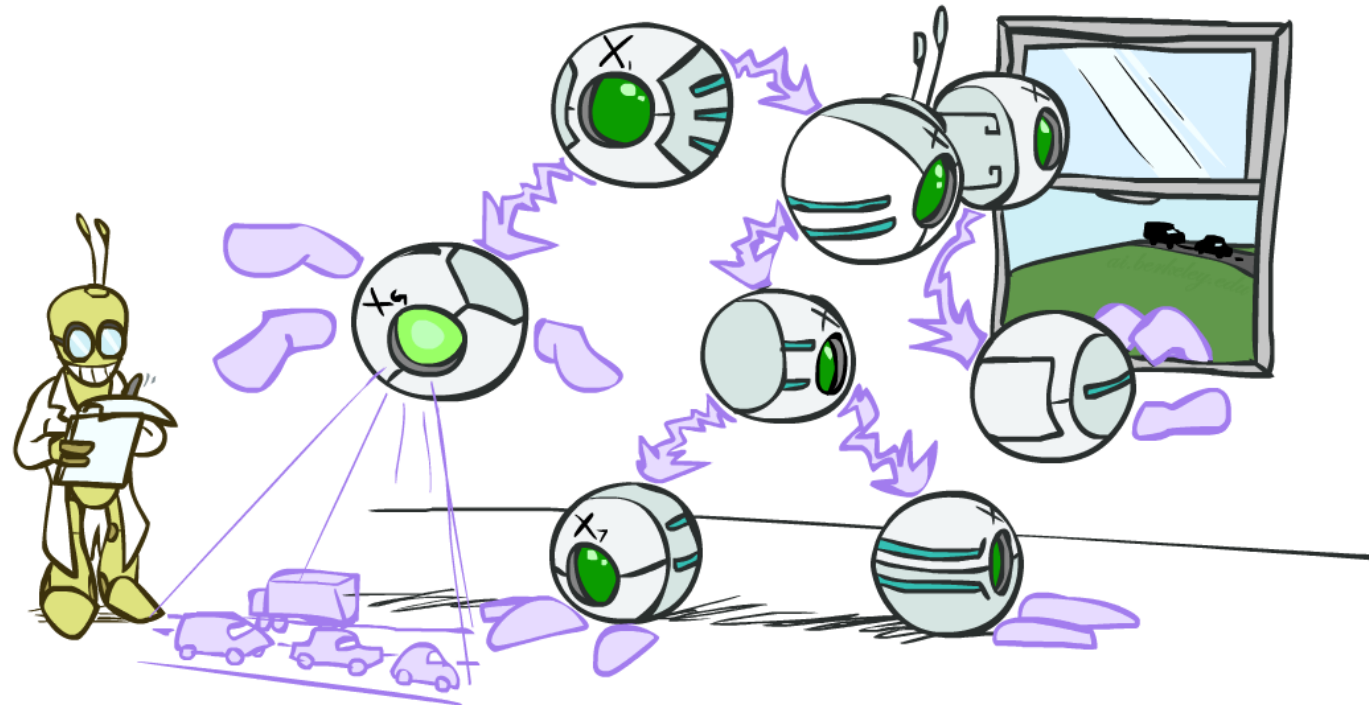


CSE 573: Artificial Intelligence

Bayes Nets: Inference



slides adapted from
Stuart Russel, Dan Klein, Pieter Abbeel from ai.berkeley.edu
And Hanna Hajishirzi, Jared Moore, Dan Weld

Bayes Nets

✓ Part I: Representation

✓ Part II: Independence

Part III: Exact inference

- Enumeration (always exponential complexity)
- Variable elimination (worst-case exponential complexity, often better)
- Inference is NP-hard in general

