

# Probabilistic planning based on Markov decision processes

Jim Blythe

November 13<sup>th</sup>

(slides due to Craig Boutilier)

# Markov Decision Processes

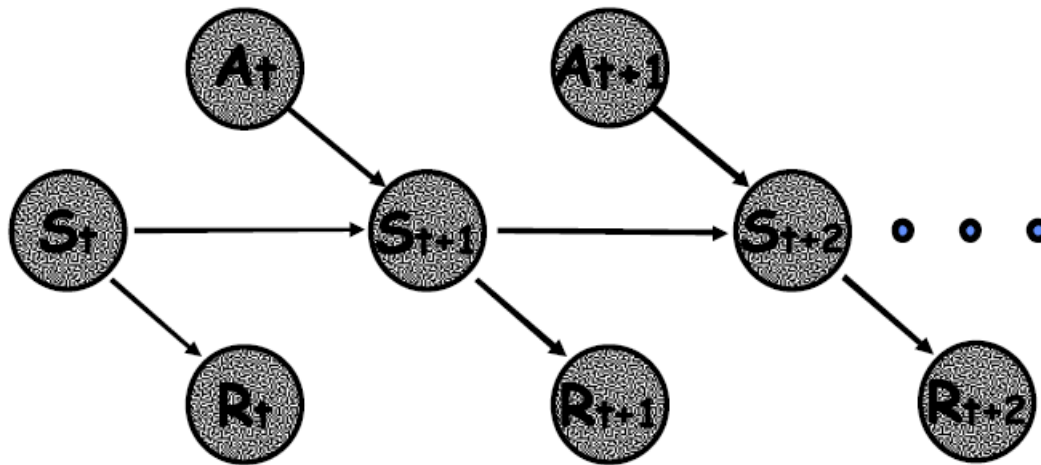
- Components of a fully observable MDP:
  - states  $S$  ( $|S| = n$ )
  - actions  $A$
  - transition function  $\text{Pr}(s,a,t)$ 
    - ↗ represented by set of  $n \times n$  stochastic matrices
  - reward function  $R(s)$ 
    - ↗ represented by  $n$ -vector

|       | $s_1$ | $s_2$ | ... | $s_n$ |
|-------|-------|-------|-----|-------|
| $s_1$ | 0.9   | 0.05  | ... | 0.0   |
| $s_2$ | 0.0   | 0.20  | ... | 0.1   |
| ⋮     |       |       |     |       |
| $s_n$ | 0.1   | 0.0   | ... | 0.0   |

|       | $R$ |
|-------|-----|
| $s_1$ | 12  |
| $s_2$ | 0.5 |
| ⋮     | ⋮   |
| $s_n$ | 10  |

# Graphical View of MDP

---



# Policies

---

- Stationary policy  $\pi$  (does not depend on time)
  - $\pi: S \rightarrow A$
  - $\pi(s)$  = action chosen at state  $s$   
(universal plan)

# Value of a Policy

---

- How good is a policy  $\pi$ ? How do we measure “accumulated” reward?
- **Value function**  $V: S \rightarrow R$  associates value with each state
- $V(\pi, s)$  denotes value of policy  $\pi$  at state  $s$ 
  - expected accumulated reward over horizon of interest
  - note  $V(\pi, s)$  is not  $R(s)$ ; it measures utility
- **Common formulations of value:**
  - Finite horizon  $n$ : total expected reward given  $\pi$
  - Infinite horizon discounted: discounting keeps total bounded

# Finite Horizon Problems

- Utility (value) depends on stage-to-go
  - hence so should policy: nonstationary  $\pi(s,k)$
- $V_{\pi}^k(s)$  is k-stage-to-go value function for  $\pi$

$$V_{\pi}^k(s) = E \left[ \sum_{t=0}^k R^t \mid \pi, s \right]$$

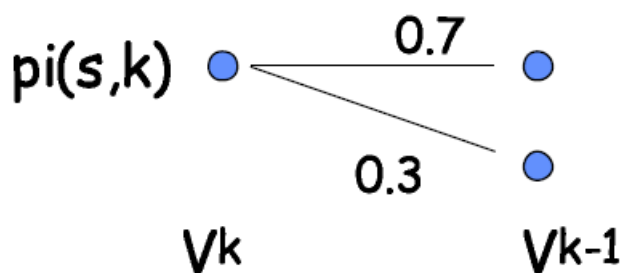
- Here  $R^t$  is a random variable denoting reward received at stage  $t$

# Compute $V$ iteratively

- Compute  $V_{\pi}^k(s)$  by dynamic programming:

(a)  $V_{\pi}^0(s) = R(s), \quad \forall s$

(b)  $V_{\pi}^k(s) = R(s) + \sum_{s'} \Pr(s, \pi(s, k), s') \cdot V_{\pi}^{k-1}(s')$



# Value Iteration

- Markov property allows exploitation of DP principle for optimal policy construction

