Lecture 10

June 1, 2006

CS 486/686

Outline

- Decision making
  - Utility Theory
  - Decision Trees

- Chapter 16 in R&N
  - Note: Some of the material we are covering today is not in the textbook

Decision Making under Uncertainty

- I give robot a planning problem: I want coffee
  - but coffee maker is broken: robot reports "No plan!"
- If I want more robust behavior – if I want robot to know what to do if my primary goal can't be satisfied – I should provide it with some indication of my *preferences over alternatives*
  - e.g., coffee better than tea, tea better than water, water better than nothing, etc.

Decision Making under Uncertainty

- But it's more complex:
  - it could wait 45 minutes for coffee maker to be fixed
  - what's better: tea now? coffee in 45 minutes?
  - could express preferences for <beverage,time> pairs

Preferences

- A *preference ordering* $\succcurlyeq$ is a ranking of all possible states of affairs (worlds) S
  - these could be outcomes of actions, truth assts, states in a search problem, etc.
  - s $\succcurlyeq$ t: means that state s is *at least as good as* t
  - s $\succ$ t: means that state s is *strictly preferred to* t
  - s~t: means that the agent is *indifferent* between states s and t

Preferences

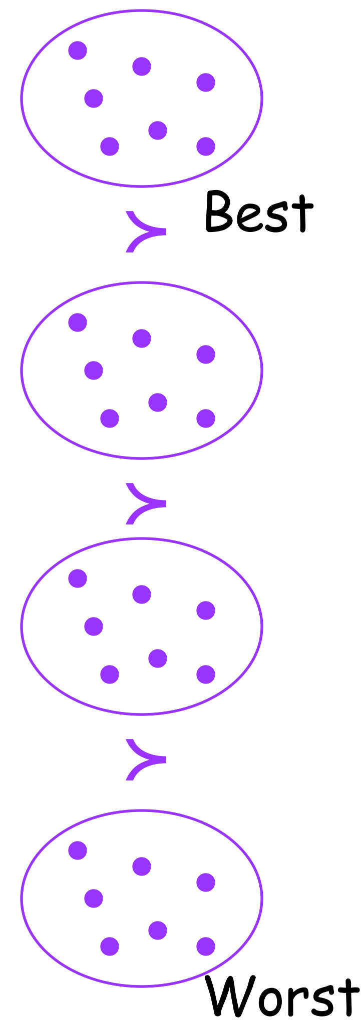
- If an agent's actions are deterministic then we know what states will occur
- If an agent's actions are not deterministic then we represent this by lotteries
  - Probability distribution over outcomes
  - Lottery $L=[p_1,s_1;p_2,s_2;\ldots;p_n,s_n]$
  - s_1 occurs with prob p_1, s_2 occurs with prob p_2,…

Axioms

- **Orderability:** Given 2 states A and B
  - $(A \succ B) \vee (B \succ A) \vee (A \sim B)$
- **Transitivity:** Given 3 states, A, B, and C
  - $(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$
- **Continuity:**
  - $A \succ B \succ C \Rightarrow \exists p\ [p,A;1-p,C] \sim B$
- **Substitutability:**
  - $A \sim B \rightarrow [p,A;1-p,C] \sim [p,B;1-p,C]$
- **Monotonicity:**
  - $A \succ B \Rightarrow (p \geq q \Leftrightarrow [p,A;1-p,B] \succsim [q,A;1-q,B]$
- **Decomposibility:**
  - $[p,A;1-p,[q,B;1-q,C]] \sim [p,A;(1-p)q,B;\ (1-p)(1-q),C]$

Why Impose These Conditions?

- Structure of preference ordering imposes certain "rationality requirements" (it is a weak ordering)
- E.g., why transitivity?
  - Suppose you (strictly) prefer coffee to tea, tea to OJ, OJ to coffee
  - If you prefer X to Y, you'll trade me Y plus $1 for X
  - I can construct a "money pump" and extract arbitrary amounts of money from you



Decision Problems: Certainty

- A *decision problem under certainty* is:
  - a set of *decisions* D
    - e.g., paths in search graph, plans, actions, etc.
  - a set of *outcomes* or states S
    - e.g., states you could reach by executing a plan
  - an *outcome function* $f : D \rightarrow S$
    - the outcome of any decision
  - a preference ordering $\succcurlyeq$ over S
- A *solution* to a decision problem is any $d^* \in D$ such that $f(d^*) \succcurlyeq f(d)$ for all $d \in D$

Computational Issues

- At some level, solution to a dec. prob. is trivial
  - complexity lies in the fact that the decisions and outcome function are rarely specified explicitly
  - e.g., in planning or search problems, you *construct* the set of decisions by constructing paths or exploring search paths -- don't know outcomes in advance!