Part IV: Approximate Inference

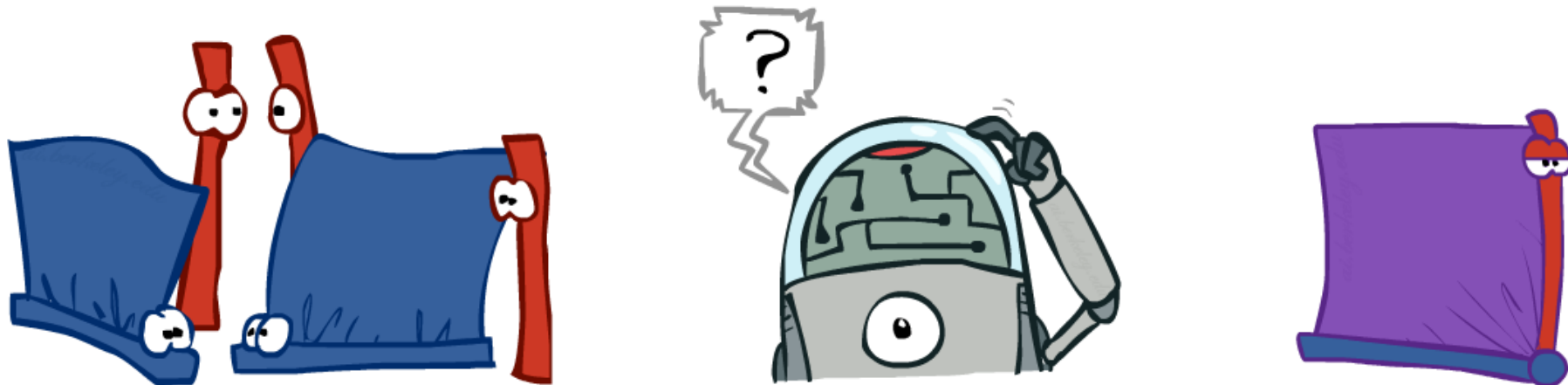
Probabilistic Inference

- Probabilistic inference: compute a desired probability from a probability model
 - Typically for a *query variable* given *evidence*
 - E.g., $P(\text{airport on time} \mid \text{no accidents}) = 0.90$
 - These represent the agent's *beliefs* given the evidence
- Probabilities change with new evidence:
 - $P(\text{airport on time} \mid \text{no accidents, 5 a.m.}) = 0.95$
 - $P(\text{airport on time} \mid \text{no accidents, 5 a.m., raining}) = 0.80$
 - Observing new evidence causes *beliefs to be updated*



Many Types of Inference

- Inference: calculating some useful quantity from a probability model (joint probability distribution)
- Examples:
 - Posterior marginal probability
 - $P(Q|e_1, \dots, e_k)$
 - E.g., what disease might I have?
 - Most likely explanation:
 - $\operatorname{argmax}_{q,r,s} P(Q=q, R=r, S=s | e_1, \dots, e_k)$
 - E.g., what did he say?



Inference by Enumeration

- General case:

- Evidence variables: $E_1, \dots, E_k = e_1, \dots, e_k$
- Query* variable: Q
- Hidden variables: H_1, \dots, H_r

} X_1, \dots, X_n
All variables

- We want:

$$P(Q \mid e_1, \dots, e_k)$$

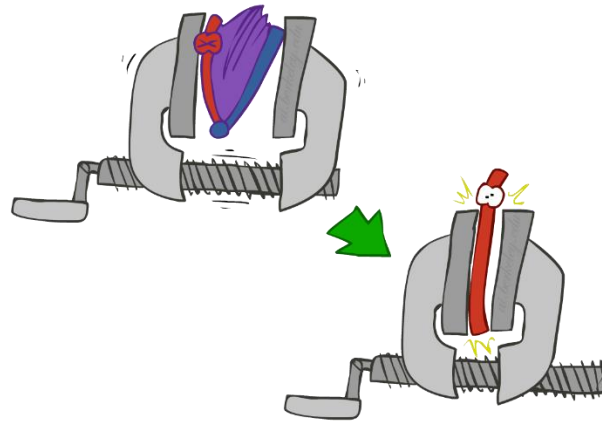
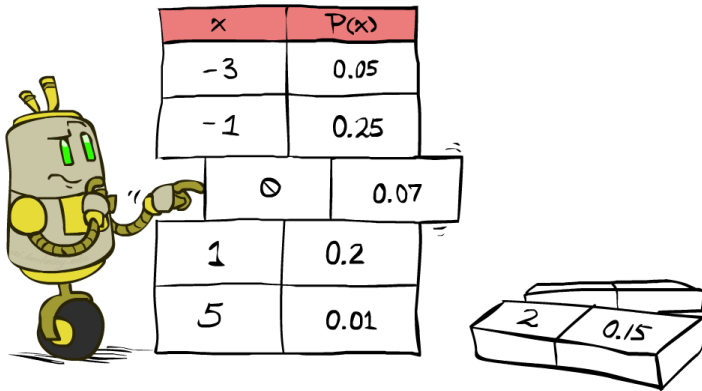
* Works fine with multiple query variables, too

Probability model $P(X_1, \dots, X_n)$ is given

- Step 1: Select the entries consistent with the evidence

- Step 2: Sum out H from model to get joint of Query and evidence

- Step 3: Normalize

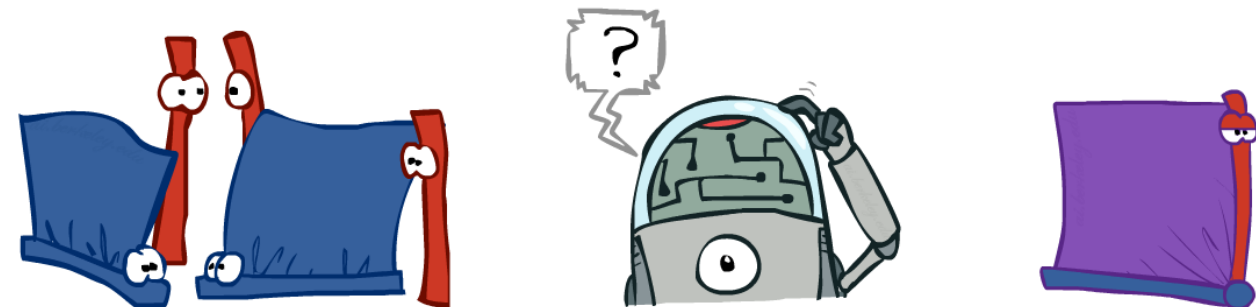
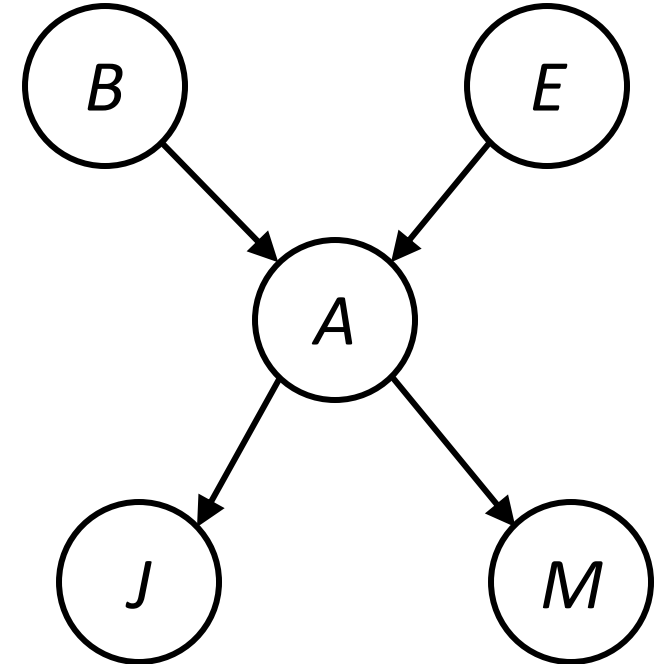