↗ no need to enumerate  $|A|^{Tn}$  possible policies

- Value Iteration

$$V^0(s) = R(s), \quad \forall s$$

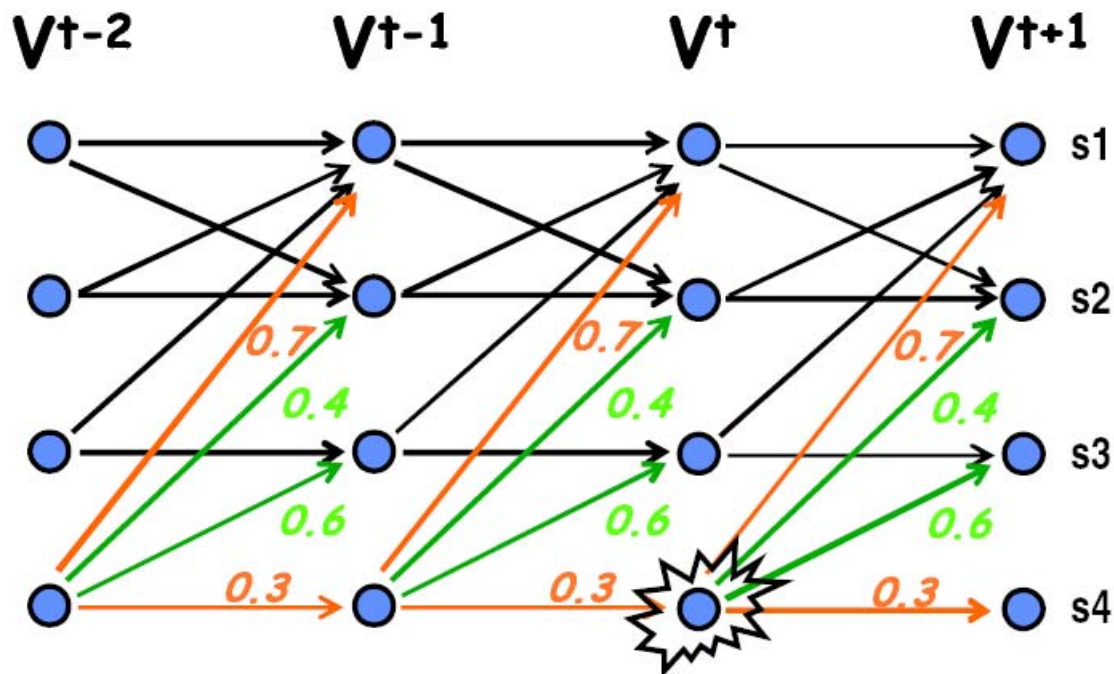
Bellman backup

$$V^k(s) = R(s) + \max_a \sum_{s'} \Pr(s, a, s') \cdot V^{k-1}(s')$$

$$\pi^*(s, k) = \arg \max_a \sum_{s'} \Pr(s, a, s') \cdot V^{k-1}(s')$$

$V^k$  is optimal k-stage-to-go value function

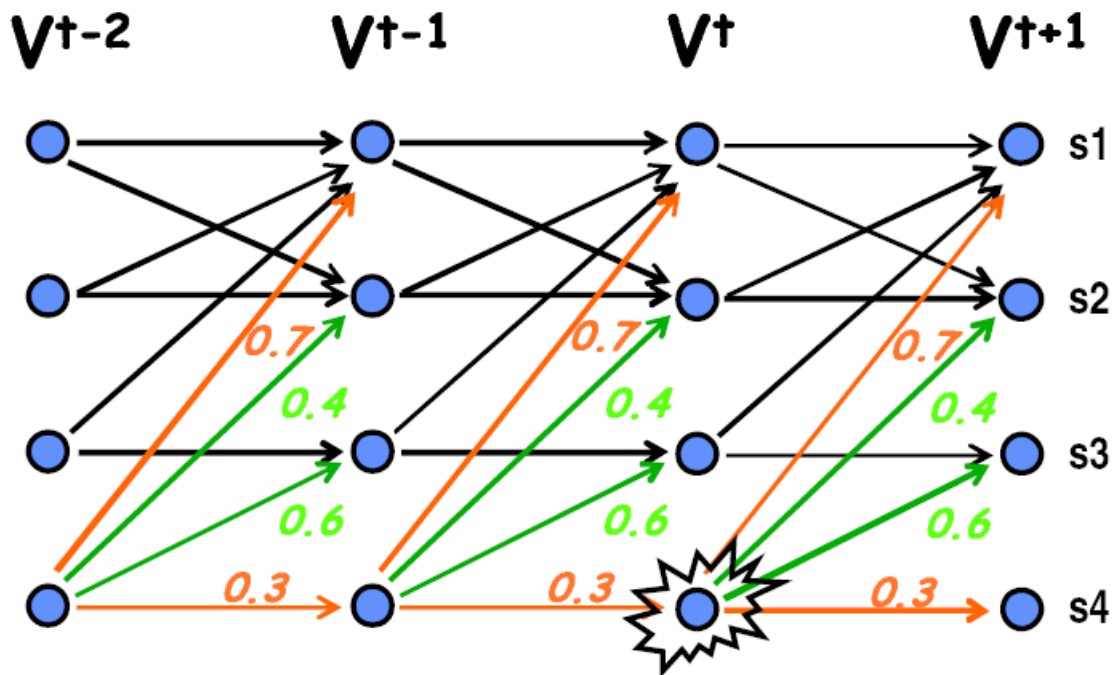
# Value iteration



$$V^t(s_4) = R(s_4) + \max \left\{ \begin{array}{l} 0.7 V^{t+1}(s_1) + 0.3 V^{t+1}(s_4) \\ 0.4 V^{t+1}(s_2) + 0.6 V^{t+1}(s_3) \end{array} \right\}$$

■     ■

# Value iteration: implied policy



$$\Pi^t(s_4) = \max \{ \blacksquare \blacksquare \}$$

## This gives an optimal policy

---

$$V_{\pi^*}^k(s) \geq V_{\pi}^k(s), \quad \forall \pi, s, k$$

- Note: optimal value function is unique, but optimal policy is not

# Discounted Infinite Horizon MDPs

- Total reward problematic (usually)
  - many or all policies have infinite expected reward
- “Trick”: introduce discount factor  $0 \leq \beta < 1$ 
  - Future rewards discounted by  $\beta$  per time step

$$V_{\pi}^k(s) = E \left[ \sum_{t=0}^{\infty} \beta^t R^t \mid \pi, s \right]$$

# Things about policies

---

- Optimal policy maximizes value at each state
- Optimal policies guaranteed to exist (Howard60)
- We define  $V^*(s) = V(\pi, s)$  for some optimal  $\pi$ .

# Value Iteration (again)

---

- Can compute optimal policy using value iteration, just like FH problems (just include discount term)

$$V^k(s) = R(s) + \beta \max_a \sum_{s'} \Pr(s, a, s') \cdot V^{k-1}(s')$$

# How to act based on an approximation to $V^*$

---

- Given  $V^*$  (or approximation), use *greedy* policy:

$$\pi^*(s) = \arg \max_a \sum_{s'} \Pr(s, a, s') \cdot V^*(s')$$

- if  $V$  within  $\epsilon$  of  $V^*$ , then  $V(\pi)$  within  $2\epsilon$  of  $V^*$
- There exists an s.t. optimal policy is returned
  - even if value estimate is off, greedy policy is optimal

# Policy Iteration

---

1. Choose a random policy  $\pi$
  2. Loop:
    - (a) Evaluate  $V_\pi$
    - (b) For each  $s$  in  $S$ , set  $\pi'(s) = \arg \max_a \sum_{s'} \Pr(s, a, s') \cdot V_\pi(s')$
    - (c) Replace  $\pi$  with  $\pi'$
- Until no improving action possible at any state

