Making Decisions Under Uncertainty

What an agent should do depends on:

- The agent's ability — what options are available to it.
- The agent's beliefs — the ways the world could be, given the agent's knowledge.
  Sensing updates the agent's beliefs.
- The agent's preferences — what the agent wants and tradeoffs when there are risks.

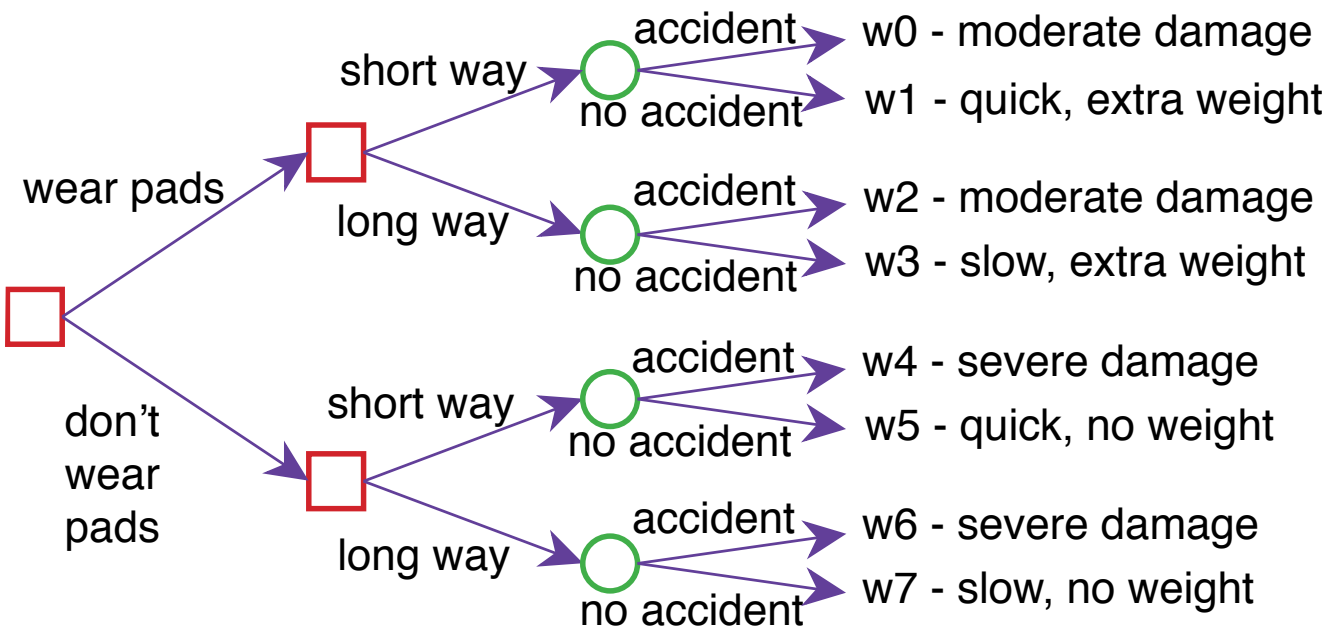
Decision theory specifies how to trade off the desirability and probabilities of the possible outcomes for competing actions.

Decision Variables

- Decision variables are like random variables that an agent gets to choose a value for.
- A possible world specifies a value for each decision variable and each random variable.
- For each assignment of values to all decision variables, the measure of the set of worlds satisfying that assignment sum to 1.
- The probability of a proposition is undefined unless the agent condition on the values of all decision variables.

Decision Tree for Delivery Robot

The robot can choose to wear pads to protect itself or not.
The robot can choose to go the short way past the stairs or a long way that reduces the chance of an accident.
There is one random variable of whether there is an accident.



Expected Values

- The expected value of a function of possible worlds is its average value, weighting possible worlds by their probability.
- Suppose $f(\omega)$ is the value of function f on world ω.
  - The expected value of f is
$$\mathcal{E}(f) = \sum_{\omega \in \Omega} P(\omega) \times f(\omega).$$
  - The conditional expected value of f given e is
$$\mathcal{E}(f|e) = \sum_{\omega \models e} P(\omega|e) \times f(\omega).$$

Utility

- Utility is a measure of desirability of worlds to an agent.
- Let $u(\omega)$ be the utility of world ω to the agent.
- Simple goals can be specified by: worlds that satisfy the goal have utility 1; other worlds have utility 0.
- Often utilities are more complicated: for example some function of the amount of damage to a robot, how much energy is left, what goals are achieved, and how much time it has taken.

