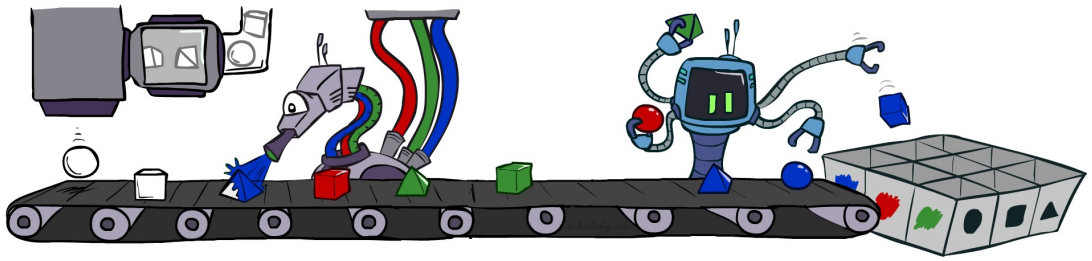
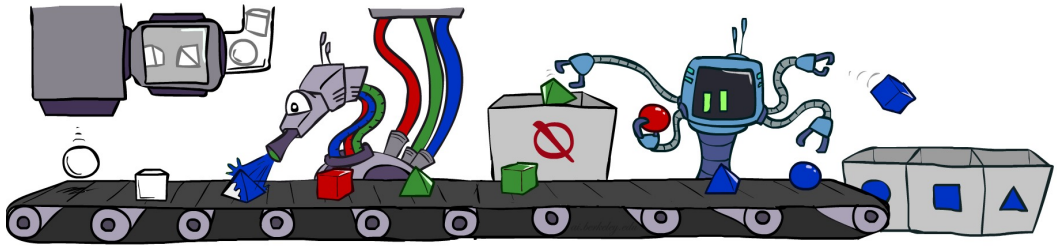
Bayes Net Sampling

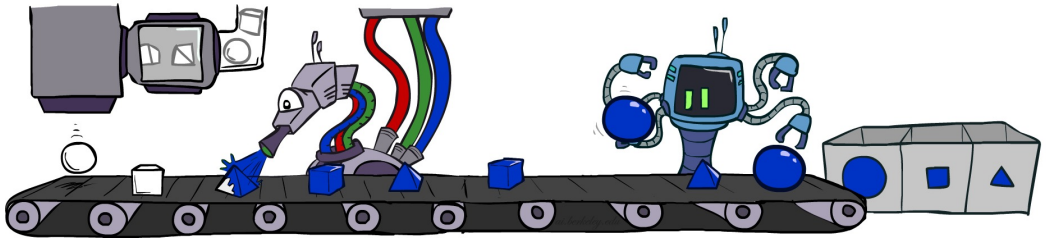
- Prior Sampling P :
  - Generate complete samples from $P(x_1,\ldots,x_n)$



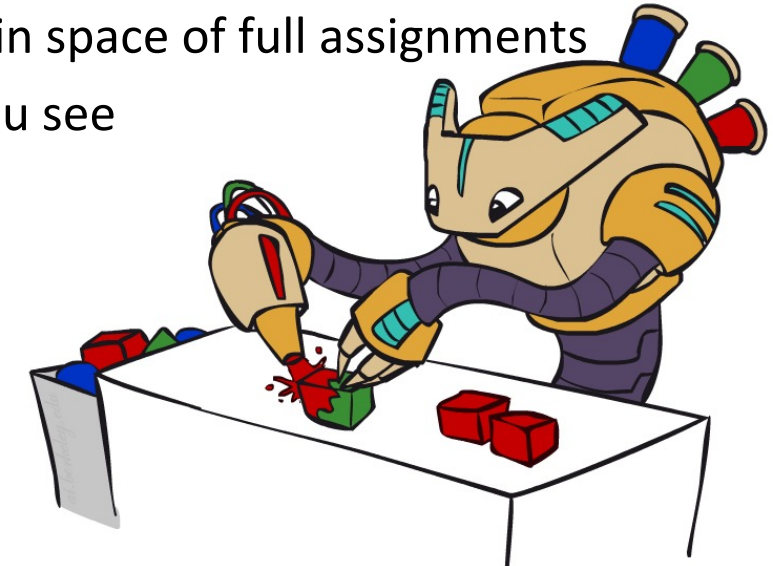
- Rejection Sampling $P(Q \mid \boldsymbol{e})$:
  - Reject samples that don't match $\boldsymbol{e}$



- Likelihood Weighting $P(Q \mid \boldsymbol{e})$:
  - Weight samples by how well they predict $\boldsymbol{e}$

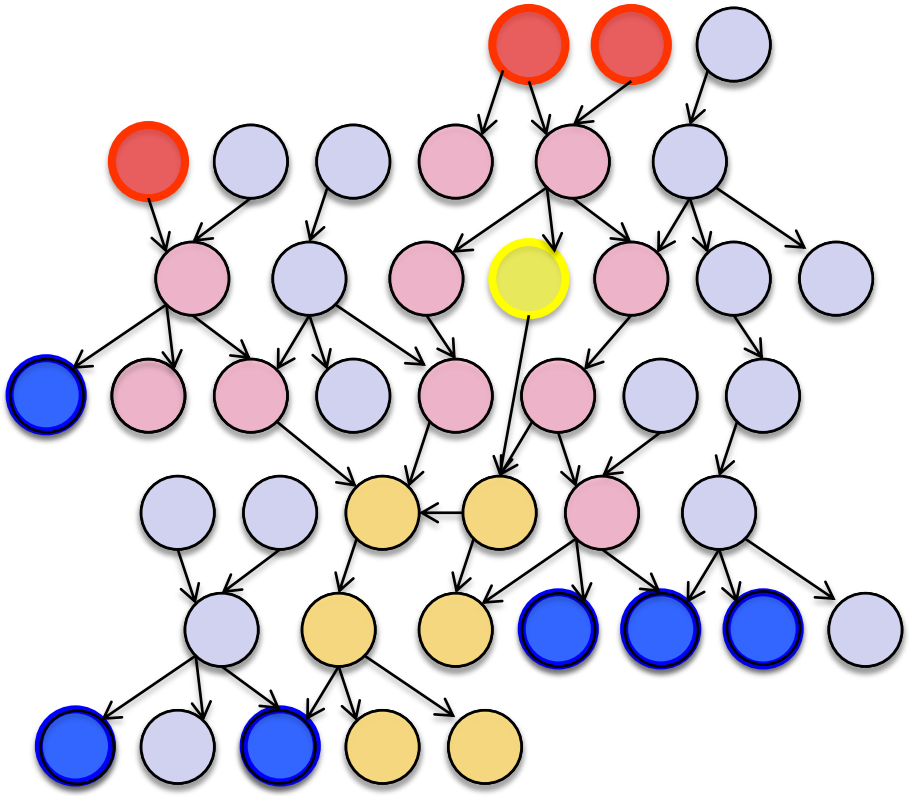


- Gibbs sampling $P(Q \mid \boldsymbol{e})$:
  - Wander around in space of full assignments
  - Average what you see



Likelihood weighting

- Likelihood weighting is good
  - All samples are used
  - The values of *downstream* variables are influenced by *upstream* evidence

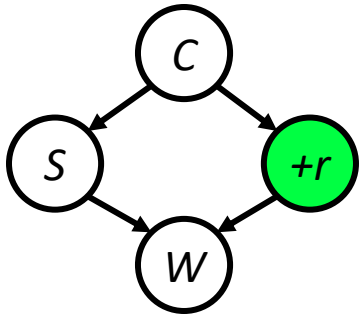


- Likelihood weighting still has weaknesses
  - The values of *upstream* variables are unaffected by *downstream* evidence
  - With evidence in k leaf nodes, weights will be $O(2^{-k})$
  - With high probability, one lucky sample will have much larger weight than the others, dominating the result

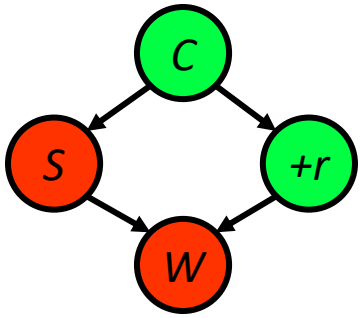
- We would like each variable to "see" ***all*** the evidence!

Gibbs Sampling Example: P(S | +r)

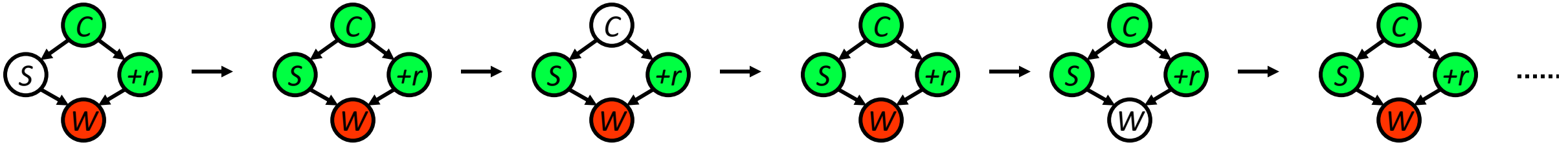
- Step 1: Fix evidence
  - R = +r



- Step 2: Initialize other variables
  - Randomly



- Steps 3: Repeat
  - Choose a *non-evidence* variable X
  - *Resample* X from P(X | all other variables)*

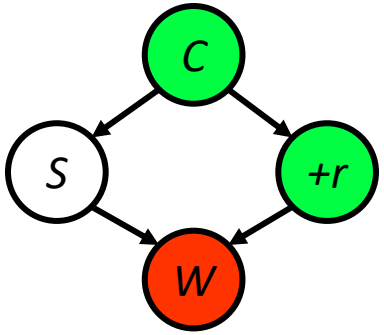


Sample from $P(S|+c,-w,+r)$ Sample from $P(C|+s,-w,+r)$ Sample from $P(W|+s,+c,+r)$

How to compute this?