c, m, ~bc

$\succ$

c, m, bc

$\succ$

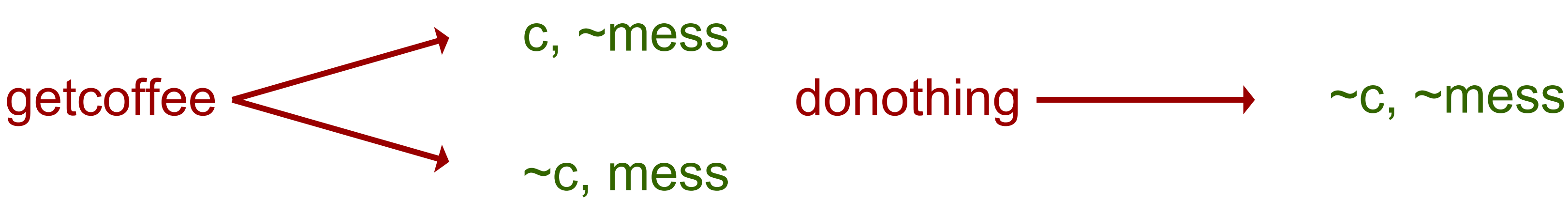
c, ~m, ~bc

$\succ$

c, ~m, bc

- E.g., my robot domain
  - We find a plan satisfying c, m, bc
  - Can we stop searching?
  - Must convince ourselves no better plan exists (nothing can reach best)
  - Generally requires searching entire plan space, unless we have some clever tricks

Decision Making under Uncertainty



- Suppose actions don't have deterministic outcomes
  - e.g., when robot pours coffee, it spills 20% of time, making a mess
  - preferences: c, ~mess ≻ ~c,~mess ≻ ~c, mess
- What should robot do?
  - decision *getcoffee* leads to a good outcome and a bad outcome with some probability
  - decision *donothing* leads to a medium outcome for sure
- Should robot be optimistic? pessimistic?
- Really odds of success should influence decision
  - but how?

Utilities

- Rather than just ranking outcomes, we must quantify our degree of preference
  - e.g., how much more important is c than ~mess
- A *utility function* $U:S \rightarrow \mathbb{R}$ associates a real-valued *utility* with each outcome.
  - U(s) measures your *degree* of preference for s
- Note: U induces a preference ordering $\succcurlyeq_U$ over S defined as: $s \succcurlyeq_U t$ iff $U(s) \geq U(t)$
  - obviously $\succcurlyeq_U$ will be reflexive, transitive, connected

Expected Utility

- Under conditions of uncertainty, each decision d induces a distribution Pr_d over possible outcomes
  - $Pr_d(s)$ is probability of outcome s under decision d

- The *expected utility* of decision d is defined

$$EU(d) = \sum_{s \in S} \Pr_d(s) U(s)$$

Expected Utility

getcoffee → c, ~mess
getcoffee → ~c, mess

donothing → ~c, ~mess

When robot pours coffee, it spills 20% of time, making a mess

If U(c,~ms) = 10, U(~c,~ms) = 5, U(~c,ms) = 0,
then EU(getcoffee) = (0.8)(10)+(0.2)(0)=8
and EU(donothing) = 5

If U(c,~ms) = 10, U(~c,~ms) = 9, U(~c,ms) = 0,
then EU(getcoffee) = (0.8)(10)+(0.2)(0)=8
and EU(donothing) = 9

The MEU Principle

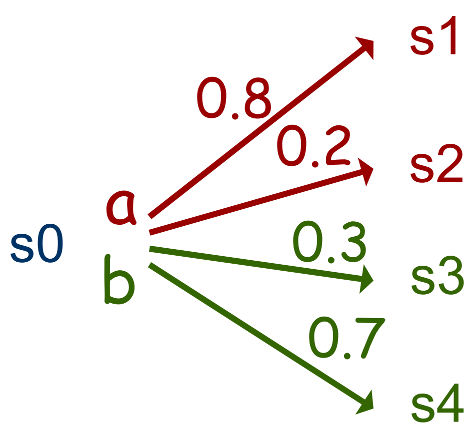
- The *principle of maximum expected utility (MEU)* states that the optimal decision under conditions of uncertainty is that with the greatest expected utility.
- In our example
  - if my utility function is the first one, my robot should get coffee
  - if your utility function is the second one, your robot should do nothing

