

Module 3

Utility Theory

CS 886 Sequential Decision Making
and Reinforcement Learning
University of Waterloo

Decision Making under Uncertainty

- I give planning problem to robot: I want coffee
 - but coffee maker is broken: robot reports "No plan!"
- For more robust behaviour, I should provide 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 $\langle \text{beverage}, \text{time} \rangle$ pairs

Preferences

- A *preference ordering* \succsim 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.
 - $s \succsim t$: means that state s is *at least as good as* t
 - $s \succ t$: means that state s is *strictly preferred to* t
 - $s \sim t$: means that the agent is *indifferent* between states s and t

Lotteries

- 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 $L=[p_1, s_1; p_2, s_2; \dots; p_n, s_n]$
 - s_1 occurs with prob p_1 , s_2 occurs with prob p_2, \dots

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:**

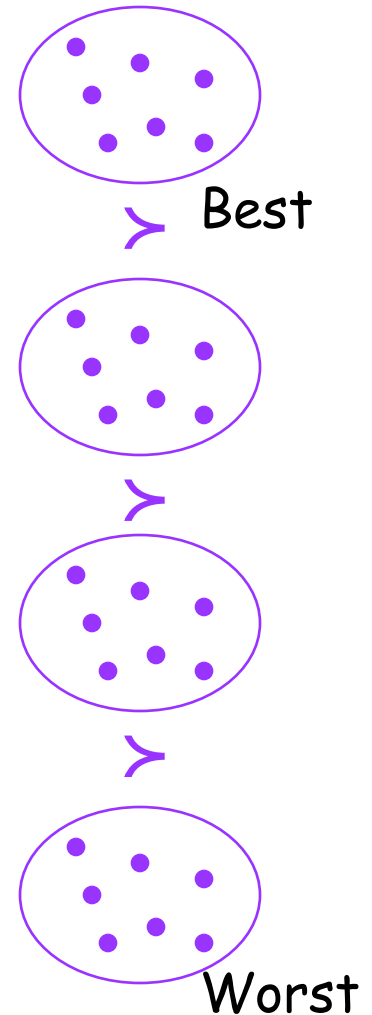
$$A \succ B \Rightarrow (p \geq q \Leftrightarrow [p, A; 1-p, B] \succcurlyeq [q, A; 1-q, B])$$

- **Decomposability:**

$$[p, A; 1-p, [q, B; 1-q, C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C]$$

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 \succsim over S
- A *solution* to a decision problem is any $d^* \in D$ such that $f(d^*) \succsim f(d)$ for all $d \in D$

Decision Making under Uncertainty



- Suppose actions don't have deterministic outcomes
 - e.g., when robot pours coffee, it spills 20% of time (mess)
 - preferences: $c, \sim\text{mess} > \sim c, \sim\text{mess} > \sim c, \text{mess}$
- What should robot do?
 - *getcoffee* leads to good/bad outcome with some probability
 - *donothing* leads to medium outcome for sure
- Should robot be optimistic? pessimistic?
- Odds of success should influence decision
 - but how?

Utilities

- Instead of ranking outcomes, quantify degrees of preference
 - e.g., how much more important is c than \sim mess
- A *utility function* $U:S \rightarrow \mathbb{R}$ associates a real-valued *utility* with each outcome.
 - $U(s)$ measures *degree* of preference for s
- Note: U induces a preference ordering \succsim_U over S defined as: $s \succsim_U t$ iff $U(s) \geq U(t)$
 - obviously \succsim_U is reflexive and transitive

Expected Utility

- Under uncertainty, each decision d induces a distribution Pr_d over possible outcomes
 - $\text{Pr}_d(s)$ is probability of outcome s under decision d
- The *expected utility* of decision d is defined

$$EU(d) = \sum_{s \in \mathcal{S}} \text{Pr}_d(s) U(s)$$

Expected Utility



When robot pours coffee, it spills 20% of time (mess)

If $U(c, \sim ms) = 10$, $U(\sim c, \sim ms) = 5$, $U(\sim c, ms) = 0$,
then $EU(\text{getcoffee}) = (0.8)(10) + (0.2)(0) = 8$
and $EU(\text{donothing}) = 5$

If $U(c, \sim ms) = 10$, $U(\sim c, \sim ms) = 9$, $U(\sim c, ms) = 0$,
then $EU(\text{getcoffee}) = (0.8)(10) + (0.2)(0) = 8$
and $EU(\text{donothing}) = 9$