Resampling Probability

- Sample from P(S | +c, +r, -w)



$$P(S|+c,+r,-w) = \frac{P(S,+c,+r,-w)}{P(+c,+r,-w)}$$ *Normalization trick*

$$= \frac{P(S,+c,+r,-w)}{\sum_s P(s,+c,+r,-w)}$$ *Definition of marginal*

$$= \frac{P(+c)P(S|+c)P(+r|+c)P(-w|S,+r)}{\sum_s P(+c)P(s|+c)P(+r|+c)P(-w|s,+r)}$$ *BN joint probabilities*

$$= \frac{P(+c)P(S|+c)P(+r|+c)P(-w|S,+r)}{P(+c)P(+r|+c)\sum_s P(s|+c)P(-w|s,+r)}$$ *Sum early*

$$= \frac{P(S|+c)P(-w|S,+r)}{\sum_s P(s|+c)P(-w|s,+r)}$$ *Only factors with S remain!*

- More generally: join all factors involving *the resampled variable* and normalize
- Better way: $P(X_i \mid x_1,..,x_{i-1},x_{i+1},..,x_n) = P(X_i \mid \text{markov\_blanket}(X_i))$

Gibbs sampling algorithm

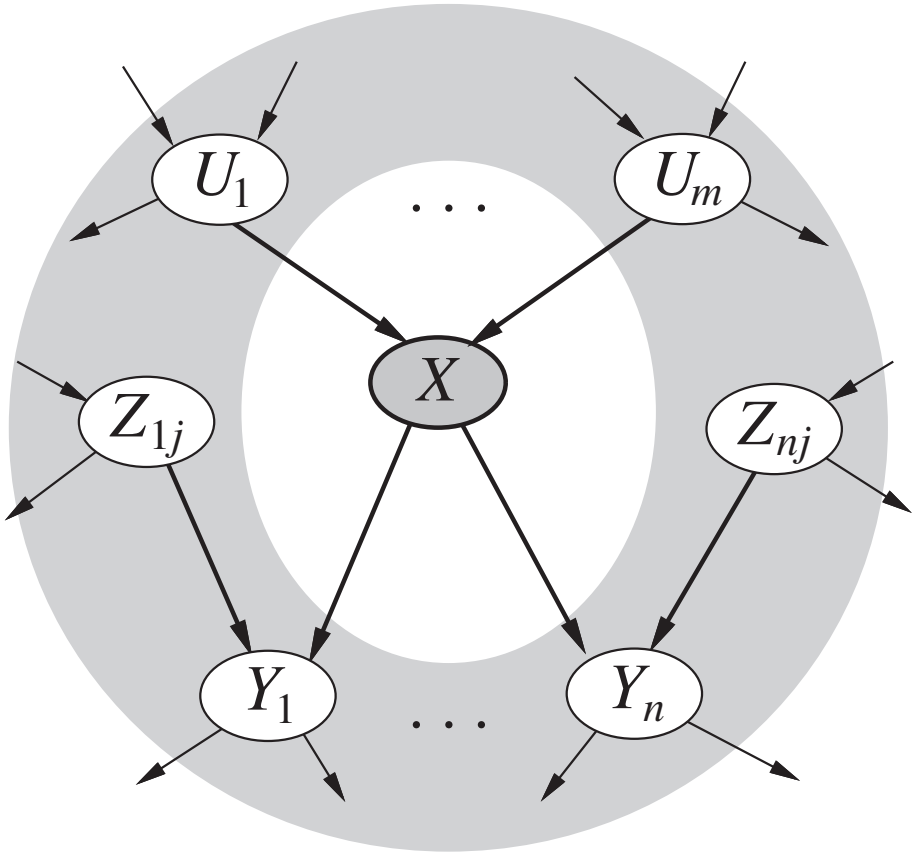
- Repeat many times
  - Sample a non-evidence variable X_i from

$P(X_i \mid x_1,..,x_{i-1},x_{i+1},..,x_n) = P(X_i \mid markov\_blanket(X_i))$

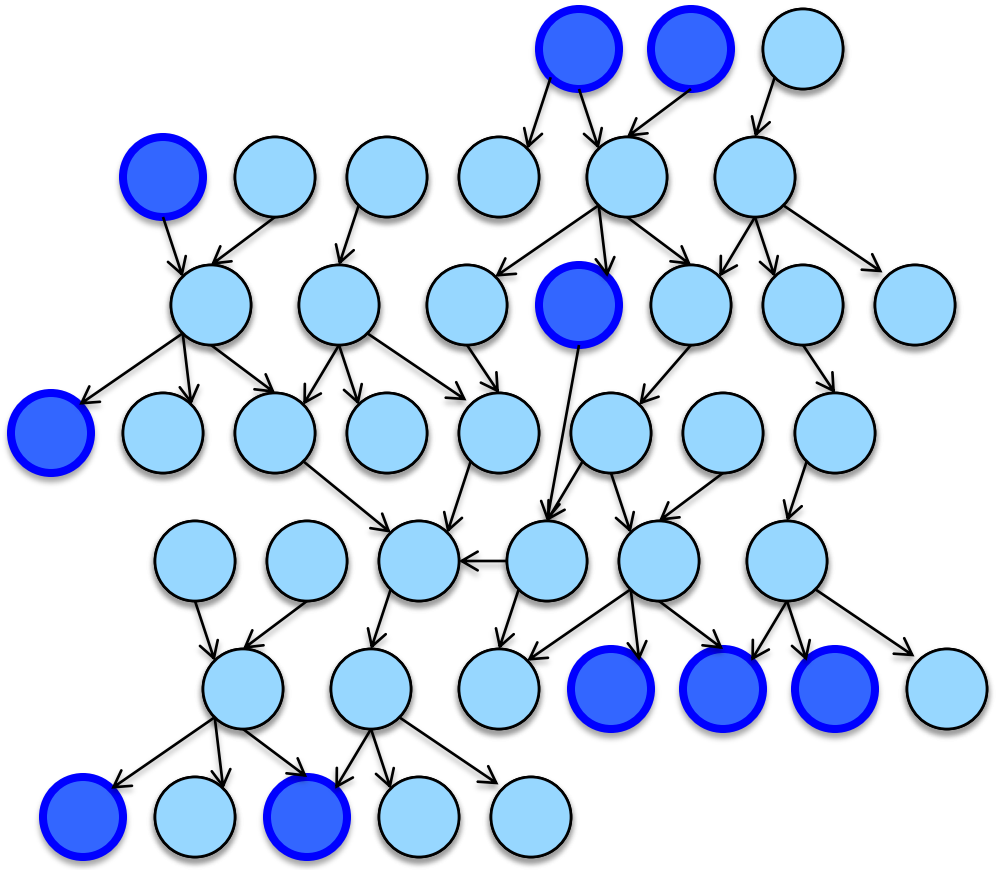
$= \alpha\ P(X_i \mid parents\ (X_i))\ \prod_j P(y_j \mid parents(Y_j))$

join all factors involving X_i and normalize

- Influence of evidences propagates locally, and gradually becomes global



Advantages of Gibbs Sampling



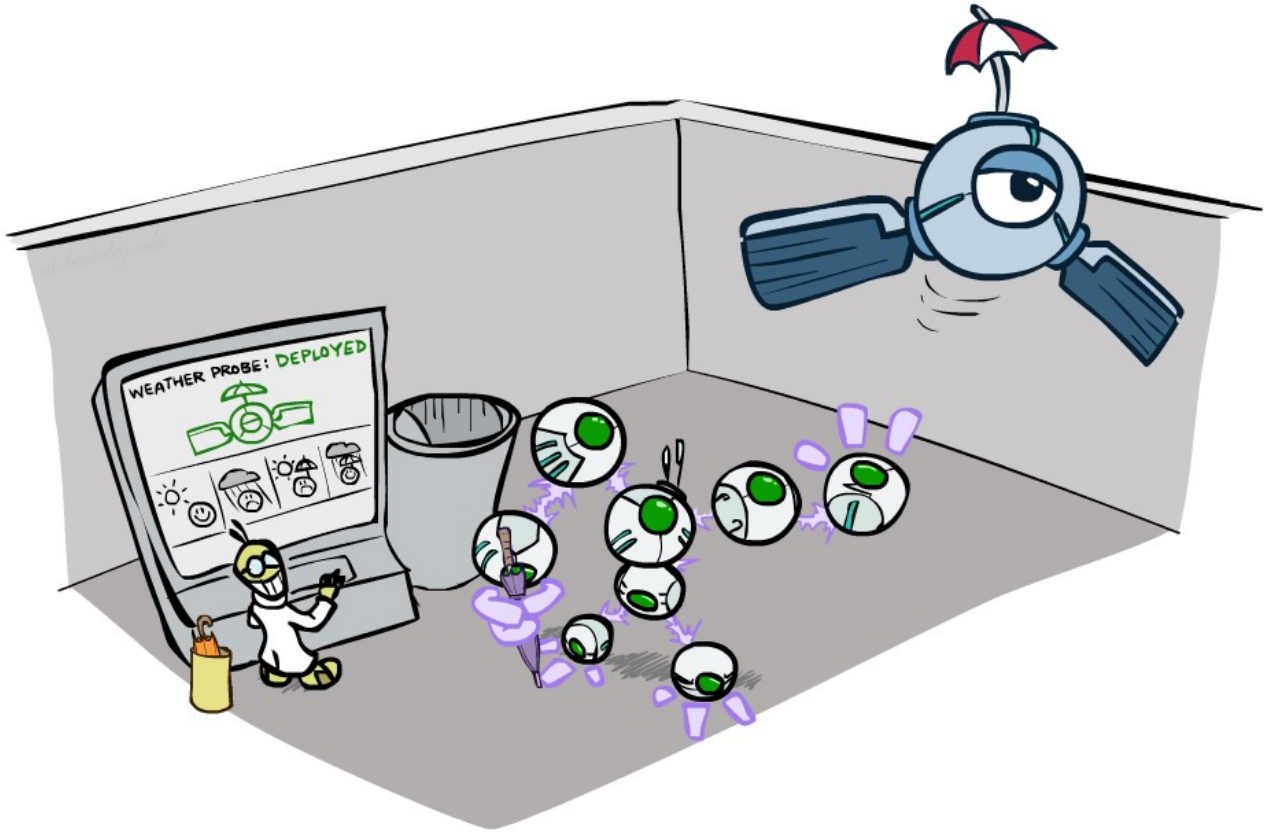
- Samples soon begin to reflect all the evidence in the network
- Eventually samples will be drawn from the true posterior, under moderate conditions:
  - Intuitively, all variables should eventually be resampled for infinite number of times
  - -> Gibbs sampling is *consistent*!

