

# Lecture 13

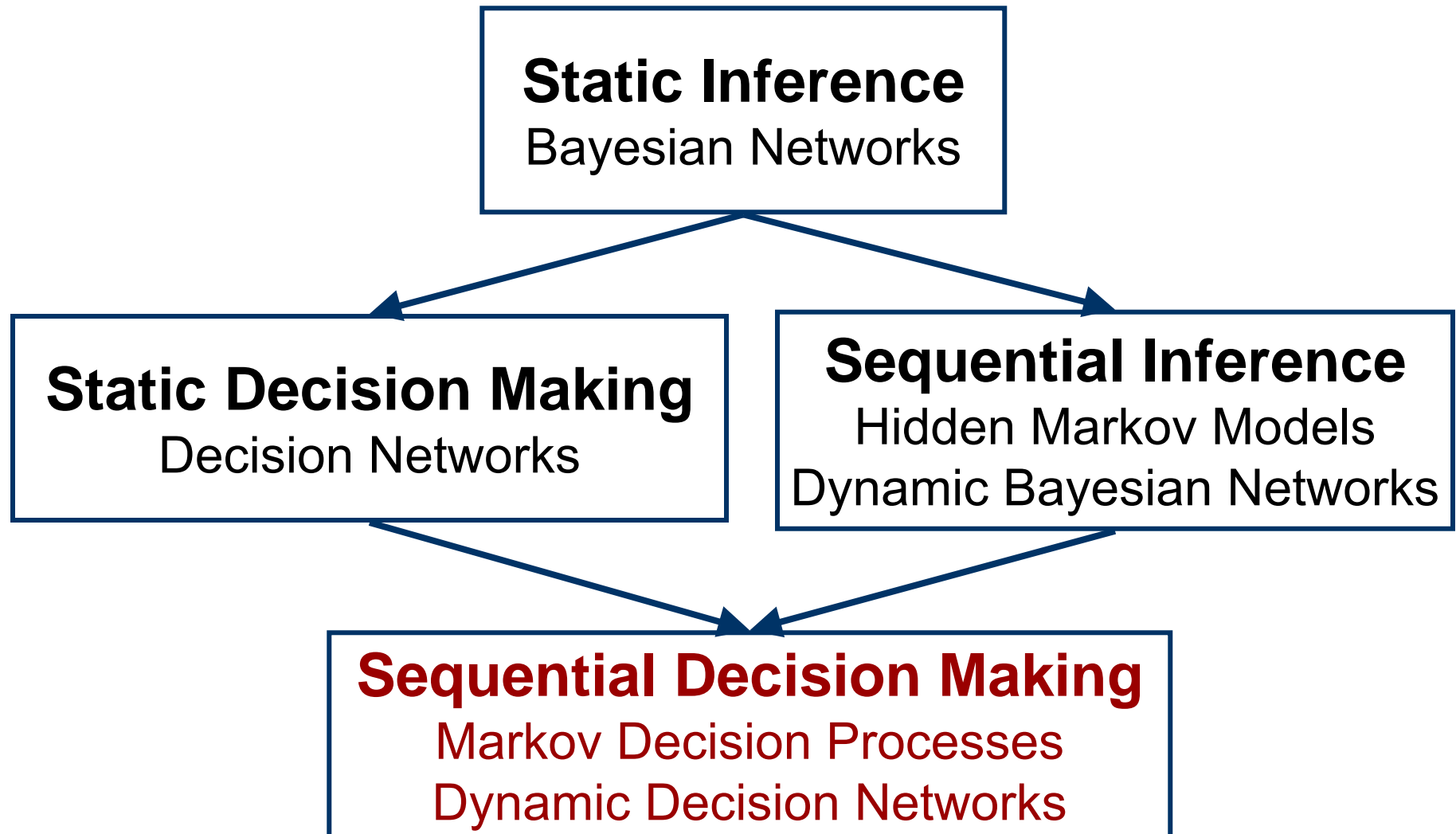
October 27, 2009

CS 486/686

# Outline

- Markov Decision Processes
- Dynamic Decision Networks
- Russell and Norvig: Sect 17.1, 17.2 (up to p. 620), 17.4, 17.5

