Lecture 10: Decision Making

CS 486/686: Introduction to Artificial Intelligence

Plan for Today

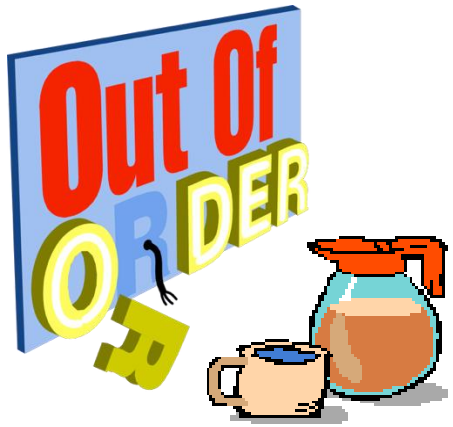
- Introduction to Preferences and Utility Theory
- Principle of Maximum Expected Utility
- Decision Networks

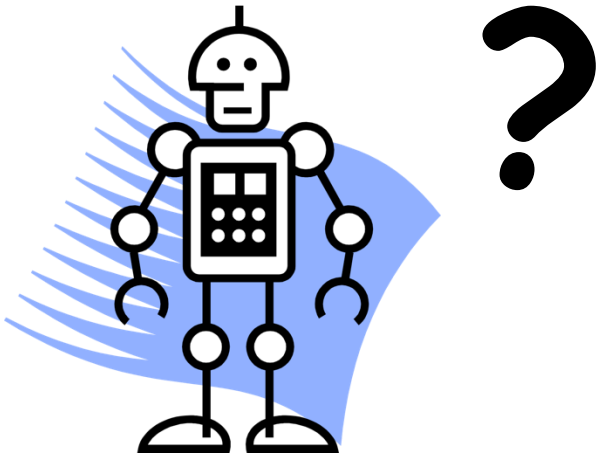
Plan for Today

- Introduction to Preferences and 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 so my robot answers "No plan!"





Decision Making Under Uncertainty

- This is unacceptable
  - I want my robot to know what to do if my primary goal is unsatisfied
    - Maybe tea now is OK
    - Maybe the machine will be fixed in 45 minutes and I am OK with waiting
    - Maybe I am simply thirsty and so water would be fine
    - …
- I need the agent/robot to reason about the state of the world and my preferences

Beliefs and desires under uncertainty

- Suppose we have agent
  - Available actions *a*
  - Probability of current state *P(s)* to account for state uncertainty
  - Transition model *P(s'|s, a)* to account for transition uncertainty
- Probability of going to a specific state after taking an action

$$P(\text{RESULT}(a)=s') = \sum_s P(s)P(s'|s,a)\,.$$

Decision Theory

- Choose actions based on desirability of their *immediate* outcomes
  - Utility function *U(s)* assigns number to express desirability of state
  - Expected Utility *EU(a)* weighs *U(s')* by probability of *s'*

$$P(\text{RESULT}(a)=s') = \sum_s P(s)P(s'|s,a). \quad EU(a) = \sum_{s'} P(\text{RESULT}(a)=s')U(s'). \qquad (15.1)$$

  - *Rational agents* choose actions with *maximum expected utility (MEU)*

$$action = \underset{a}{\text{argmax}}\, EU(a).$$

An argument for maximizing expected utility

- Suppose agent has following preferences over states

$A \succ B$	the agent prefers A over B.
$A \sim B$	the agent is indifferent between A and B.
$A \succsim B$	the agent prefers A over B or is indifferent between them.