Gibbs sampling in practice

- Gibbs sampling belongs to a family of sampling methods called Markov chain Monte Carlo (MCMC)
  - Specifically, it is a special case of a subset of MCMC methods called Metropolis-Hastings
- The most commonly used method for large Bayes nets
- Can be *compiled* to run very fast
  - Eliminate all data structure references, just multiply and sample
  - ~100 million samples per second on a laptop
- Parallelization (one processor per variable)
- Many cognitive scientists suggest the brain runs on MCMC

CS 3317: Artificial Intelligence

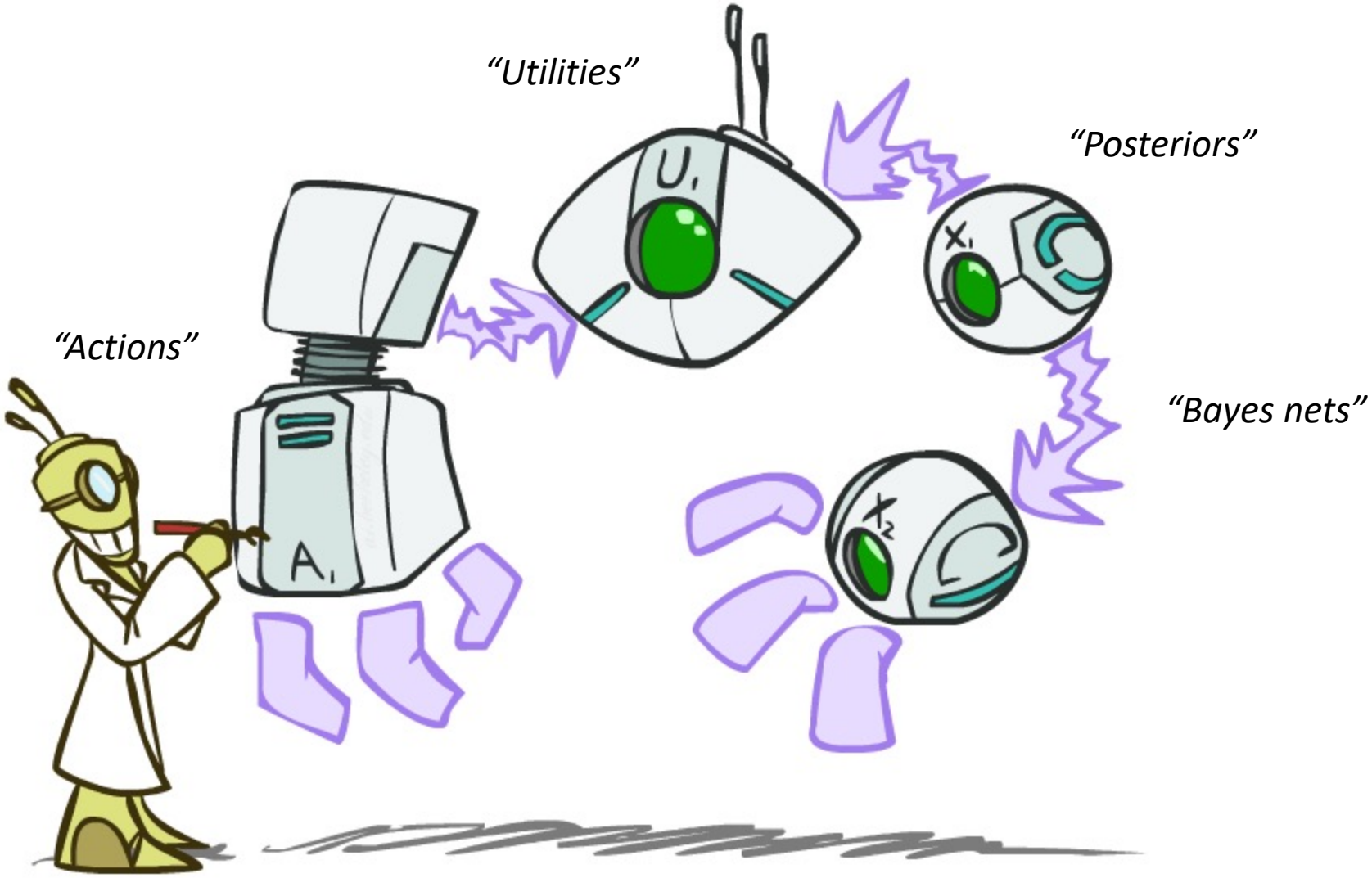
Decision Networks and Value of Information



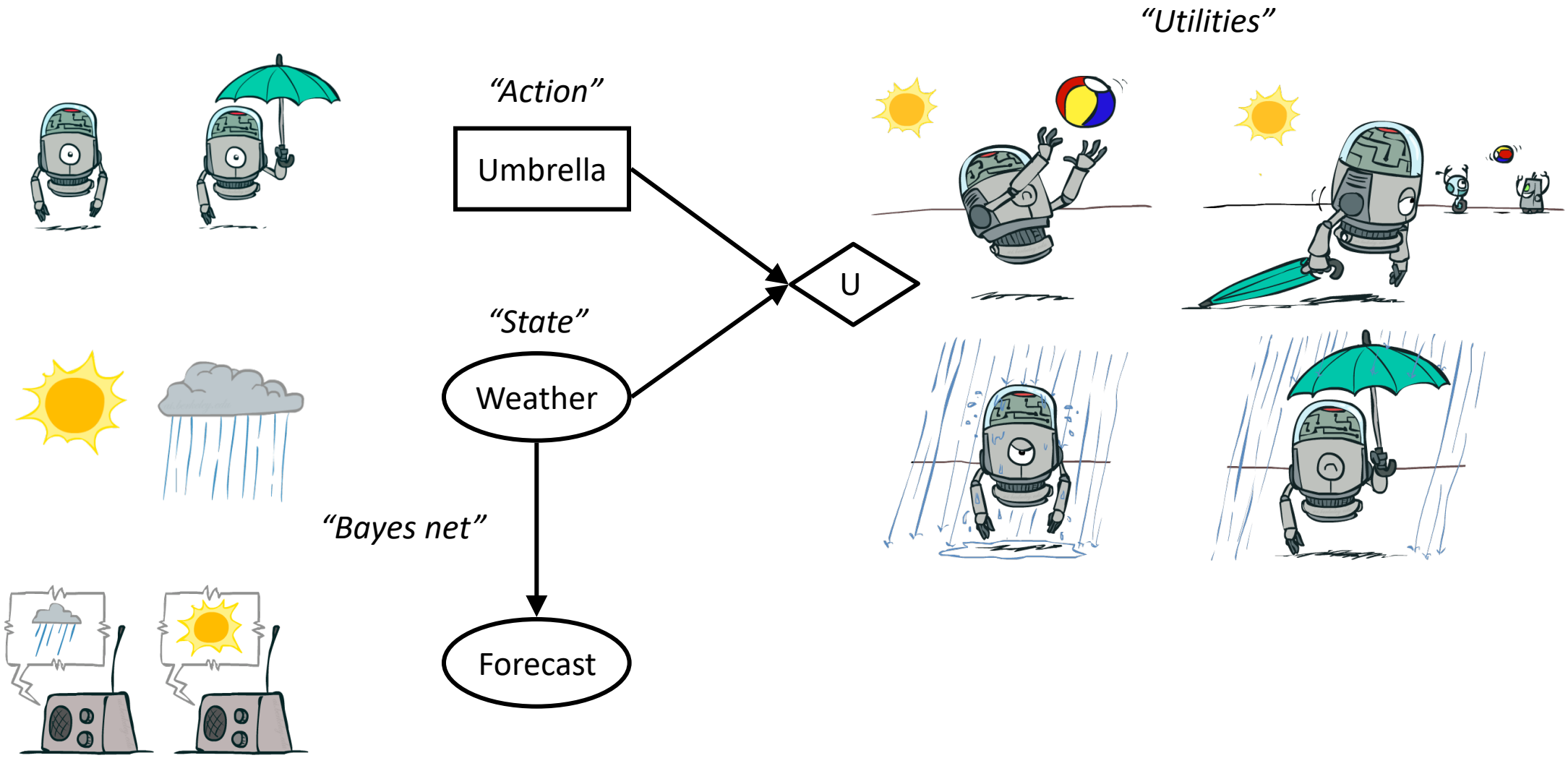
Instructor: Panpan Cai

[Slides adapted from UC Berkeley CS188]

