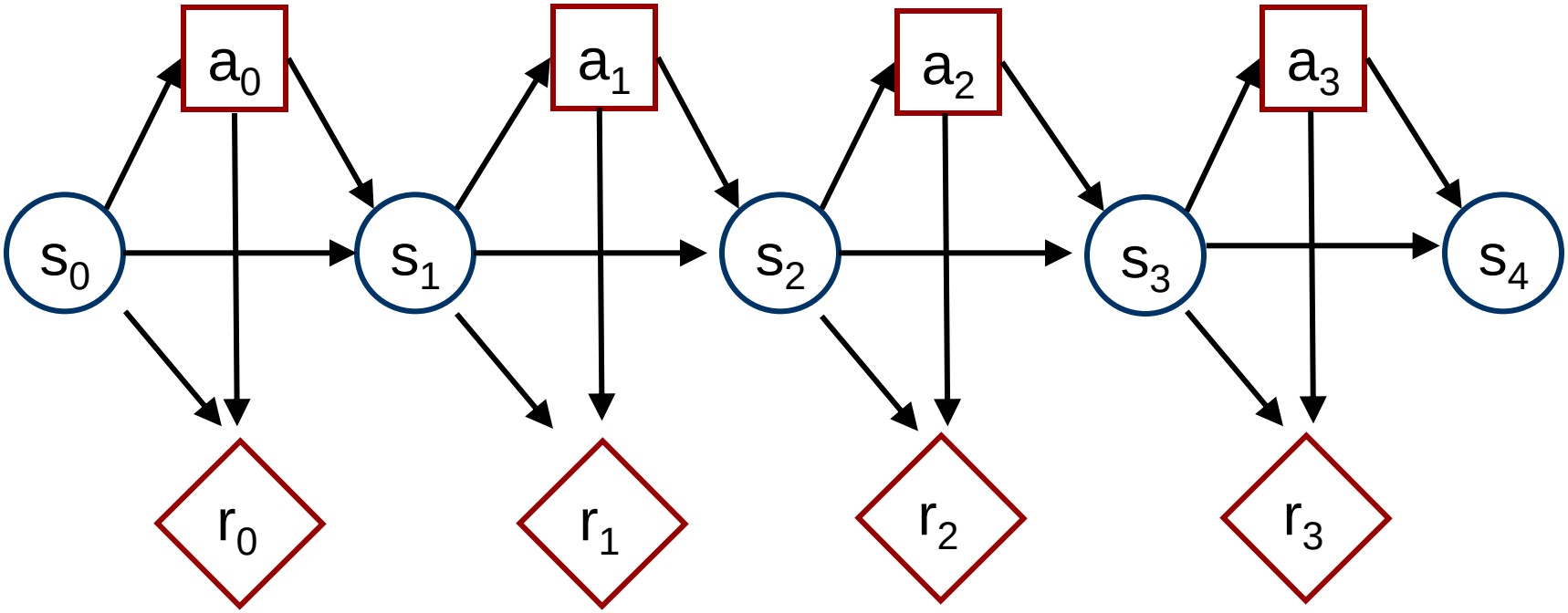
Assumptions

- Uncertainty: stochastic process
- Time: sequential process
- Observability: full observability
- No learning: complete model
- # of agents: single agent
- Variable type: discrete

CS886 (c) 2013 Pascal Poupart

Markov Decision Process

- Markov process augmented with…
  - Decisions e.g., a_t
  - Utilities e.g., r_t



CS886 (c) 2013 Pascal Poupart

Stationary Preferences

- Hum… but why many utility nodes?

- $U(s_0, a_0, s_1, a_1, s_2, a_2, \dots)$
  - Infinite process → infinite utility function

- Solution:
  - Assume stationary and additive preferences
  - $U(s_0, a_0, s_1, a_1, s_2, a_2, \dots) = \sum_t R(s_t, a_t)$

CS886 (c) 2013 Pascal Poupart

Discounted/Average Rewards

- If process infinite, isn't $\sum_t R(s_t, a_t)$ infinite?