- Treat outcome of each action as a *lottery* with outcomes $S_1 \ldots S_n$

$$L = [p_1, S_1;\ p_2, S_2;\ \ldots\ p_n, S_n]\,.$$

An argument for maximizing expected utility

- We define a rational agent as follows $L = [p_1, S_1;\ p_2, S_2;\ \ldots\ p_n, S_n]$.
  - Orderability: given 2 lotteries, must prefer one or rate as equal

$$\text{Exactly one of } (A \succ B), (B \succ A), \text{ or } (A \sim B) \text{ holds.}$$

  - Transitivity: $A \sim B \Rightarrow [p, A;\ 1-p, C] \sim [p, B;\ 1-p, C]$.

$$(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C).$$

  - Continuity: if *A* > *B* > *C* then there is some *p* for which agent indifferent between getting $A \succ B \succ C \Rightarrow \exists p\,[\mathrm{p}, A;\ 1-p, C] \sim B.$ with p and C w/ *1-p*

An argument for maximizing expected utility

- We define a rational agent as follows $L = [p_1, S_1;\ p_2, S_2;\ \ldots\ p_n, S_n]$.
  - *Substitutability:* If agent indifferent b/w A and B, then also indifferent wrt substitutions b/w A and B in lotteries

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

  - *Monotonicity:* If two L's have same outcomes, and agent prefers outcome *O*,* then it must prefer lottery w/ higher *P(O*)*

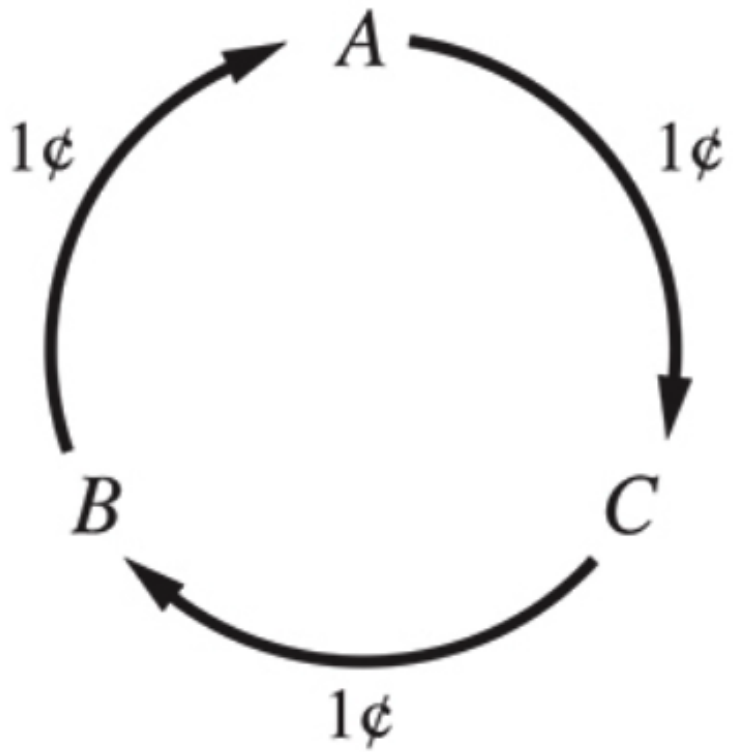
$$A \succ B \Rightarrow (p > q \Leftrightarrow [p, A;\ 1-p, B] \succ [q, A; 1-q, B]).$$

  - *Decomposability:* compound lotteries can be reduced to simpler ones using probability

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

An argument for maximizing expected utility

- Why are these important?
  - Example:
    - A > B > C > A causes irrational behaviour
    - If agent has A
      - Agent prefers C > A: We trade C for A + 1 cent
      - Agent prefers B > C: We trade B for C + 1 cent
      - Agent prefers A > B: We trade A for B + 1 cent
      - Agent back where it started and lost 3 cents



An argument for maximizing expected utility

- Suppose agent preferences obey axioms of utility
  - Then there exists *U* such that *U(A) > U(B)* iff A is preferred to B
    - *U(A) = U(B)* if agent is indifferent

$$U(A) > U(B) \Leftrightarrow A \succ B \text{ and } U(A) = U(B) \Leftrightarrow A \sim B.$$