Decision Networks

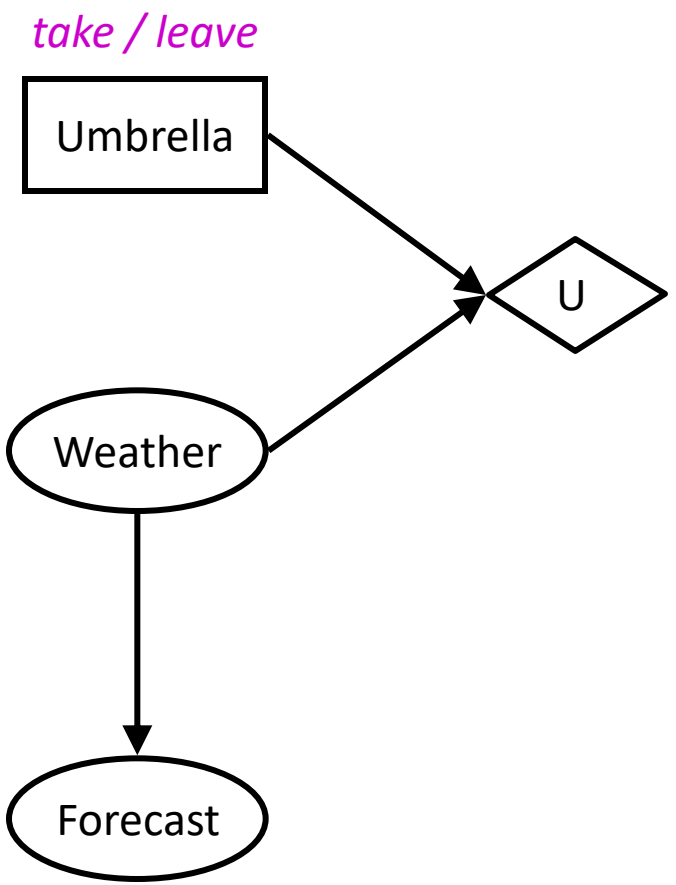


Decision Networks



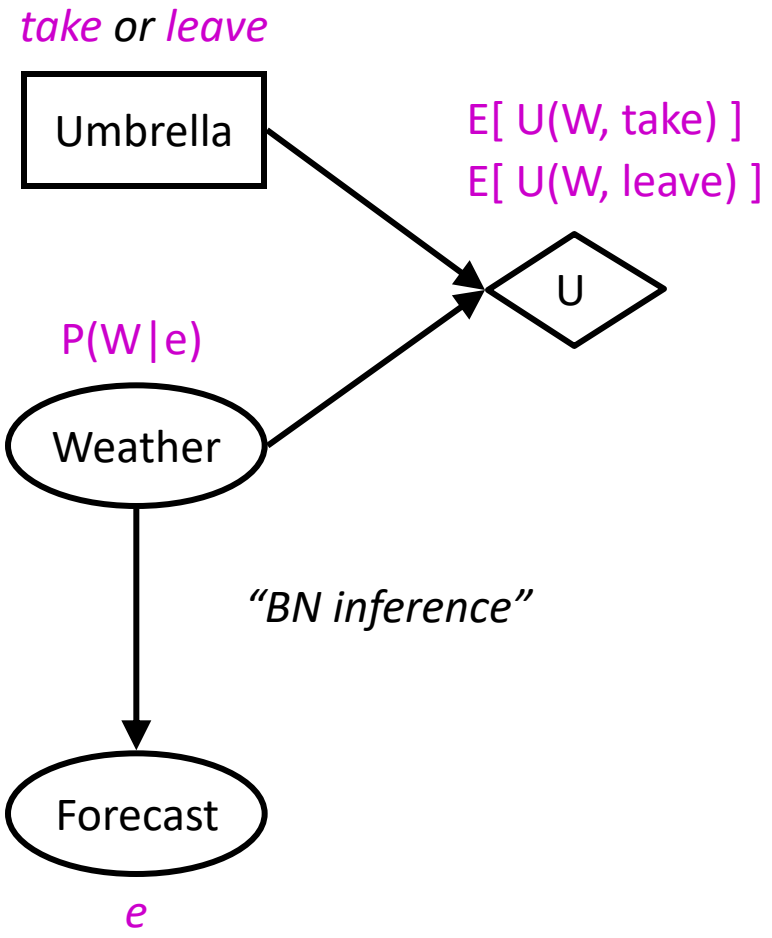
Decision Networks

- **MEU: choose the action which maximizes the expected utility given the evidence**
- Using decision networks:
  - Bayes nets with nodes for utility and actions
  - Lets us calculate the expected utility for each action
- New node types:
  - Chance nodes (just like BNs)
  - Actions (rectangles, *cannot* have parents, act as observed evidence)
  - Utility node (diamond, depends on action and chance nodes)



Decision Networks

- Action selection
  - Instantiate all evidence
  - Set action node(s) each possible way
  - Calculate posterior for all parents of utility node, given the evidence
  - Calculate expected utility for each action
  - Choose maximizing action



Example: Umbrella Decision Network

Umbrella = leave

$$\mathrm{EU}(\text{leave}) = \sum_w P(w)U(\text{leave}, w)$$

$$= 0.7 \cdot 100 + 0.3 \cdot 0 = 70$$

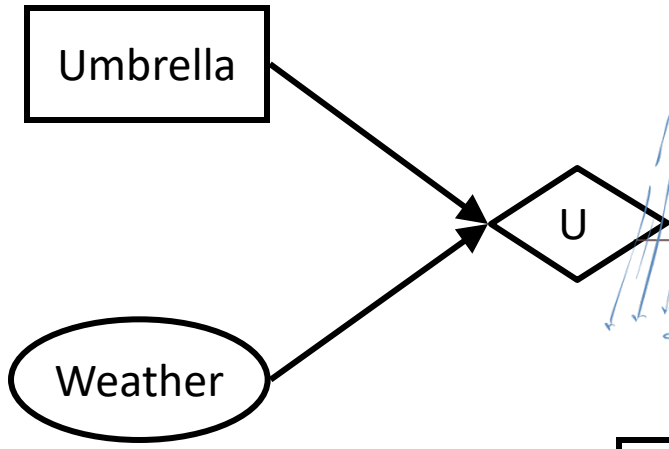
Umbrella = take

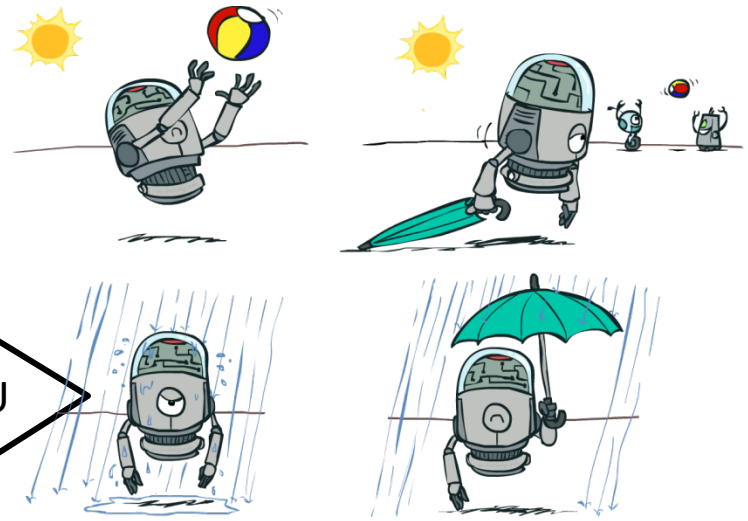
$$\mathrm{EU}(\text{take}) = \sum_w P(w)U(\text{take}, w)$$

$$= 0.7 \cdot 20 + 0.3 \cdot 70 = 35$$

Optimal decision = leave

$$\mathrm{MEU}(\emptyset) = \max_a \mathrm{EU}(a) = 70$$





W	P(W)
sun	0.7
rain	0.3

A	W	U(A,W)
leave	sun	100
leave	rain	0
take	sun	20
take	rain	70

Decision Networks: Notation

Umbrella = leave

$$\text{EU}(\text{leave}) = \sum_w P(w)U(\text{leave}, w)$$

$$= 0.7 \cdot 100 + 0.3 \cdot 0 = 70$$

Umbrella = take

$$\text{EU}(\text{take}) = \sum_w P(w)U(\text{take}, w)$$

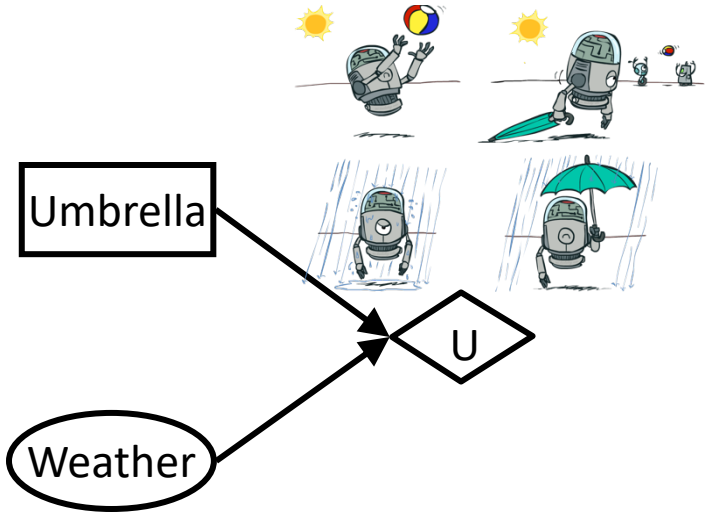
$$= 0.7 \cdot 20 + 0.3 \cdot 70 = 35$$

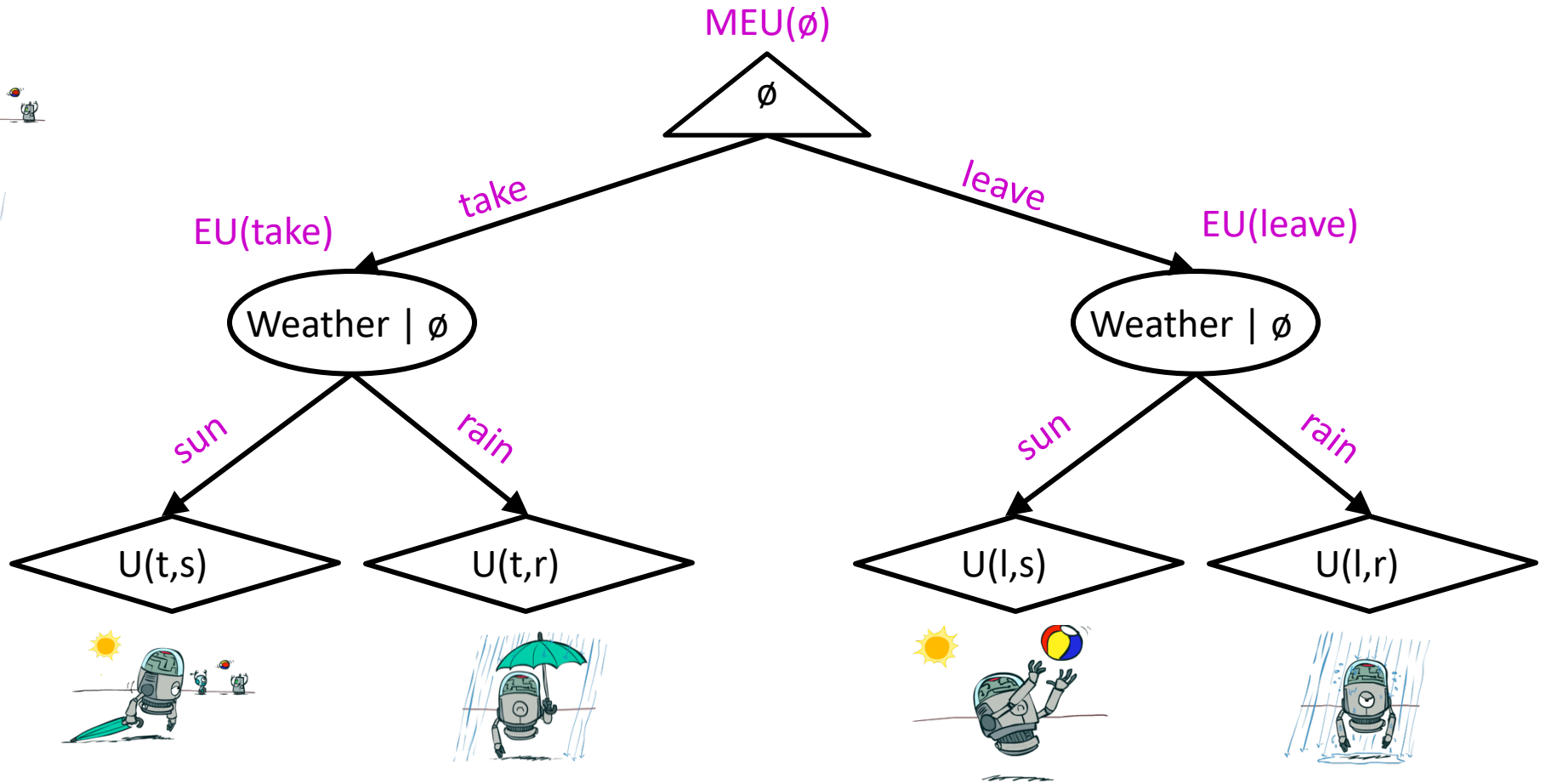
Optimal decision = leave

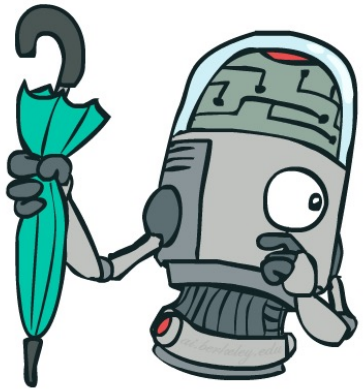
$$\text{MEU}(\text{ø}) = \max_a \text{EU}(a) = 70$$

- EU(leave) = Expected Utility of taking action leave
  - In the parentheses, we write an action
  - Calculating EU requires taking an expectation over chance node outcomes
- MEU(ø) = Maximum Expected Utility, given no information
  - In the parentheses, we write the evidence (which nodes we know)
  - Calculating MEU requires taking a maximum over several expectations (one EU per action)

Decisions as Outcome Trees







- Almost exactly like expectimax / MDPs
- What's changed?

