

Lecture 4

Jan 19, 2010

CS 886

Outline

- Decision making
 - Utility Theory
 - Decision Networks
- 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.

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

Preferences

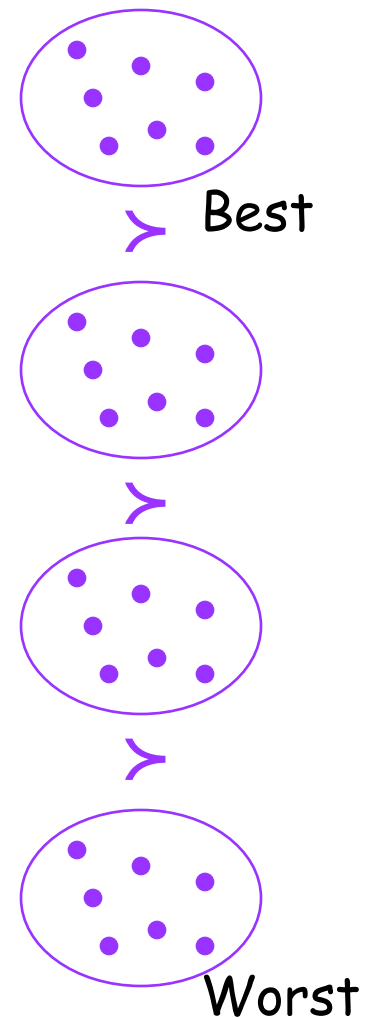
- If an agent's actions are deterministic then we know what states will occur
- If an agent's actions are not deterministic then we represent this by lotteries
 - Probability distribution over outcomes
 - Lottery $L = [p_1, s_1; p_2, s_2; \dots; p_n, s_n]$
 - s_1 occurs with prob p_1 , s_2 occurs with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

Preference 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 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: $c, \sim\text{mess} > \sim c, \sim\text{mess} > \sim c, \text{mess}$
- 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 \sim 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 \succsim_U over S defined as: $s \succsim_U t$ iff $U(s) \geq U(t)$
 - obviously \succsim_U will be reflexive, transitive, connected

Expected Utility

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

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

- The *expected utility* of decision d is defined

Expected Utility



When robot pours coffee, it spills 20% of time, making a 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* to a decision problem under uncertainty is any $d^* \in D$ such that $EU(d^*) \geq EU(d)$ for all $d \in D$
- Again, for single-shot problems, this is trivial

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*

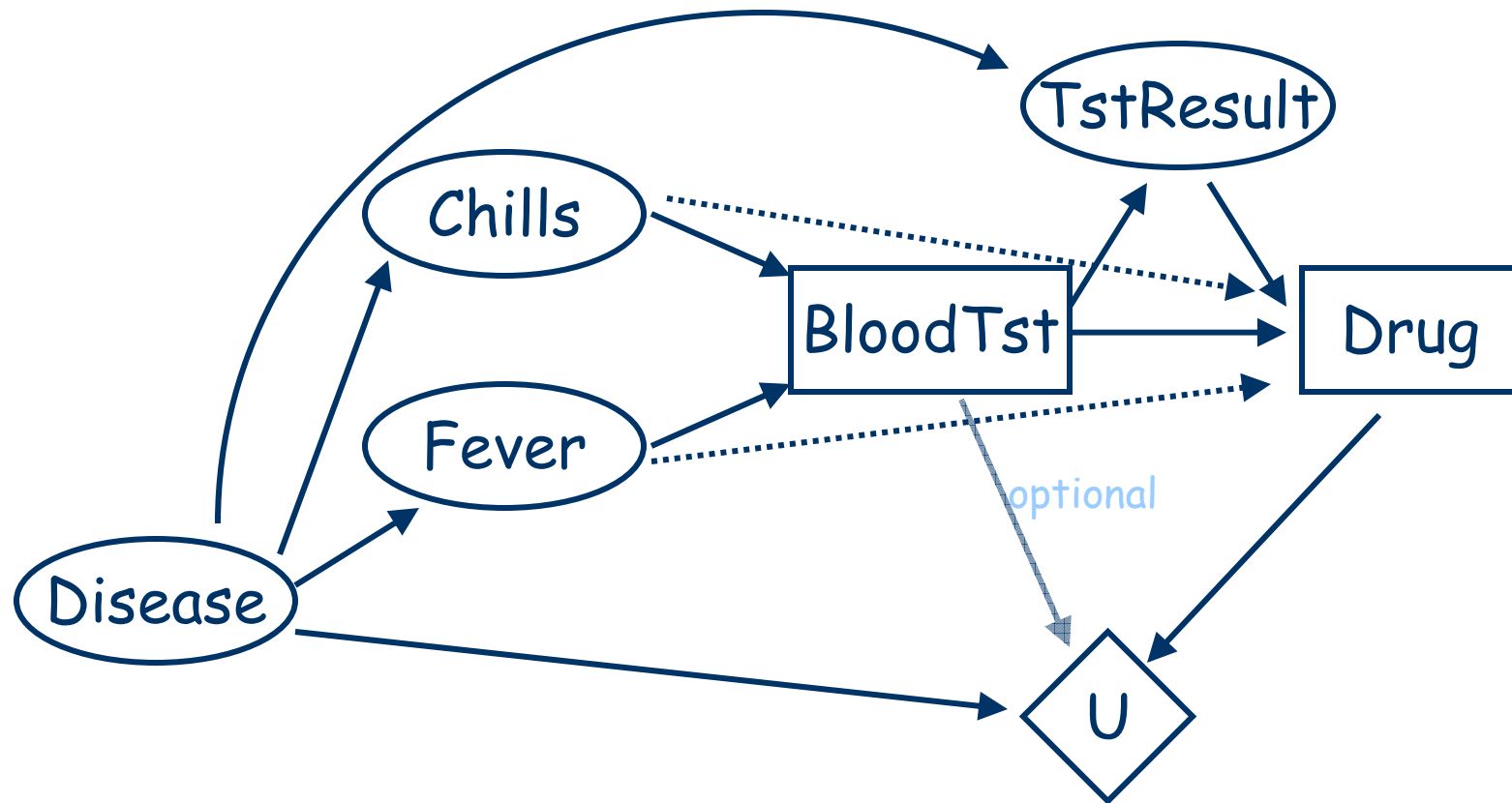
So What are the Complications?