# Sequential Decision Making

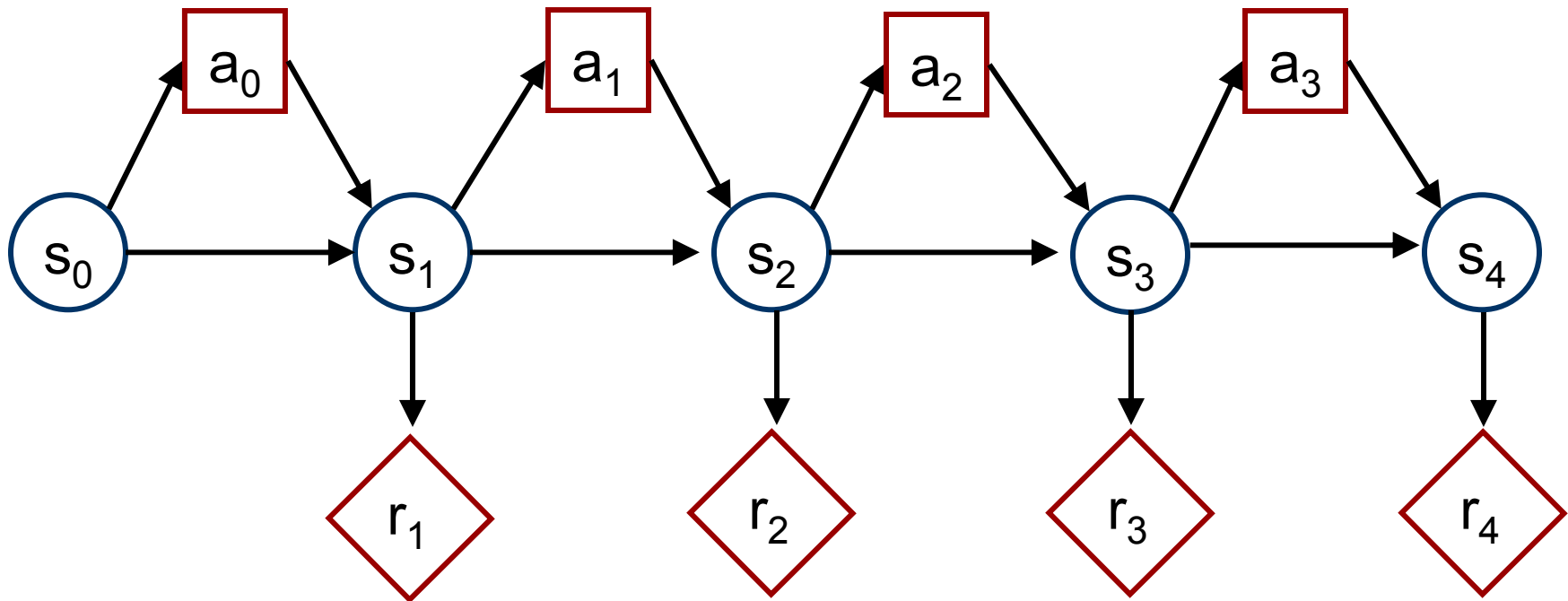


# Sequential Decision Making

- Wide range of applications
  - Robotics (e.g., control)
  - Investments (e.g., portfolio management)
  - Computational linguistics (e.g., dialogue management)
  - Operations research (e.g., inventory management, resource allocation, call admission control)
  - Assistive technologies (e.g., patient monitoring and support)

# Markov Decision Process

- Intuition: Markov Process with...
  - Decision nodes
  - Utility nodes



# Stationary Preferences

- Hum... but why many utility nodes?
- $U(s_0, s_1, s_2, \dots)$ 
  - Infinite process  $\rightarrow$  infinite utility function
- Solution:
  - Assume stationary and additive preferences
  - $U(s_0, s_1, s_2, \dots) = \sum_t R(s_t)$

# Discounted/Average Rewards

- If process infinite, isn't  $\sum_t R(s_t)$  infinite?
- Solution 1: **discounted rewards**
  - Discount factor:  $0 \leq \gamma \leq 1$
  - Finite utility:  $\sum_t \gamma^t R(s_t)$  is a geometric sum
  - $\gamma$  is like 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

# Markov Decision Process

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



# 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:  *$\infty$*
- Tradeoff: *increasing supplies decreases odds of missed sales but increases storage costs*

# Policy

- Choice of action at each time step
- Formally:
  - Mapping from states to actions
  - i.e.,  $\delta(s_t) = a_t$
  - Assumption: **fully observable states**
    - Allows  $a_t$  to be chosen only based on current state  $s_t$ . Why?

# Policy Optimization

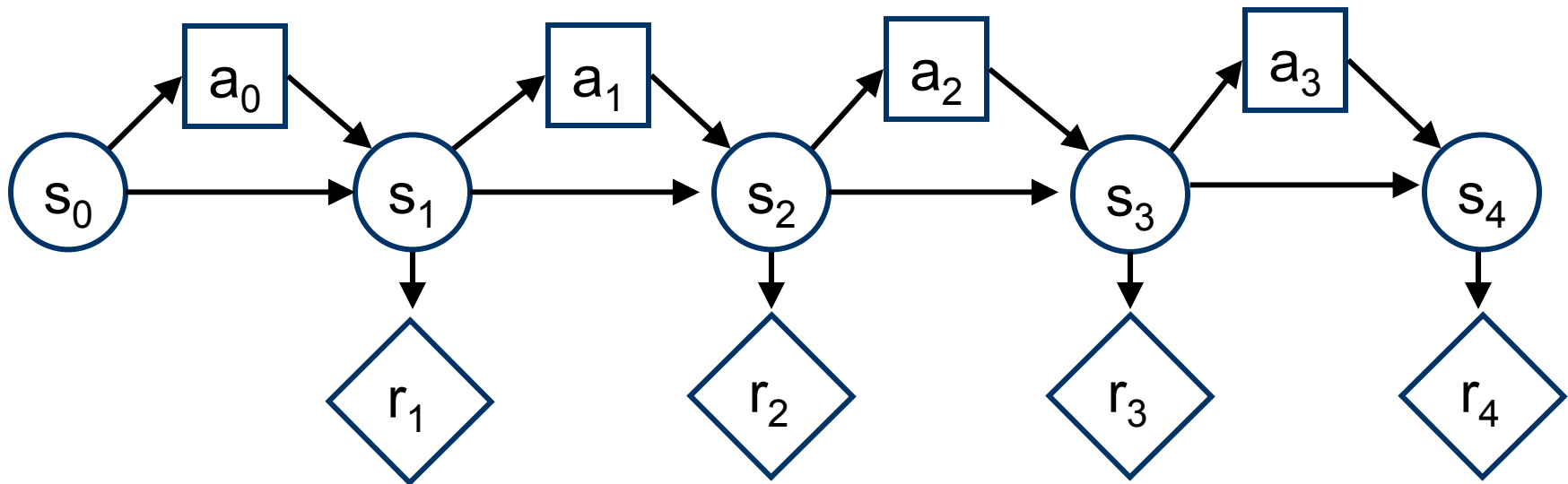
- Policy evaluation:
  - Compute expected utility
  - $EU(\delta) = \sum_{t=0}^h \gamma^t \Pr(s_t|\delta) R(s_t)$
- Optimal policy:
  - Policy with highest expected utility
  - $EU(\delta) \leq EU(\delta^*)$  for all  $\delta$