  - Utility of lottery is sum of *P(outcome) x U(outcome)*

$$U([p_1, S_1; \ldots; p_n, S_n]) = \sum_i p_i U(s_i).$$

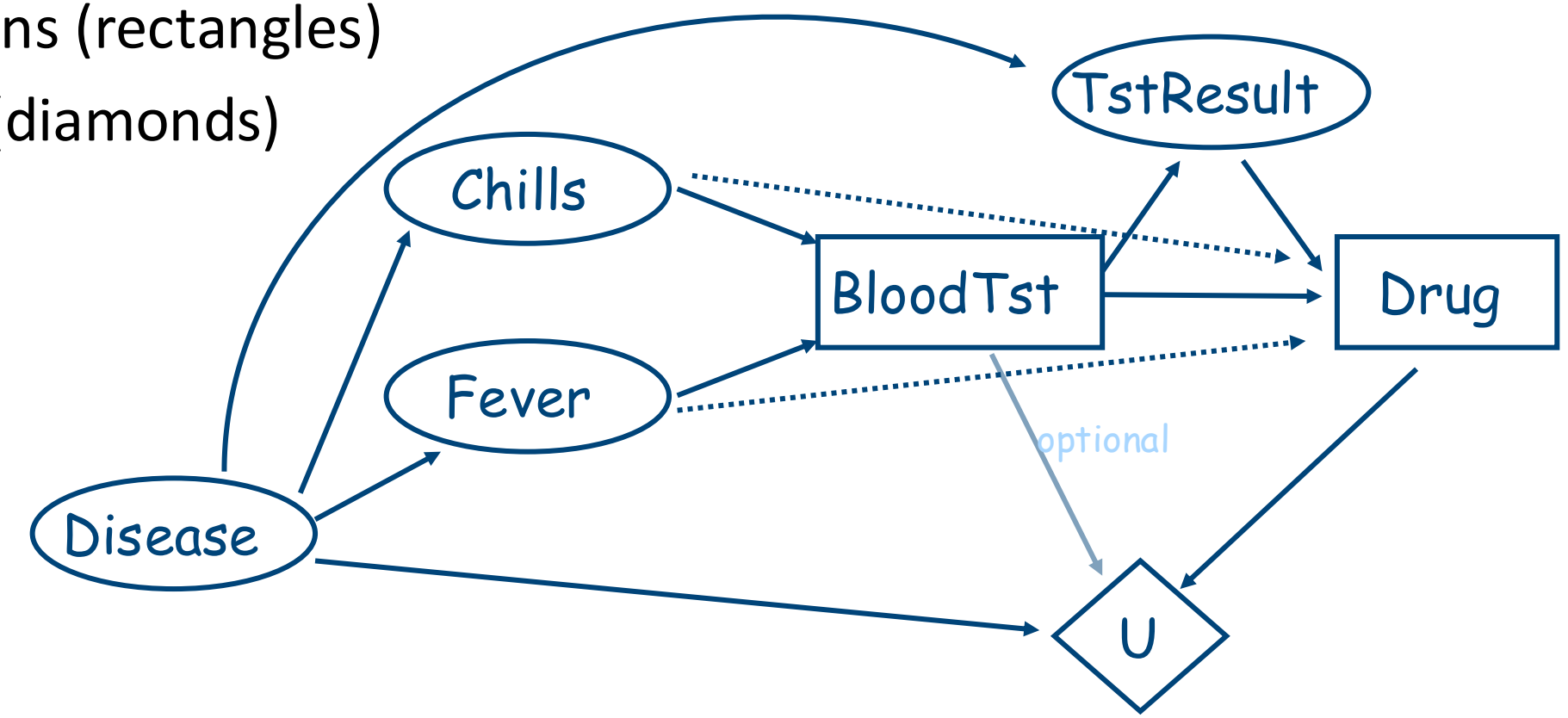
- Agent can act rationally once probabilities and utilities of outcome are defined, consistent with its preferences

How can we make rational decisions?