# Policy iteration notes

---

- Very flexible algorithm
  - need only improve policy at one state (not each state)
- Gives exact value of optimal policy
- Generally converges much faster than VI
  - each iteration more complex, but fewer iterations
  - quadratic rather than linear rate of convergence

# So why not just use MDPs?

---

- Most AI problems are **feature-based**
  - $|S|$  exponential in number of variables
  - Spec./Rep'n of problem in state form impractical
  - Explicit state-based DP impractical
- Require structured representations
  - exploit regularities in probabilities, rewards
- Require structured computation
  - exploit regularities in policies, value functions
  - can aid in approximation (anytime computation)

# Structured representation

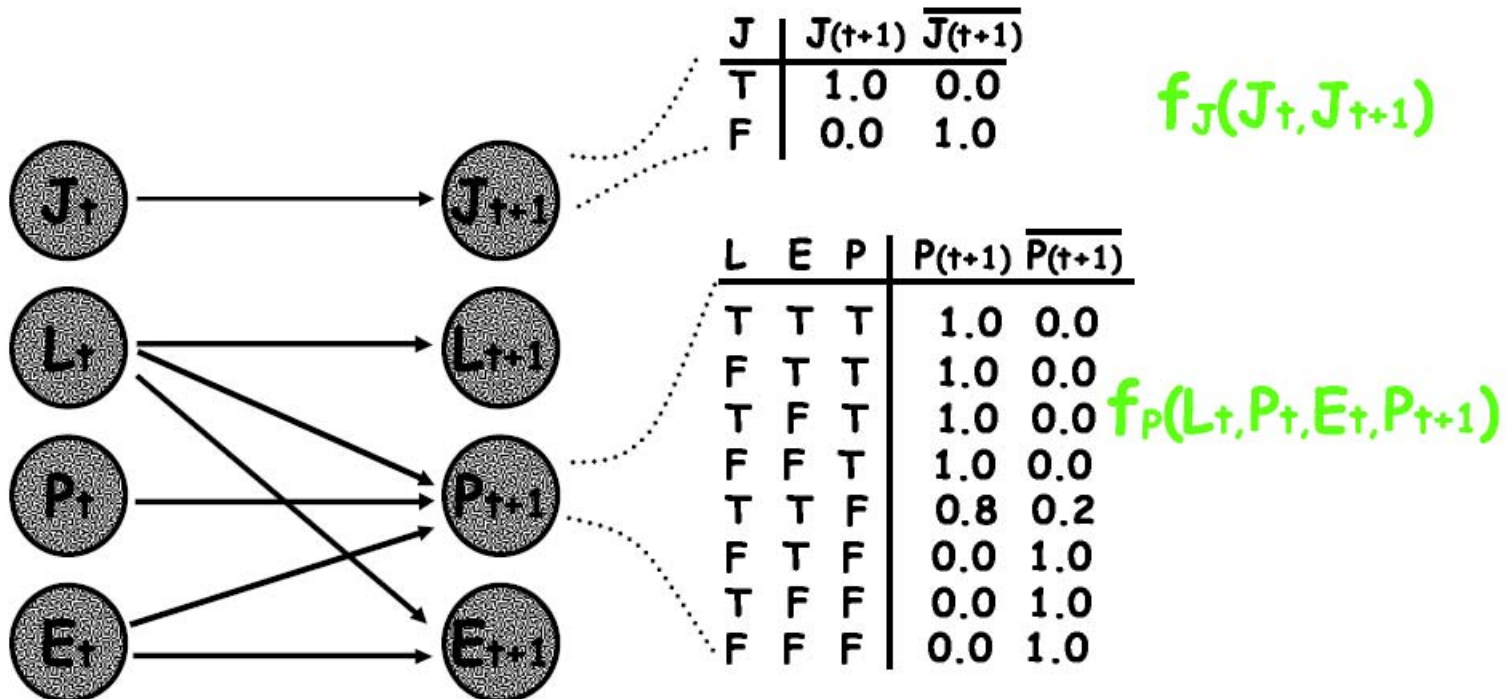
- States decomposable into *state variables*

$$S = X_1 \times X_2 \times \dots \times X_n$$

- *Structured* representations the norm in AI
  - STRIPS, Sit-Calc., Bayesian networks, etc.
  - Describe *how actions affect/depend on features*
  - Natural, concise, can be exploited computationally
- Same ideas can be used for MDPs
  - actions, rewards, policies, value functions, etc.
  - dynamic Bayes nets [DeanKanazawa89,BouDeaGol95]
  - decision trees and diagrams [BouDeaGol95,Hoeyetal99]

# Action Representation – DBN/ADD

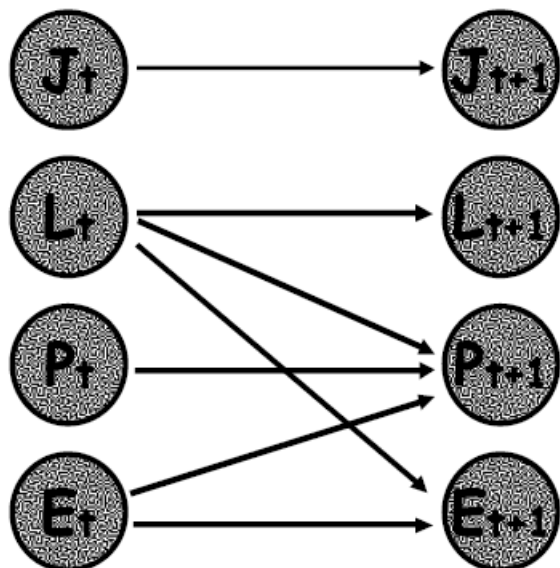
## Pickup Printout



# Action Representation – DBN/ADD

$$\Pr(J_{t+1}, L_{t+1}, P_{t+1}, E_{t+1} \mid J_t, L_t, P_t, E_t)$$

$$= f_J(J_t, J_{t+1}) * f_P(L_t, P_t, E_t, P_{t+1}) * f_L(L_t, L_{t+1}) * f_E(E_t, E_{t+1})$$