- 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)

Decision Networks

- *Decision networks* (also known as *influence diagrams*) provide a way of representing sequential decision problems
 - basic idea: represent the variables in the problem as you would in a BN
 - add decision variables - variables that you "control"
 - add utility variables - how good different states are

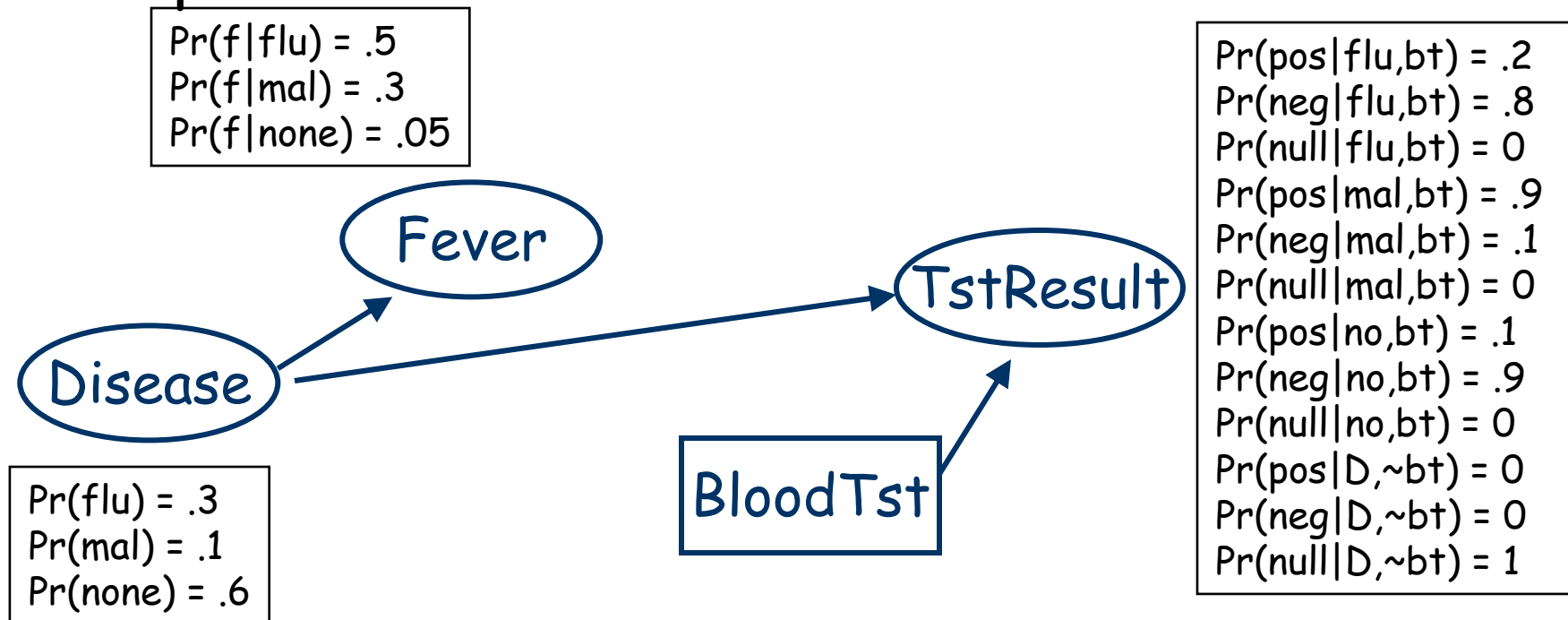
Sample Decision Network



Decision Networks: Chance Nodes

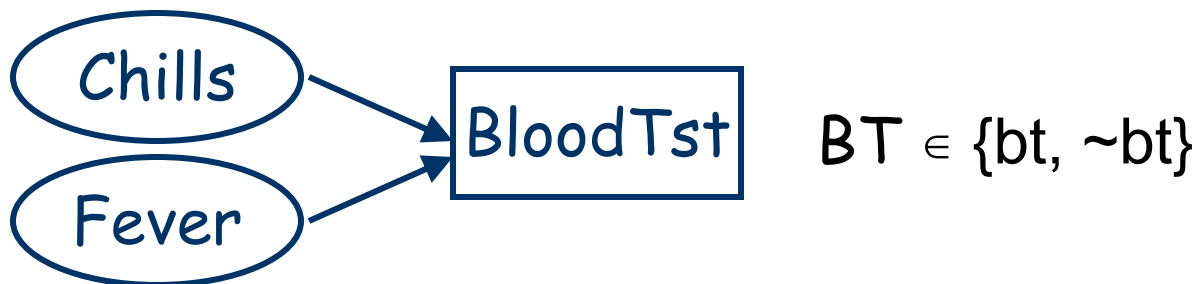
- **Chance nodes**

- random variables, denoted by circles
- as in a BN, probabilistic dependence on parents



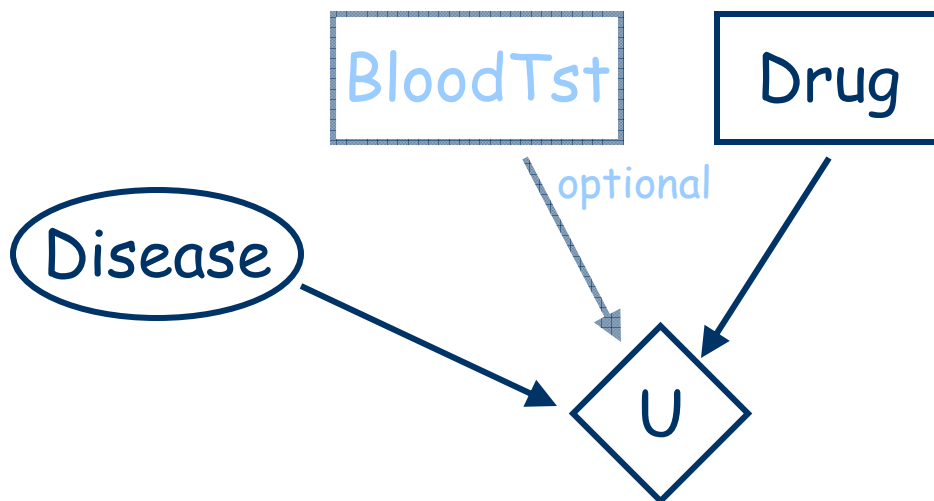