Umbrella Decision Network (Evidence)

Umbrella = leave

$$\mathrm{EU}(\text{leave}|\text{bad}) = \sum_w P(w|\text{bad})U(\text{leave}, w)$$

$$= 0.34 \cdot 100 + 0.66 \cdot 0 = 34$$

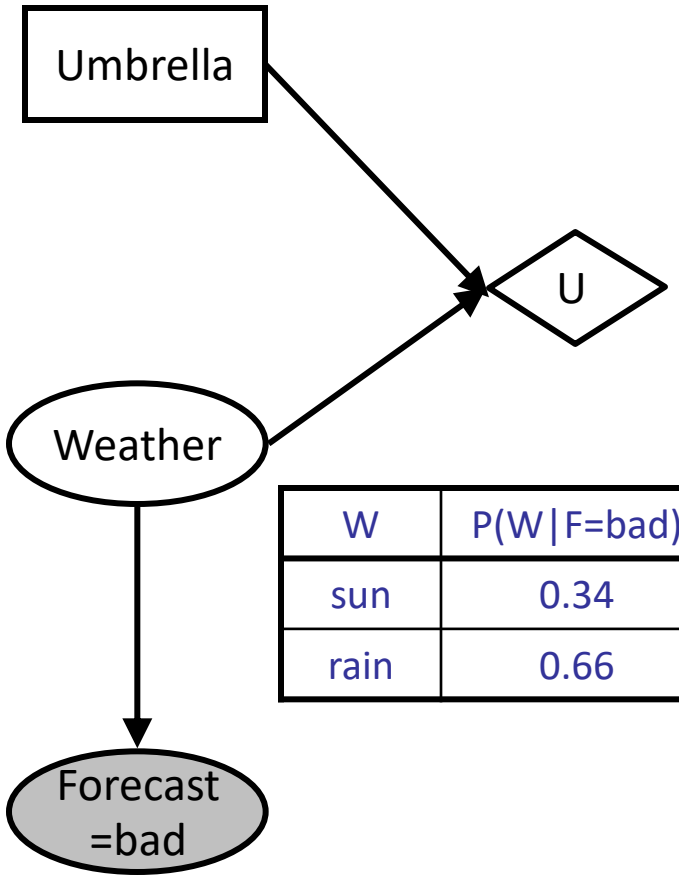
Umbrella = take

$$\mathrm{EU}(\text{take}|\text{bad}) = \sum_w P(w|\text{bad})U(\text{take}, w)$$

$$= 0.34 \cdot 20 + 0.66 \cdot 70 = 53$$

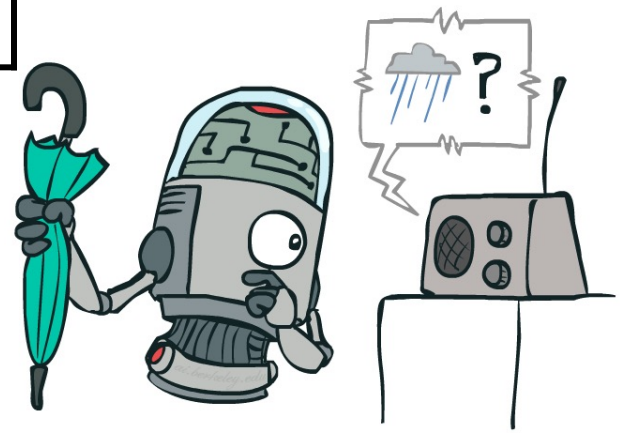
Optimal decision = take

$$\mathrm{MEU}(F = \text{bad}) = \max_a \mathrm{EU}(a|\text{bad}) = 53$$



A	W	U(A,W)
leave	sun	100
leave	rain	0
take	sun	20
take	rain	70

W	P(W|F=bad)
sun	0.34
rain	0.66



Decision Networks: Notation

Umbrella = leave

$$\mathrm{EU}(\text{leave}|\text{bad}) = \sum_w P(w|\text{bad})U(\text{leave}, w)$$

$$= 0.34 \cdot 100 + 0.66 \cdot 0 = 34$$

Umbrella = take

$$\mathrm{EU}(\text{take}|\text{bad}) = \sum_w P(w|\text{bad})U(\text{take}, w)$$

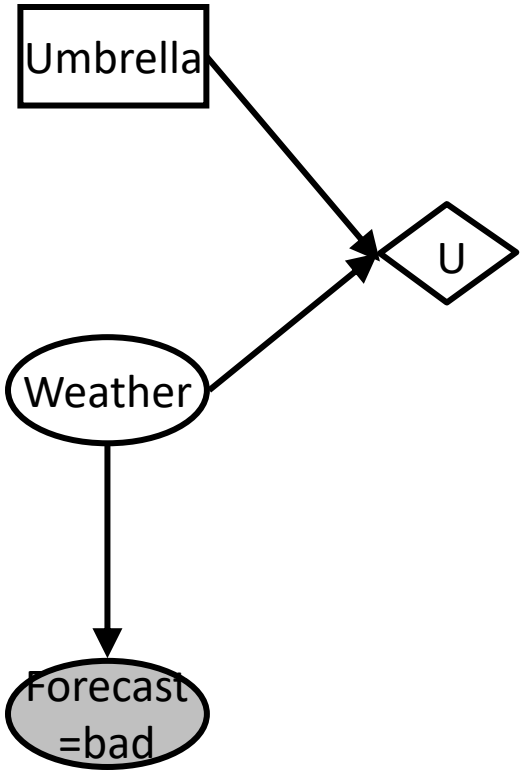
$$= 0.34 \cdot 20 + 0.66 \cdot 70 = 53$$

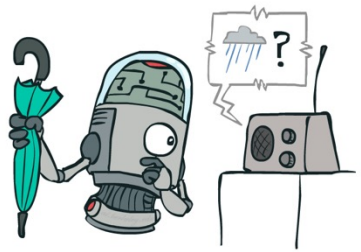
Optimal decision = take

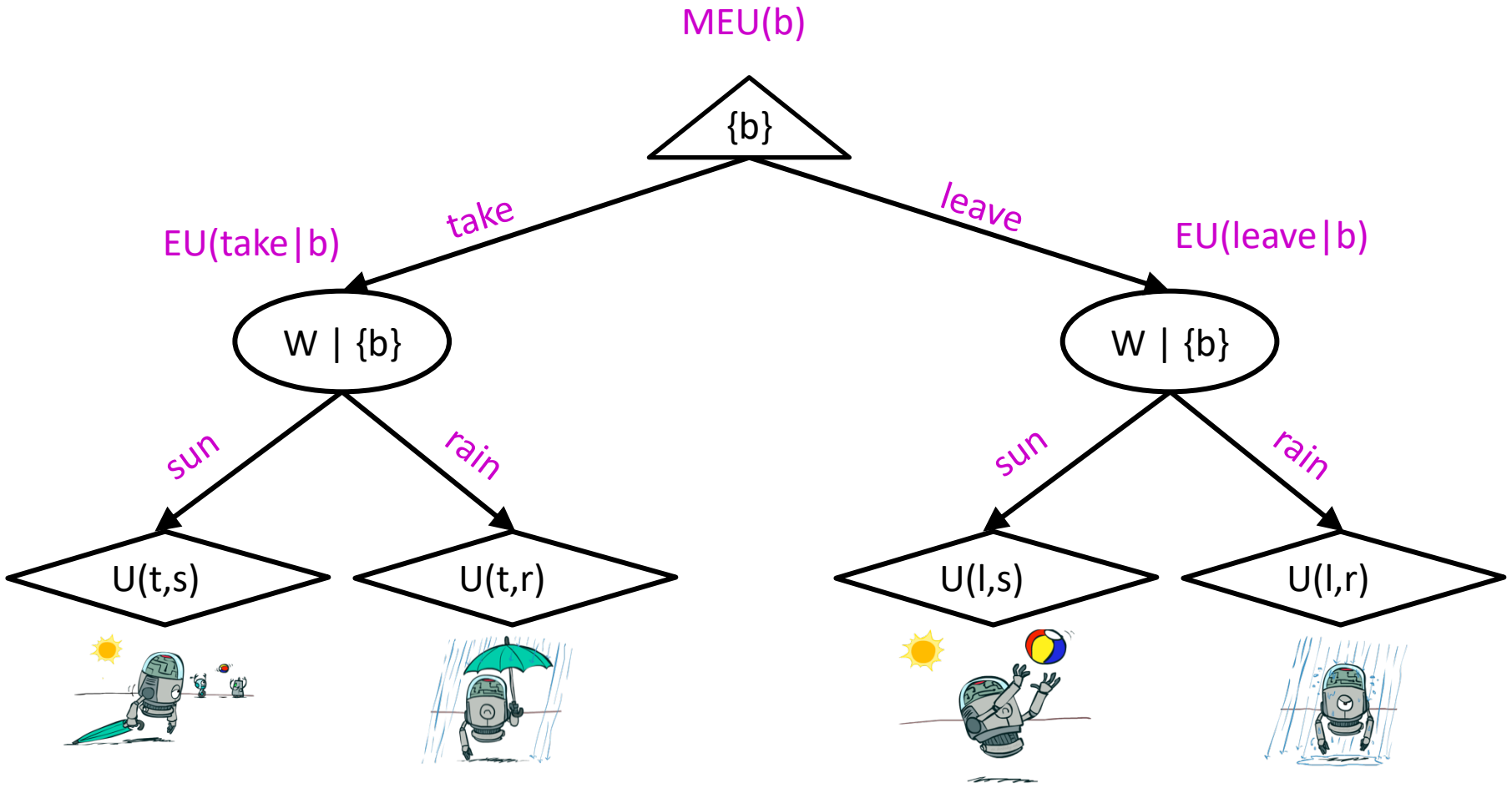
$$\mathrm{MEU}(F = \text{bad}) = \max_a \mathrm{EU}(a|\text{bad}) = 53$$

- EU(leave|bad) = Expected Utility of taking action leave, given you know the forecast is bad
  - Left side of conditioning bar: Action being taken
  - Right side of conditioning bar: The random variable(s) we know the value of (evidence)
- MEU(F=bad) = Maximum Expected Utility, given you know the forecast is bad
  - In the parentheses, we write the evidence (which nodes we know)
- Remind you of anything?

