CS 486/686: Introduction to Artificial Intelligence

Decision Theory

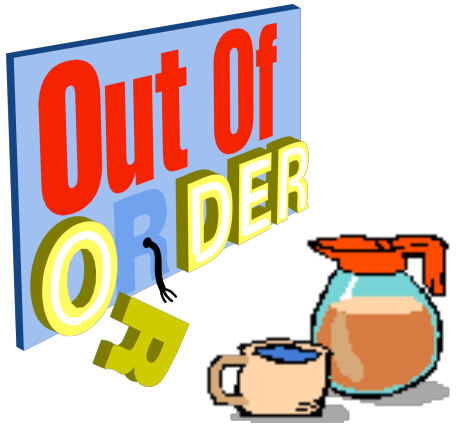
Plan for Today

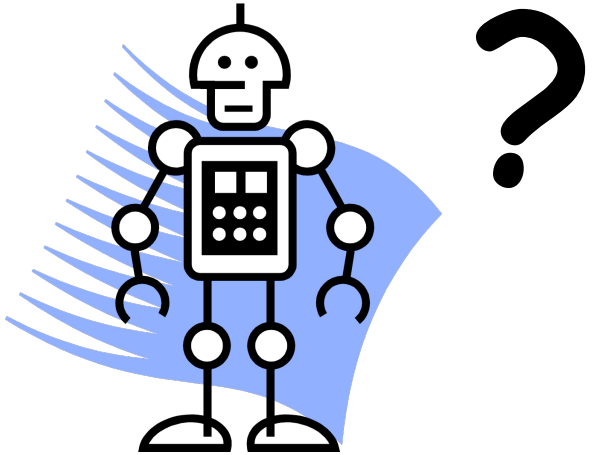
- Introduction to Utility Theory
- Principle of Maximum Expected Utility
- Decision Networks

Decision Making Under Uncertainty

I give a robot a planning problem: “ I want coffee”

But the coffee maker is broken: Robot reports “No plan!”





Decision Making Under Uncertainty

I want more robust behaviour

I want my robot to know what to do when my primary goal is not satisfied

Provide it with some indication of my preferences over alternatives

e.g. coffee better than tea, tea better than water, water better than nothing,...

Decision Making Under Uncertainty

- But it is more complicated than that
  - The robot could wait 45 minutes for the coffee machine to be fixed

- What is better?
  - Tea now?
  - Coffee in 45 minutes?

Preferences

A preference ordering $\succsim$ is a ranking over all possible states of the world s

These could be outcomes of actions, truth assignments, states in a search problem, etc

- $s \succsim t$: state s is **at least as good as** state t
- $s > t$: state s is **strictly preferred to** state t
- $s \sim t$: agent is **ambivalent between states** s

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 probability p_1, s_2 occurs with probability p_2, ...

Axioms

Orderability: Given 2 states A and B

$(A \succsim B) \vee (B \succsim A) \vee (A \sim B)$

Transitivity: Given 3 states A, B, C

$(A \succsim B) \wedge (B \succsim C) \rightarrow (A \succsim C)$

Continuity:

$A \succsim B \succsim C \rightarrow \text{Exists } p, [p,A;(1-p),C] \sim B$

Substitutability

$A \sim B \rightarrow [p,A;1-p,C] \sim [p,B,1-p,C]$

Monotonicity:

$(A \succsim B) \rightarrow (p \geq q \leftrightarrow [p,A;1-p,B] \succsim [q,A;1-q,B]$