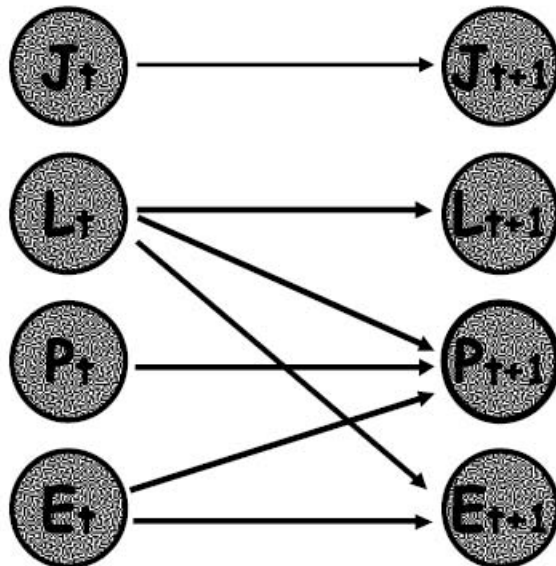
- Only 28 parameters vs. 256 for matrix

|       | $s_1$ | $s_2$ | ... | $s_{256}$ |
|-------|-------|-------|-----|-----------|
| $s_1$ | 0.9   | 0.05  | ... | 0.0       |
| $s_2$ | 0.0   | 0.20  | ... | 0.1       |
| ...   |       |       |     |           |
| $s_6$ | 0.1   | 0.0   | ... | 0.0       |

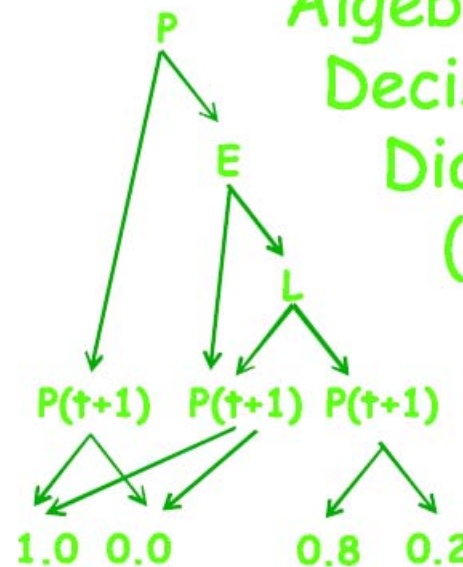
- Removes global exponential dependence

# Action Representation – DBN/ADD

Pickup Printout



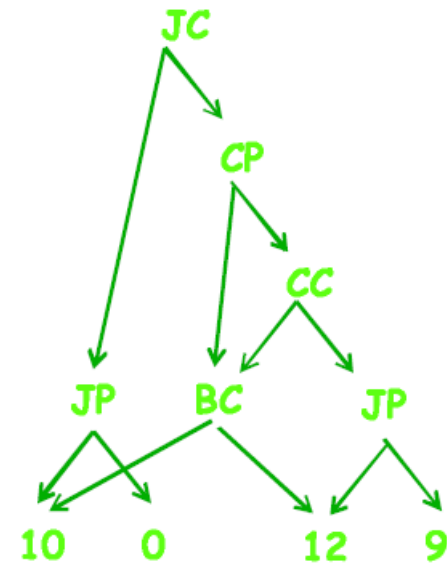
Algebraic Decision Diagram (ADD)



- ADDs, decision trees, Horn rules, ... (natural)

# Reward Representation

- Rewards represented similarly
  - save on  $2n$  size of vector rep'n



# Reward Representation

- Additive independent reward also very common
  - as in multiattribute utility theory
  - offers more natural and concise representation for many types of problems



# Key Question:

## Structured computation

---

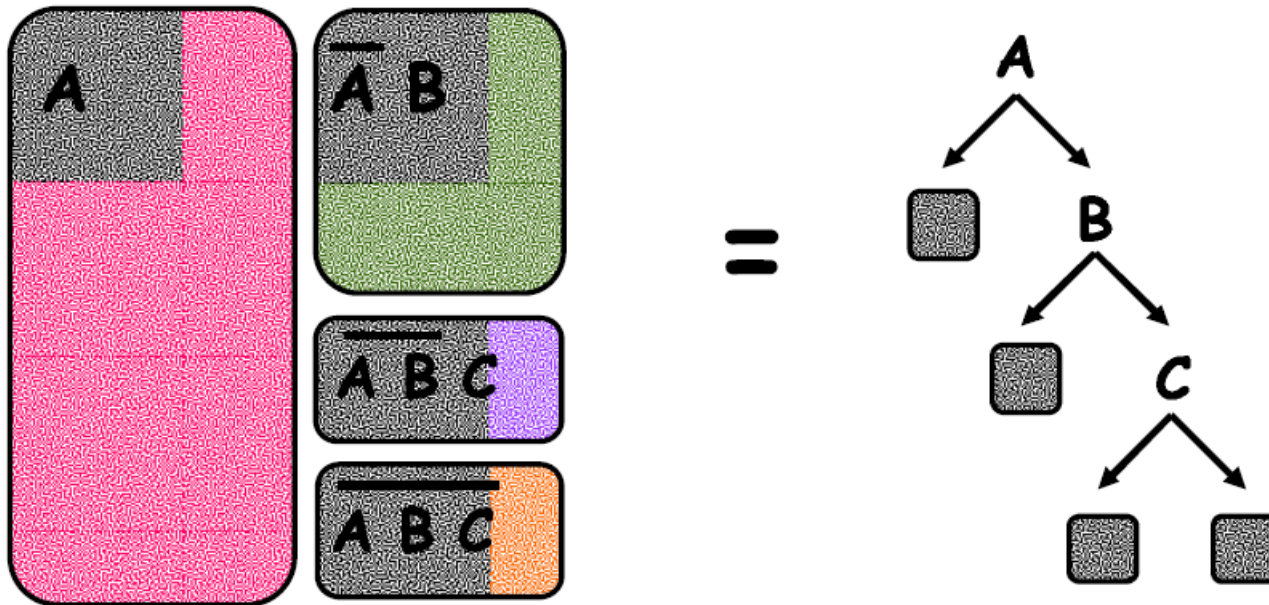
- Given compact representation, can we solve MDP without explicit state space enumeration?
- Can we avoid  $O(|S|)$ -computations by exploiting regularities made explicit by DBNs/ADDs?