- Solution 1: discounted rewards
  - Discount factor: $0 \leq \gamma \leq 1$
  - Finite utility: $\sum_t \gamma^t R(s_t, a_t)$ is a geometric sum
  - γ induces an inflation rate of $1/\gamma - 1$
  - Intuition: prefer utility sooner than later

- Solution 2: average rewards
  - More complicated computationally
  - Beyond the scope of this course

CS886 (c) 2013 Pascal Poupart

Markov Decision Process

- Definition
  - Set of states: S
  - Set of actions (i.e., decisions): A
  - Transition model: $\Pr(s_t|s_{t-1}, a_{t-1})$
  - Reward model (i.e., utility): $R(s_t, a_t)$
  - Discount factor: $0 \leq \gamma \leq 1$
  - Horizon (i.e., # of time steps): h

- Goal: find optimal policy

CS886 (c) 2013 Pascal Poupart

Inventory Management

- Markov Decision Process
  - States: inventory levels
  - Actions: {doNothing, orderWidgets}
  - Transition model: stochastic demand
  - Reward model: Sales – Costs - Storage
  - Discount factor: 0.999
  - Horizon: ∞

- Tradeoff: increasing supplies decreases odds of missed sales but increases storage costs

CS886 (c) 2013 Pascal Poupart

Policy

- Choice of action at each time step

- Formally:
  - Mapping from states to actions
  - i.e., $\pi(s_t) = a_t$
  - Assumption: fully observable states
    - Allows a_t to be chosen only based on current state s_t. Why?

CS886 (c) 2013 Pascal Poupart

Policy Optimization

- Policy evaluation:
  - Compute expected utility

$$V^{\pi}(s_0) = \sum_{t=0}^{h} \gamma^t \Pr(s_t | s_0, \pi) R(s_t, \pi(s_t))$$

- Optimal policy:
  - Policy with highest expected utility

$$V^{\pi^*}(s_0) \geq V^{\pi}(s_0) \quad \forall \pi$$

CS886 (c) 2013 Pascal Poupart

Policy Optimization

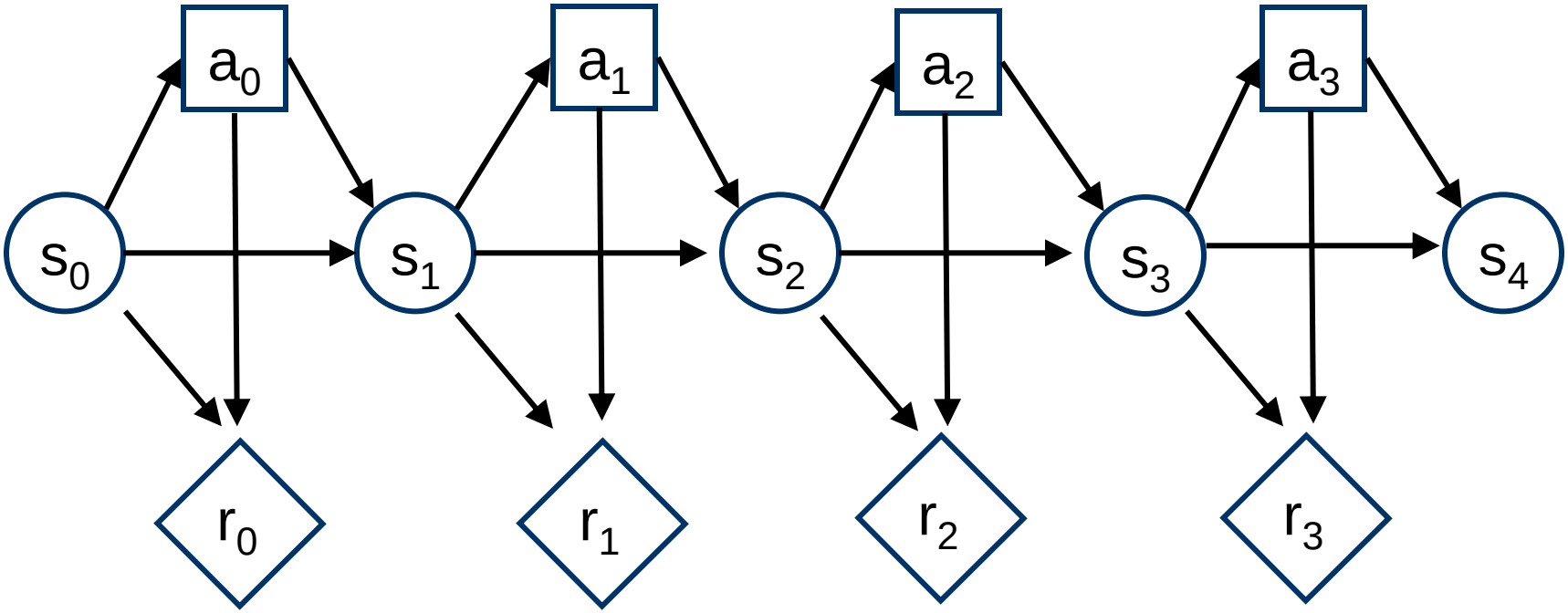
- Several classes of algorithms:
  - Value iteration
  - Policy iteration
  - Linear Programming
  - Search techniques

- Computation may be done
  - Offline: before the process starts
  - Online: as the process evolves