Decomposability

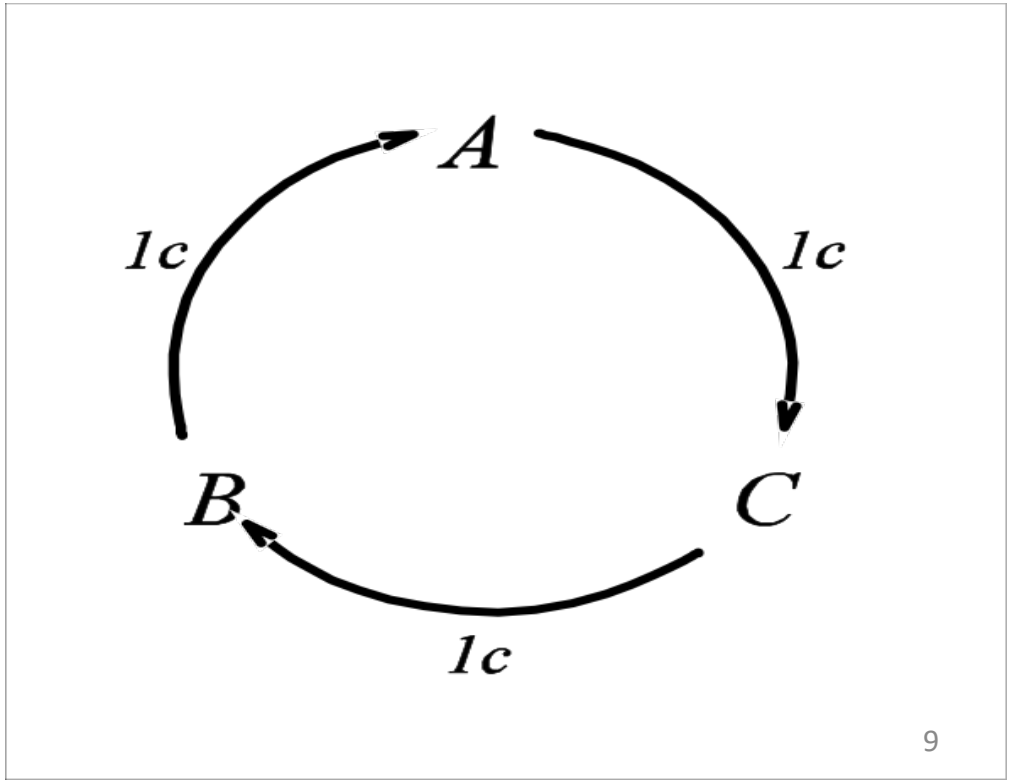
$[p,A;1-p[q,B;1-q,C]] \sim [p,A;\ (1-p)q,B;(1-p)(1-q),C]$

Why Impose These Conditions?

- Structure of preference ordering imposes certain “rationality requirements”
  - It is a weak ordering

- Example: Why transitivity?

A>B>C>A



Utilities

Rather than just ranking outcomes, we need to quantify our degree of preference

- How much more we prefer one outcome to another (e.g c to ~mess)

A utility function U:S→**R** associates a real-valued utility to each outcome

- Utility measures your degree of preference for s

U induces a preference ordering $\succsim_U$ over S where $s \succsim_U t$ if and only if U(s)≥U(t)

Expected Utility

- If there is uncertainty, then we use expected utility
  - $P_d(s)$ is the probability of outcome s under decision d
  - The **expected utility** of decision d is $EU(d)=\sum_{s \text{ in } S} P_d(s)U(s)$

- The Principle of Maximum Expected Utility: the optimal decision under conditions of uncertainty is the one with the greatest expected utility.