Decision Problems: Uncertainty

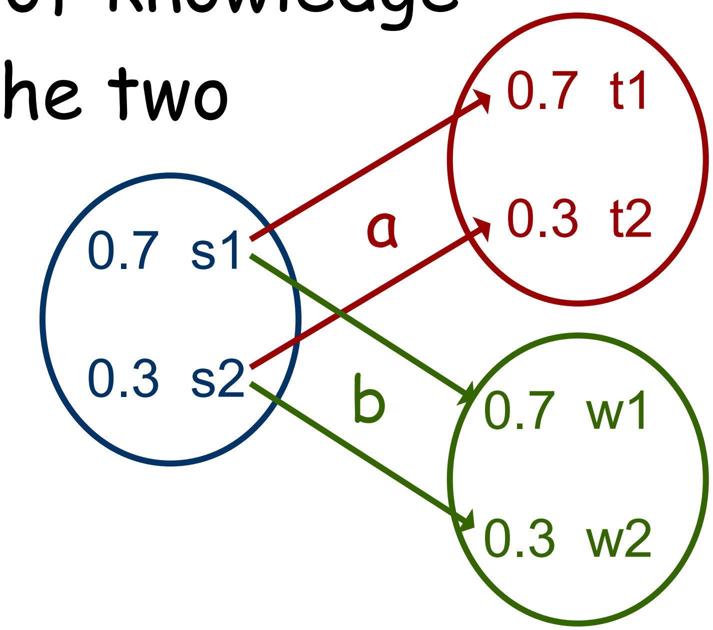
- A *decision problem under uncertainty* is:
  - a set of *decisions* D
  - a set of *outcomes* or states S
  - an *outcome function* $Pr : D \rightarrow \Delta(S)$
    - $\Delta(S)$ is the set of distributions over S (e.g., Pr_d)
  - a utility function U over S
- A *solution* to a decision problem under uncertainty is any $d^* \in D$ such that $EU(d^*) \succcurlyeq EU(d)$ for all $d \in D$
- Again, for single-shot problems, this is trivial

Expected Utility: Notes

- Note that this viewpoint accounts for both:
  - uncertainty in action outcomes
  - uncertainty in state of knowledge
  - any combination of the two



Stochastic actions



Uncertain knowledge

Expected Utility: Notes

- Why MEU? Where do utilities come from?
  - underlying foundations of utility theory tightly couple utility with action/choice
  - a utility function can be determined by asking someone about their preferences for actions in specific scenarios (or "lotteries" over outcomes)
- Utility functions needn't be unique
  - if I multiply U by a positive constant, all decisions have same relative utility
  - if I add a constant to U, same thing
  - *U is unique up to positive affine transformation*

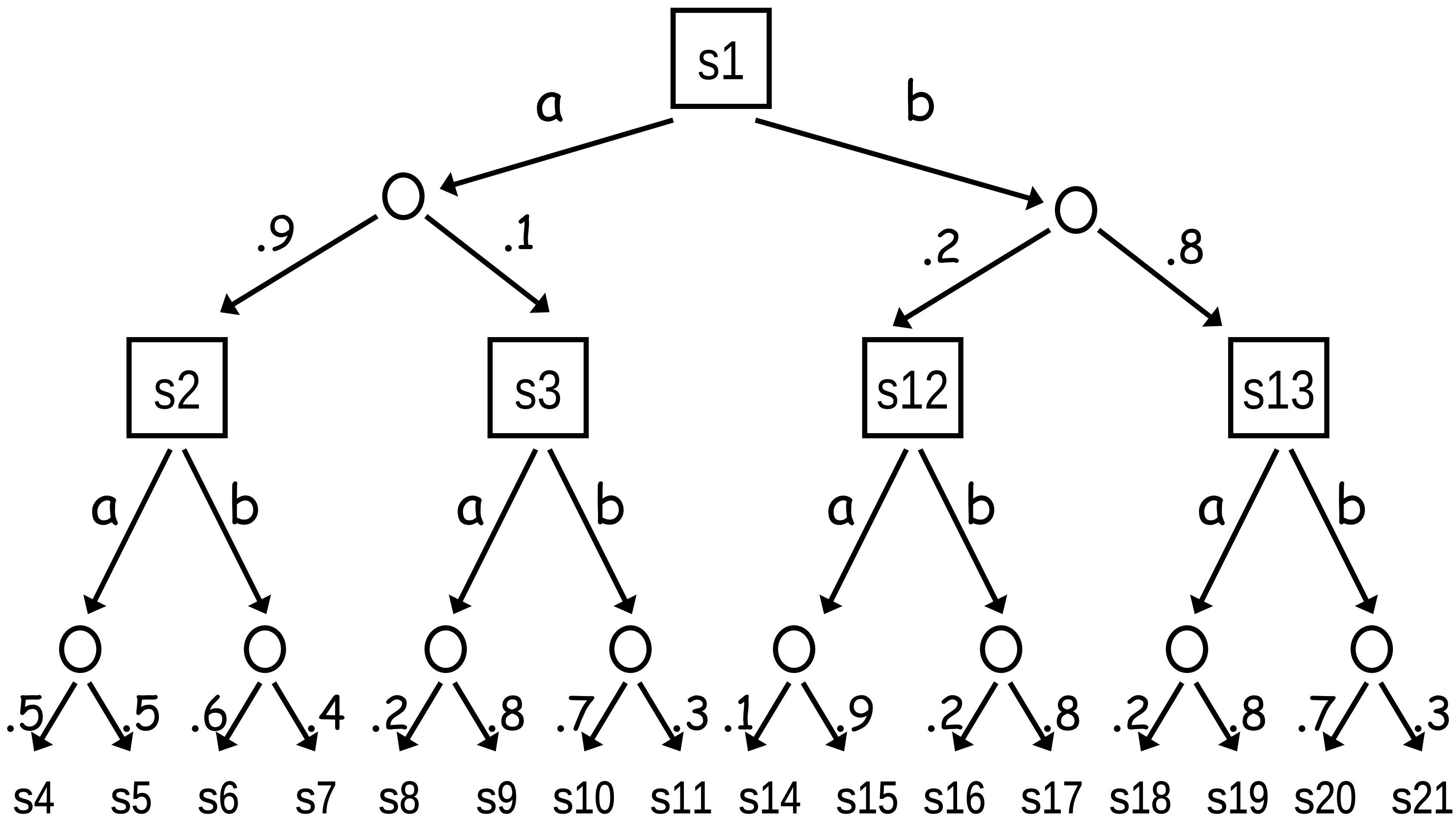
So What are the Complications?