# Policy Optimization

- Three algorithms to optimize policy:
  - Value iteration
  - Policy iteration
  - Linear Programming
- Value iteration:
  - Equivalent to variable elimination

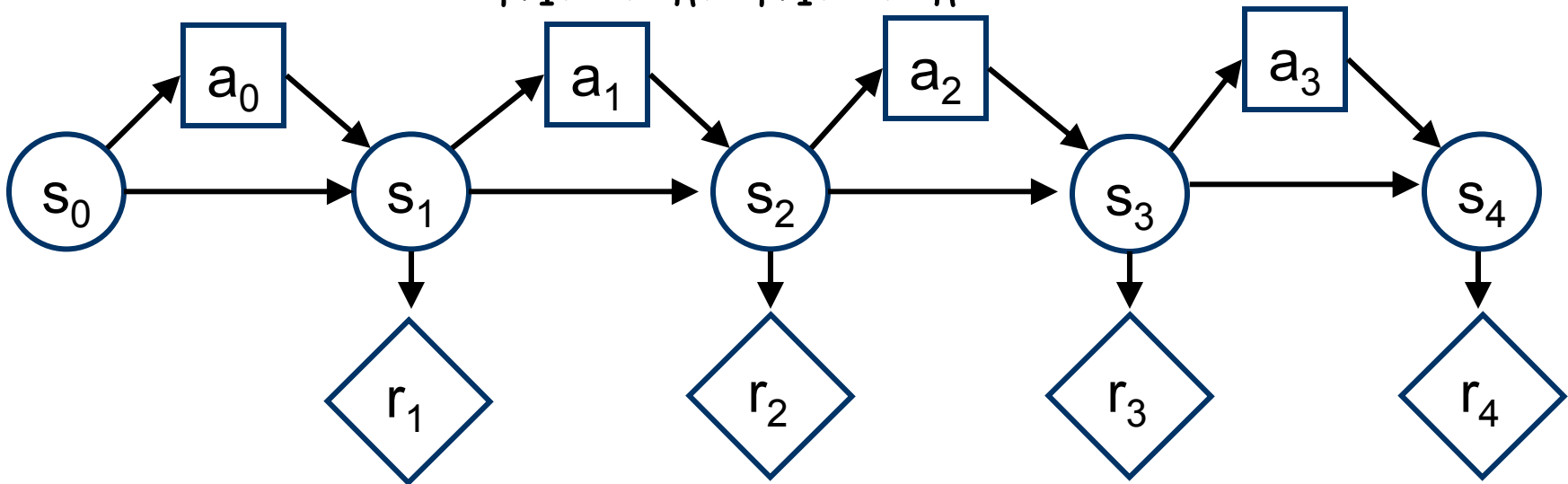
# Value Iteration

- Nothing more than variable elimination
- Performs dynamic programming
- Optimize decisions in reverse order