The MEU Principle

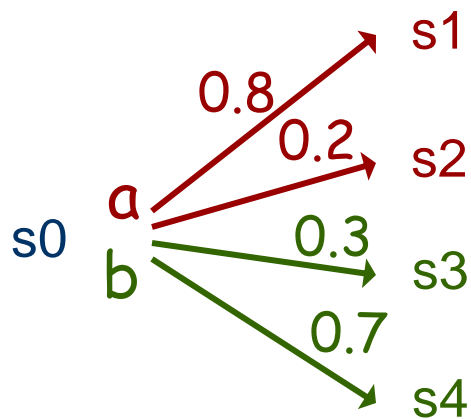
- The principle of *maximum expected utility (MEU)* states that the optimal decision under conditions of uncertainty is that with the greatest expected utility.
- In our example
 - if my utility function is the first one, my robot should get coffee
 - if your utility function is the second one, your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

Decision Problems: Uncertainty

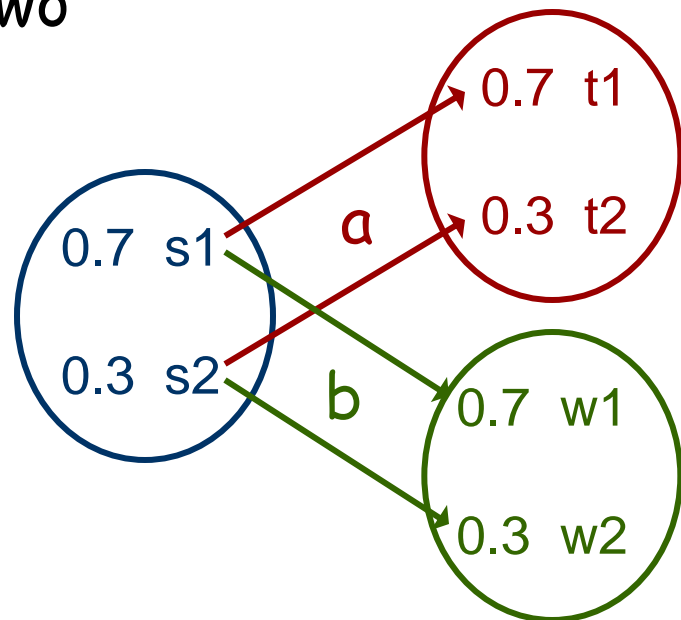
- A *decision problem under uncertainty* is:
 - a set of *decisions* D
 - a set of *outcomes* or states S
 - an *outcome function* $Pr : D \rightarrow \Delta(S)$
 - $\Delta(S)$ is the set of distributions over S (e.g., Pr_d)
 - a *utility function* U over S
- A *solution* is any $d^* \in D$ such that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

Expected Utility: Notes

- This viewpoint accounts for:
 - 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 need not 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 transformations*