- Outcome space is large
  - like all of our problems, states spaces can be huge
  - don't want to spell out distributions like Pr_d explicitly
  - Soln: Bayes nets (or related: *influence diagrams*)
- Decision space is large
  - usually our decisions are not one-shot actions
  - rather they involve sequential choices (like plans)
  - if we treat each plan as a distinct decision, decision space is too large to handle directly
  - Soln: use dynamic programming methods to *construct* optimal plans (actually generalizations of plans, called policies… like in game trees)

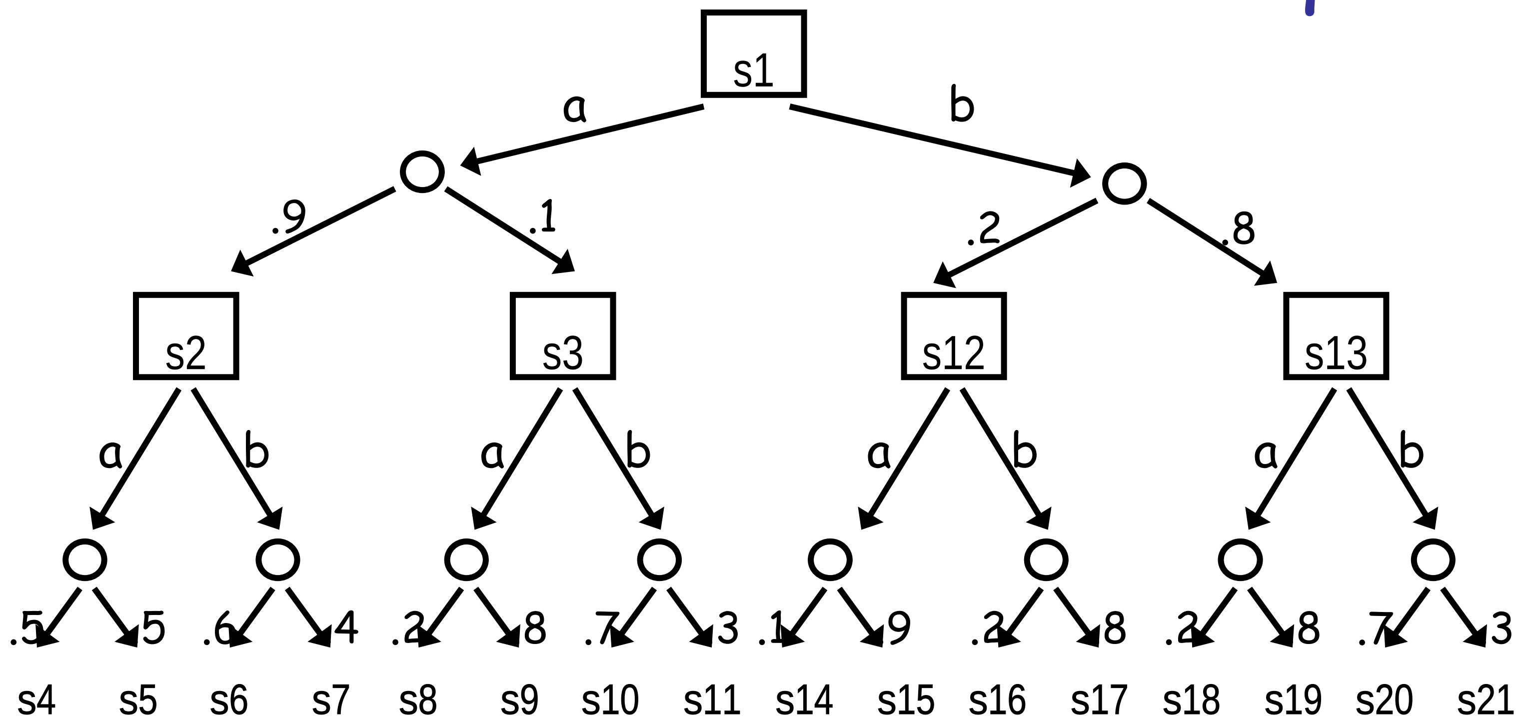
A Simple Example

- Suppose we have two actions: a, b
- We have time to execute *two* actions in sequence