# Value Iteration

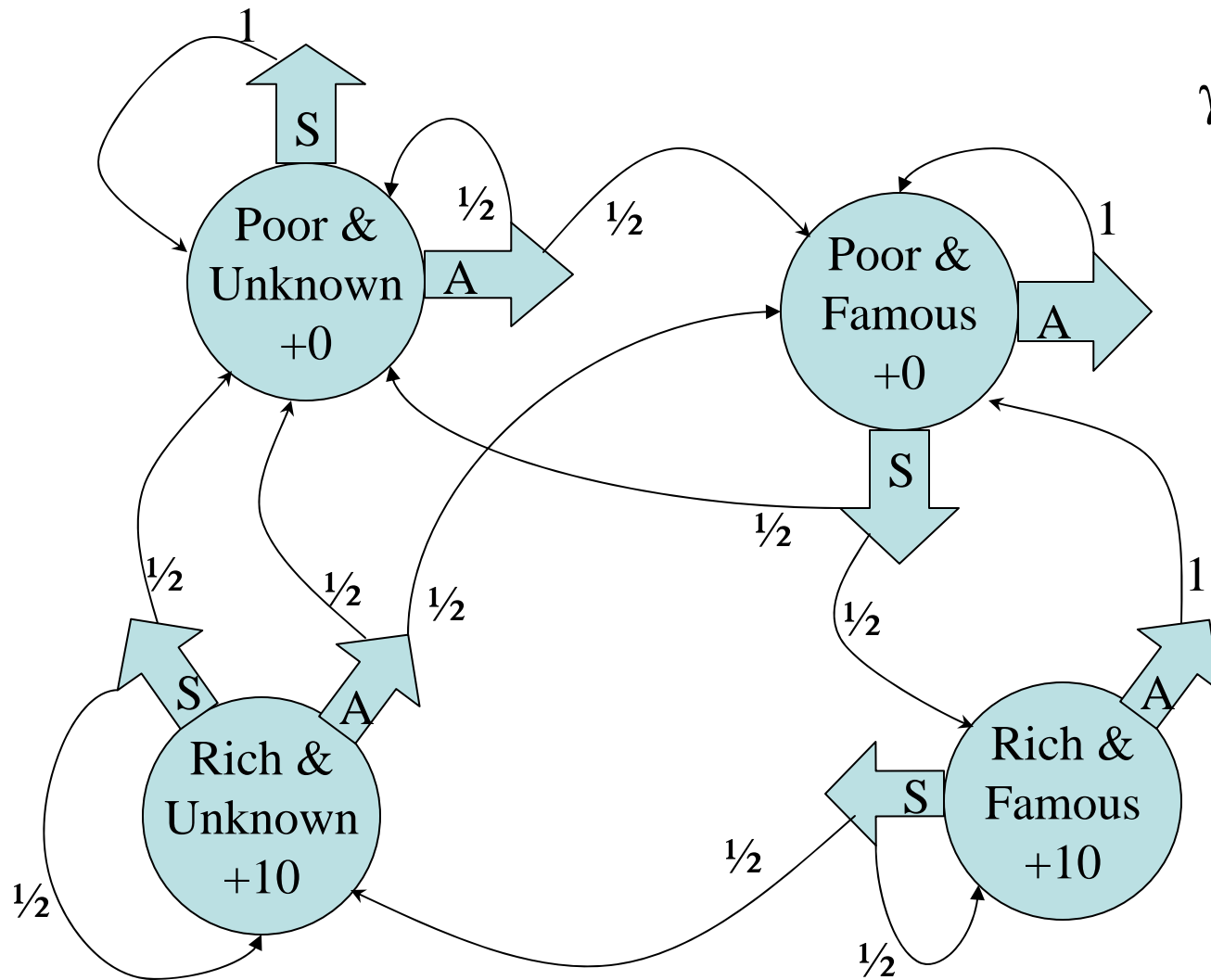
- At each  $t$ , starting from  $t=h$  down to  $0$ :
  - Optimize  $a_t$ :  $EU(a_t|s_t)$ ?
  - Factors:  $Pr(s_{i+1}|a_i,s_i)$ ,  $R(s_i)$ , for  $0 \leq i \leq h$
  - Restrict  $s_t$
  - Eliminate  $s_{t+1}, \dots, s_h, a_{t+1}, \dots, a_h$



# Value Iteration

- Value when no time left:
  - $V(s_h) = R(s_h)$
- Value with one time step left:
  - $V(s_{h-1}) = \max_{a_{h-1}} R(s_{h-1}) + \gamma \sum_{s_h} \Pr(s_h | s_{h-1}, a_{h-1}) V(s_h)$
- Value with two time steps left:
  - $V(s_{h-2}) = \max_{a_{h-2}} R(s_{h-2}) + \gamma \sum_{s_{h-1}} \Pr(s_{h-1} | s_{h-2}, a_{h-2}) V(s_{h-1})$
- ...
- **Bellman's equation:**
  - $V(s_t) = \max_{a_t} R(s_t) + \gamma \sum_{s_{t+1}} \Pr(s_{t+1} | s_t, a_t) V(s_{t+1})$
  - $a_t^* = \operatorname{argmax}_{a_t} R(s_t) + \gamma \sum_{s_{t+1}} \Pr(s_{t+1} | s_t, a_t) V(s_{t+1})$

# A Markov Decision Process

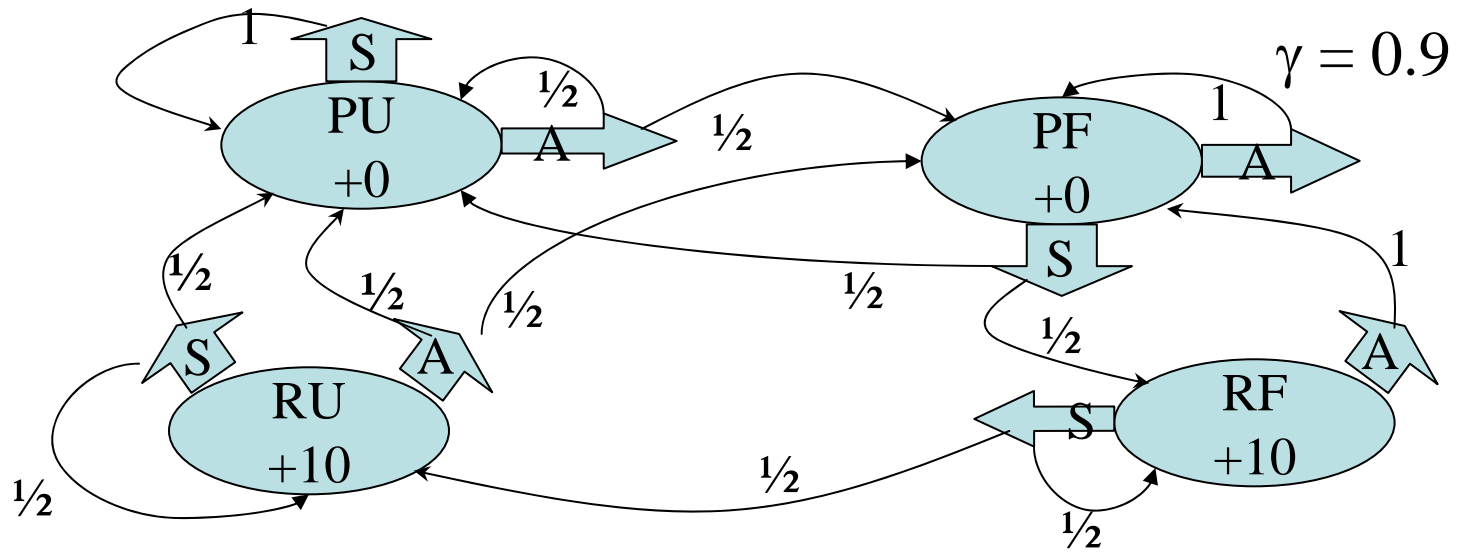


$$\gamma = 0.9$$

You own a company

In every state you must choose between **Saving money** or **Advertising**





$t$	$V(\text{PU})$	$V(\text{PF})$	$V(\text{RU})$	$V(\text{RF})$
$h$	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