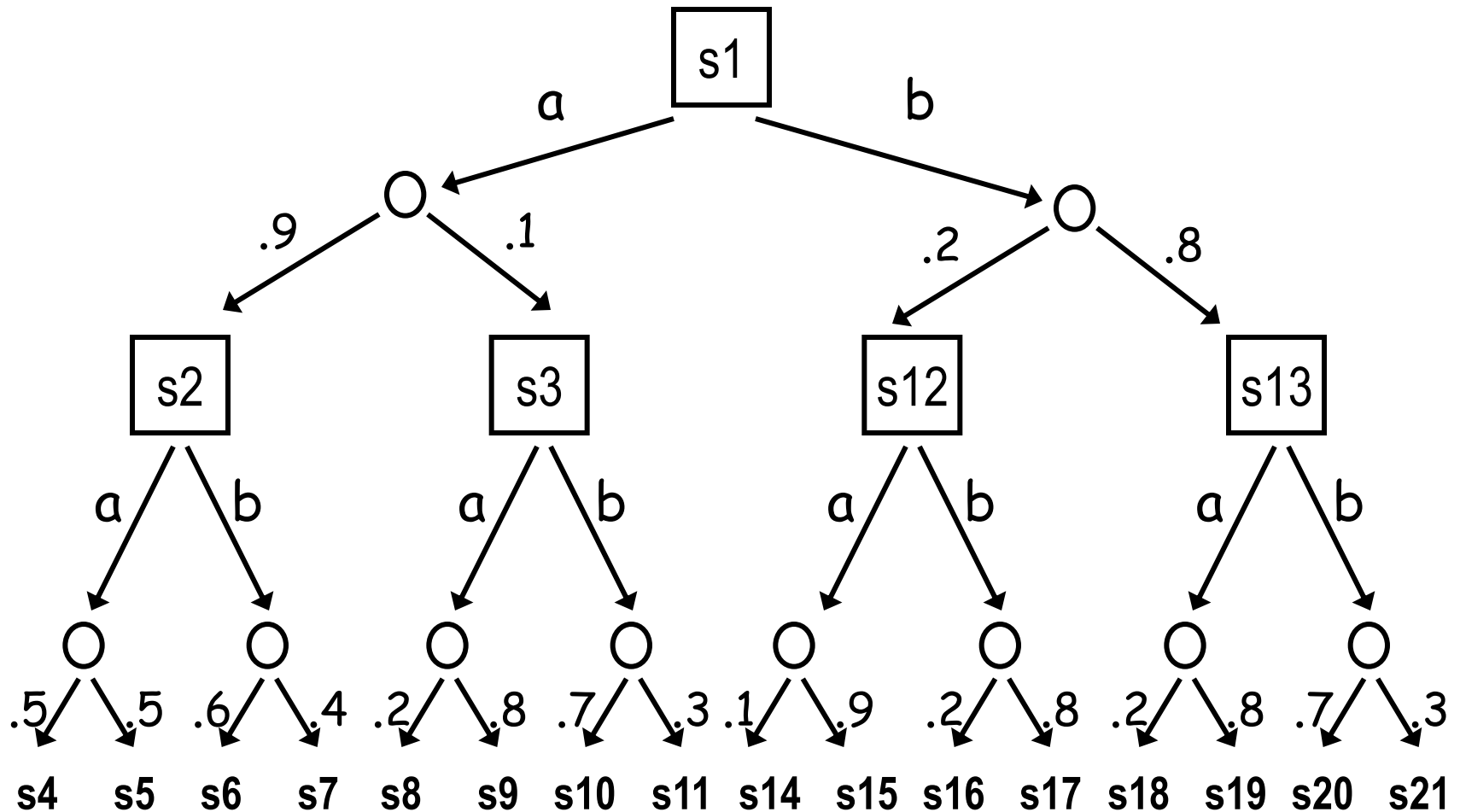
So What are the Complications?

- Outcome space is large
 - states spaces can be huge
 - don't want to spell out distributions like Pr_d explicitly
- Decision space is large
 - usually 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)