$$P(Q \mid e_1, \dots, e_k) = \alpha P(Q, e_1, \dots, e_k)$$

$$P(Q, e_1, \dots, e_k) = \sum_{h_1, \dots, h_r} \underbrace{P(Q, h_1, \dots, h_r, e_1, \dots, e_k)}_{X_1, \dots, X_n}$$

Inference by Enumeration in Bayes' Nets

- Reminder of inference by enumeration:
 - Any probability of interest can be computed by summing entries from the joint distribution: $P(Q | e) = \alpha \sum_h P(Q, h, e)$
 - Entries from the joint distribution can be obtained from a BN by multiplying the corresponding conditional probabilities
- $P(B | j, m) = \alpha \sum_{e,a} P(B, e, a, j, m)$
 $= \alpha \sum_{e,a} P(B) P(e) P(a | B, e) P(j | a) P(m | a)$
- So inference in Bayes nets means computing sums of products of numbers: sounds easy!
- Problem: sums of **exponentially many** products!



Can we do better?

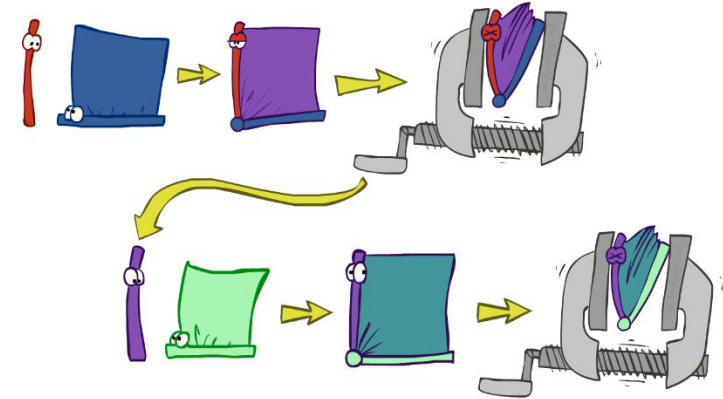
- Consider $uwy + uwz + uxy + uxz + vwy + vwz + vxy + vxz$
 - 16 multiplies, 7 adds
 - Lots of repeated subexpressions!
- Rewrite as $(u+v)(w+x)(y+z)$
 - 2 multiplies, 3 adds
- $\sum_{e,a} P(B) P(e) P(a | B, e) P(j | a) P(m | a)$
- $= P(B)P(e)P(a | B, e)P(j | a)P(m | a) + P(B)P(\neg e)P(a | B, \neg e)P(j | a)P(m | a)$
 $+ P(B)P(e)P(\neg a | B, e)P(j | \neg a)P(m | \neg a) + P(B)P(\neg e)P(\neg a | B, \neg e)P(j | \neg a)P(m | \neg a)$

Lots of repeated subexpressions!