Decision Networks: Decision Nodes

- **Decision nodes**
 - variables decision maker sets, denoted by squares
 - parents reflect *information available* at time decision is to be made
- In example decision node: the actual values of Ch and Fev will be observed before the decision to take test must be made
 - agent can make *different decisions* for each instantiation of parents (i.e., policies)



Decision Networks: Value Node

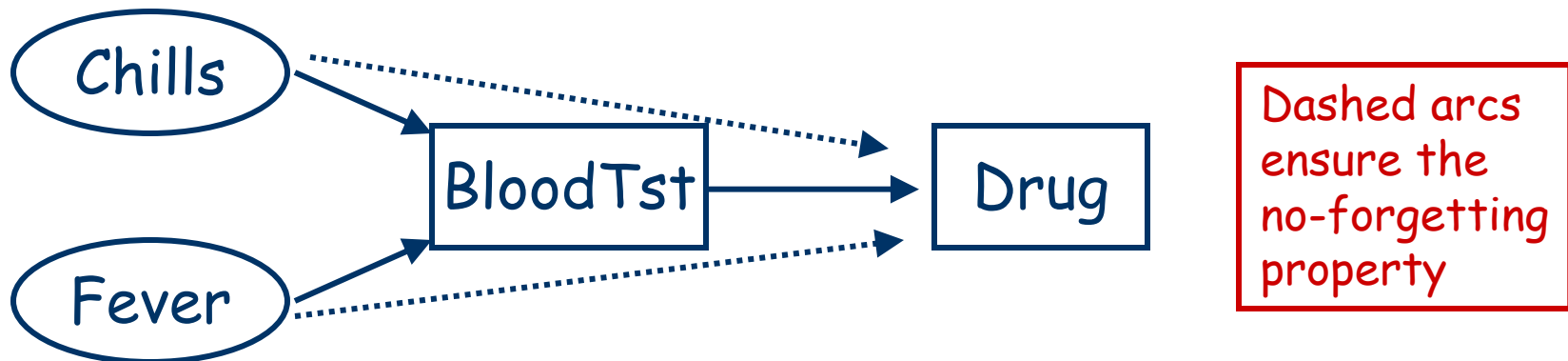
- **Value node**
 - specifies utility of a state, denoted by a diamond
 - utility depends *only on state of parents* of value node
 - generally: only one value node in a decision network
- Utility depends only on disease and drug