- This means we can do either:
  - [a,a], [a,b], [b,a], [b,b]
- Actions are stochastic: action a induces distribution $Pr_a(s_i \mid s_j)$ over states
  - e.g., $Pr_a(s_2 \mid s_1) = .9$ means prob. of moving to state s_2 when a is performed at s_1 is .9
  - similar distribution for action b
- How good is a particular sequence of actions?

Distributions for Action Sequences



Distributions for Action Sequences



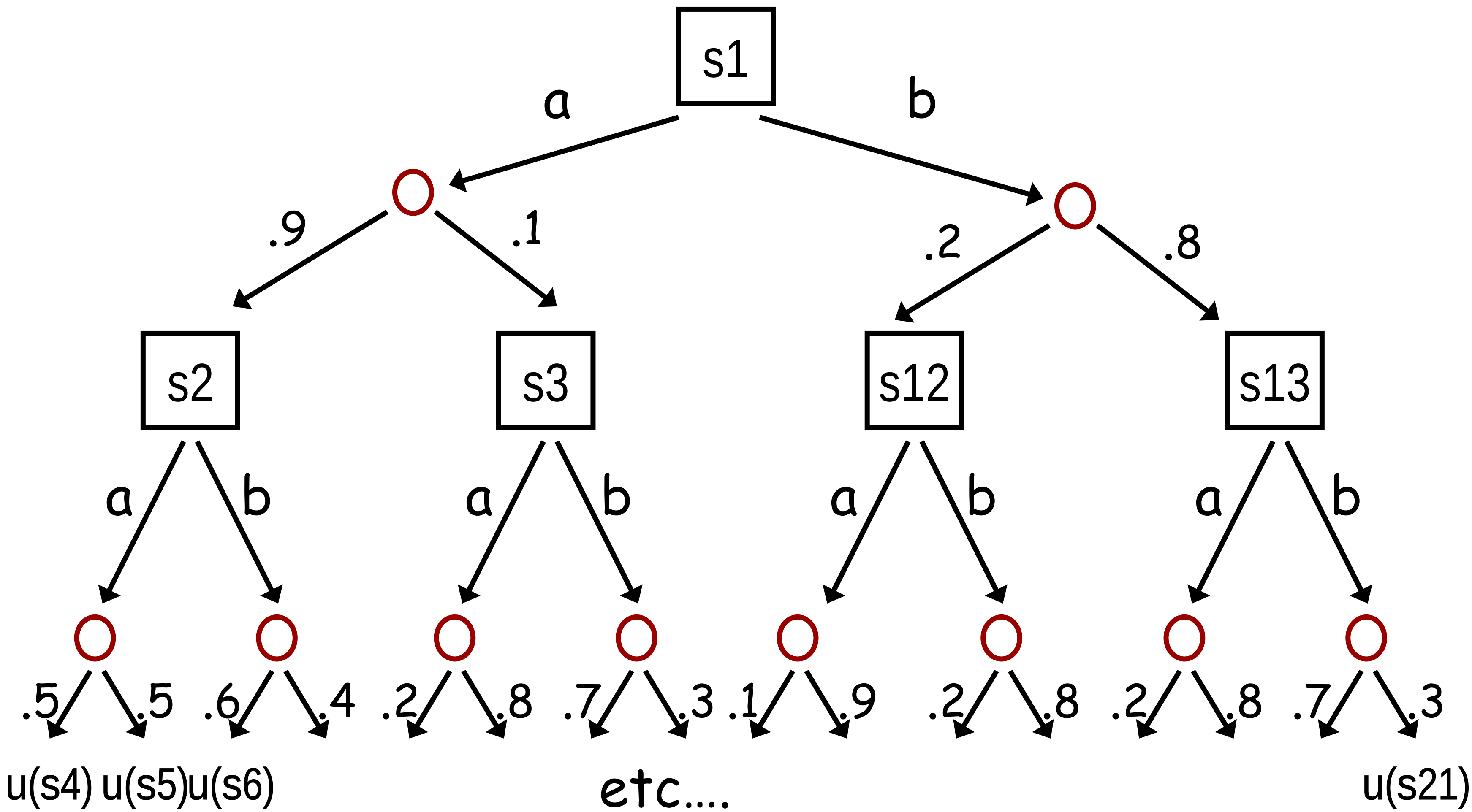
- Sequence [a,a] gives distribution over "final states"
  - Pr(s4) = .45, Pr(s5) = .45, Pr(s8) = .02, Pr(s9) = .08
- Similarly:
  - [a,b]: Pr(s6) = .54, Pr(s7) = .36, Pr(s10) = .07, Pr(s11) = .03
  - and similar distributions for sequences [b,a] and [b,b]

How Good is a Sequence?