Variable elimination: The basic ideas

- Move summations inwards as far as possible

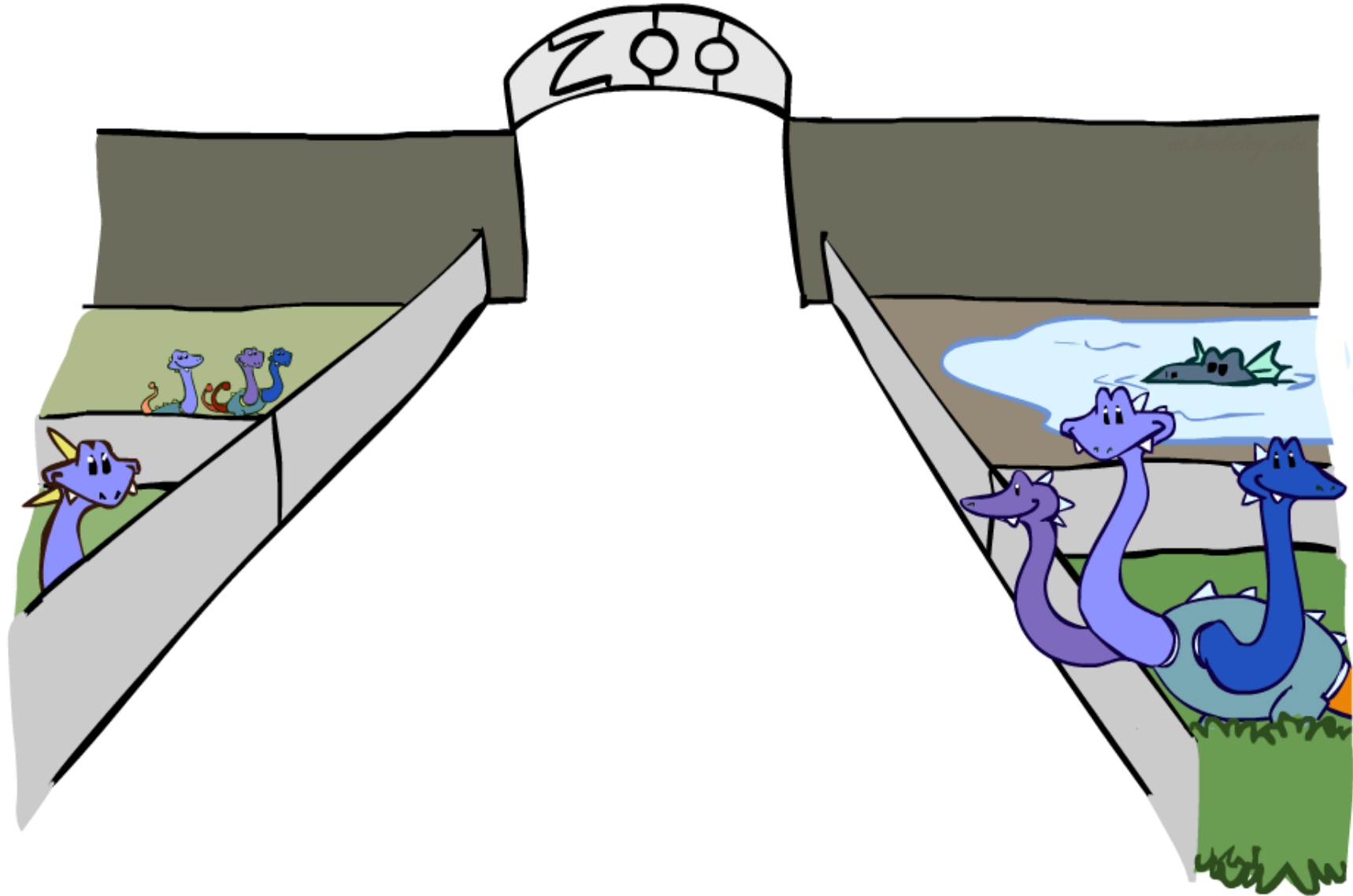
- $$P(B | j, m) = \alpha \sum_{e,a} P(B) P(e) P(a|B,e) P(j|a) P(m|a)$$
$$= \alpha P(B) \sum_e P(e) \sum_a P(a|B,e) P(j|a) P(m|a)$$



- Do the calculation from the inside out

- I.e., sum over a first, then sum over e
- Problem: $P(a|B,e)$ isn't a single number, it's a bunch of different numbers depending on the values of B and e
- Solution: use arrays of numbers (of various dimensions) with appropriate operations on them
 - These are called **factors**

Factor Zoo



Factor Zoo I

- Joint distribution: $P(X,Y)$

- Entries $P(x,y)$ for all x, y
- $|X| \times |Y|$ matrix
- Sums to 1

$P(A,J)$

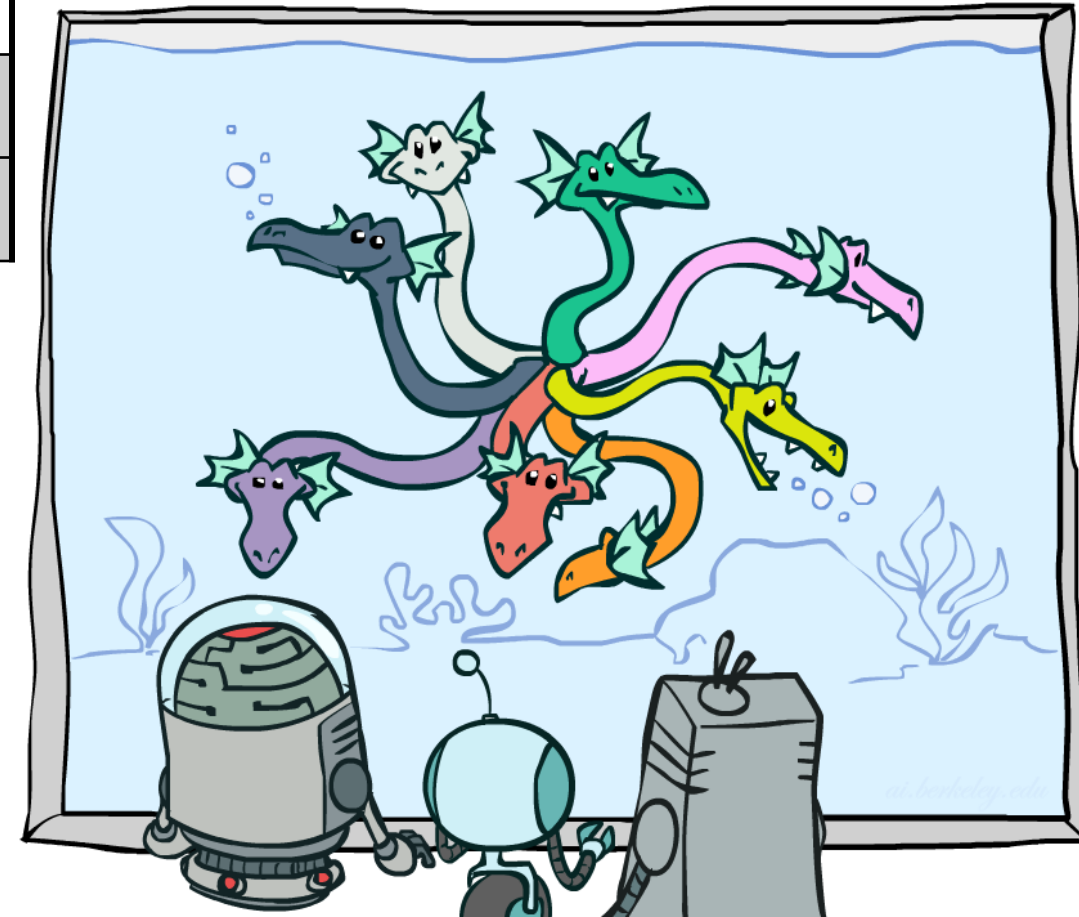
A \ J	true	false
true	0.09	0.01
false	0.045	0.855