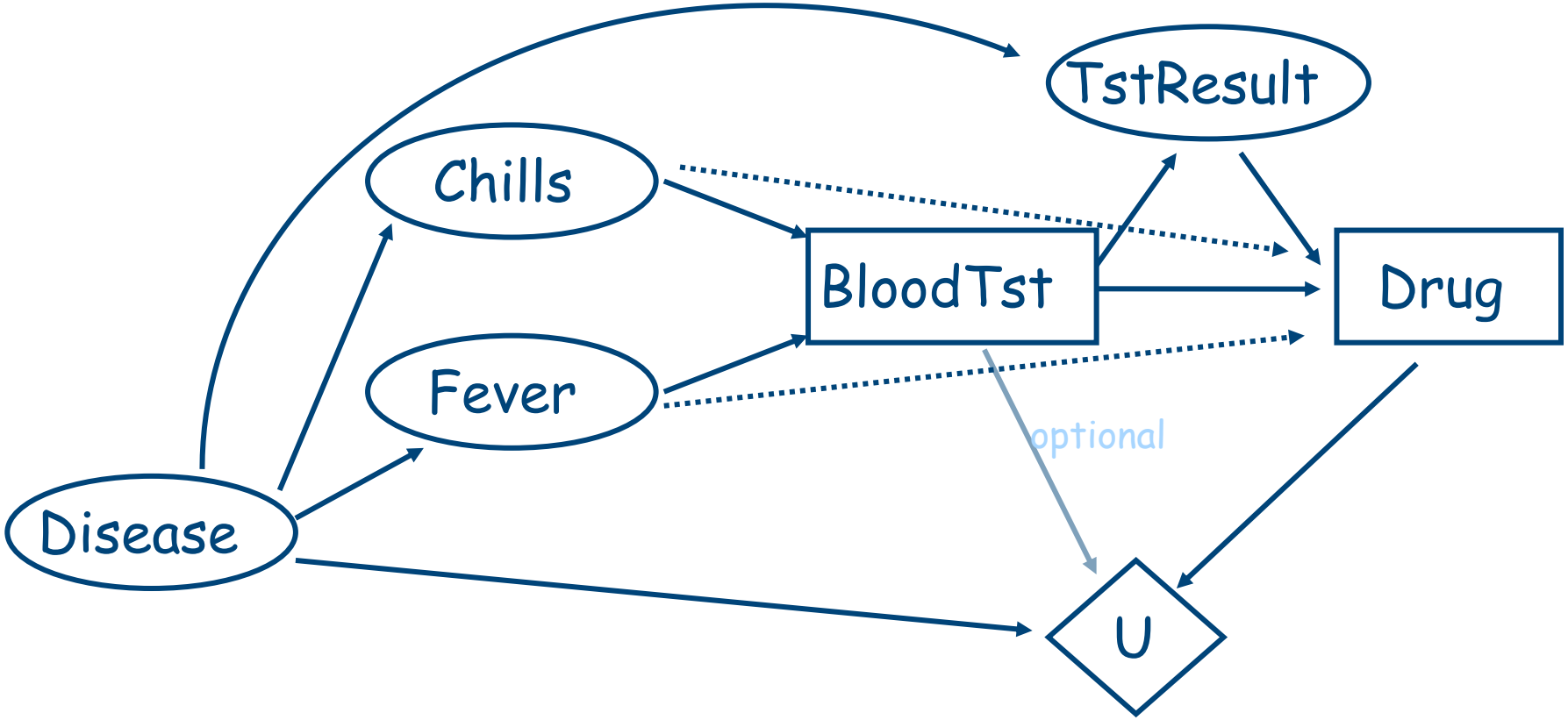
Notes on Expected Utility

- Utility functions need not be unique
  - If you multiply U by a positive constant, all decisions have the same relative utility
  - If you add a constant to U, then the same thing is true
- U is unique up to a positive affine transformation

$$\text{If } d^* = \text{argmax}_d \Pr(d)U(d)$$
$$\text{then}$$
$$d^* = \text{argmax}_d \Pr(d)[aU(d)+b]$$
$$a>0$$

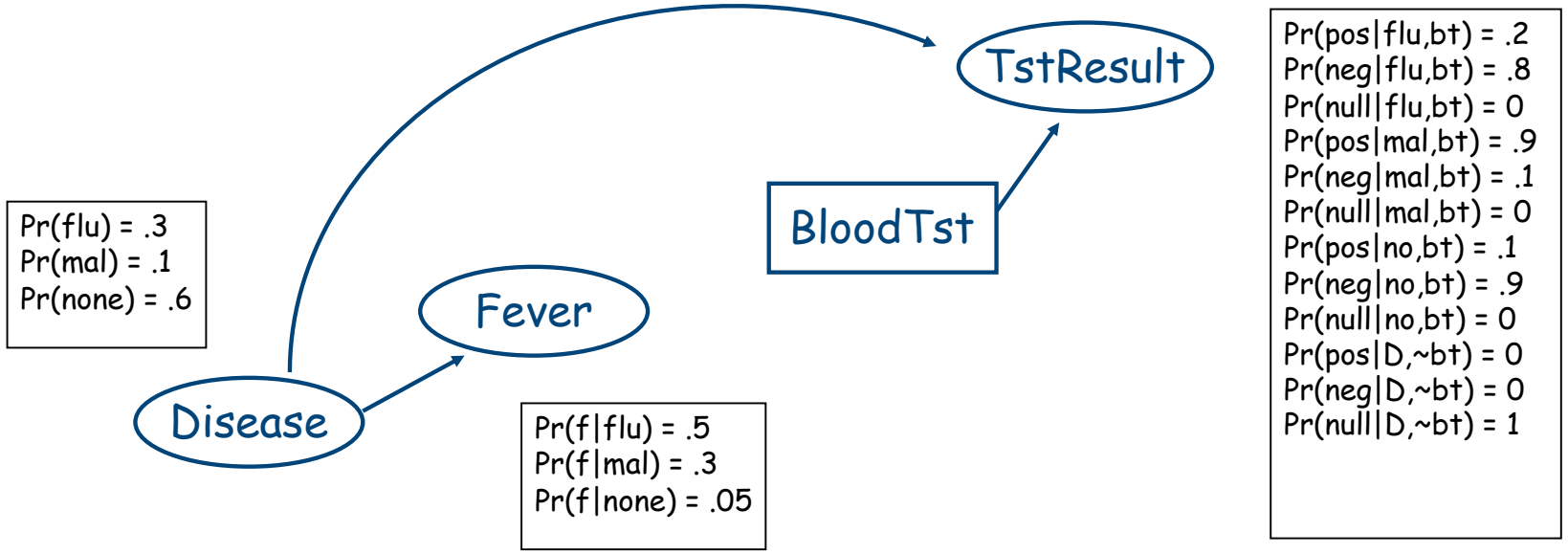
Decision Networks (Influence Diagrams)