# 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

# 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...

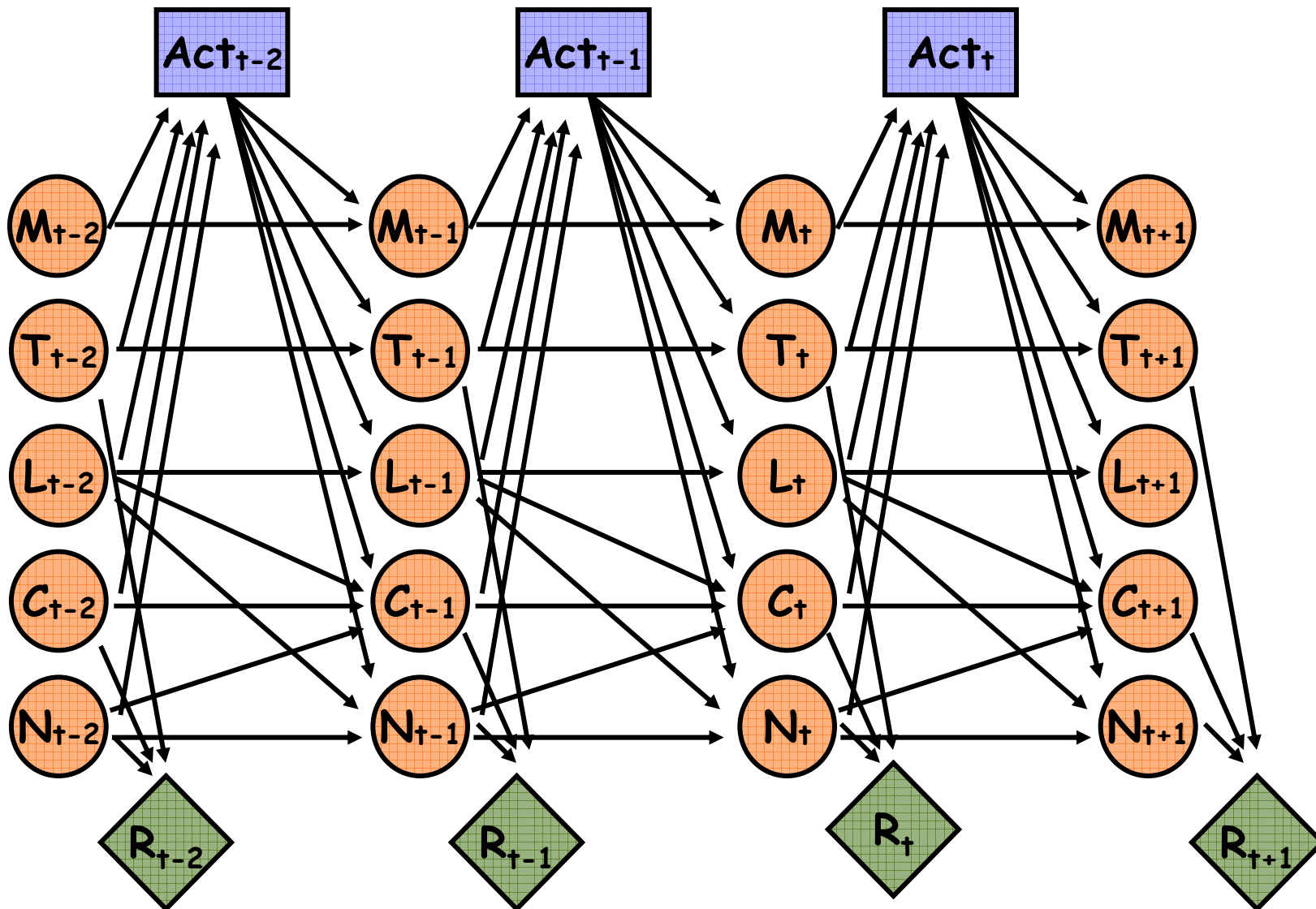
# Infinite Horizon

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

# 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 several random variables and consequently exponential?
- Solution: exploit conditional independence
  - **Dynamic decision network**