- Projected joint: $P(x,Y)$

- A slice of the joint distribution
- Entries $P(x,y)$ for one x , all y
- $|Y|$ -element vector
- Sums to $P(x)$

$P(a,J)$

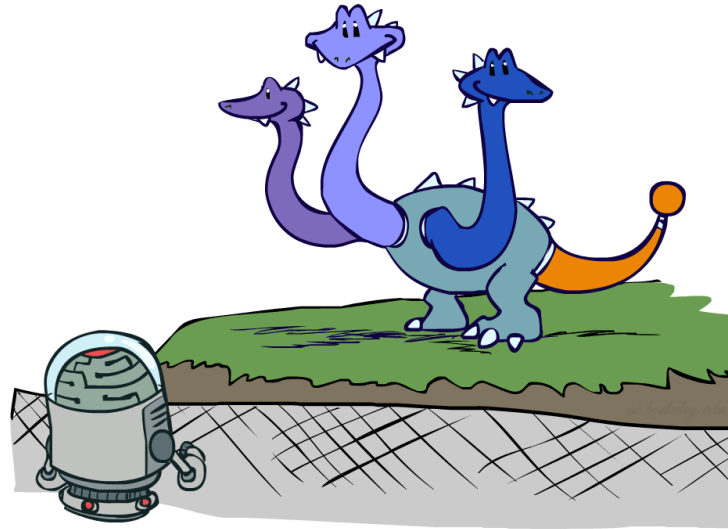
A \ J	true	false
true	0.09	0.01



Number of variables (capitals) = dimensionality of the table

Factor Zoo II

- Single conditional: $P(Y | x)$
 - Entries $P(y | x)$ for fixed x , all y
 - Sums to 1

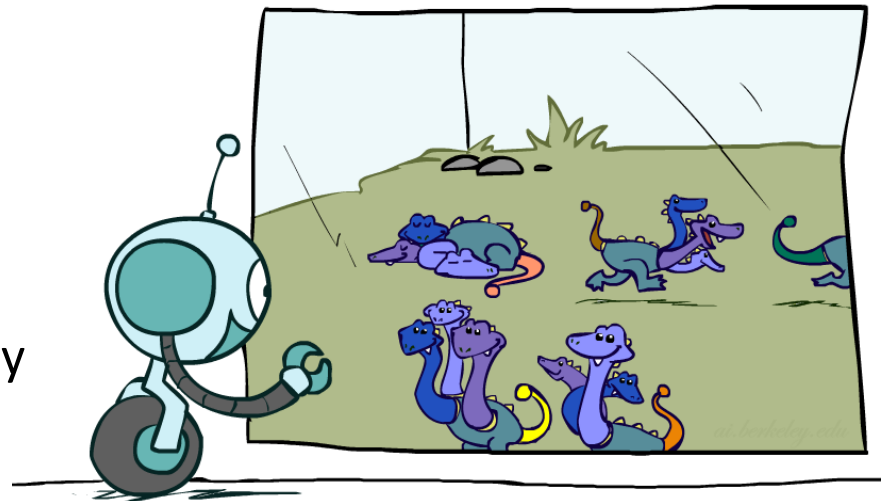


$P(J|a)$

A \ J	true	false
true	0.9	0.1

- Family of conditionals: $P(X | Y)$

- Multiple conditionals
- Entries $P(x | y)$ for all x, y
- Sums to $|Y|$



$P(J|A)$

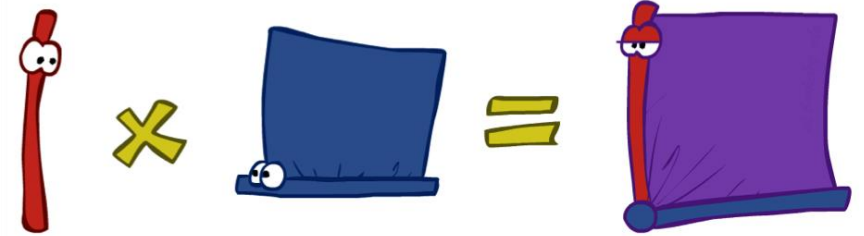
A \ J	true	false
true	0.9	0.1
false	0.05	0.95