Single decisions

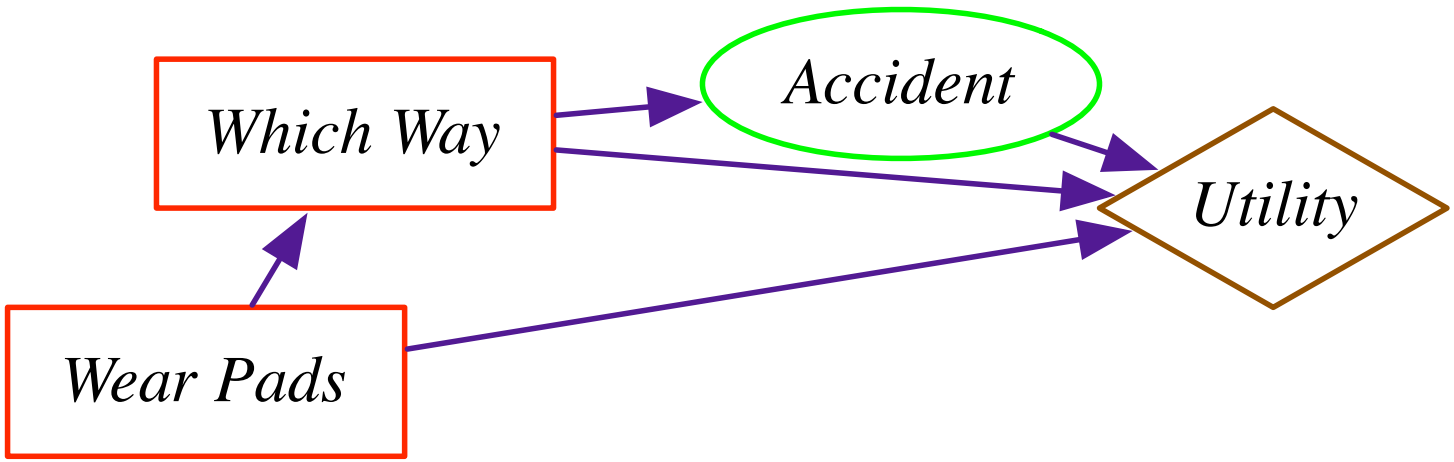
- In a single decision variable, the agent can choose $D = d_i$ for any $d_i \in dom(D)$.
- The expected utility of decision $D = d_i$ is $\mathcal{E}(u|D = d_i)$.
- An optimal single decision is the decision $D = d_{max}$ whose expected utility is maximal:

$$\mathcal{E}(u|D = d_{max}) = \max_{d_i \in dom(D)} \mathcal{E}(u|D = d_i).$$

Single-stage decision networks

Extend belief networks with:

- Decision nodes, that the agent chooses the value for. Domain is the set of possible actions. Drawn as rectangle.
- Utility node, the parents are the variables on which the utility depends. Drawn as a diamond.



This shows explicitly which nodes affect whether there is an accident.

Finding the optimal decision

- Suppose the random variables are $X_1, \ldots, X_n$, and utility depends on $X_{i_1}, \ldots, X_{i_k}$

$$\begin{aligned}\mathcal{E}(u|D) &= \sum_{X_1,\ldots,X_n} P(X_1, \ldots, X_n|D) \times u(X_{i_1}, \ldots, X_{i_k}) \\ &= \sum_{X_1,\ldots,X_n} \prod_{i=1}^{n} P(X_i|parents(X_i)) \times u(X_{i_1}, \ldots, X_{i_k})\end{aligned}$$

  To find the optimal decision:
  - Create a factor for each conditional probability and for the utility
  - Sum out all of the random variables
  - This creates a factor on D that gives the expected utility for each D
  - Choose the D with the maximum value in the factor.