- Decision networks (aka influence diagrams) provide a representation for sequential decision making
- Basic idea
  - Random variables like in Bayes Nets
  - Decision variables that you “control”
  - Utility variables which state how good certain states are


TstResult
Chills
BloodTst
Drug
Fever
optional
Disease
U

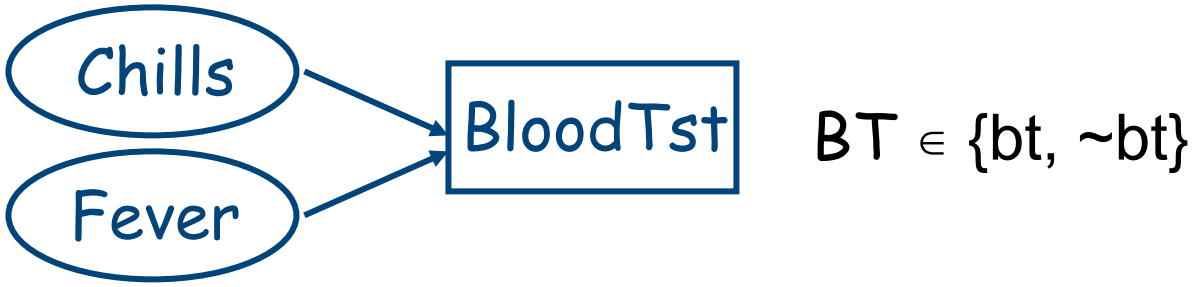
Chance Nodes

- Random variables (denoted by circles)
- Like as in a BN, probabilistic dependence on parents



Decision Nodes

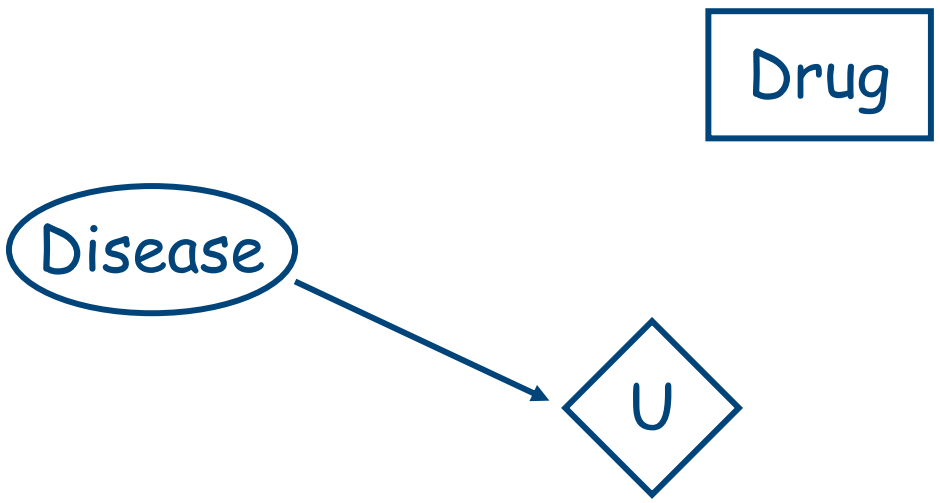
- Variables the decision maker sets (denoted by squares)
- Parents reflect information available at time of decision



BT ∈ {bt, ~bt}

Value Node

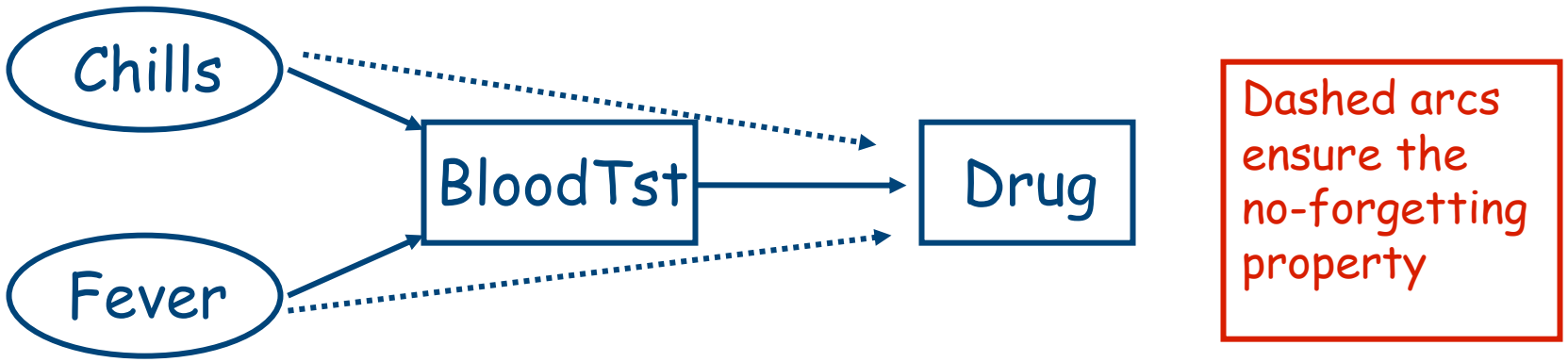
- Specifies the utility of a state (denoted by a diamond)
- Utility depends only on state of parents
- Generally, only one value node in a network