} - $P(J|a)$

} - $P(J|\neg a)$

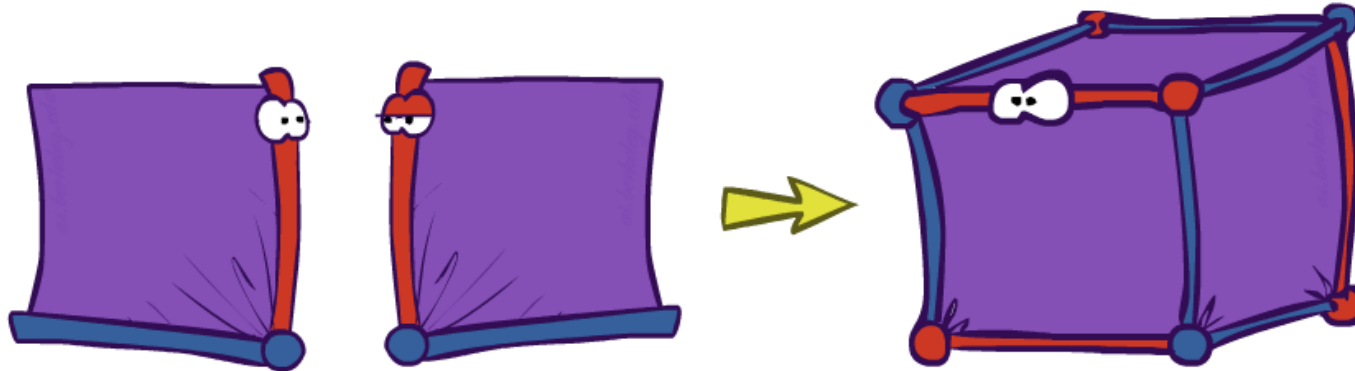
Operation 1: Pointwise product

- First basic operation: **pointwise product** of factors (similar to a **database join**, **not** matrix multiply!)
 - New factor has **union** of variables of the two original factors
 - Each entry is the product of the corresponding entries from the original factors
- Example: $P(A) \times P(J|A) = P(A,J)$



$P(A)$		$P(J A)$			$P(A,J)$		
		A \ J	true	false	A \ J	true	false
true	0.1	true	0.9	0.1	true	0.09	0.01
false	0.9	false	0.05	0.95	false	0.045	0.855

Example: Making larger factors



- Example: $P(A,J) \times P(A,M) = P(A,J,M)$

$P(A,J)$

A \ J	true	false
true	0.09	0.01
false	0.045	0.855

\times

$P(A,M)$

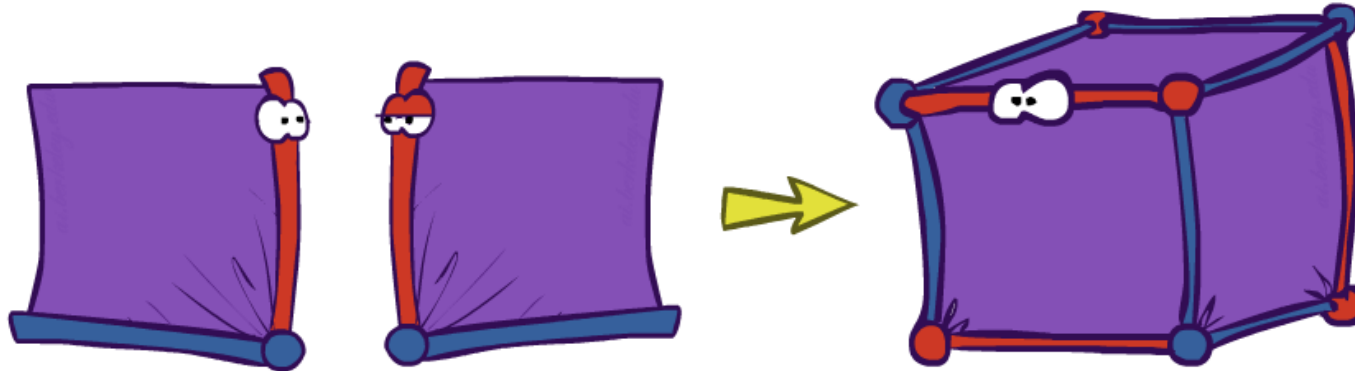
A \ M	true	false
true	0.07	0.03
false	0.009	0.891

$=$

$P(A,J,M)$

		J \ M	true	false	
J \ M	true	false			
	true				18
	false			.0003	
					A=true
					A=false

Example: Making larger factors



- Example: $P(U,V) \times P(V,W) \times P(W,X) = P(U,V,W,X)$
- Sizes: $[10,10] \times [10,10] \times [10,10] = [10,10,10,10]$
 - I.e., 300 numbers blows up to 10,000 numbers!
- Factor blowup can make Variable Elimination very expensive