# Dynamic Decision Network

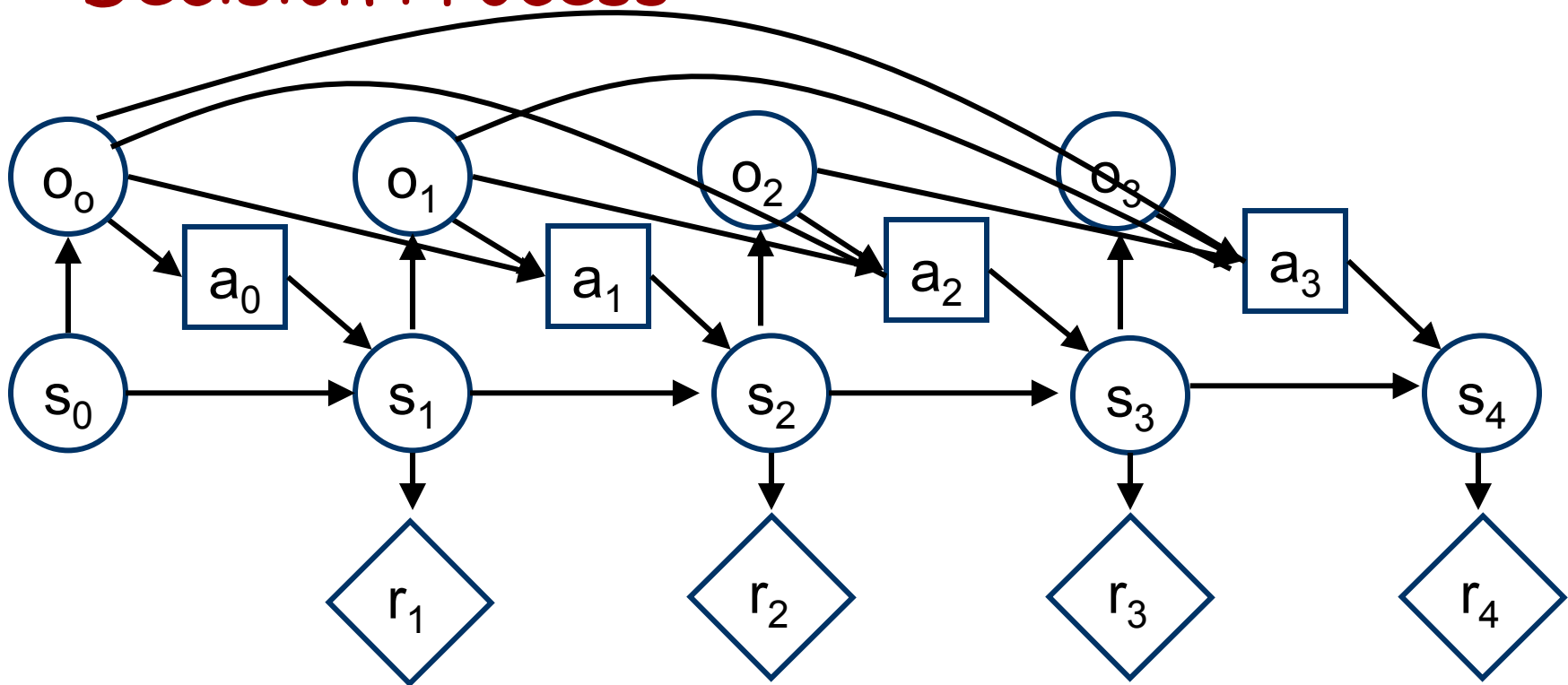


# Dynamic Decision Network

- Similarly to dynamic Bayes nets:
  - Compact representation 😊
  - Exponential time for decision making 😞

# Partial Observability

- What if states are not fully observable?
- Solution: **Partially Observable Markov Decision Process**





# Partially Observable Markov Decision Process (POMDP)

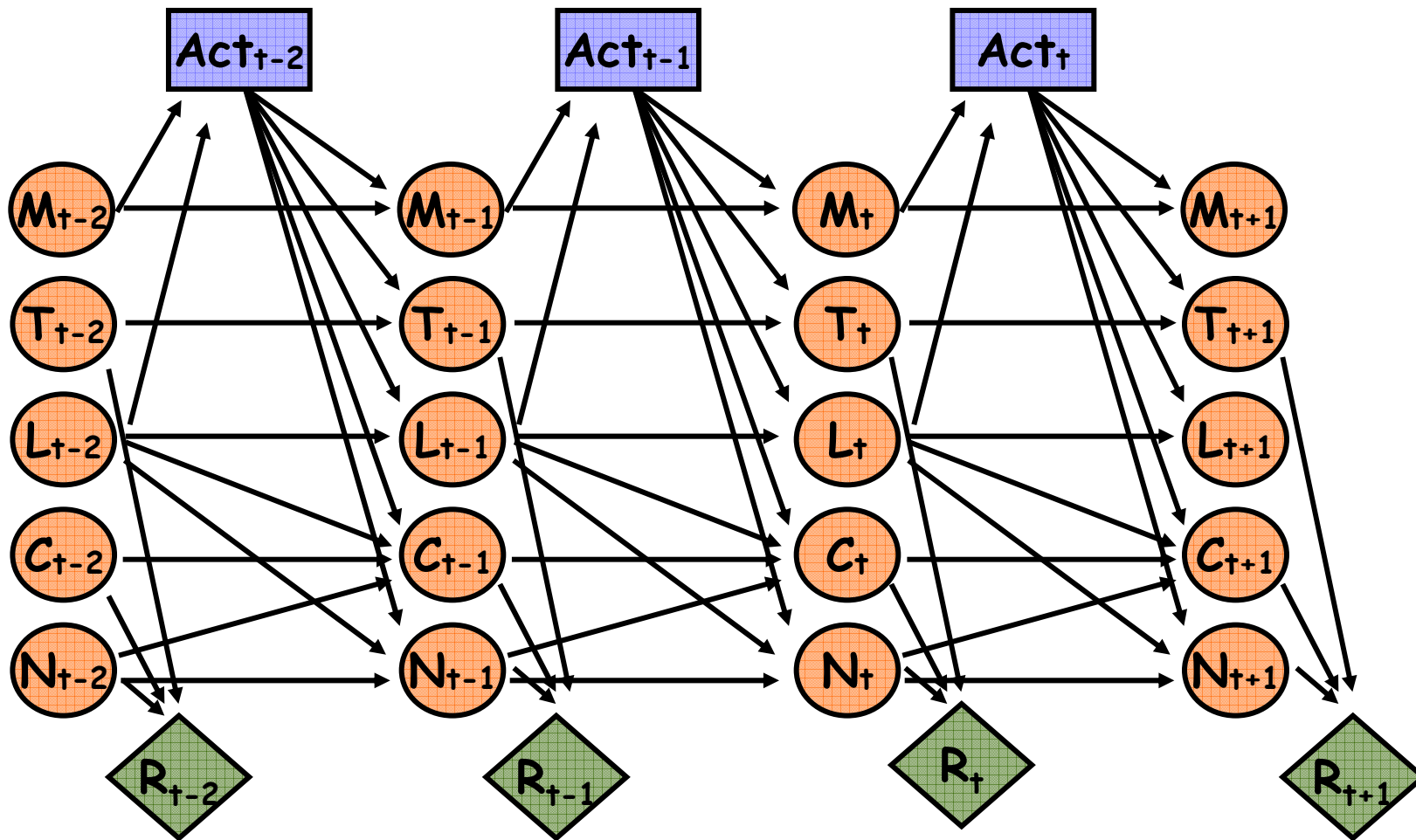
- Definition
  - Set of states:  $S$
  - Set of actions (i.e., decisions):  $A$
  - Set of observations:  $O$
  - Transition model:  $\Pr(s_t | a_{t-1}, s_{t-1})$
  - Observation model:  $\Pr(o_t | s_t)$
  - Reward model (i.e., utility):  $R(s_t)$
  - Discount factor:  $0 \leq \gamma \leq 1$
  - Horizon (i.e., # of time steps):  $h$
- Policy: mapping from past obs. to actions