- Making rational decisions involves maximizing expected utility
- Represent decision problem with a *decision network*
  - Bayesian network augmented with actions and utilities
  - Contains
    - Agent state
    - Possible actions
    - Resulting state
    - Utility of state

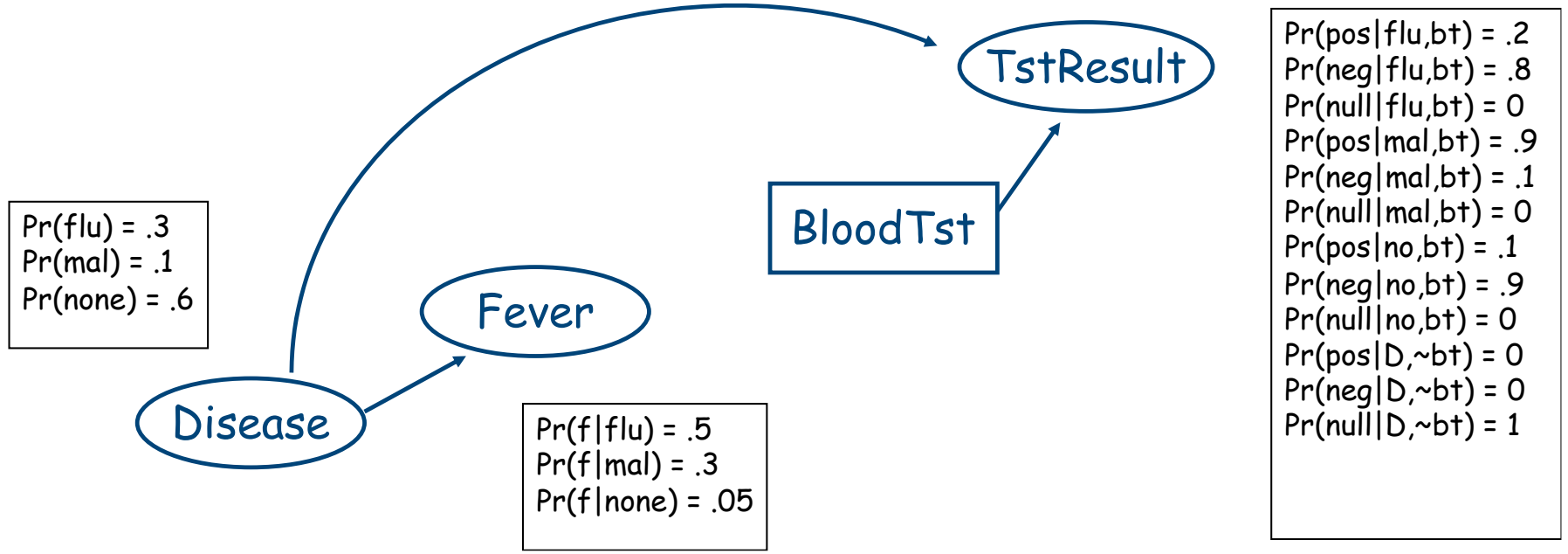
How can we make rational decisions?

- Nodes
  - Random variables (oval)
  - Decisions (rectangles)
  - Utility (diamonds)