Operation 2: Summing out a variable

- Second basic operation: **summing out** (or eliminating) a variable from a factor
 - Shrinks a factor to a smaller one
- Example: $\sum_j P(A,J) = P(A,j) + P(A,\neg j) = P(A)$

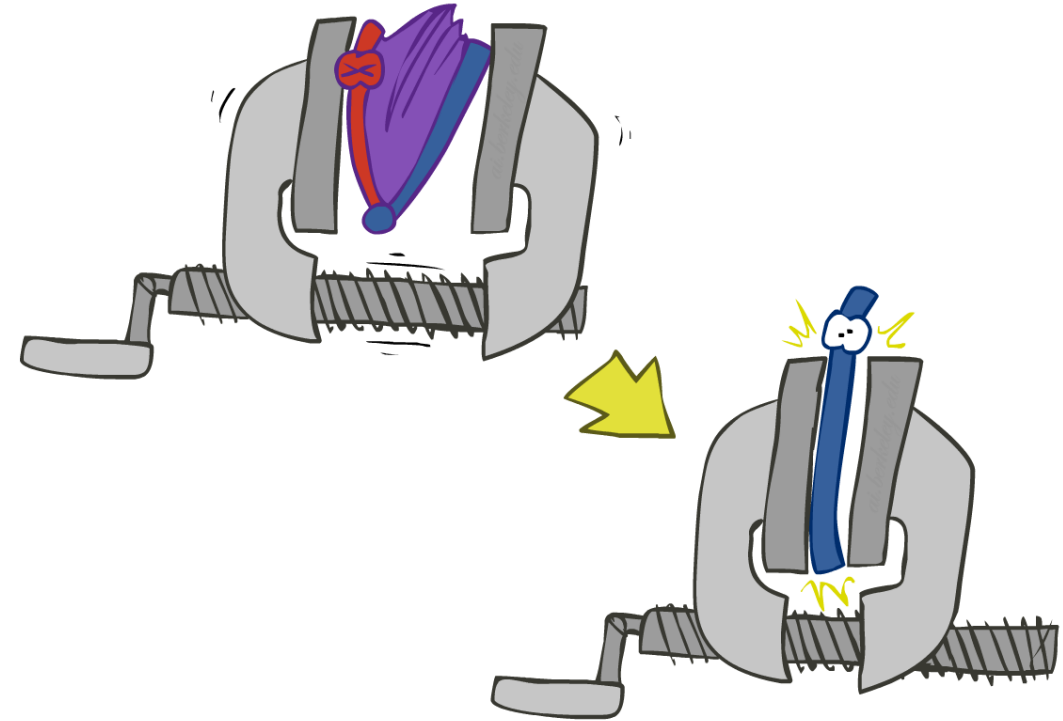
$P(A,J)$

A \ J	true	false
true	0.09	0.01
false	0.045	0.855

Sum out J
→

$P(A)$

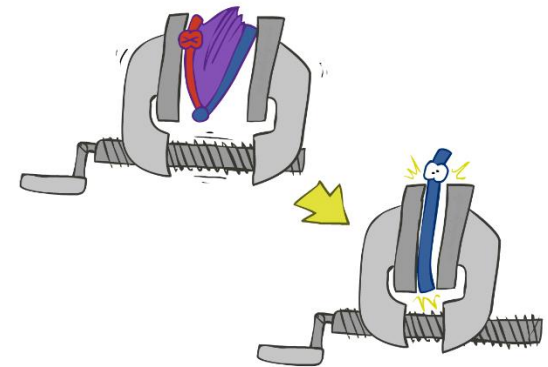
true	0.1
false	0.9



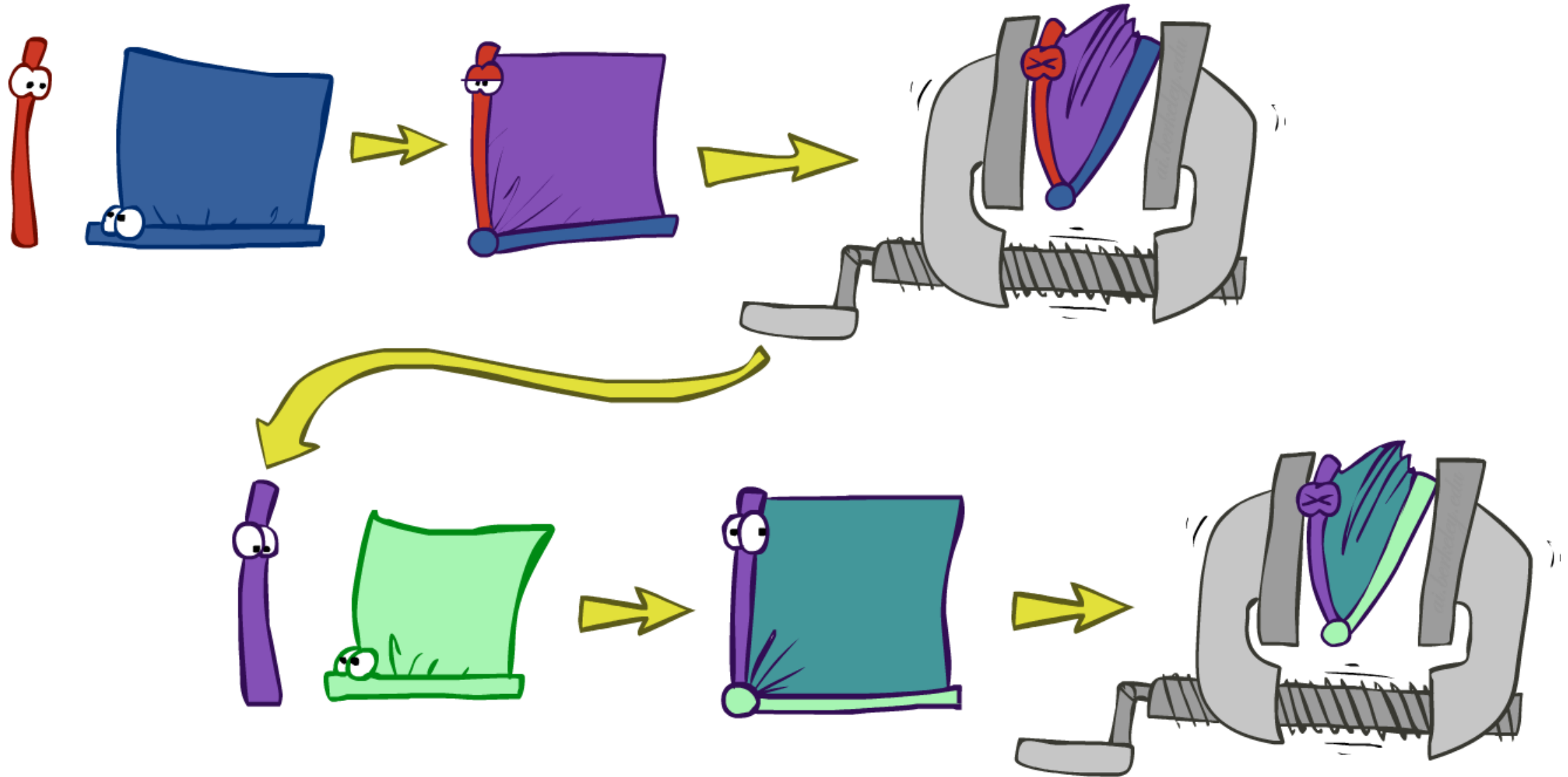
Summing out from a product of factors

- Project the factors each way first, then sum the products

- Example: $\sum_a P(a | B, e) \times P(j | a) \times P(m | a)$
 $= P(a | B, e) \times P(j | a) \times P(m | a) +$
 $P(\neg a | B, e) \times P(j | \neg a) \times P(m | \neg a)$



Variable Elimination

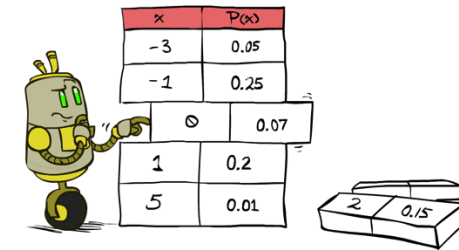