U(fludrug, flu) = 20
U(fludrug, mal) = -300
U(fludrug, none) = -5
U(maldrug, flu) = -30
U(maldrug, mal) = 10
U(maldrug, none) = -20
U(no drug, flu) = -10
U(no drug, mal) = -285
U(no drug, none) = 30

Assumptions

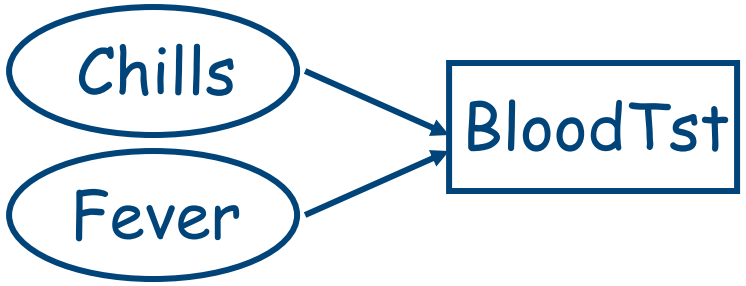
- Decision nodes are totally ordered
  - Given decision variables $D_1,..., D_n$, decisions are made in sequence
- No forgetting property
  - Any information available for decision D_i remains available for decision D_j where j>i
  - All parents of D_i are also parents for D_j



Policies

- Let Par(D_i) be the parents of decision node D_i
  - Dom(Par(D_i)) is the set of assignments to Par(D_i)
- A policy δ is a set of mappings δ_i, one for each decision node D_i
  - $\delta_i(D_i)$ associates a decision for each parent assignment
  - δ_i:Dom(Par(D_i))→Dom(D_i)

$\delta_{BT}(c,f)=bt$
$\delta_{BT}(c,\sim f)=\sim bt$
$\delta_{BT}(\sim c,f)=bt$
$\delta_{BT}(\sim c,\sim f)=\sim bt$



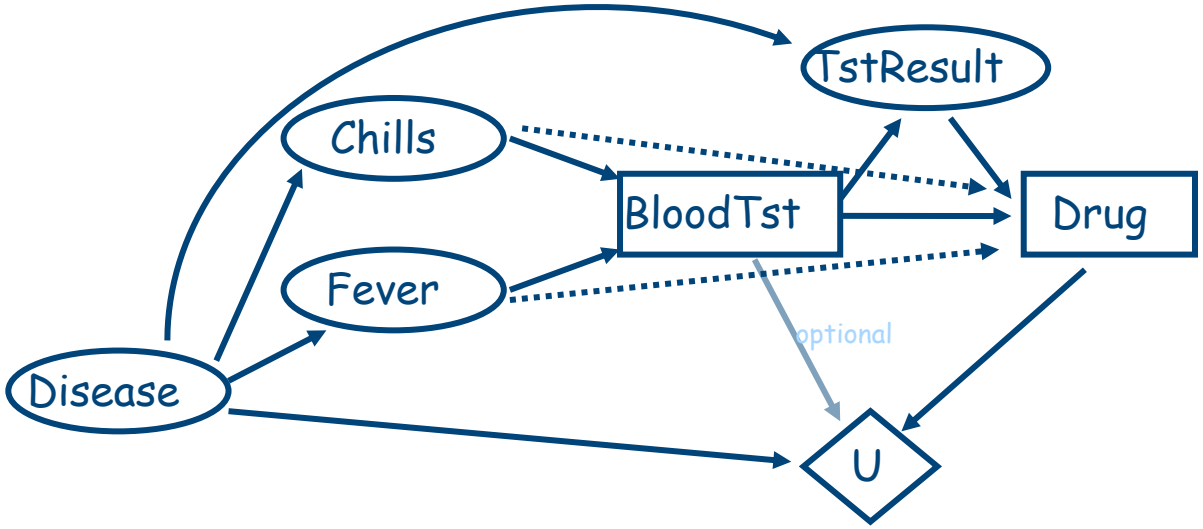
Value of a Policy

- Expected utility given that all decision nodes are executed according to the policy