Decisions as Outcome Trees (Evidence)

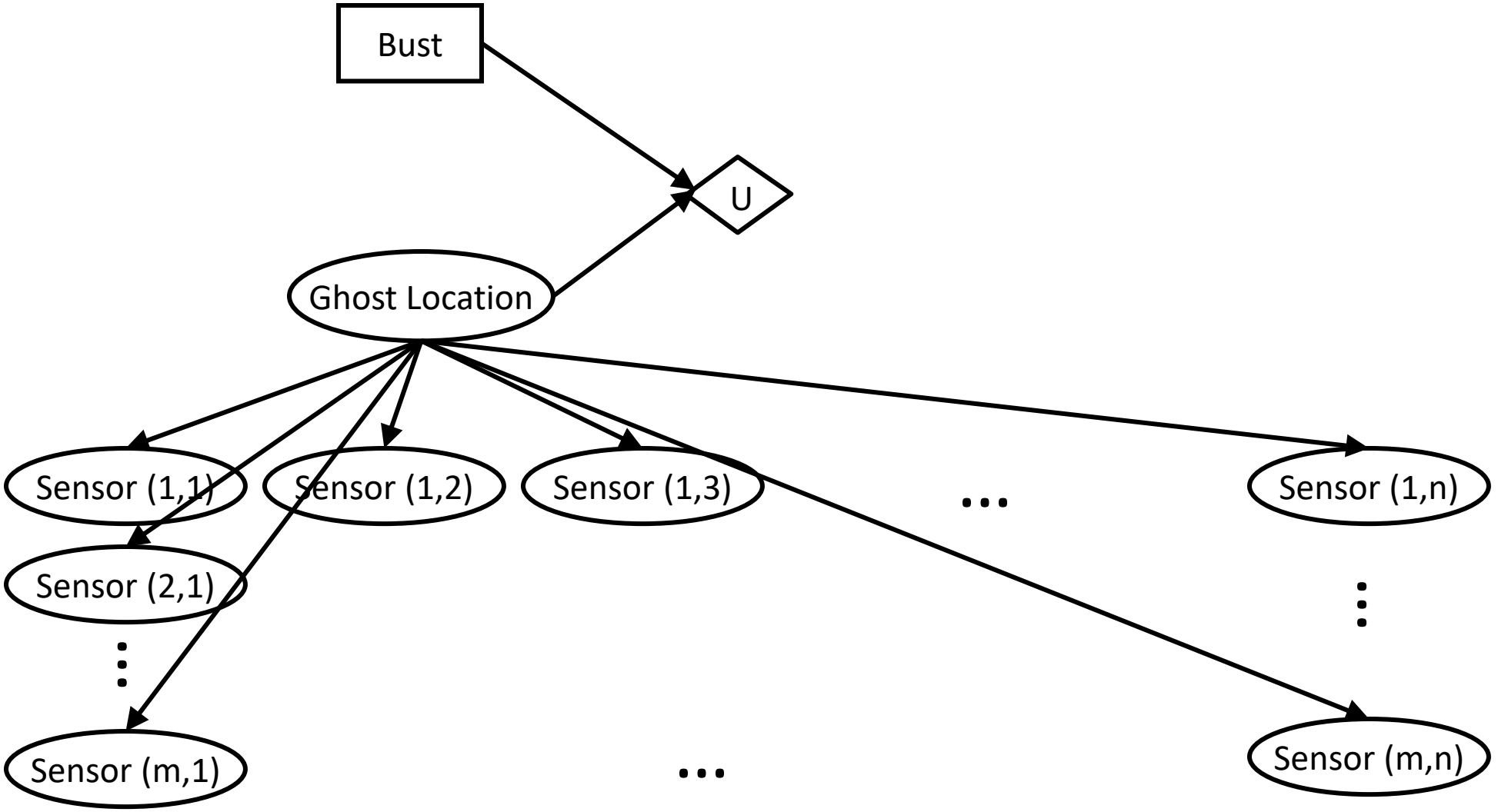




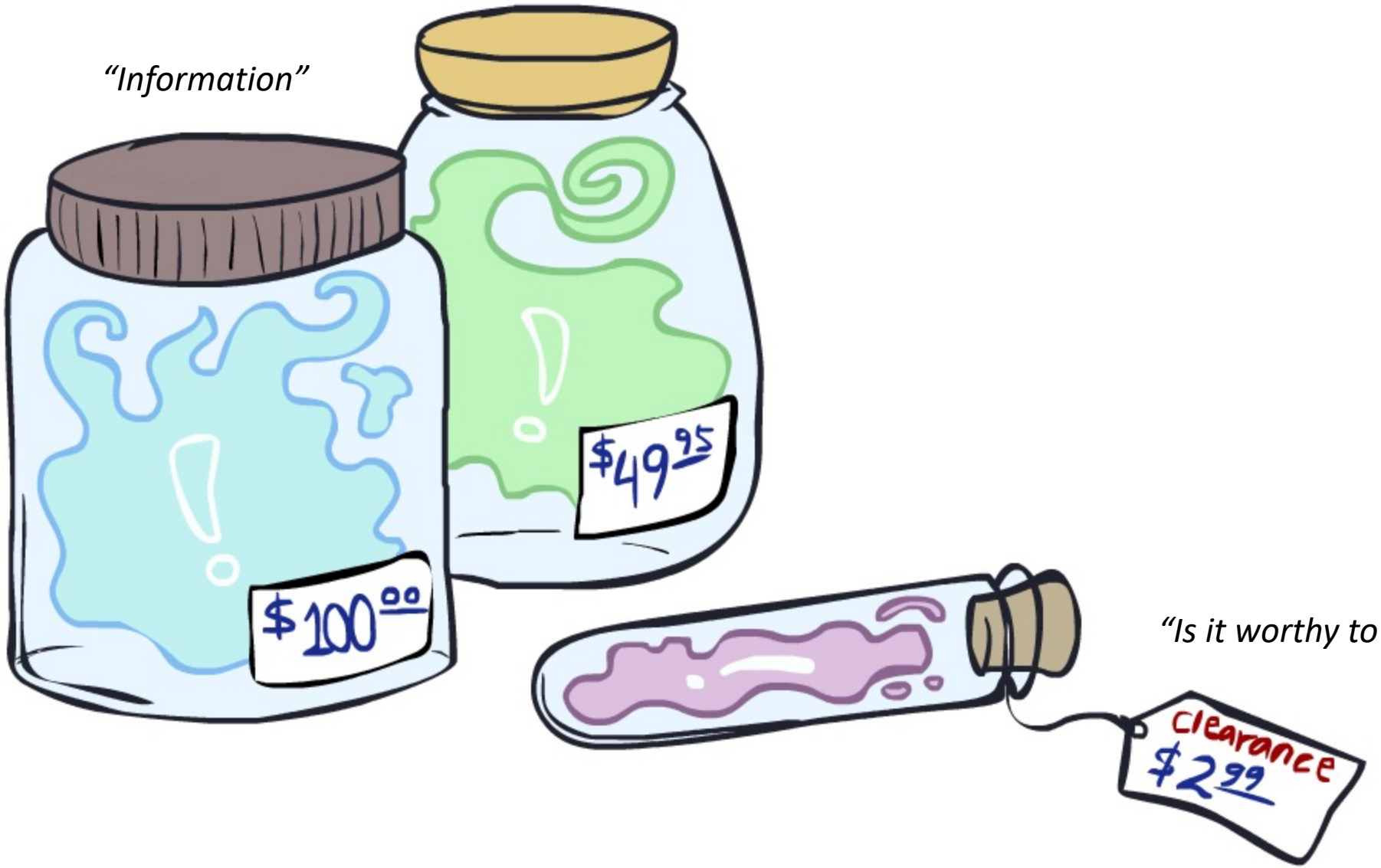


Video of Demo Ghostbusters with Probability

Ghostbusters Decision Network

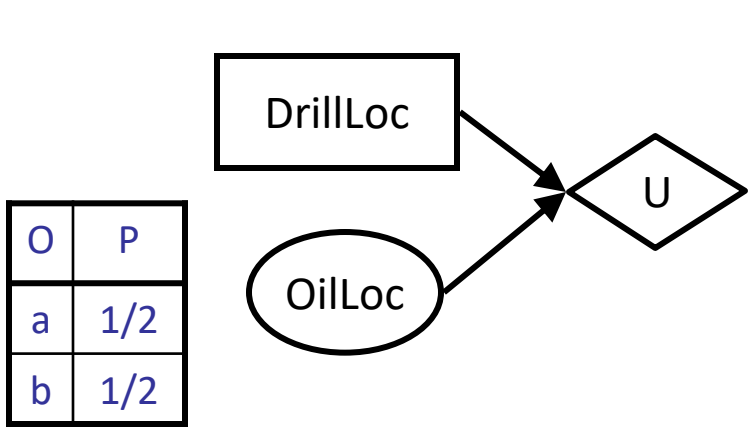


Value of Information



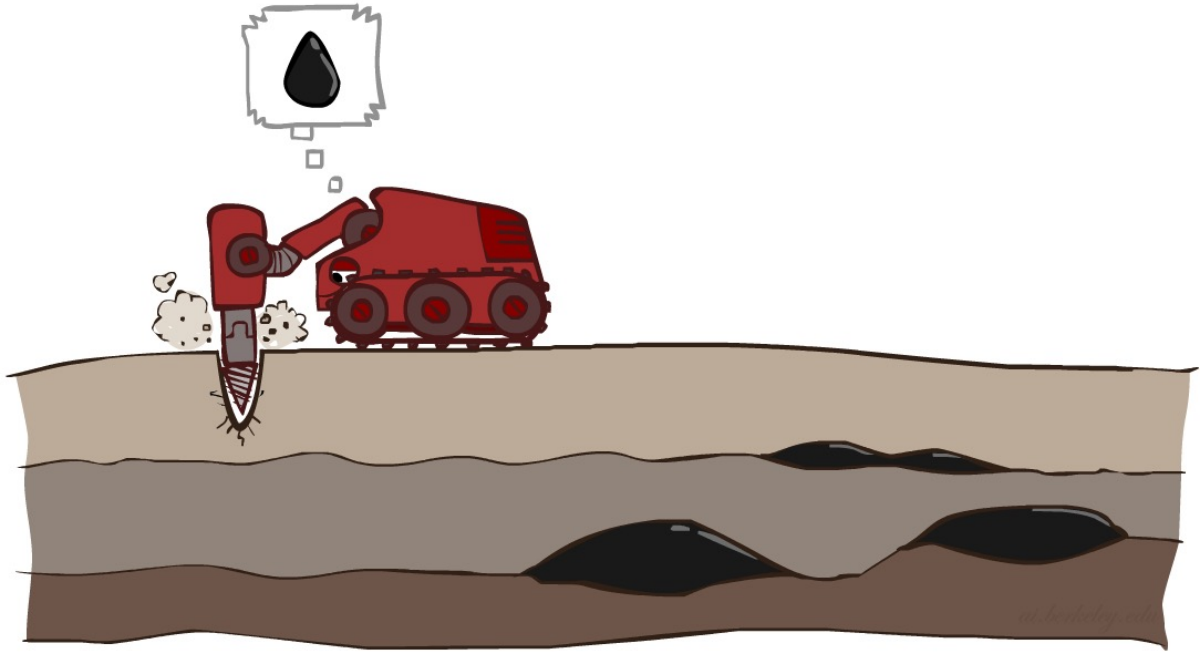
Value of Information

- Idea: compute value of acquiring evidence
  - Can be done directly from decision network
- Example: buying oil drilling rights
  - Two blocks A and B, exactly one has oil, worth k
  - You can drill in one location
  - Prior probabilities 0.5 each, & mutually exclusive
  - Drilling in either A or B has EU = k/2, MEU = k/2
- Question: what's the value of information of O?
  - Value of knowing which of A or B has oil from a survey
  - Value is expected gain in MEU from new info
  - Survey may say "oil in a" or "oil in b", prob 0.5 each
  - If we know OilLoc, MEU is k (either way)
  - Gain in MEU from knowing OilLoc?
  - VPI(OilLoc) = k/2
  - Fair price of information: k/2



O	P
a	1/2
b	1/2

D	O	U
a	a	k
a	b	0
b	a	0
b	b	k



VPI Example: Weather

MEU with no evidence

$$\mathrm{MEU}(\emptyset) = \max_a \mathrm{EU}(a) = 70$$

MEU if forecast is bad

$$\mathrm{MEU}(F = \mathrm{bad}) = \max_a \mathrm{EU}(a|\mathrm{bad}) = 53$$

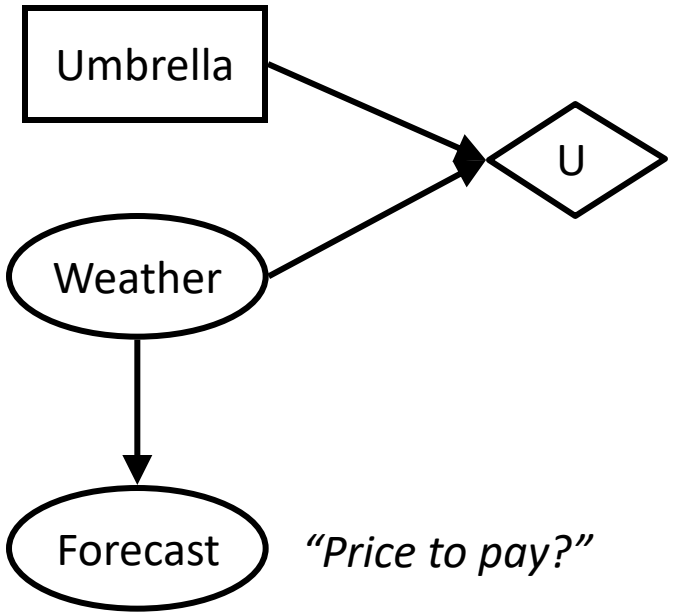
MEU if forecast is good

$$\mathrm{MEU}(F = \mathrm{good}) = \max_a \mathrm{EU}(a|\mathrm{good}) = 95$$

Forecast distribution

F	P(F)
good	0.59
bad	0.41

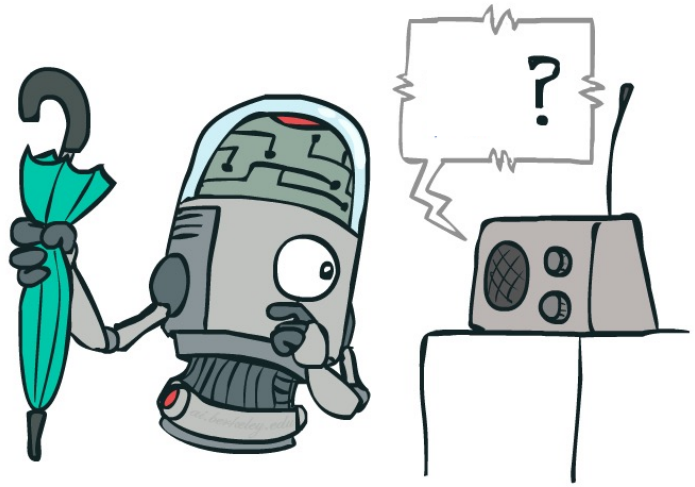
$$0.59 \cdot (95) + 0.41 \cdot (53) - 70$$
$$77.8 - 70 = 7.8$$



"Price to pay?"

A	W	U
leave	sun	100
leave	rain	0
take	sun	20
take	rain	70

$$\mathrm{VPI}(E'|e) = \left(\sum_{e'} P(e'|e)\mathrm{MEU}(e, e')\right) - \mathrm{MEU}(e)$$