# State Space Abstraction

---

- General method: ***state aggregation***
  - group states, treat aggregate as single state
- ***Abstraction*** is a specific aggregation technique
  - aggregate by ignoring details (features)
  - ideally, focus on **relevant** features

# Graphical view of abstraction

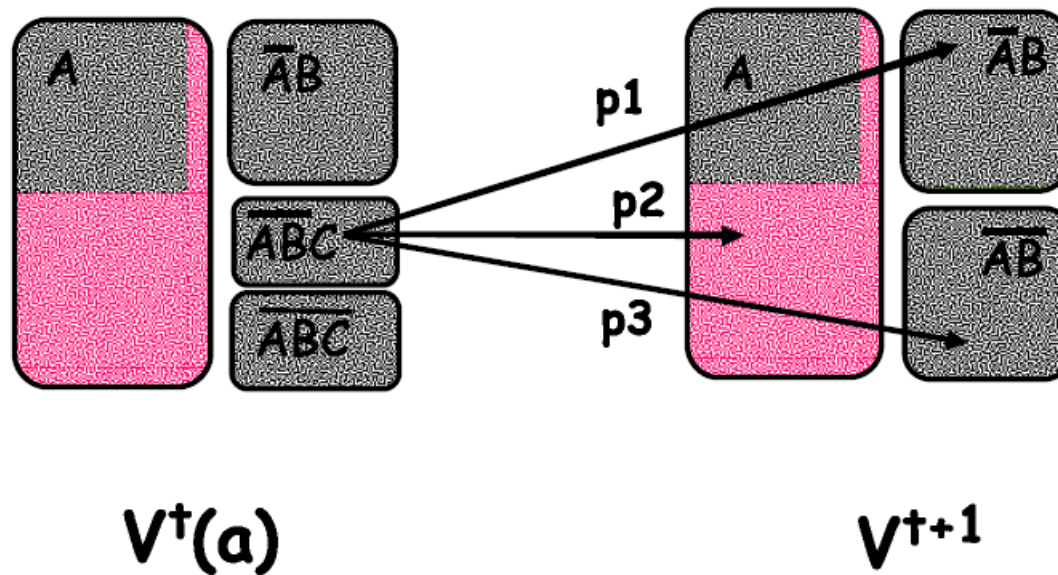


# Decision-Theoretic Regression

---

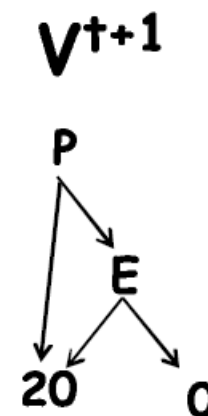
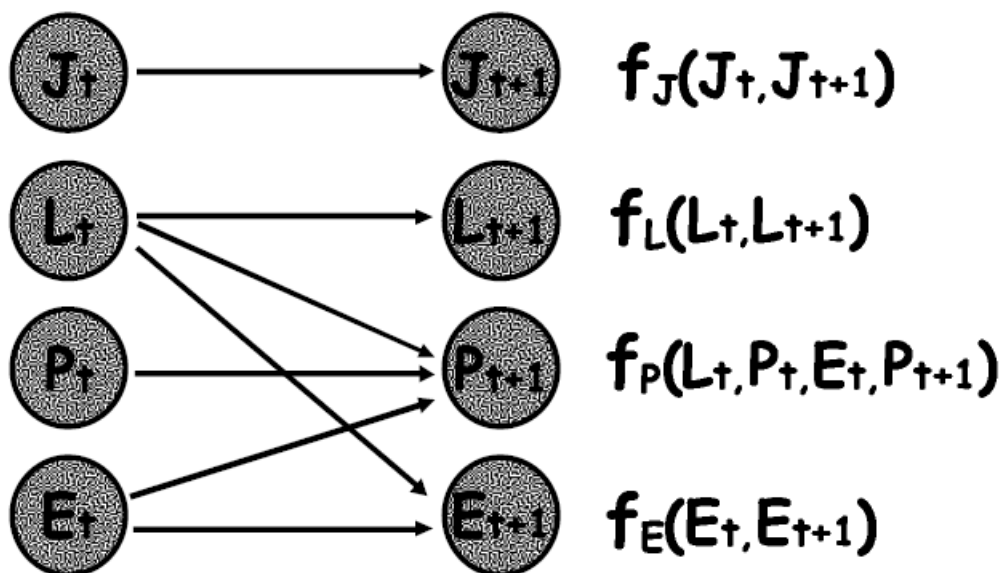
- Goal regression a classical abstraction method
  - $Regr(G,a)$  is a logical condition  $C$  under which  $a$  leads to  $G$  (aggregates  $C$  states and  $\sim C$  states)
- Decision-theoretic analog: given “logical description” of  $V_{t+1}$ , produce such a description of  $V_t$  or optimal policy (e.g., using ADDs)
- Cluster together states at any point in calculation with **same best action** (policy), or with **same value** (VF)

# Graphical view of decision-theoretic regression



# Functional View of DTR

- Generally,  $V_{t+1}$  depends on only a subset of variables (usually in a structured way)
- What is value of action  $a$  at time  $t$  (at any  $s$ )?



# Decision-theoretic refinement

---

- Assume  $V^a$   $V_{t+1}$  is structured: what is value of doing action  $a$  at time  $t$ ?

$$V_t^a(J_t, L_t, P_t, E_t)$$

# Decision-theoretic refinement

- Assume  $V_{t+1}$  is structured: what is value of doing action  $a$  at time  $t$ ?

$$V_t^a(J_t, L_t, P_t, E_t) \\ = R + \sum_{J, L, P, E(t+1)} \Pr^a(J_{t+1}, L_{t+1}, P_{t+1}, E_{t+1} \mid J_t, L_t, P_t, E_t) V_{t+1}(J_{t+1}, L_{t+1}, P_{t+1}, E_{t+1})$$

# Decision-theoretic refinement

- Assume  $V_{t+1}$  is structured: what is value of doing action  $a$  at time  $t$ ?

$$V_t^a(J_t, L_t, P_t, E_t)$$

$$= R + \sum_{J, L, P, E(t+1)} \Pr^a(J_{t+1}, L_{t+1}, P_{t+1}, E_{t+1} \mid J_t, L_t, P_t, E_t) V_{t+1}(J_{t+1}, L_{t+1}, P_{t+1}, E_{t+1})$$

$$= R + \sum_{J, L, P, E(t+1)} f_J(J_t, J_{t+1}) f_P(L_t, P_t, E_t, P_{t+1}) f_L(L_t, L_{t+1}) f_E(E_t, E_{t+1}) V_{t+1}(P_{t+1}, E_{t+1})$$