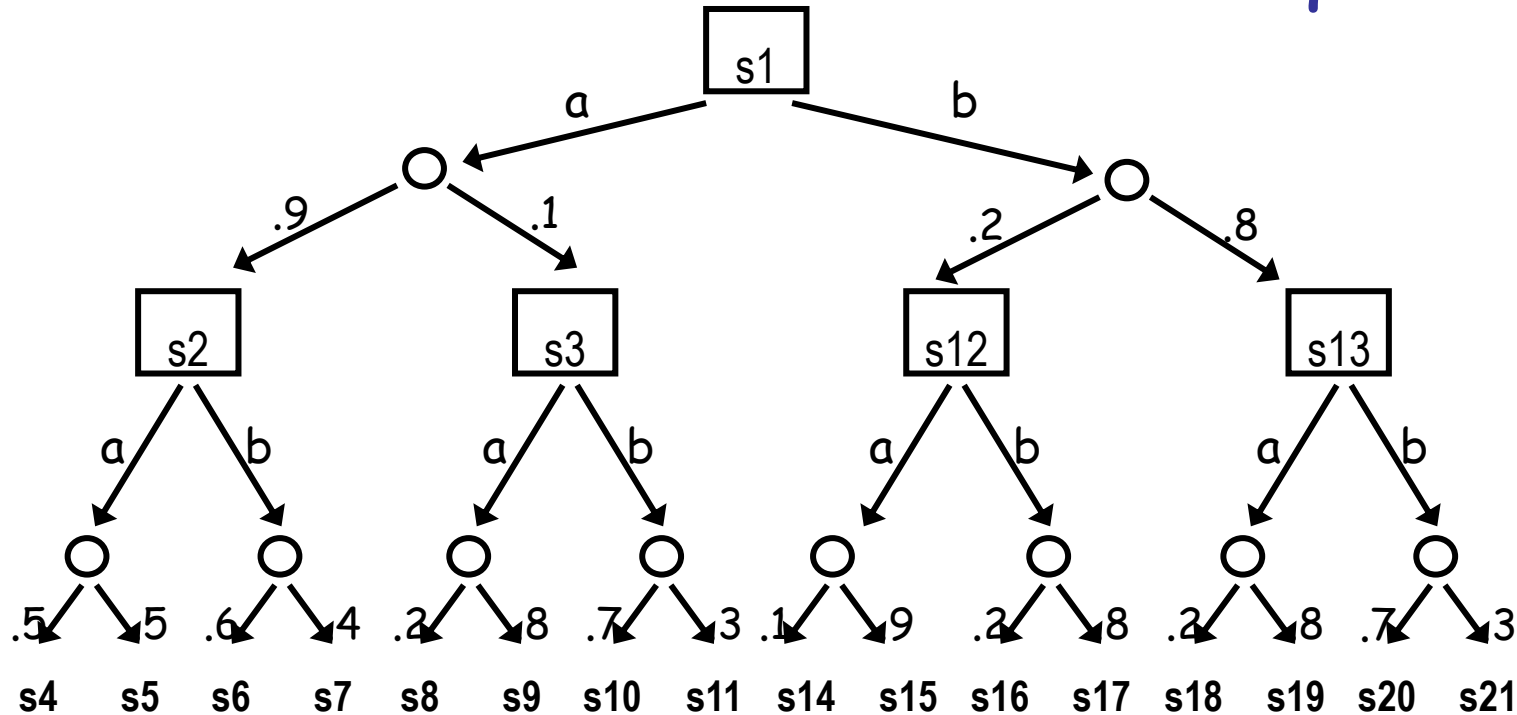
A Simple Example

- Suppose we have two actions: a, b
- We have time to execute *two* actions in sequence
 - [a,a], [a,b], [b,a], [b,b]
- Actions are stochastic: $\Pr_a(s_i | s_j)$
 - e.g., $\Pr_a(s_2 | s_1) = .9$ 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