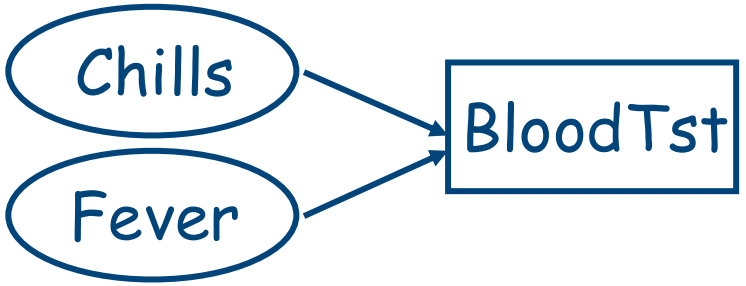
Chance Nodes

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



Decision Nodes

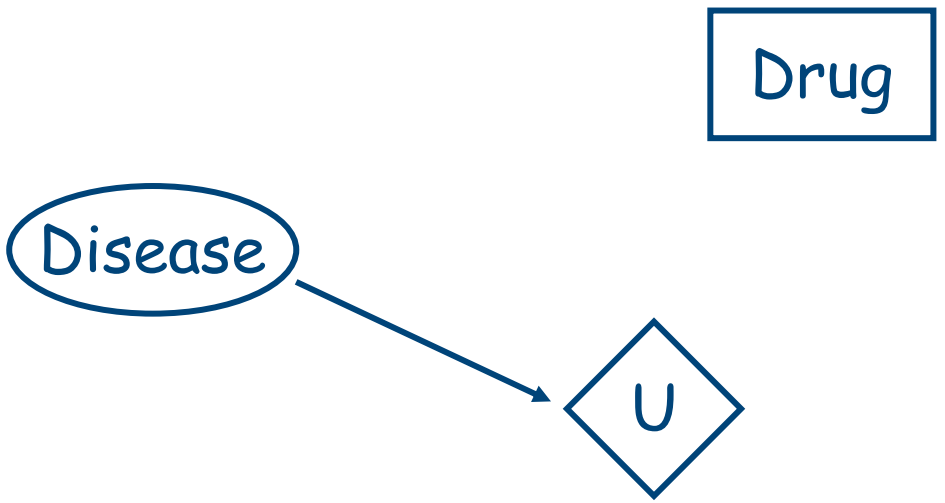
- Variables that the decision-maker sets
- Parents reflect information available at the time of the decision



BT ∈ {bt, ~bt}

Value Nodes

- Specifies the utility of a state, depending on the state of parents
- Generally, only one value node in the network

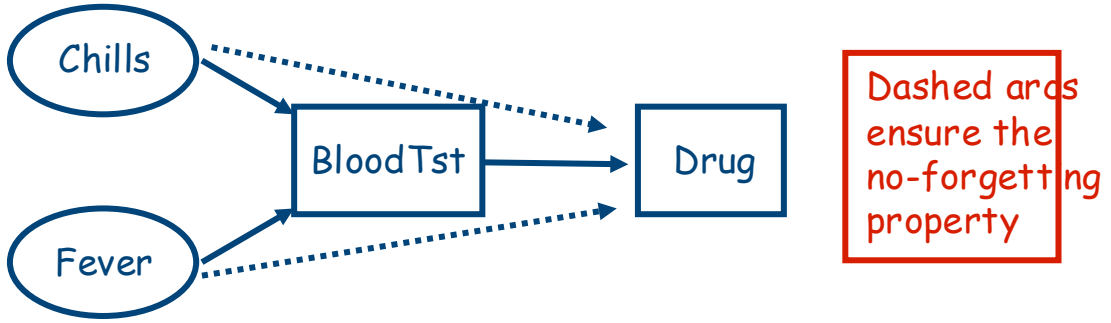


U(fludrug, flu) = 20
U(fludrug, mal) = -300
U(fludrug, none) = -5
U(maldrug, flu) = -30
U(maldrug, mal) = 10
U(maldrug, none) = -20
U(no drug, flu) = -10
U(no drug, mal) = -285
U(no drug, none) = 30

Policies

- Decision nodes are totally ordered
- No forgetting property
  - Any information that is available at D_i is available for D_{i+1}
    - Parents of D_i are implicit parents of D_{i+1}

- A **policy, ∏,** is a mapping from decision nodes to actions
- The value of a policy is the expected utility obtained from following ∏



Optimal Policy

- An Optimal Policy $\prod^*$ is $\prod^* = \arg\max_{\prod} EU(\prod)$

- Note that we might have a sequence of decisions $D_1, \ldots, D_i, D_{i+1}, \ldots$
  - A policy has to specify what action to take for each D_i, given Parents(D_i)
  - We can compute the optimal policy by using dynamic programming and leveraging the Bayes Net structures (and Variable Elimination) to handle the uncertainty