- We associate *utilities with the "final" outcomes*
  - how good is it to end up at s4, s5, s6, …
  - note: we could assign utilities to the intermediate states s2, s3, s12, and s13 also. We ignore this for now. Technically, think if utility u(s4) as utility of entire *trajectory* or sequence of states we pass through.

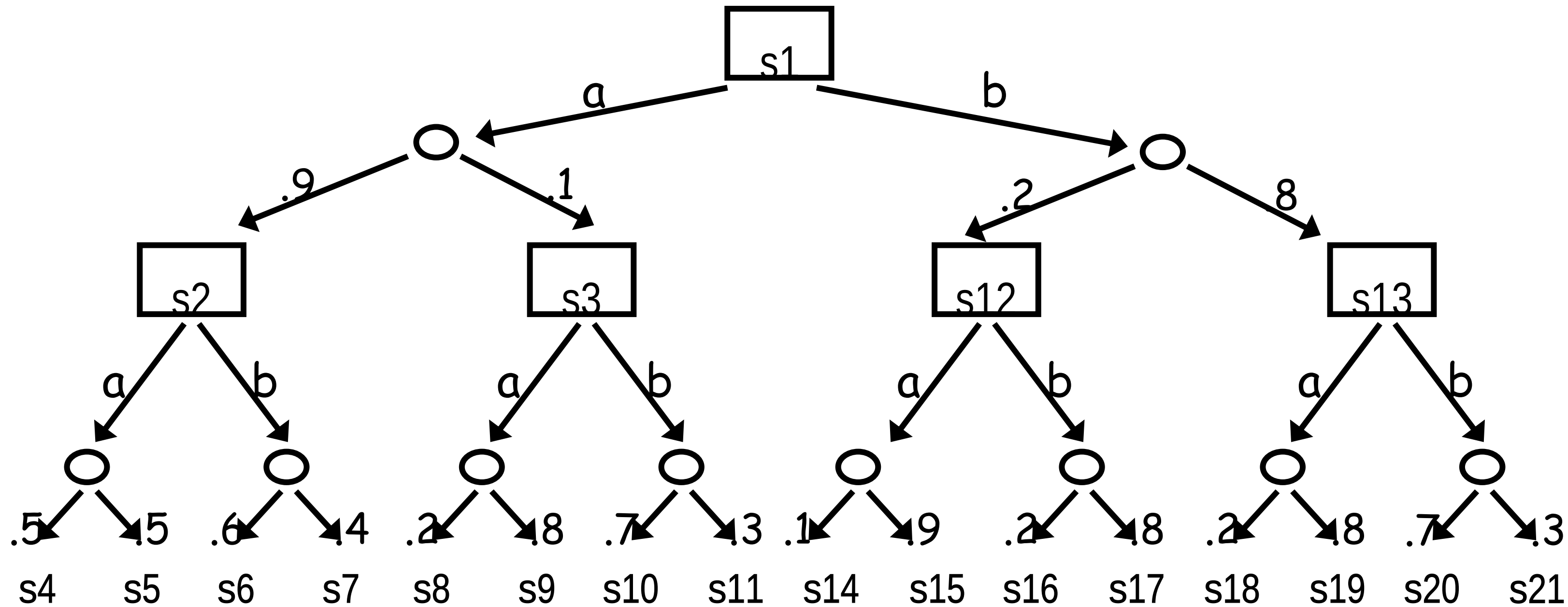
- Now we have:
  - EU(aa) = .45u(s4) + .45u(s5) + .02u(s8) + .08u(s9)
  - EU(ab) = .54u(s6) + .36u(s7) + .07u(s10) + .03u(s11)
  - etc…

Utilities for Action Sequences



Looks a lot like a game tree, but with chance nodes instead of min nodes. (We average instead of minimizing)

Why *Sequences* might be bad



- Suppose we do *a* first; we could reach s2 or s3:
  - At s2, assume: EU(a) = .5u(s4) + .5u(s5) > EU(b) = .6u(s6) + .4u(s7)
  - At s3: EU(a) = .2u(s8) + .8u(s9) < EU(b) = .7u(s10) + .3u(s11)