Value of Information (Breakdown)

- Assume we have evidence E=e. MEU if we act now:

$$\text{MEU}(e) = \max_a \sum_s P(s|e)\, U(s,a)$$

- Assume we see that E' = e'. MEU if we act then:

$$\text{MEU}(e,e') = \max_a \sum_s P(s|e,e')\, U(s,a)$$

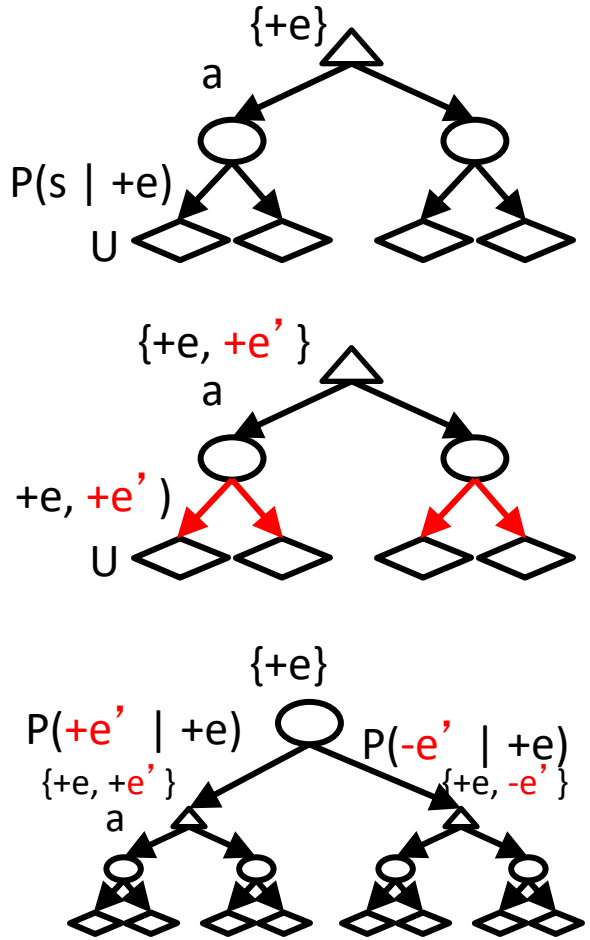
- BUT E' is a random variable whose value is *unknown*, so we don't know what e' will be

- Expected MEU if E' is revealed and then we act:

$$\text{MEU}(e,E') = \sum_{e'} P(e'|e)\text{MEU}(e,e')$$

- Value of information: how much MEU goes up by revealing E' first then acting, over acting now:

$$\text{VPI}(E'|e) = \text{MEU}(e,E') - \text{MEU}(e)$$



VPI: Notation

- MEU(e) = Maximum Expected Utility, given evidence E=e
  - In the parentheses, we write the evidence (which nodes we know)
  - Calculating MEU requires taking a maximum over several expectations (one EU per action)
- VPI(E'|e) = Expected gain in utility for knowing the value of E', given that I already know the value of e so far
  - Left side of conditioning bar: The random variable(s) we want to know the value of revealing
  - Right side of conditioning bar: The random variable(s) we already know the value of
  - Calculating VPI requires taking an expectation over several MEUs (one MEU per possible outcome of E', because we don't know the value of E')

$$\text{MEU}(e) = \max_a \sum_s P(s|e)\, U(s,a)$$

$$\text{MEU}(e,e') = \max_a \sum_s P(s|e,e')\, U(s,a)$$

$$\text{VPI}(E'|e) = \left(\sum_{e'} P(e'|e)\text{MEU}(e,e')\right) - \text{MEU}(e)$$

VPI: Computation Workflow

$\text{MEU}(e, E')$ $\text{MEU}(e, E') = \sum_{e'} P(e'|e)\text{MEU}(e, e')$ $\text{MEU}(e, e') = \max_a \text{EU}(a)$