Example Initial Factors

Which Way	Accident	Value
long	true	0.01
long	false	0.99
short	true	0.2
short	false	0.8

Which Way	Accident	Wear Pads	Value
long	true	true	30
long	true	false	0
long	false	true	75
long	false	false	80
short	true	true	35
short	true	false	3
short	false	true	95
short	false	false	100

Sequential Decisions

- An intelligent agent doesn't carry out a multi-step plan ignoring information it receives in between steps.
- A more typical scenario is where the agent:
  observes, acts, observes, acts, ...
- Subsequent actions can depend on what is observed.
  What is observed depends on previous actions.
- Often the sole reason for carrying out an action is to provide information for future actions.
  For example: diagnostic tests, spying.

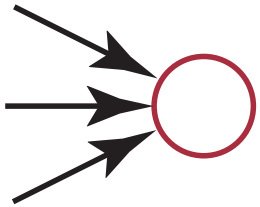
Sequential decision problems

- A sequential decision problem consists of a sequence of decision variables $D_1, \ldots, D_n$.
- Each D_i has an information set of variables $parents(D_i)$, whose value will be known at the time decision D_i is made.

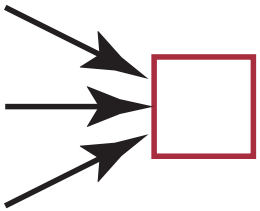
Decision Networks

- A decision network is a graphical representation of a finite sequential decision problem.
- Decision networks extend belief networks to include decision variables and utility.
- A decision network specifies what information is available when the agent has to act.
- A decision network specifies which variables the utility depends on.

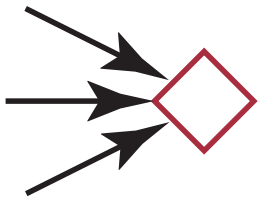
Decisions Networks



- A random variable is drawn as an ellipse. Arcs into the node represent probabilistic dependence.

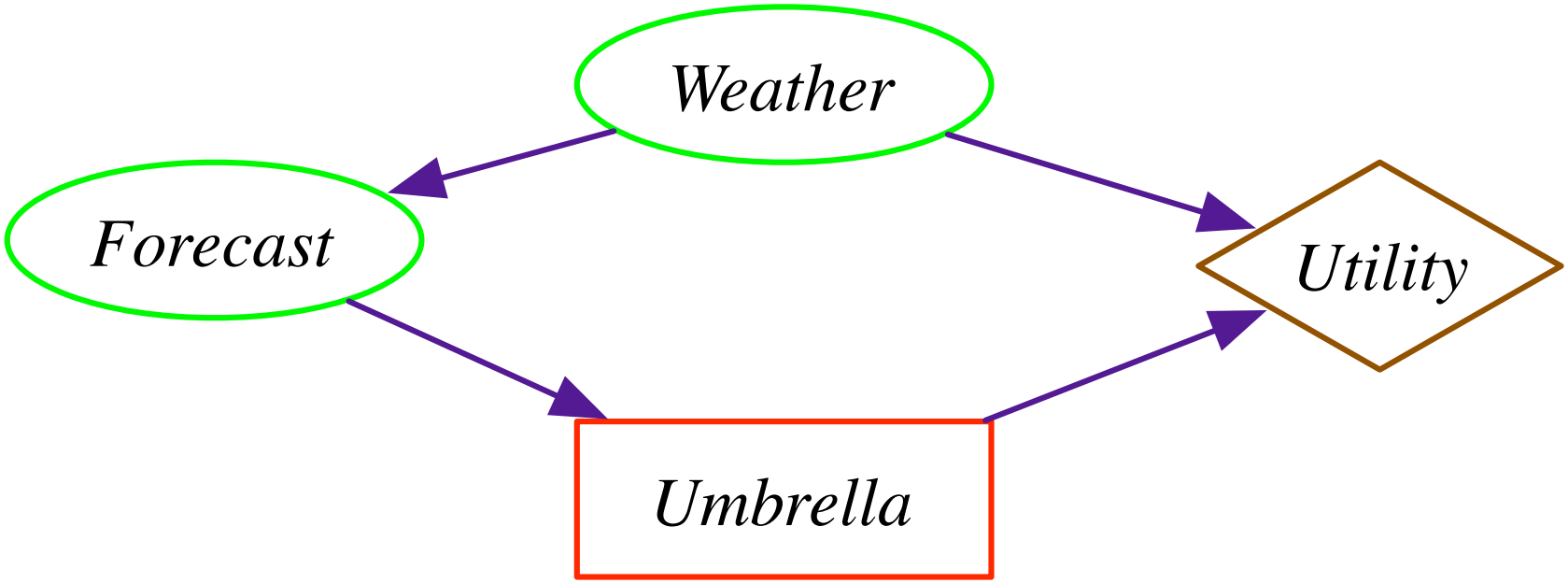


- A decision variable is drawn as an rectangle. Arcs into the node represent information available when the decision is make.