- After doing *a* first, we want to do *a* next *if we reach s2*, but we want to do *b* second *if we reach s3*

Policies

- This suggests that we want to consider *policies*, **not** sequences of actions (plans)
- We have eight policies for this decision tree:

  [a; if s2 a, if s3 a] [b; if s12 a, if s13 a]
  [a; if s2 a, if s3 b] [b; if s12 a, if s13 b]
  [a; if s2 b, if s3 a] [b; if s12 b, if s13 a]
  [a; if s2 b, if s3 b] [b; if s12 b, if s13 b]

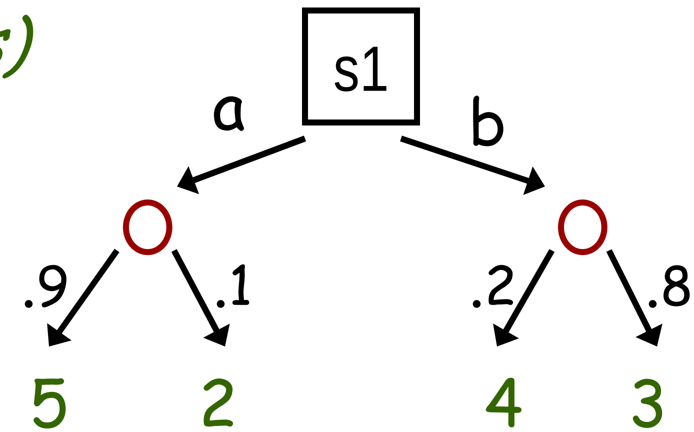
- Contrast this with four "plans"
  - [a; a], [a; b], [b; a], [b; b]
  - note: each plan corresponds to a policy, so we can only *gain* by allowing decision maker to use policies

Evaluating Policies

- Number of plans (sequences) of length *k*
  - exponential in *k*: $|A|^k$ if *A* is our action set
- Number of policies is even much larger
  - if we have $n=|A|$ actions and $m=|O|$ outcomes per action, then we have $(nm)^k$ policies
- Fortunately, *dynamic programming* can be used
  - e.g., suppose EU(a) > EU(b) at *s2*
  - never consider a policy that does anything else at s2
- How to do this?
  - back values up the tree much like minimax search

Decision Trees

- Squares denote *choice* nodes
  - these denote action choices by decision maker *(decision nodes)*
- Circles denote *chance* nodes
  - these denote uncertainty regarding action effects
  - "nature" will choose the child with specified probability
- Terminal nodes labeled with *utilities*
  - denote utility of "trajectory" (branch) to decision maker