$U(\text{fludrug}, \text{flu}) = 20$
$U(\text{fludrug}, \text{mal}) = -300$
$U(\text{fludrug}, \text{none}) = -5$
$U(\text{maldrug}, \text{flu}) = -30$
$U(\text{maldrug}, \text{mal}) = 10$
$U(\text{maldrug}, \text{none}) = -20$
$U(\text{no drug}, \text{flu}) = -10$
$U(\text{no drug}, \text{mal}) = -285$
$U(\text{no drug}, \text{none}) = 30$

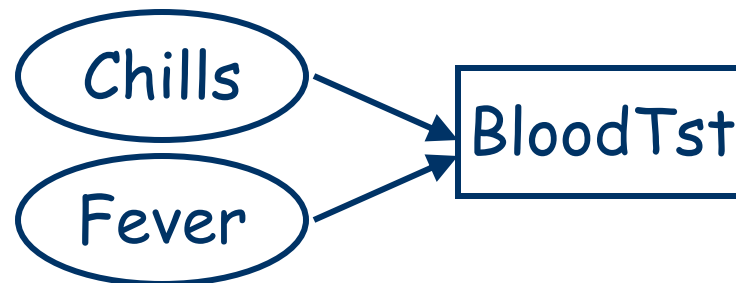
Decision Networks: Assumptions

- Decision nodes are totally ordered
 - decision variables D_1, D_2, \dots, D_n
 - decisions are made in sequence
 - e.g., BloodTst (yes,no) decided before Drug (fd,md,no)
- *No-forgetting property*
 - any information available when decision D_i is made is available when decision D_j is made (for $i < j$)
 - thus all parents of D_i are parents of D_j



Policies

- Let $Par(D_i)$ be the parents of decision node D_i
 - $Dom(Par(D_i))$ is the set of assignments to parents
- A policy δ is a set of mappings δ_i , one for each decision node D_i
 - $\delta_i : Dom(Par(D_i)) \rightarrow Dom(D_i)$
 - δ_i associates a decision with each parent asst for D_i
- For example, a policy for BT might be:
 - $\delta_{BT}(c, f) = bt$
 - $\delta_{BT}(c, \sim f) = \sim bt$
 - $\delta_{BT}(\sim c, f) = bt$
 - $\delta_{BT}(\sim c, \sim f) = \sim bt$



Value of a Policy

- *Value of a policy* δ is the expected utility given that decision nodes are executed according to δ
- Given asst \mathbf{x} to the set \mathbf{X} of all chance variables, let $\delta(\mathbf{x})$ denote the asst to decision variables dictated by δ
 - e.g., asst to D_1 determined by it's parents' asst in \mathbf{x}
 - e.g., asst to D_2 determined by it's parents' asst in \mathbf{x} along with whatever was assigned to D_1
 - etc.
- Value of δ :

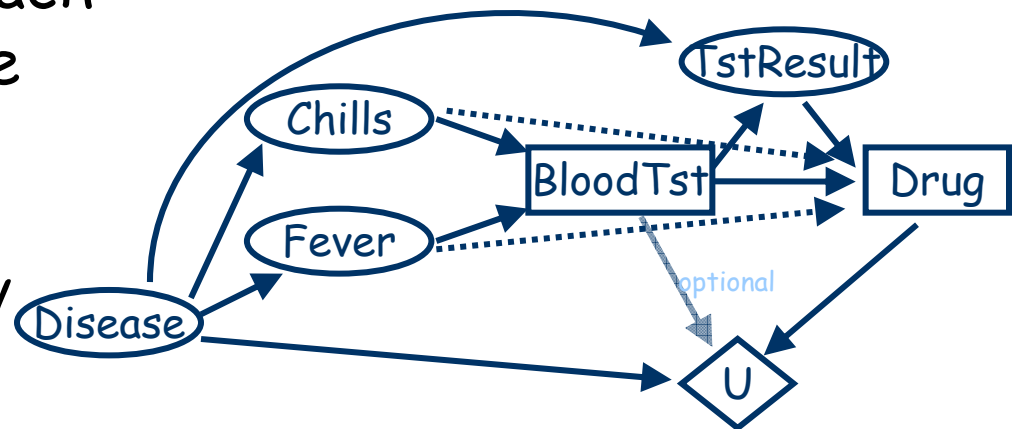
$$EU(\delta) = \sum_{\mathbf{x}} P(\mathbf{X}, \delta(\mathbf{X})) U(\mathbf{X}, \delta(\mathbf{X}))$$

Optimal Policies

- An *optimal policy* is a policy δ^* such that $EU(\delta^*) \geq EU(\delta)$ for all policies δ
- We can use the dynamic programming principle yet again to avoid enumerating all policies
- We can also use the structure of the decision network to use **variable elimination** to aid in the computation