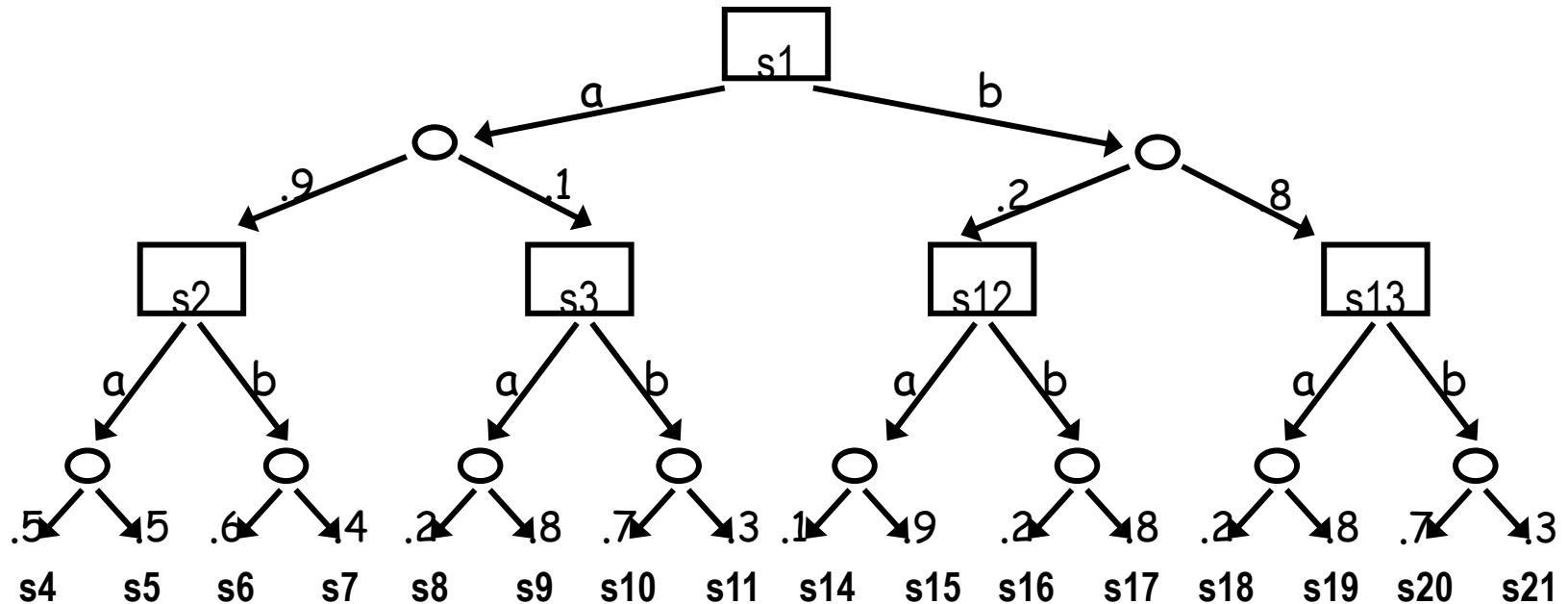


- Sequence $[a,a]$ gives distribution over "final states"
 - $\Pr(s_4) = .45, \Pr(s_5) = .45, \Pr(s_8) = .02, \Pr(s_9) = .08$
- Similarly:
 - $[a,b]: \Pr(s_6) = .54, \Pr(s_7) = .36, \Pr(s_{10}) = .07, \Pr(s_{11}) = .03$
 - and similar distributions for sequences $[b,a]$ and $[b,b]$

How Good is a Sequence?

- We associate *utilities with the "final" outcomes*
 - how good is it to end up at s_4, s_5, s_6, \dots
 - note: we could assign utilities to the intermediate states s_2, s_3, s_{12} , and s_{13} also. We ignore this for now. Technically, think of utility $u(s_4)$ as utility of entire *trajectory* or sequence of states we pass through.
- Now we have:
 - $EU(aa) = .45u(s_4) + .45u(s_5) + .02u(s_8) + .08u(s_9)$
 - $EU(ab) = .54u(s_6) + .36u(s_7) + .07u(s_{10}) + .03u(s_{11})$
 - etc...

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

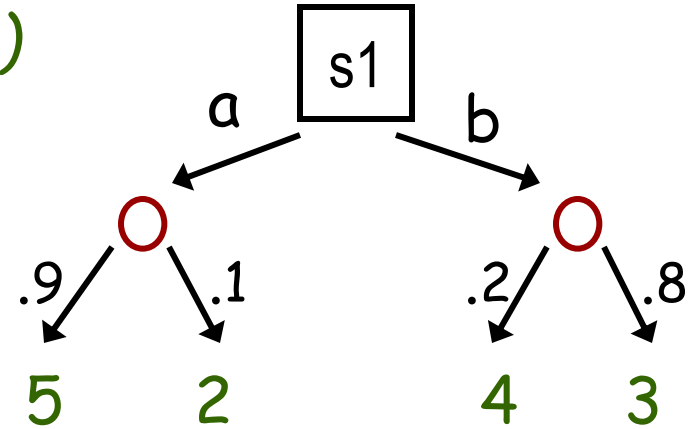
- This suggests that we want to consider *policies*, **not** sequences of actions (plans)
- We have eight policies for this decision tree:
 - [a; if s2 a, if s3 a] [b; if s12 a, if s13 a]
 - [a; if s2 a, if s3 b] [b; if s12 a, if s13 b]
 - [a; if s2 b, if s3 a] [b; if s12 b, if s13 a]
 - [a; if s2 b, if s3 b] [b; if s12 b, if s13 b]
- Contrast this with four “plans”
 - [a; a], [a; b], [b; a], [b; b]
 - note: 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 s_2
 - never consider a policy that does anything else at s_2
- How to do this?
 - back values up the tree