- An **optimal policy** δ^* is such that $EU(\delta^*) \geq EU(\delta)$ for all δ
- We can use dynamic programming to avoid enumerating all possible policies
- We can also use the BN structure and Variable Elimination to aid the computation

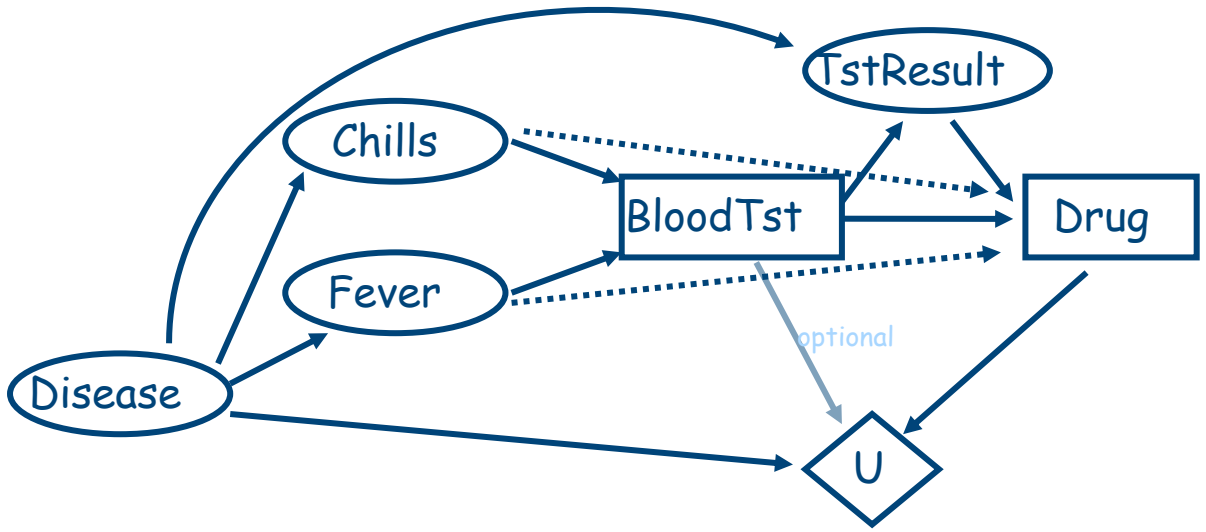
Computing the Optimal Policy

- Work backwards as follows
  - Compute optimal policy for Drug
  - 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



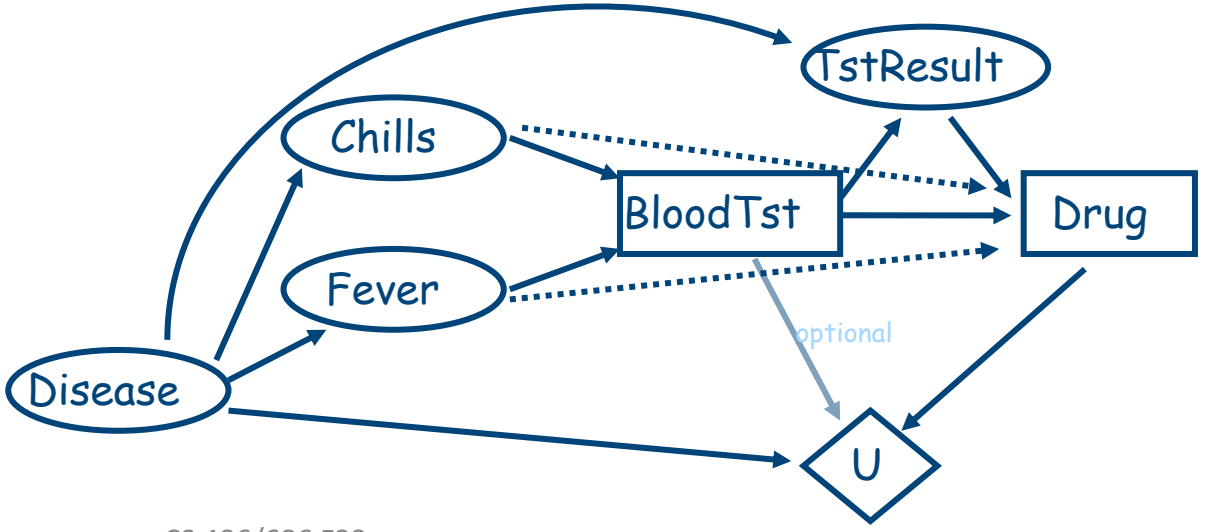
Computing the Optimal Policy

- Next compute policy for BT, given policy δD(C,F,BT,TR) just computed
  - Since δ_D is fixed, we treat D as a random variable with deterministic probabilities
  - Solve for BT just like you did for D