Computing the Best Policy

- We can work backwards as follows
- First compute optimal policy for Drug (last dec'n)
 - for each asst to parents (C,F,BT,TR) and for each decision value (D = md,fd,none), *compute the expected value* of choosing that value of D
 - set policy choice for each value of parents to be the value of D that has max value
 - eg: $\delta_D(c,f,bt,pos) = md$

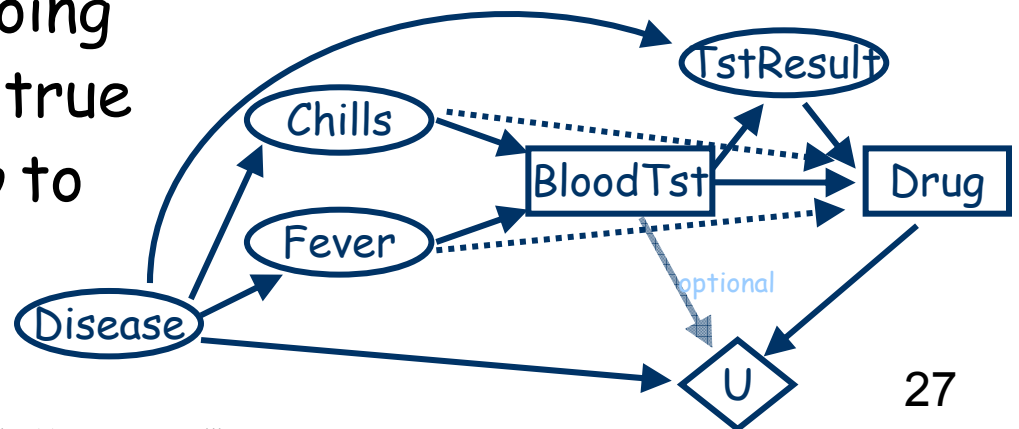


Computing the Best Policy

- Next compute policy for BT given policy $\delta_D(C, F, BT, TR)$ just determined for Drug
 - since $\delta_D(C, F, BT, TR)$ is fixed, we can treat Drug as a normal random variable with deterministic probabilities
 - i.e., for any instantiation of parents, value of Drug is fixed by policy δ_D
 - this means we can solve for optimal policy for BT just as before
 - only uninstantiated vars are random vars (once we fix *its* parents)

Computing the Best Policy

- How do we compute these expected values?
 - suppose we have asst $\langle c, f, bt, pos \rangle$ to parents of *Drug*
 - we want to compute EU of deciding to set $Drug = md$
 - we can run **variable elimination!**
- Treat C, F, BT, TR, Dr as evidence
 - this reduces factors (e.g., U restricted to bt, md : depends on *Dis*)
 - eliminate remaining variables (e.g., only *Disease* left)
 - left with factor: **$EU(md|c, f, bt, pos) = \sum_{Dis} P(Dis|c, f, bt, pos, md) U(Dis, bt, md)$**
- We now know EU of doing $Dr=md$ when c, f, bt, pos true
- Can do same for fd, no to decide which is best

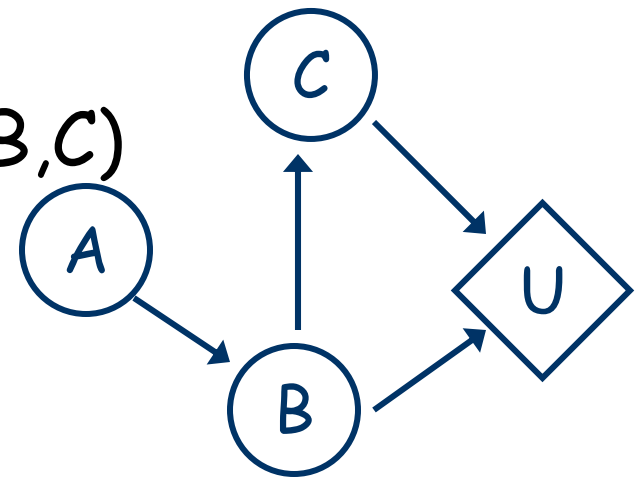


Computing Expected Utilities

- The previous example illustrates a general phenomenon
 - computing expected utilities with BNs is quite easy
 - utility nodes are just factors that can be dealt with using variable elimination

$$\begin{aligned} EU &= \sum_{A,B,C} P(A,B,C) U(B,C) \\ &= \sum_{A,B,C} P(C|B) P(B|A) P(A) U(B,C) \end{aligned}$$

- Just eliminate variables in the usual way



Optimizing Policies: Key Points

- If a decision node D has no decisions that follow it, we can find its policy by instantiating each of its parents and computing the expected utility of each decision for each parent instantiation
 - no-forgetting means that all other decisions are instantiated (they must be parents)
 - its easy to compute the expected utility using VE
 - the number of computations is quite large: we run expected utility calculations (VE) for each parent instantiation together with each possible decision D might allow
 - policy: choose max decision for each parent instant'n