# POMDP

- Problem: action choice generally depends on **all previous observations...**
- Two solutions:
  - Consider only policies that depend on a finite history of observations
  - Find **stationary sufficient statistics** encoding relevant past observations

# Partially Observable DDN

- Actions do not depend on all state variables



# Policy Optimization

- Policy optimization:
  - Value iteration (variable elimination)
  - Policy iteration
- POMDP and PODDN complexity:
  - Exponential in  $|O|$  and  $k$  when action choice depends on all previous observations ☹️
  - In practice, good policies based on subset of past observations can still be found

# COACH project

- Automated prompting system to help elderly persons wash their hands
- IATSL: Alex Mihailidis, Pascal Poupart, Jennifer Boger, Jesse Hoey, Geoff Fernie and Craig Boutilier



# Aging Population

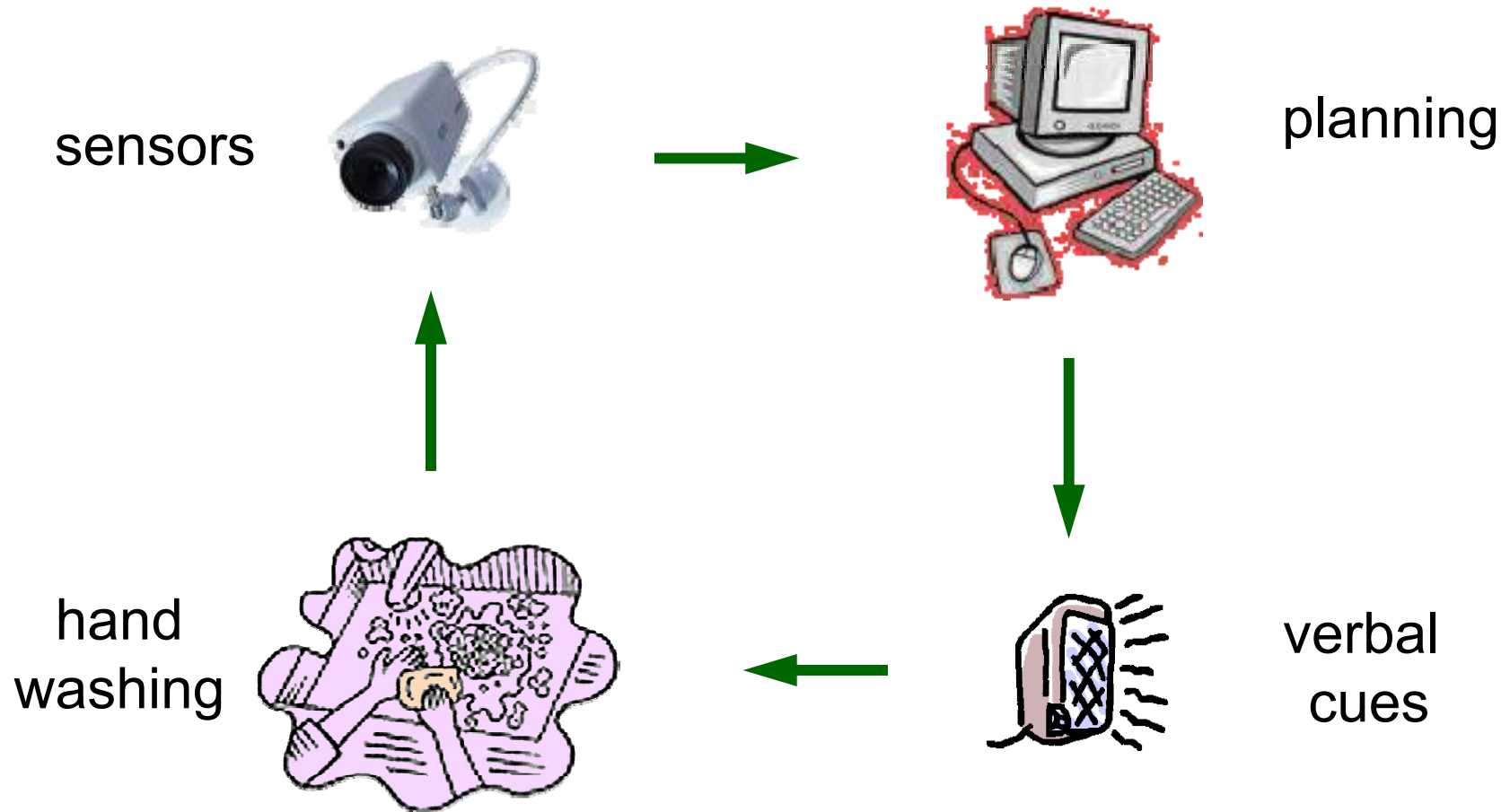
- Dementia
  - Deterioration of intellectual faculties
  - Confusion
  - Memory losses (e.g., Alzheimer's disease)
- Consequences:
  - Loss of autonomy
  - Continual and expensive care required