CS886 (c) 2013 Pascal Poupart

Value Iteration

- Performs dynamic programming
- Optimizes decisions in reverse order



CS886 (c) 2013 Pascal Poupart

Value Iteration

- Value when no time left:

  $V(s_h) = \max_{a_h} R(s_h, a_h)$

- Value with one time step left:

  $V(s_{h-1}) = \max_{a_{h-1}} R(s_{h-1}, a_{h-1}) + \gamma \sum_{s_h} \Pr(s_h | s_{h-1}, a_{h-1}) V(s_h)$

- Value with two time steps left:

  $V(s_{h-2}) = \max_{a_{h-2}} R(s_{h-2}, a_{h-2}) + \gamma \sum_{s_{h-1}} \Pr(s_{h-1} | s_{h-2}, a_{h-2}) V(s_{h-1})$

- …

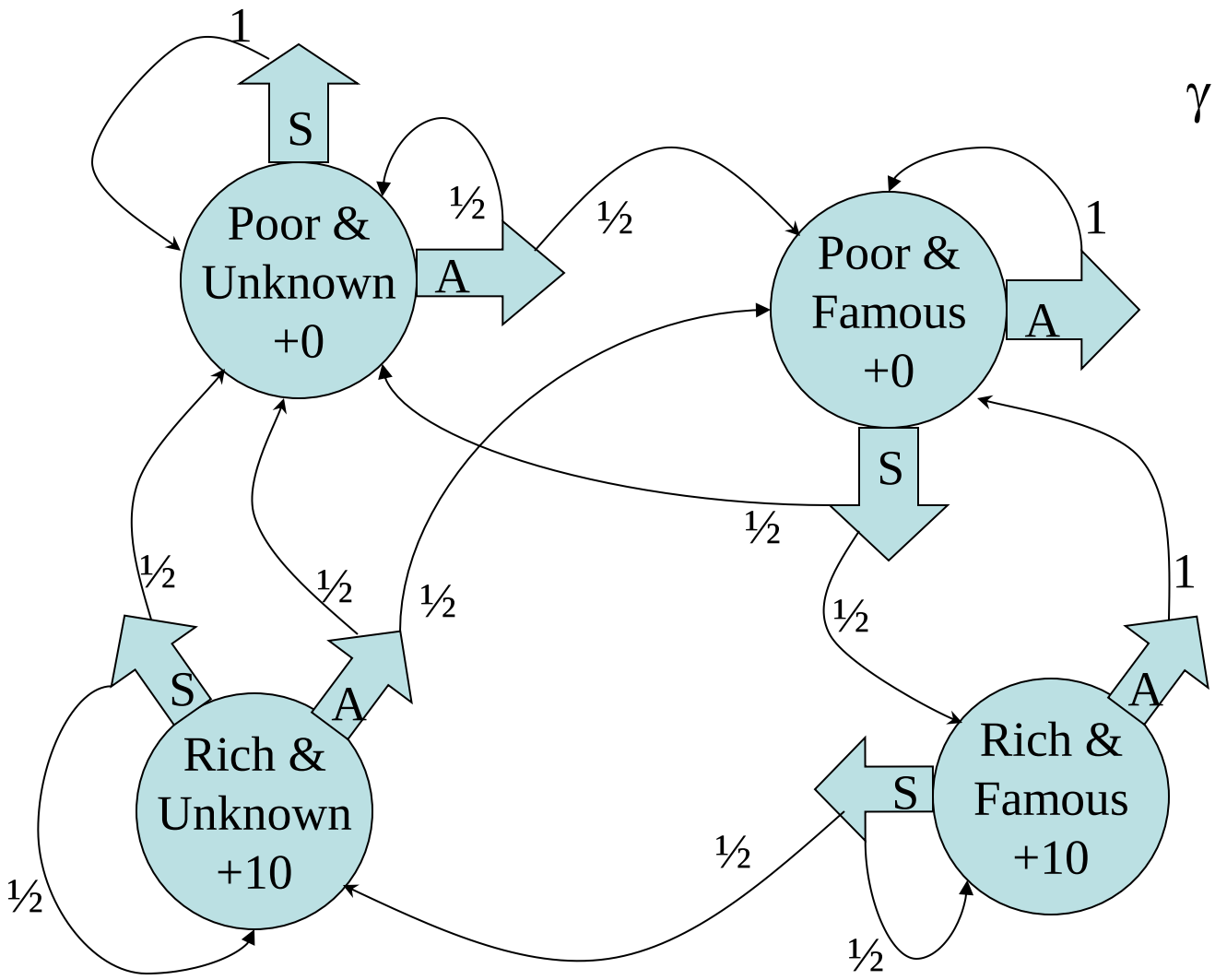
- Bellman's equation:

  $V(s_t) = \max_{a_t} R(s_t, a_t) + \gamma \sum_{s_{t+1}} \Pr(s_{t+1} | s_t, a_t) V(s_{t+1})$

  $a_t^* = \operatorname{argmax}_{a_t} R(s_t, a_t) + \gamma \sum_{s_{t+1}} \Pr(s_{t+1} | s_t, a_t) V(s_{t+1})$

CS886 (c) 2013 Pascal Poupart

A Markov Decision Process

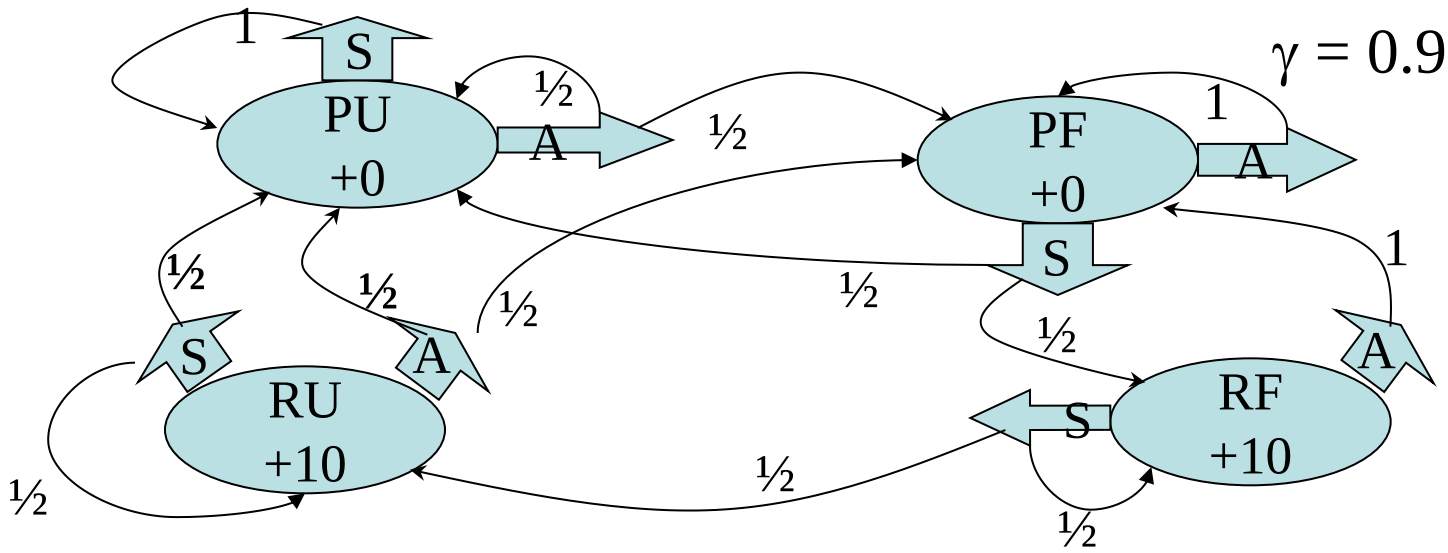


$\gamma = 0.9$

You own a company

In every state you must choose between **S**aving money or **A**dvertising

CS886 (c) 2013 Pascal Poupart



t	V(PU)	V(PF)	V(RU)	V(RF)
h	0	0	10	10
h-1	0	4.5	14.5	19
h-2	2.03	8.55	16.53	25.08
h-3	4.76	12.20	18.35	28.72
h-4	7.63	15.07	20.40	31.18
h-5	10.21	17.46	22.61	33.21

CS886 (c) 2013 Pascal Poupart

Finite Horizon

- When h is finite,
- Non-stationary optimal policy
- Best action different at each time step
- Intuition: best action varies with the amount of time left

CS886 (c) 2013 Pascal Poupart

Infinite Horizon

- When h is infinite,
- Stationary optimal policy
- Same best action at each time step
- Intuition: same (infinite) amount of time left at each time step, hence same best action

- Problem: value iteration does an infinite number of iterations…

CS886 (c) 2013 Pascal Poupart

Infinite Horizon

- Assuming a discount factor γ, after k time steps, rewards are scaled down by γ^k
- For large enough k, rewards become insignificant since $\gamma^k \to 0$
- Solution:
  - pick large enough k
  - run value iteration for k steps
  - Execute policy found at the k^{th} iteration

CS886 (c) 2013 Pascal Poupart

Computational Complexity

- Space and time: $O(k|A||S|^2)$ ☺
  - Here k is the number of iterations

- But what if $|A|$ and $|S|$ are defined by the joint instantiation of several random variables and consequently exponential?

- Solutions:
  - Exploit structure
  - Approximate

CS886 (c) 2013 Pascal Poupart