Optimizing Policies: Key Points

- When a decision D node is optimized, it can be treated as a random variable
 - for each instantiation of its parents we now know what value the decision should take
 - just treat policy as a new CPT: for a given parent instantiation \mathbf{x} , D gets $\delta(\mathbf{x})$ with probability 1 (all other decisions get probability zero)
- If we optimize from last decision to first, at each point we can optimize a specific decision by (a bunch of) simple VE calculations
 - it's successor decisions (optimized) are just normal nodes in the BNs (with CPTs)

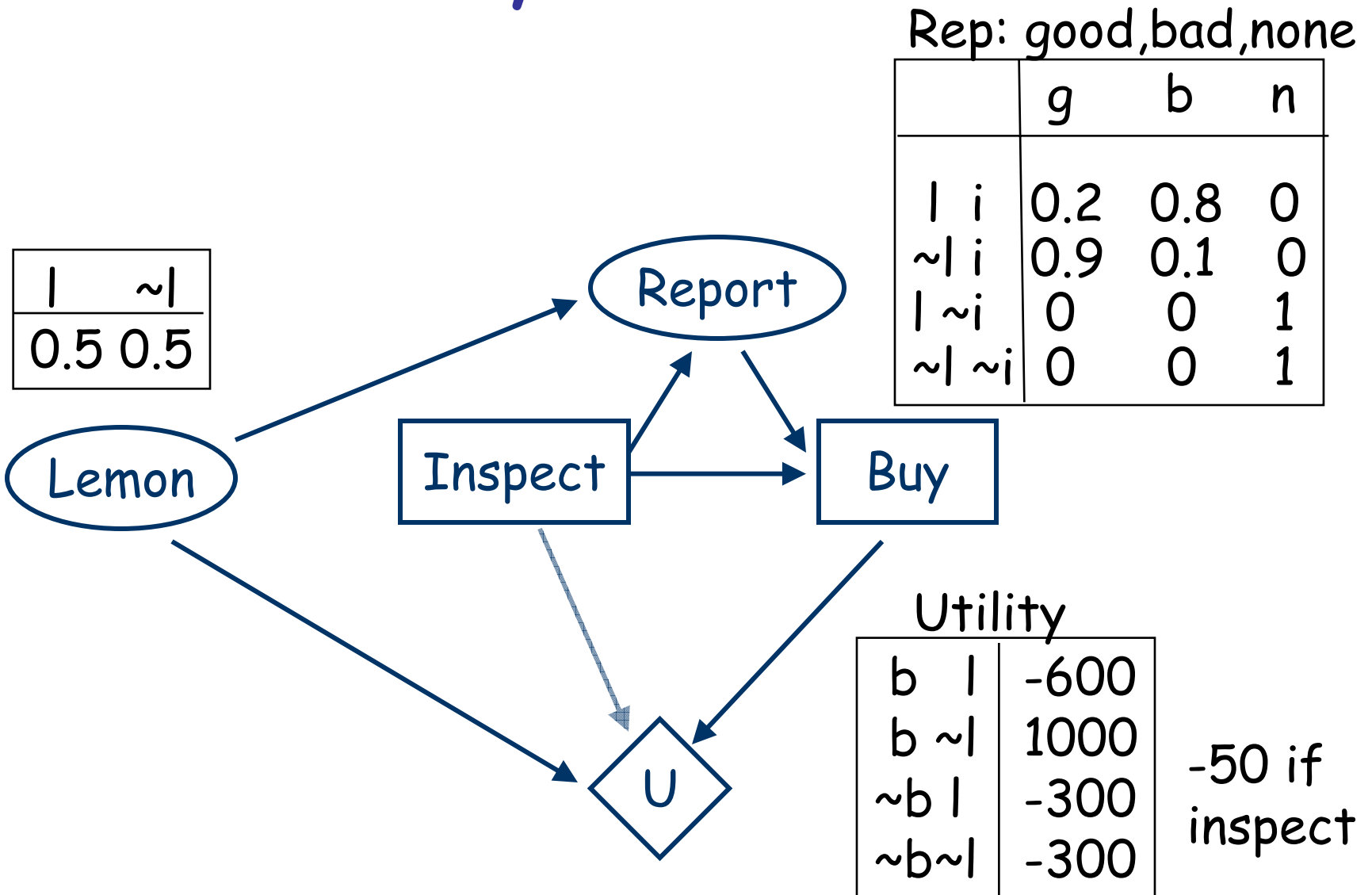
Decision Network Notes

- Decision networks commonly used by decision analysts to help structure decision problems
- Much work put into computationally effective techniques to solve these
 - common trick: replace the decision nodes with random variables at outset and solve a plain Bayes net (a subtle but useful transformation)
- Complexity much greater than BN inference
 - we need to solve a number of BN inference problems
 - one BN problem for each setting of decision node parents and decision node value

A Decision Net Example

- Setting: you want to buy a used car, but there's a good chance it is a "lemon" (i.e., prone to breakdown). Before deciding to buy it, you can take it to a mechanic for inspection. S/he will give you a report on the car, labelling it either "good" or "bad". A good report is positively correlated with the car being sound, while a bad report is positively correlated with the car being a lemon.
- The report costs \$50 however. So you could risk it, and buy the car without the report.
- Owning a sound car is better than having no car, which is better than owning a lemon.