# Decision-theoretic refinement

- Assume  $V^a$   $V_{t+1}$  is structured: what is value of doing action  $a$  at time  $t$  ?

$$V_t^a(J_t, L_t, P_t, E_t)$$

$$= R + \sum_{J, L, P, E(t+1)} \Pr^a(J_{t+1}, L_{t+1}, P_{t+1}, E_{t+1} \mid J_t, L_t, P_t, E_t) V_{t+1}(J_{t+1}, L_{t+1}, P_{t+1}, E_{t+1})$$

$$= R + \sum_{J, L, P, E(t+1)} f_J(J_t, J_{t+1}) f_P(L_t, P_t, E_t, P_{t+1}) f_L(L_t, L_{t+1}) f_E(E_t, E_{t+1}) V_{t+1}(P_{t+1}, E_{t+1})$$

$$= R + \sum_{L, P, E(t+1)} f_P(L_t, P_t, E_t, P_{t+1}) f_L(L_t, L_{t+1}) f_E(E_t, E_{t+1}) V_{t+1}(P_{t+1}, E_{t+1})$$

# Decision-theoretic refinement

- Assume  $V^a$   $V_{t+1}$  is structured: what is value of doing action  $a$  at time  $t$ ?

$$V_t^a(J_t, L_t, P_t, E_t)$$

$$= R + \sum_{J, L, P, E(t+1)} \Pr^a(J_{t+1}, L_{t+1}, P_{t+1}, E_{t+1} \mid J_t, L_t, P_t, E_t) V_{t+1}(J_{t+1}, L_{t+1}, P_{t+1}, E_{t+1})$$

$$= R + \sum_{J, L, P, E(t+1)} f_J(J_t, J_{t+1}) f_P(L_t, P_t, E_t, P_{t+1}) f_L(L_t, L_{t+1}) f_E(E_t, E_{t+1}) V_{t+1}(P_{t+1}, E_{t+1})$$

$$= R + \sum_{L, P, E(t+1)} f_P(L_t, P_t, E_t, P_{t+1}) f_L(L_t, L_{t+1}) f_E(E_t, E_{t+1}) V_{t+1}(P_{t+1}, E_{t+1})$$

- $V_t(a)$  depends on subset of variables as well
  - Each component function represented as ADD
  - ADD operations allow structure to be preserved

# Planning by DTR

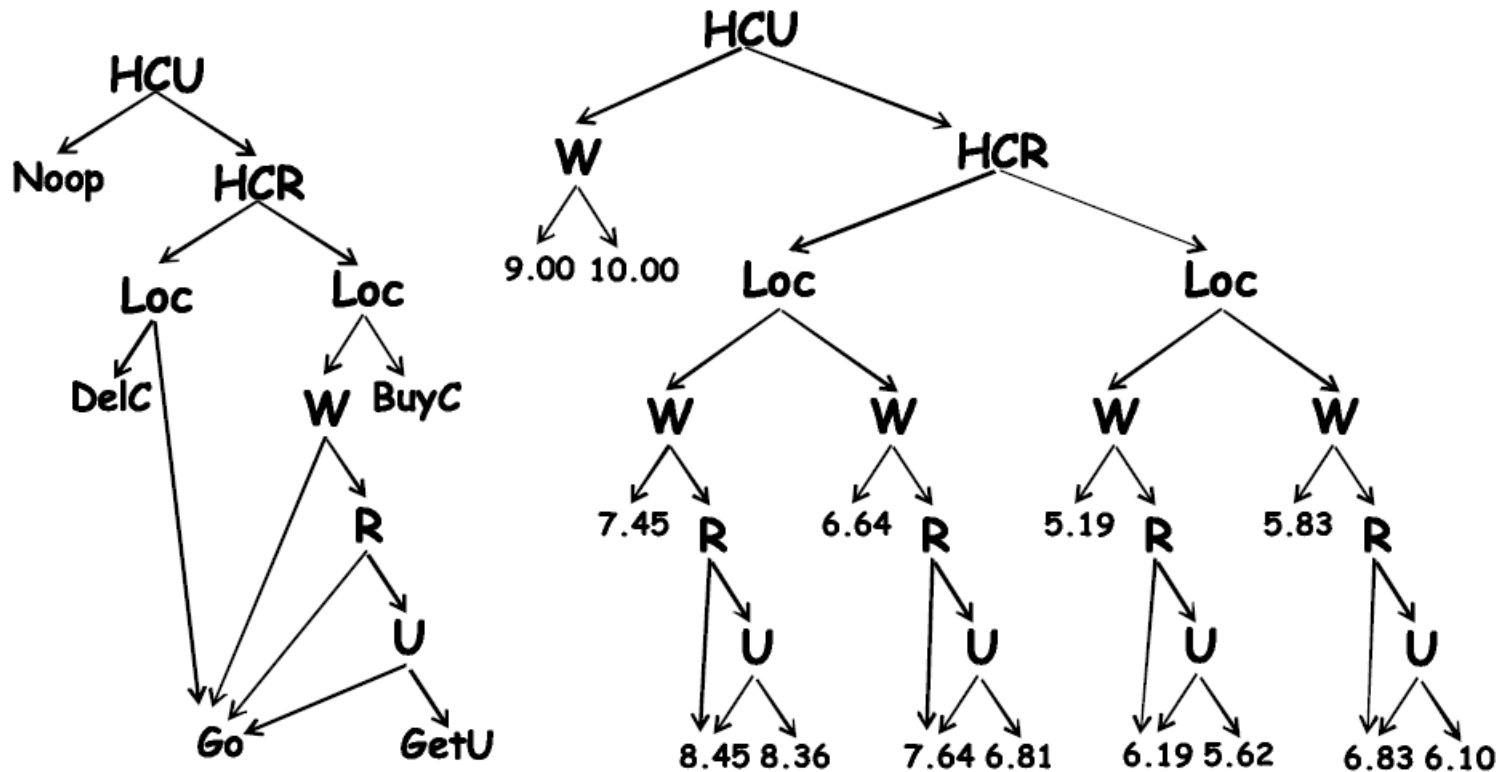
---

- Standard DP algorithms can be implemented using structured DTR
- All operations exploit ADD rep'n and algorithms
  - multiplication, summation, maximization of functions
  - standard ADD packages very fast
- Several variants possible
  - MPI/VI with decision trees [**BouDeaGol95,00; Bou97;BouDearden96**]
  - MPI/VI with ADDs [**HoeyStAubinHuBoutilier99, 00**]