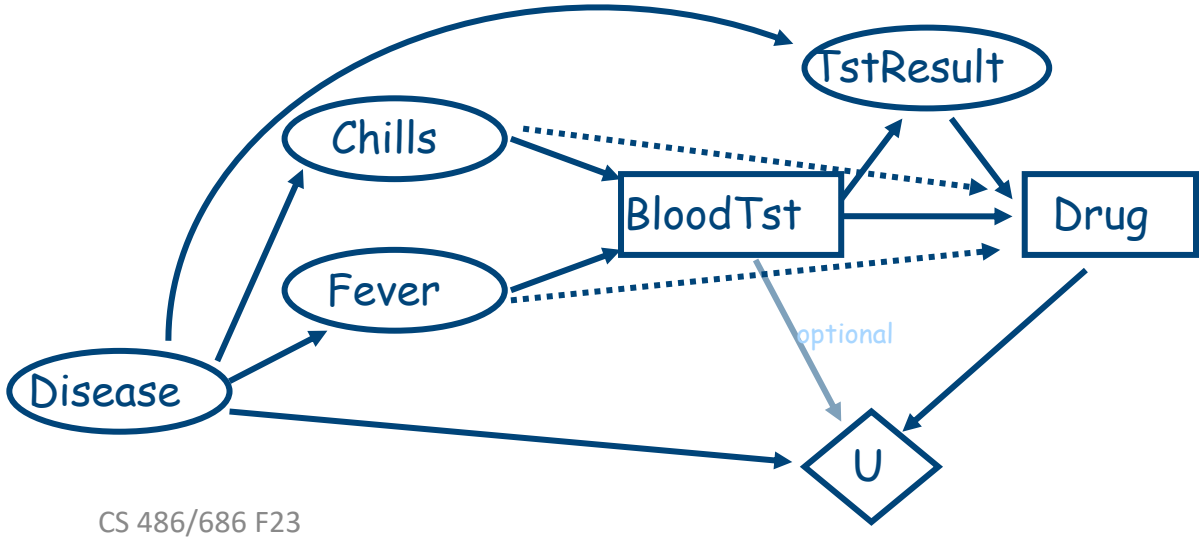
Computing the Optimal Policy

- How do we compute these expected values?
  - Suppose we have asst *<c,f,bt,pos>* to parents of *Drug*
  - We want to compute EU of deciding to set *Drug = md*
  - We can run variable elimination!



Computing the Optimal Policy

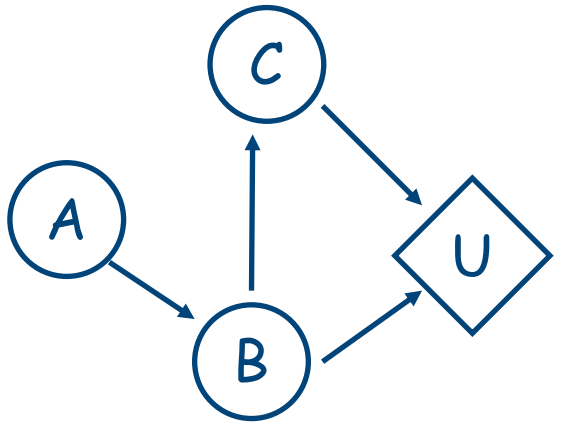
- Treat C, F, BT, Tr, Dr as evidence
  - This reduces the factors
  - Eliminate remaining variables (Dis)
  - Left with factor U()=ΣDis P(Dis | c,f,bt,pos,md)U(Dis,md,bt)
- We now know EU of doing Dr=md when c,f,bt,pos



Computing the Optimal Policy

- Computing expected utilities with BNs is straightforward
- Utility nodes are just factors that can be dealt with using variable elimination