Car Buyer's Network



Evaluate Last Decision: Buy (1)

- $EU(B|I,R) = \sum_L P(L|I,R,B) U(L,I,B)$
- $I = i, R = g$:
 - $EU(\text{buy}) = P(l|i,g,\text{buy}) U(l,i,\text{buy}) + P(\sim l|i,g,\text{buy}) U(\sim l,i,\text{buy})$
 $= .18 * -650 + .82 * 950 = 662$
 - $EU(\sim\text{buy}) = P(l|i,g,\sim\text{buy}) U(l,i,\sim\text{buy}) + P(\sim l|i,g,\sim\text{buy}) U(\sim l,i,\sim\text{buy})$
 $= -300 - 50 = -350$ (-300 indep. of lemon)
 - So optimal $\delta_{Buy}(i,g) = \text{buy}$

Evaluate Last Decision: Buy (2)

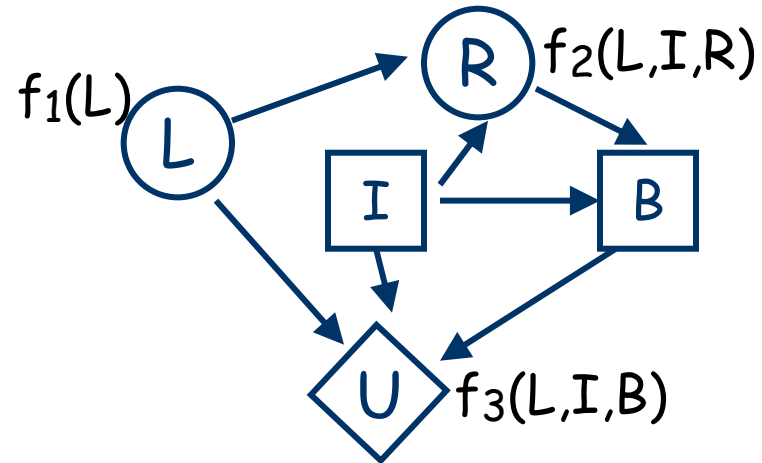
- $I = i, R = b$:
 - $EU(\text{buy}) = P(l|i,b,\text{buy}) U(l,i,\text{buy}) + P(\sim l|i,b,\text{buy}) U(\sim l,i,\text{buy})$
 $= .89 * -650 + .11 * 950 = -474$
 - $EU(\sim\text{buy}) = P(l|i,b,\sim\text{buy}) U(l,i,\sim\text{buy}) + P(\sim l|i,b,\sim\text{buy}) U(\sim l,i,\sim\text{buy})$
 $= -300 - 50 = -350$ (-300 indep. of lemon)
 - So optimal $\delta_{Buy}(i,b) = \sim\text{buy}$

Evaluate Last Decision: Buy (3)

- $I = \sim i, R = n$
 - $EU(\text{buy}) = P(l|\sim i, n, \text{buy}) U(l, \sim i, \text{buy}) + P(\sim l|\sim i, n, \text{buy}) U(\sim l, \sim i, \text{buy})$
 $= .5 * -600 + .5 * 1000 = 200$
 - $EU(\sim \text{buy}) = P(l|\sim i, n, \sim \text{buy}) U(l, \sim i, \sim \text{buy}) + P(\sim l|\sim i, n, \sim \text{buy}) U(\sim l, \sim i, \sim \text{buy})$
 $= -300$ (-300 indep. of lemon)
 - So optimal $\delta_{Buy}(\sim i, n) = \text{buy}$
- So optimal policy for Buy is:
 - $\delta_{Buy}(i, g) = \text{buy}$; $\delta_{Buy}(i, b) = \sim \text{buy}$; $\delta_{Buy}(\sim i, n) = \text{buy}$
- Note: we don't bother computing policy for $(i, \sim n)$, $(\sim i, g)$, or $(\sim i, b)$, since these occur with probability 0

Using Variable Elimination

Factors: $f_1(L)$ $f_2(L,I,R)$
 $f_3(L,I,B)$
Query: $EU(B)?$
Evidence: $I = i, R = g$
Elim. Order: L



Restriction: replace $f_2(L,I,R)$ by $f_4(L) = f_2(L,i,g)$
replace $f_3(L,I,B)$ by $f_5(L,B) = f_2(L,i,B)$

Step 1: Add $f_6(B) = \sum_L f_1(L) f_4(L) f_5(L,B)$
Remove: $f_1(L), f_4(L), f_5(L,B)$

Last factor: $f_6(B)$ is the unscaled expected utility of buy and ~buy. Select action with highest (unscaled) expected utility.