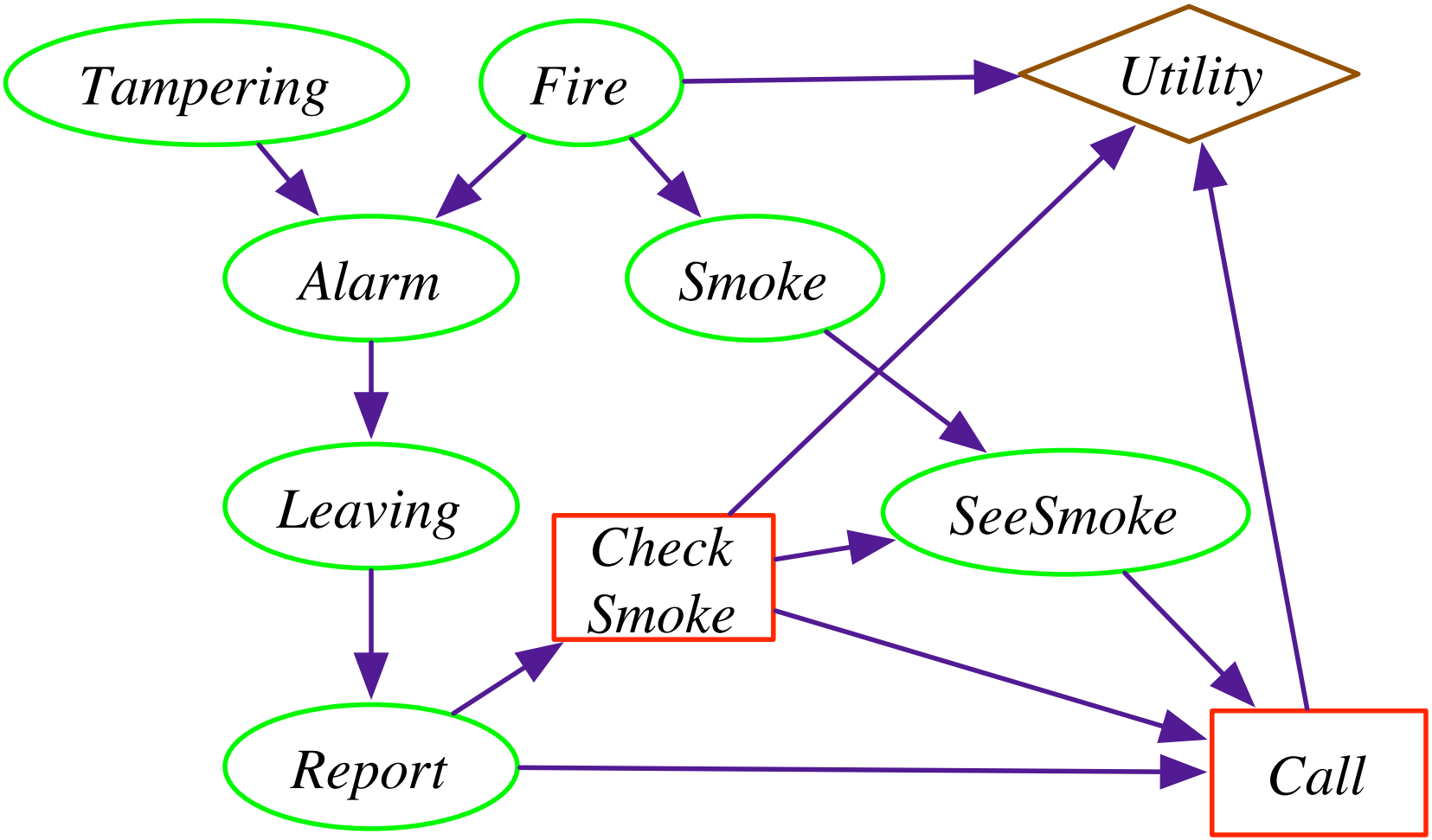
- A utility node is drawn as a diamond. Arcs into the node represent variables that the utility depends on.