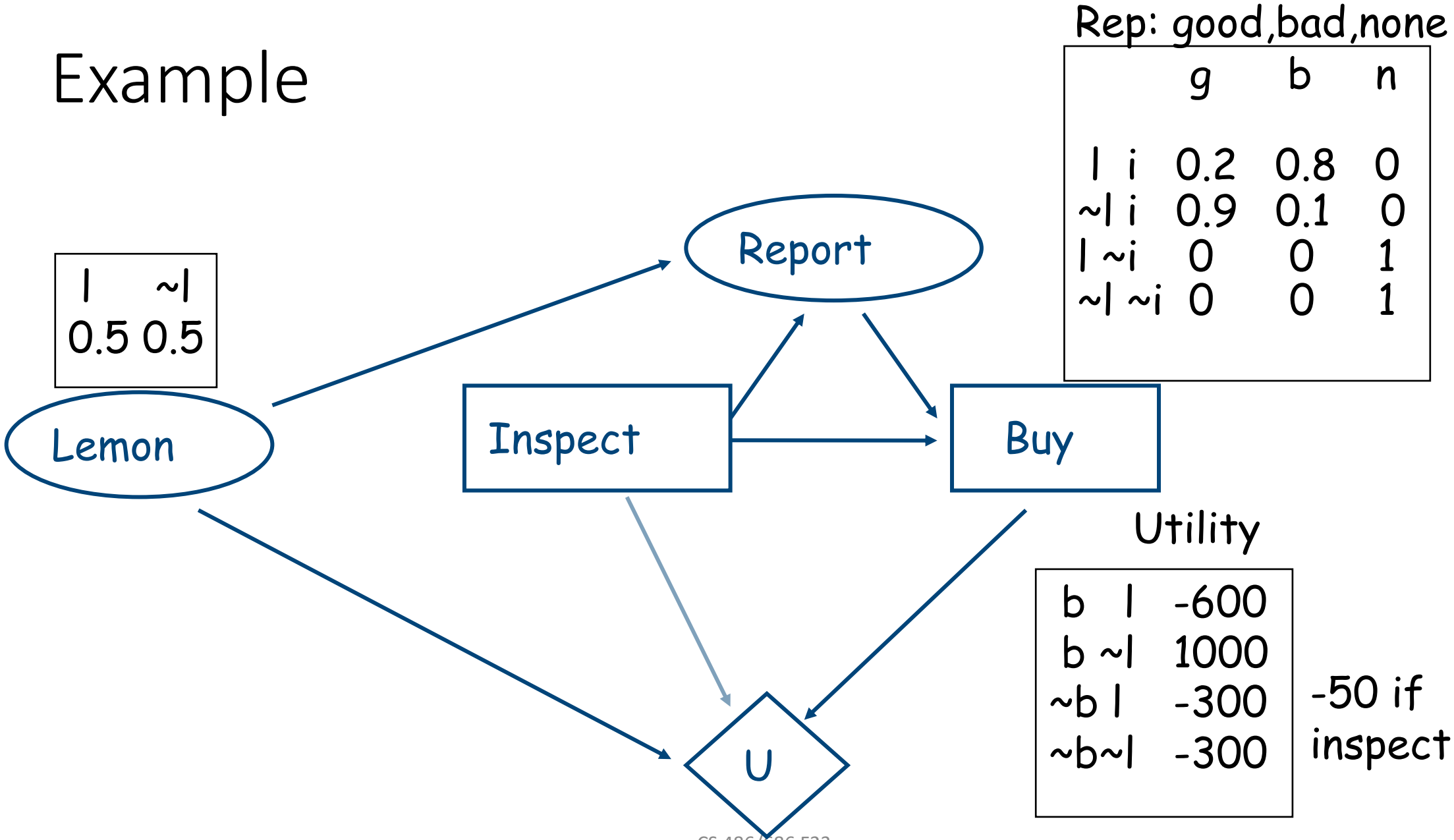
$$EU = \Sigma_{A,B,C} P(A,B,C)\, U(B,C)$$
$$= \Sigma_{A,B,C} P(C|B)\, P(B|A)\, P(A)\, U(B,C)$$



Example

- You want to buy a used car, but there is some chance it is a “lemon” (i.e. it breaks down often). Before deciding to buy it, you can take it to a mechanic for an inspection. S/he will give you a report, labelling the car as either “good” or “bad”. A good report is positively correlated with the car not being a lemon while a bad report is positively correlated with the car being a lemon
- The report costs $50. You could risk it and buy the car with no report.
- Owning a good car is better than no car, which is better than owning a lemon.

Example


Rep: good,bad,none
g b n
l i 0.2 0.8 0
~l i 0.9 0.1 0
l ~i 0 0 1
~l ~i 0 0 1
l ~l
0.5 0.5
Lemon
Report
Inspect
Buy
U
Utility
b l -600
b ~l 1000
~b l -300
~b~l -300
-50 if inspect

Value of Information

- Information has value
  - To the extent it is likely to cause a change of plan
  - To the extent that the new plan will be significantly better than the old plan
- The value of information is non-negative
  - This is true for any decision-theoretic agent (but not necessarily true for multi-agent settings)