# Intelligent Assistive Technology

- Let's facilitate aging in place
- Intelligent assistive technology
  - Non-obtrusive, yet pervasive
  - Adaptable
- Benefits:
  - Greater autonomy
  - Feeling of independence

# System Overview





# Prompting Strategy

- Sequential decision problem
  - Sequence of prompts
- Noisy sensors & imprecise actuators
  - Noisy image processing, uncertain prompt effects
- Partially unknown environment
  - Unknown user habits, preferences and abilities
- Tradeoff between complex concurrent goals
  - Rapid task completion vs greater autonomy
- Approach: Partially Observable Markov Decision Processes (POMDPs)

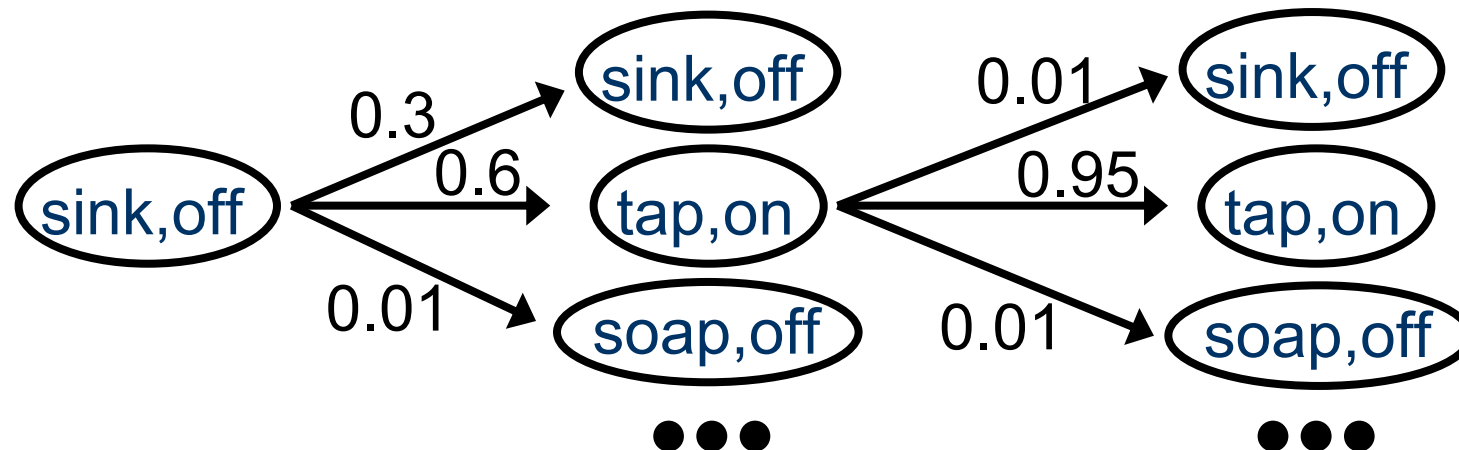
# POMDP components

- **State set  $\mathbf{S}$**  =  $\text{dom}(\text{HL}) \times \text{dom}(\text{WF}) \times \text{dom}(\text{D}) \times \dots$ 
  - Hand Location  $\in \{\text{tap}, \text{water}, \text{soap}, \text{towel}, \text{sink}, \text{away}, \dots\}$
  - Water Flow  $\in \{\text{on}, \text{off}\}$ ,
  - Dementia  $\in \{\text{high}, \text{low}\}$ , etc.
- **Observation set  $\mathbf{O}$**  =  $\text{dom}(\text{C}) \times \text{dom}(\text{FS})$ 
  - Camera  $\in \{\text{handsAtTap}, \text{handsAtTowel}, \dots\}$
  - Faucet sensor  $\in \{\text{waterOn}, \text{waterOff}\}$
- **Action set  $\mathbf{A}$** 
  - DoNothing, CallCaregiver, Prompt  $\in \{\text{turnOnWater}, \text{rinseHands}, \text{useSoap}, \dots\}$

# POMDP components

- Transition function  
 $\Pr(s'|s,a)$

- Observation function  
 $\Pr(o|s)$



- Reward function  $R(s,a)$ 
  - Task completed  $\rightarrow +100$
  - Call caregiver  $\rightarrow -30$
  - Each prompt  $\rightarrow -1, -2$  or  $-3$

# Next Class

- Machine Learning
- Decision Trees