Umbrella Decision Network



You don't get to observe the weather when you have to decide whether to take your umbrella. You do get to observe the forecast.

Decision Network for the Alarm Problem



No-forgetting

A No-forgetting decision network is a decision network where:

- The decision nodes are totally ordered. This is the order the actions will be taken.
- All decision nodes that come before D_i are parents of decision node D_i. Thus the agent remembers its previous actions.
- Any parent of a decision node is a parent of subsequent decision nodes. Thus the agent remembers its previous observations.

What should an agent do?

- What an agent should do at any time depends on what it will do in the future.
- What an agent does in the future depends on what it did before.

Policies

- A policy specifies what an agent should do under each circumstance.
- A policy is a sequence $\delta_1, \ldots, \delta_n$ of decision functions

$$\delta_i : dom(parents(D_i)) \rightarrow dom(D_i).$$

This policy means that when the agent has observed $O \in dom(parents(D_i))$, it will do $\delta_i(O)$.

Expected Utility of a Policy

- Possible world ω satisfies policy δ, written $\omega \models \delta$ if the world assigns the value to each decision node that the policy specifies.
- The expected utility of policy δ is

$$\mathcal{E}(u|\delta) = \sum_{\omega \models \delta} u(\omega) \times P(\omega),$$

- An optimal policy is one with the highest expected utility.

Finding the optimal policy

- Remove all variables that are not ancestors of the utility node
- Create a factor for each conditional probability table and a factor for the utility.
- Sum out variables that are not parents of a decision node.
- Select a variable D that is only in a factor f with (some of) its parents.
- Eliminate D by maximizing. This returns:
  - the optimal decision function for D, $\arg\max_D f$
  - a new factor to use in VE, $\max_D f$
- Repeat till there are no more decision nodes.
- Eliminate the remaining random variables. Multiply the factors: this is the expected utility of the optimal policy.

Initial factors for the Umbrella Decision

Weather	Value
norain	0.7
rain	0.3

Weather	Fcast	Value
norain	sunny	0.7
norain	cloudy	0.2
norain	rainy	0.1
rain	sunny	0.15
rain	cloudy	0.25
rain	rainy	0.6

Weather	Umb	Value
norain	take	20
norain	leave	100
rain	take	70
rain	leave	0

Eliminating By Maximizing

f:

Fcast	Umb	Val
sunny	take	12.95
sunny	leave	49.0
cloudy	take	8.05
cloudy	leave	14.0
rainy	take	14.0
rainy	leave	7.0

$\max_{Umb} f$:

Fcast	Val
sunny	49.0
cloudy	14.0
rainy	14.0

$\arg\max_{Umb} f$:

Fcast	Umb
sunny	leave
cloudy	leave
rainy	take

Complexity of finding the optimal policy

- If there are k binary parents, to a decision D, there are 2^k assignments of values to the parents.
- If there are b possible actions, there are b^{2^k} different decision functions.
- The number of policies is the product of the number decision functions.
- The number of optimizations in the dynamic programming is the sum of the number of assignments of values to parents.
- The dynamic programming algorithm is much more efficient than searching through policy space.

Value of Information

- The value of information X for decision D is the utility of the network with an arc from X to D (+ no-forgetting arcs) minus the utility of the network without the arc.
- The value of information is always non-negative.
- It is positive only if the agent changes its action depending on X.
- The value of information provides a bound on how much an agent should be prepared to pay for a sensor. How much is a better weather forecast worth?
- We need to be careful when adding an arc would create a cycle. E.g., how much would it be worth knowing whether the fire truck will arrive quickly when deciding whether to call them?

Value of Control

- The value of control of a variable X is the value of the network when you make X a decision variable (and add no-forgetting arcs) minus the value of the network when X is a random variable.
- You need to be explicit about what information is available when you control X.
- If you control X without observing, controlling X can be worse than observing X. E.g., controlling a thermometer.
- If you keep the parents the same, the value of control is always non-negative.