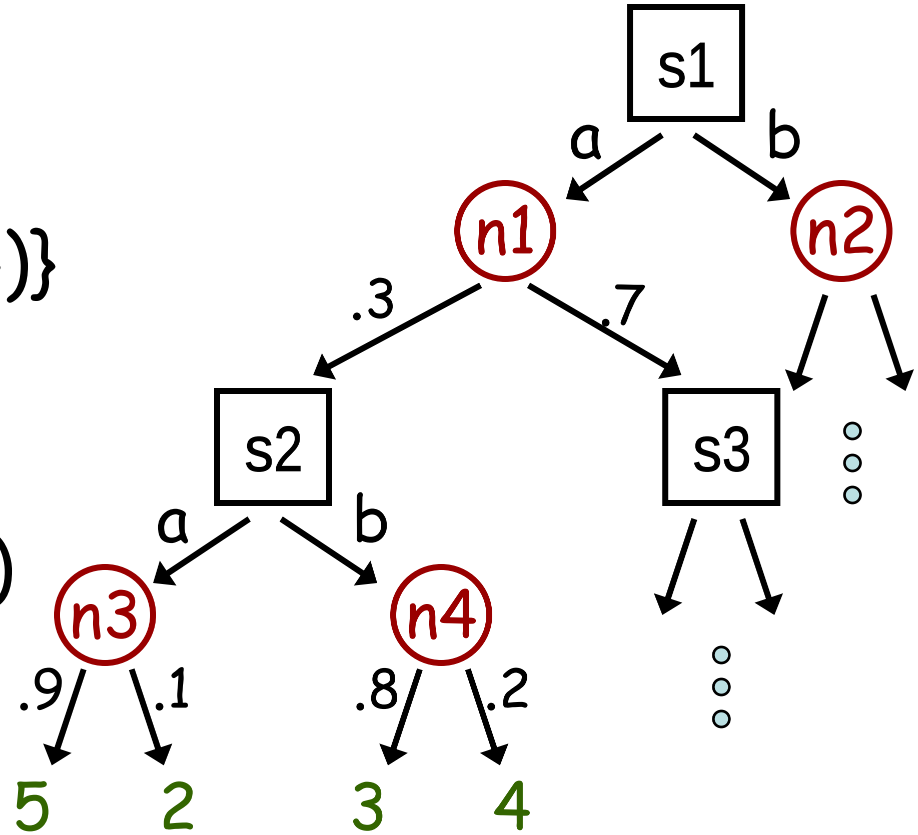
Evaluating Decision Trees

- Procedure is exactly like game trees, except…
  - key difference: the "opponent" is "nature" who simply chooses outcomes at chance nodes with specified probability: so we average instead of minimizing
- Back values *up* the tree
  - *U(t)* is defined for all terminals (part of input)
  - *U(n)* = avg {*U(c)* : *c* a child of *n*} if *n* is a chance node
  - *U(n)* = max {*U(c)* : *c* a child of *n*} if *n* is a choice node
- At any choice node (state), the decision maker chooses action that leads to *highest utility child*

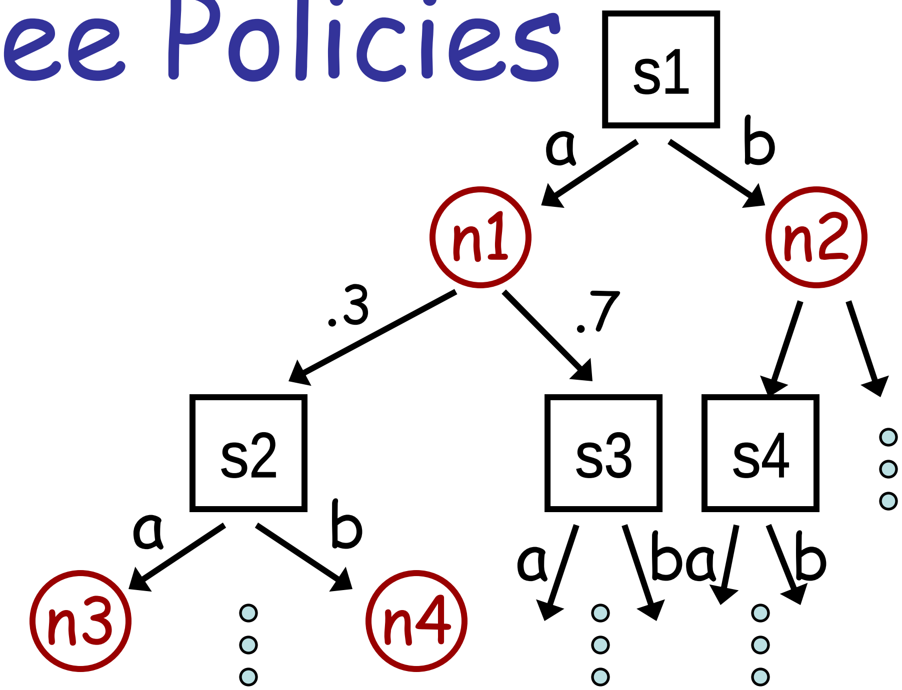
Evaluating a Decision Tree

- $U(n3) = .9*5 + .1*2$
- $U(n4) = .8*3 + .2*4$
- $U(s2) = \max\{U(n3), U(n4)\}$
  - decision a or b (whichever is max)
- $U(n1) = .3U(s2) + .7U(s3)$
- $U(s1) =$

  $\max\{U(n1), U(n2)\}$
  - decision: max of a, b



Decision Tree Policies



- Note that we don't just compute values, but policies for the tree
- A ***policy*** assigns a decision to each choice node in tree
- Some policies can't be distinguished in terms of there expected values
  - e.g., if policy chooses a at node s1, choice at s4 doesn't matter because it won't be reached
  - Two policies are *implementationally indistinguishable* if they disagree only at unreachable decision nodes
    - reachability is determined by policy themselves

Computational Issues

- Savings compared to explicit policy evaluation is substantial
- Evaluate only $O((nm)^d)$ nodes in tree of depth d
  - total computational cost is thus $O((nm)^d)$
- Note that there are also $(nm)^d$ ***policies*** and
  - evaluating a single policy explicitly requires substantial computation: $O(m^d)$
  - total computation for explicitly evaluating each policy would be $O(n^d m^{2d})$!!!
- Tremendous value to dynamic programming solution

Computational Issues

- **Tree size:** grows exponentially with depth
- Possible solutions:
  - bounded lookahead with heuristics (like game trees)
  - heuristic search procedures (like A*)
- **Full observability**: we must know the initial state and outcome of each action
- Possible solutions:
  - handcrafted decision trees for certain initial state uncertainty
  - more general policies based on *observations*

Other Issues

- **Specification**: suppose each state is an assignment to variables; then representing action probability distributions is complex (and branching factor could be immense)
- Possible solutions:
  - represent distribution using Bayes nets
  - solve problems using *decision networks* (or influence diagrams)

Next Class

- Decision networks
- Russell and Norvig Chapter 16