$- \text{MEU}(e)$

$\text{MEU}(e) = \max_a \text{EU}(a)$

$= \text{VPI}(E'|e)$

VPI Properties

- Nonnegative

$$\forall E', e : \text{VPI}(E'|e) \geq 0$$

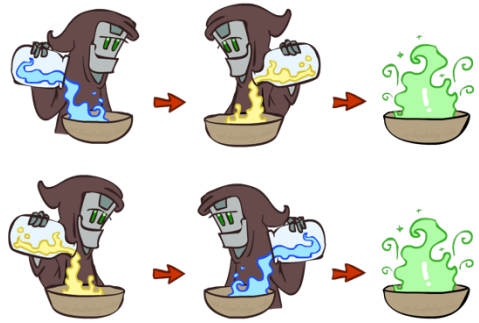
- Nonadditive

(think of observing E_j twice)

$$\text{VPI}(E_j, E_k|e) \neq \text{VPI}(E_j|e) + \text{VPI}(E_k|e)$$

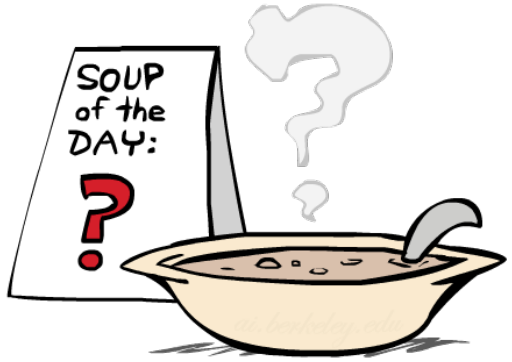
- Order-independent

$$\text{VPI}(E_j, E_k|e) = \text{VPI}(E_j|e) + \text{VPI}(E_k|e, E_j)$$
$$= \text{VPI}(E_k|e) + \text{VPI}(E_j|e, E_k)$$

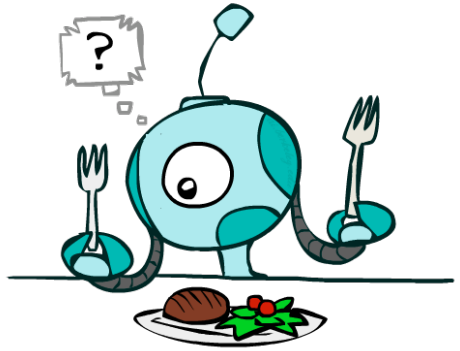


Quick VPI Questions

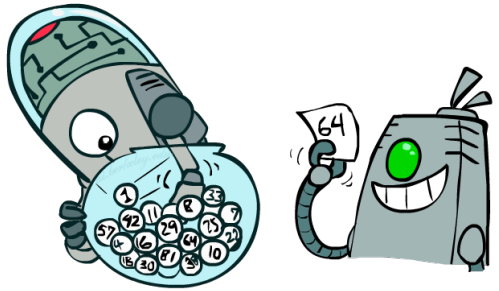
- The soup of the day is either clam chowder or split pea, but you wouldn't order either one. What's the value of knowing which it is?



- There are two kinds of plastic forks at a picnic. One kind is slightly sturdier. What's the value of knowing which?



- You're playing the lottery. The prize will be $0 or $100. You can play any number between 1 and 100 (chance of winning is 1%). What is the value of knowing the winning number?

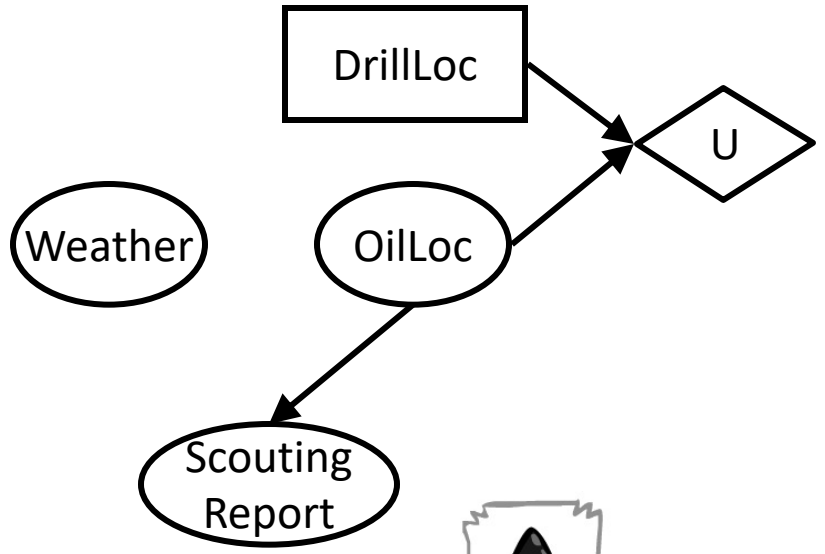


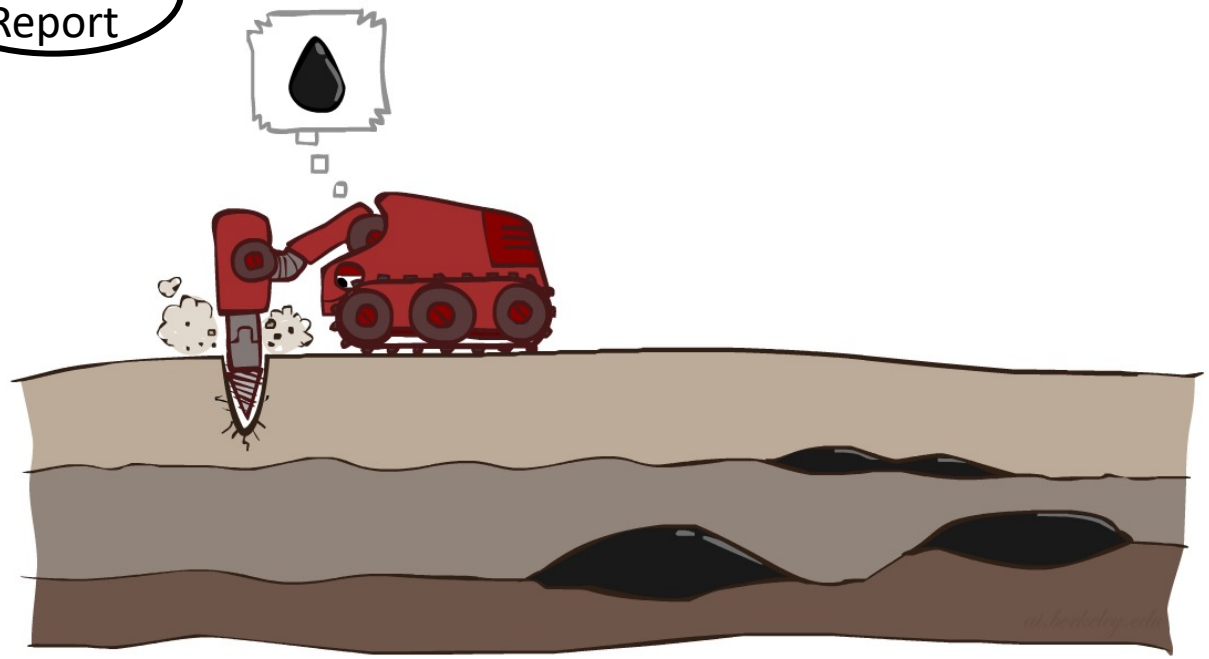
Value of Imperfect Information?

- No such thing (as we formulate it)
- Information corresponds to the observation of a node in the decision network
- If data is “noisy” that just means we don’t observe the original variable, but another variable which is a noisy version of the original one

VPI Question: Oil Net

- VPI(OilLoc) ?
- VPI(ScoutingReport) ?
- VPI(Weather) ?
- VPI(Weather | ScoutingReport) ?





- Generally:

  If Parents(U) $\perp\!\!\!\perp$ Z | CurrentEvidence

  Then VPI(Z | CurrentEvidence) = 0

Next Time: Hidden Markov Models

MCTS Pseudo-code (MDP)

Algorithm 4.9 Monte Carlo tree search

```
function SelectAction(s, d)                    s: state; d: remaining steps of episode
    loop
        Simulate(s, d, π0)                     e.g., try 1000 simulations to estimate Q(s,a)
    return arg max_a Q(s,a)
function Simulate(s, d, π0)
    if d = 0                                   Termination of a simulation (episode length exhausted, terminal state)
        return 0
    if s ∉ T                                   If s is not in the tree (new node)
        for a ∈ A(s)
            (N(s,a), Q(s,a)) ← (N0(s,a), Q0(s,a))      Initialize quantities in the new node
        T = T ∪ {s}
        return Rollout(s, d, π0)               Add to the tree, then rollout
    a ← arg max_{a∈A(s)} [ Q(s,a) + c sqrt(log N(s) / N(s,a)) ]      Select action branch to traverse (for both new/existing nodes)
    (s', r) ~ G(s,a)                           Sample transition and reward (G: step function)
    q ← r + γ Simulate(s', d − 1, π0)          Forward recursion and approximate Q-update
    N(s,a) ← N(s,a) + 1
    Q(s,a) ← Q(s,a) + (q − Q(s,a)) / N(s,a)    Backup: updating quantities in the node (running average)
    return q                                   Return the outcome of the current simulation
```

MCTS Pseudo-code (MDP)

Algorithm 4.10 Rollout evaluation

```
function ROLLOUT(s, d, π_0)
    if d = 0                                  Termination of rollout (length exhausted or terminal state)
        return 0
    a ~ π_0(s)                                Sample action from rollout policy
    (s', r) ~ G(s, a)                         Sample transition and reward from step function
    return r + γ ROLLOUT(s', d − 1, π_0)      Forward recursion and back up (accumulate rewards)
```

MCTS Pseudo-code (MiniMax)

```
1:  function SELECT_ACTION(s)
2:      loop
3:          SIMULATE(s, π0, player)
4:      return arg max_s' U(s')

5:  function SIMULATE(s, π0, player)
6:      if player wins or draw
7:          return 1 (MAX) or -1 (MIN) or 0 (draw)
8:      if s not in T
9:          U(s)=U0(s), N(s)=N0(s)
10:         T = T ∪ {s}
11:         return ROLLOUT(s, π0, player)
12: s' ← random(unexplored successors) if s is leaf
13: s' ← arg max_s' [ U(s') + c * sqrt(log(N(s)) / N(s')) ] if MAX player
14: s' ← arg min_s' [ U(s') - c * sqrt(log(N(s)) / N(s')) ] if MIN player
15: r ← SIMULATE(s', π0, -player)
16: N(s) ← N(s) + 1
17: U(s) ← U(s) + (r - U(s)) / N(s)
18: return r
```

s: state of the board

e.g., try 1000 simulations to estimate U(s'), player = MAX

Termination of a simulation (win or draw)

If s is not in the tree (new node)

Initialize quantities in the new node

Add to the tree, then rollout

Select next state to traverse (choose 1 case)

Forward recursion and update utility estimate

Backup: updating quantities in the node (running average)

Return the outcome of the current simulation

MCTS Pseudo-code (MiniMax)

```
1:  function ROLLOUT(s, π0, player)
2:      if s is a terminal state
3:          return +1/-1/0
4:      s' ← π0(s)
6:      return ROLLOUT(s', π0, -player)
```

Termination of rollout

Sample plays from rollout policy

Forward recursion and back up (accumulate rewards)