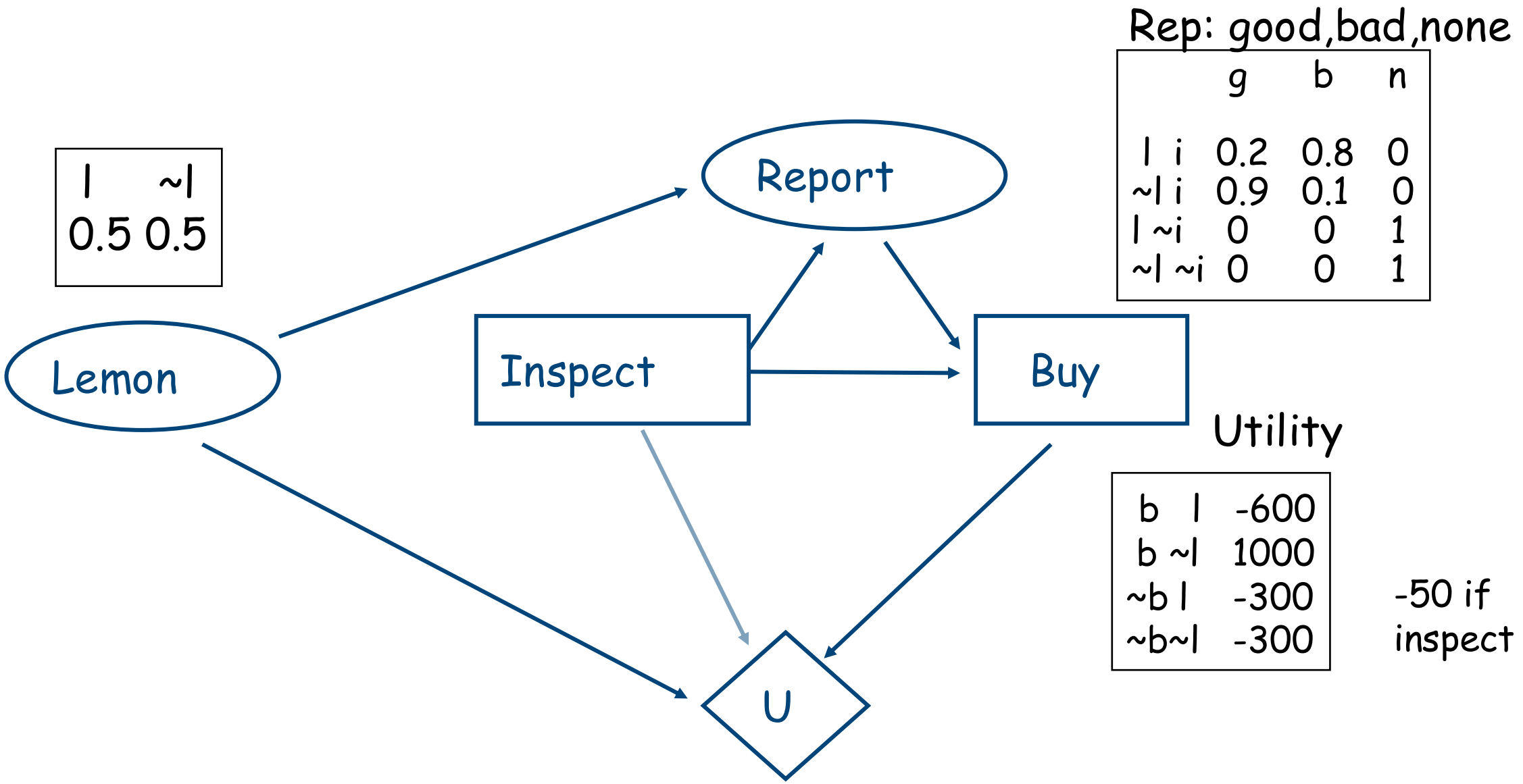
Finding an optimal policy

1. Create a factor for each conditional probability table and a factor for the utility.
2. Set remaining decision nodes ← all decision nodes
3. Multiply factors and sum out random variables that are not parents of a remaining decision node.
4. Select and remove a decision variable *D* from list of remaining decision nodes:
   - pick one that is in a factor with only itself and some of its parents (no children).
5. Eliminate *D* by maximizing. This returns:
   - the optimal decision function for *D*, $argmax_D f$
   - a new factor to use, $max_D f$
6. Repeat 3-5 till there are no more remaining decision nodes.
7. Eliminate the remaining random variables. Multiply the factors: this is the expected utility of the optimal policy.
8. If any nodes were in evidence, divide by the *P(evidence)*

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
Report
Lemon
Inspect
Buy
Utility
b l -600
b ~l 1000
~b l -300
~b~l -300
-50 if inspect
U

Value of Information

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