# Structured Value Iteration

- Assume compact representation of  $V_k$ 
  - start with  $R$  at stage-to-go 0 (say)
- For each action  $a$ , compute  $Q_{k+1}$  using variable elimination on the two-slice DBN
  - eliminate all  $k$ -variables, leaving only  $k+1$  variables
  - use ADD operations if initial rep'n allows
- Compute  $V_{k+1} = \max_a Q_{k+1}$ 
  - use ADD operations if initial representation allows
- Policy iteration can be approached similarly

# Structured Policy and Value Function

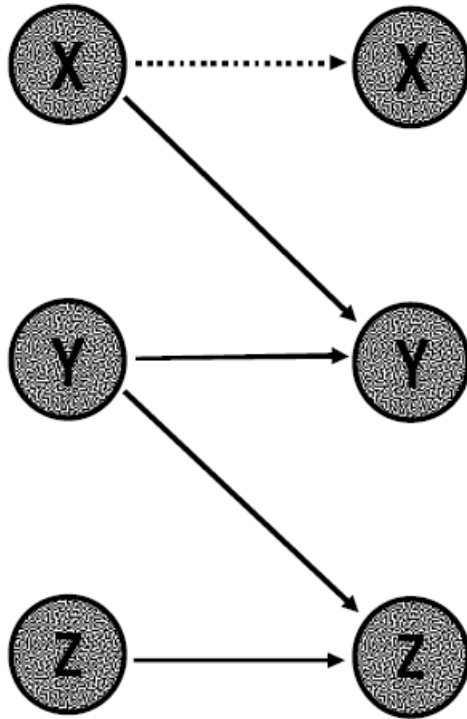


# Structured Policy Evaluation: Trees

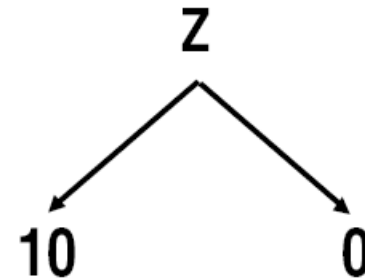
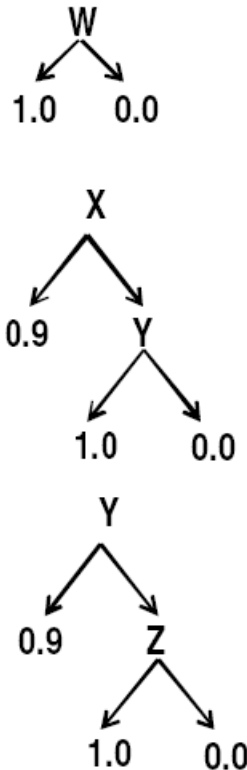
---

- Assume a tree for  $V^t$ , produce  $V^{t+1}$
- For each distinction  $Y$  in  $\text{Tree}(V^t)$ :
  - a) use 2TBN to discover conditions affecting  $Y$
  - b) piece together using the structure of  $\text{Tree}(V^t)$
- Result is a tree exactly representing  $V^{t+1}$ 
  - dictates conditions under which leaves (values) of  $\text{Tree}(V^t)$  are reached with *fixed* probability
- **A decision theoretic form of regression**