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

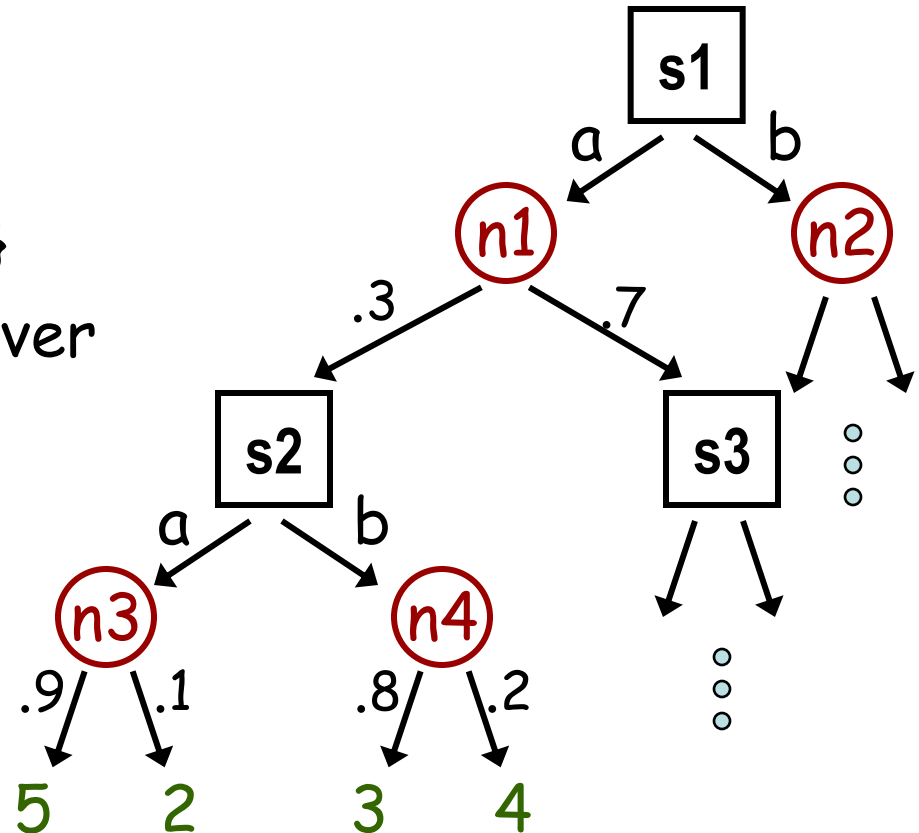


Evaluating Decision Trees

- Back values *up* the tree
 - $U(t)$ is defined for all terminals (part of input)
 - $U(n) = \text{expectation } \{U(c) : c \text{ a child of } n\}$ if n is a chance node
 - $U(n) = \max \{U(c) : c \text{ a child of } n\}$ if n is a choice node
- At any choice node (state), the decision maker chooses action that leads to *highest utility child*

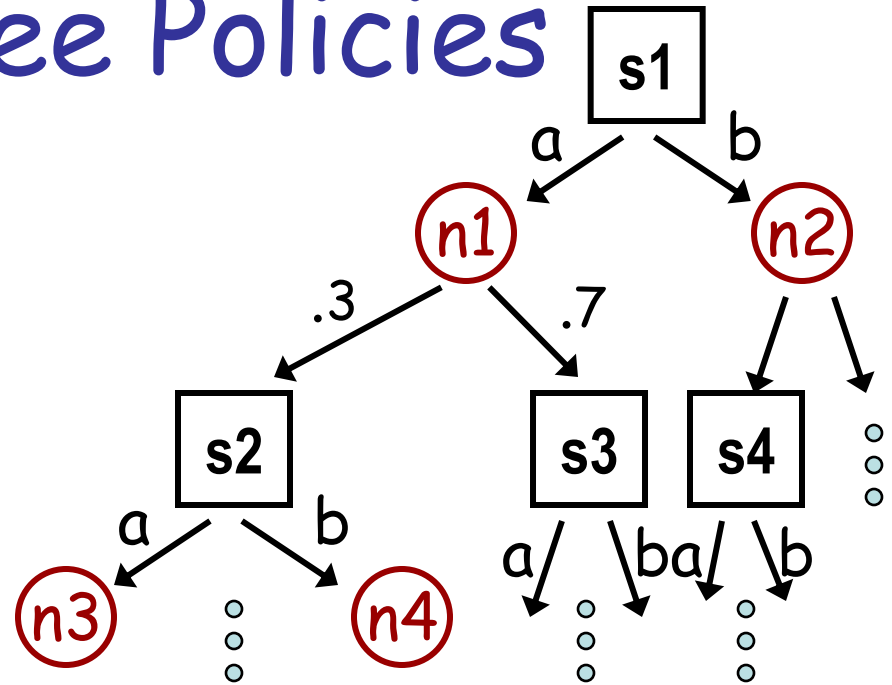
Evaluating a Decision Tree

- $U(n3) = .9*5 + .1*2$
- $U(n4) = .8*3 + .2*4$
- $U(s2) = \max\{U(n3), U(n4)\}$
 - 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

- A **policy** assigns a decision to each choice node in tree



- Some policies can't be distinguished in terms of their 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

- Evaluate $O((nm)^d)$ nodes in tree of depth d
 - total computational cost is thus $O((nm)^d)$
- Note that there are $(nm)^d$ **policies** and
 - evaluating a single policy explicitly requires substantial computation: $O(m^d)$
 - total computation for explicitly evaluating each policy would be $O(ndm^{2d})$!!!
- Dynamic programming saves computation, but takes exponential time still

Possible Solutions

- Reduce computational complexity by
 - Detecting repeated states
 - Prune by branch-and-bound
 - Approximate expectations by sampling