Repeat for $EU(B|i,b), EU(B|\sim i,n)$

Alternatively

- N.B.: variable elimination for decision networks computes expected utility that are **not scaled**...
- Can still pick best action, since utility scale is not important (relative magnitude is what matters)
- If we want exact expected utility:
 - Let $\mathbf{X} = \text{parents}(U)$
 - $\text{EU}(\text{dec}|\text{evidence}) = \sum_{\mathbf{X}} \text{Pr}(\mathbf{X}|\text{dec},\text{evidence}) U(\mathbf{X})$
 - Compute $\text{Pr}(\mathbf{X}|\text{dec},\text{evidence})$ by variable elimination
 - Multiply $\text{Pr}(\mathbf{X}|\text{dec},\text{evidence})$ by $U(\mathbf{X})$
 - Summout \mathbf{X}

Evaluate First Decision: Inspect

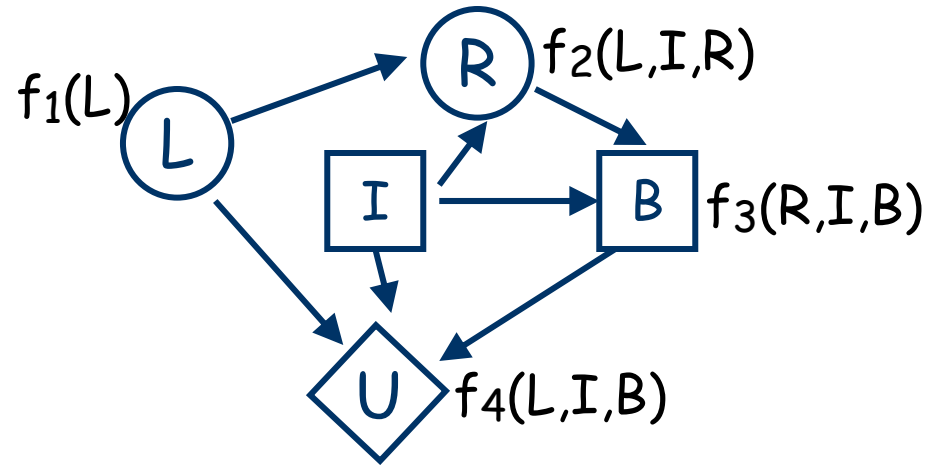
- $EU(I) = \sum_{L,R} P(L,R|i) U(L,i, \delta_{Buy} (I, R))$
 - where $P(R,L|i) = P(R|L,i)P(L|i)$
 - $EU(i) = (.1)(-650) + (.4)(-350) + (.45)(950) + (.05)(-350)$
 $= 205$
 - $EU(\sim i) = P(n,l|\sim i) U(l,\sim i, buy) + P(n,\sim l|\sim i) U(\sim l,\sim i, buy)$
 $= .5 * -600 + .5 * 1000 = 200$
 - So optimal $\delta_{Inspect} () = inspect$



	$P(R,L i)$	δ_{Buy}	$U(L, i, \delta_{Buy})$
g,l	0.1	buy	$-600 - 50 = -650$
b,l	0.4	\sim buy	$-300 - 50 = -350$
$g,\sim l$	0.45	buy	$1000 - 50 = 950$
$b,\sim l$	0.05	\sim buy	$-300 - 50 = -350$

Using Variable Elimination

Factors: $f_1(L)$ $f_2(L,I,R)$
 $f_3(R,I,B)$ $f_4(L,I,B)$
Query: $EU(I)?$
Evidence: none
Elim. Order: L, R, B



N.B. $f_3(R,I,B) = \delta_B(R,I)$

Step 1: Add $f_5(R,I,B) = \sum_L f_1(L) f_2(L,I,R) f_4(L,I,B)$
Remove: $f_1(L) f_2(L,I,R) f_4(L,I,B)$

Step 2: Add $f_6(I,B) = \sum_R f_3(R,I,B) f_5(R,I,B)$
Remove: $f_3(R,I,B) f_5(R,I,B)$

Step 3: Add $f_7(I) = \sum_B f_6(I,B)$
Remove: $f_6(I,B)$

Last factor: $f_7(I)$ is the expected utility of inspect and \sim inspect.
Select action with highest expected utility.

Value of Information

- So optimal policy is: inspect the car and if the report is good buy, otherwise don't buy
 - $EU = 205$
 - Notice that the EU of inspecting the car, then buying it iff you get a good report is 205 (i.e., $255 - 50$ (cost of inspection)) which is greater than 200. So inspection improves EU.
 - Suppose inspection cost is \$60: would it be worth it?
 - $EU = 255 - 60 = 195 < EU(\sim i)$
 - The *expected value of information* associated with inspection is 55 (it improves expected utility by this amount ignoring cost of inspection). How? Gives opportunity to change decision (\sim buy if bad).
 - You should be willing to pay up to \$55 for the report