# A Simple Action/Reward Example

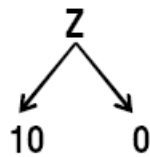


Network Rep'n for Action  $A$

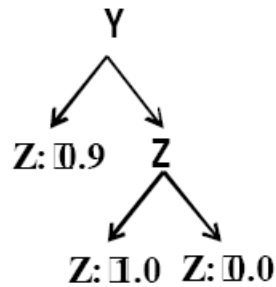


Reward Function  $R$

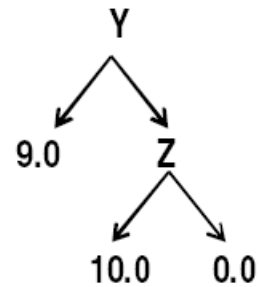
# Example: Generation of $V^1$



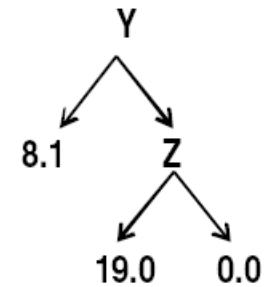
$V^0 = \mathbb{R}$



Step 1

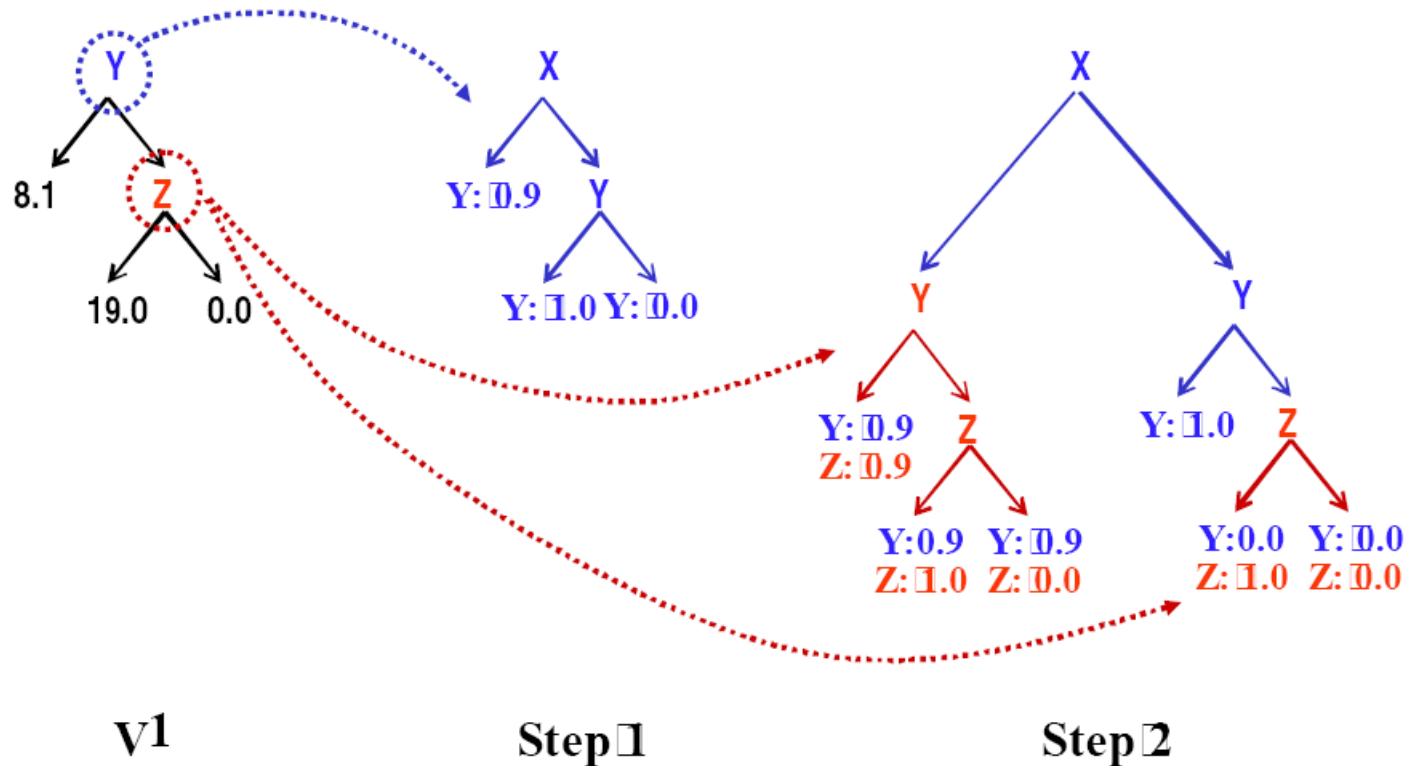


Step 2



Step 3:  $V^1$

# Example: Generation of V2



# DTR: Relative Merits

---

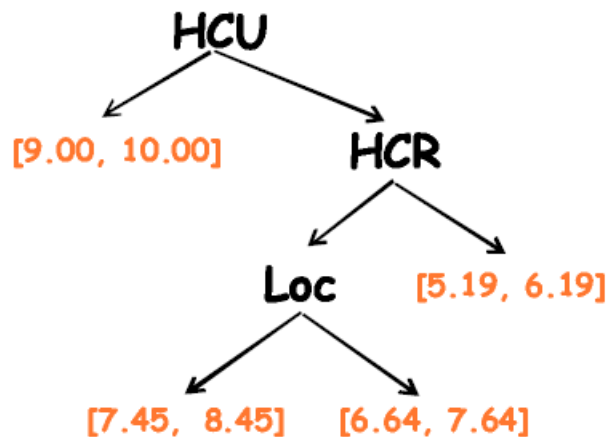
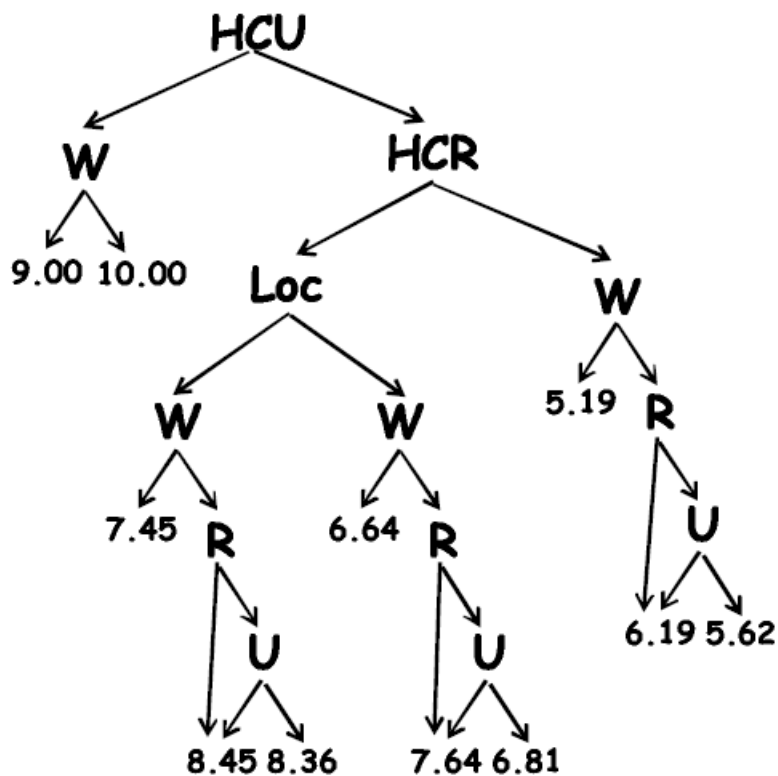
- Adaptive, nonuniform, exact abstraction method
  - provides exact solution to MDP
  - much more efficient on certain problems (time/space)
  - 400 million state problems (ADDs) in a couple hrs
  
- Some drawbacks
  - produces piecewise constant VF
  - some problems admit no compact solution representation (though ADD overhead “minimal”)
  - approximation may be desirable or necessary

# Can easily approximate DTR:

---

- Simple *pruning* of value function
  - Can prune trees [BouDearden96] or ADDs [StaubinHoeyBou00]
- Gives regions of *approximately same value*

# A pruned value ADD



# Approximate DTR: Relative Merits

---

## ■ Relative merits of ADTR

- fewer regions implies faster computation
- can provide leverage for *optimal* computation
- 30-40 billion state problems in a couple hours
- allows fine-grained control of time vs. solution quality with dynamic (*a posteriori*) error bounds
- technical challenges: variable ordering, convergence, fixed vs. adaptive tolerance, etc.

## ■ Some drawbacks

- (still) produces piecewise constant VF
- doesn't exploit additive structure of VF at all

# Summary

---

- MDPs produce optimal solutions and can exploit dynamic programming, BUT state space hard to control
- We can make the standard MDP algorithms use structured representations of actions and reward, by backing up the structure through the policy and value functions.
- Often, we still need to make approximations.