Variable Elimination Steps

■ Query: $P(Q | E_1=e_1, \dots, E_k=e_k)$

1. Start with initial factors:

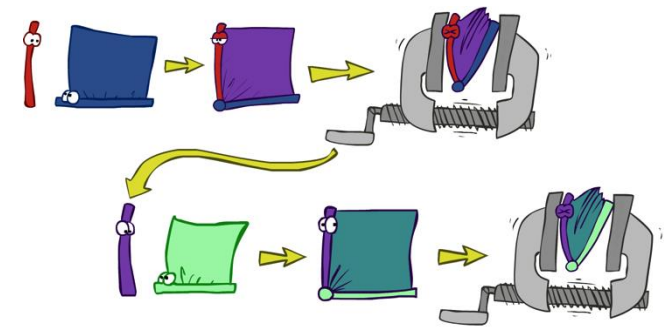
- Local CPTs (but instantiated by evidence)



x	P(x)
-3	0.05
-1	0.25
0	0.07
1	0.2
5	0.01

2. While there are still hidden variables (not Q or evidence):

- Pick a hidden variable H_j
- Eliminate (sum out) H_j from the product of all factors mentioning H_j



3. Join all remaining factors and normalize

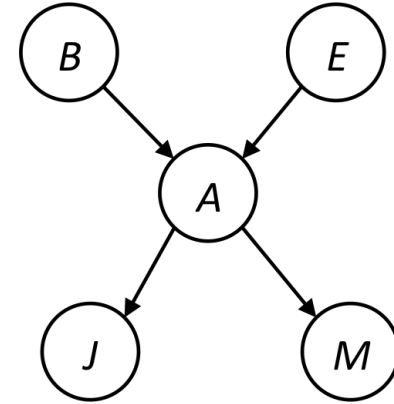

$$\text{stick} \times \text{blue square} = \text{purple square} \times \alpha$$

Var. Elim: Example

Query $P(B \mid j,m)$

initial factors:

$P(B)$ $P(E)$ $P(A \mid B,E)$ $P(j \mid A)$ $P(m \mid A)$



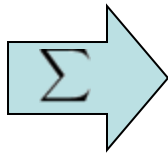
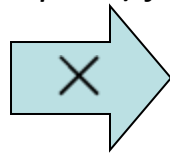
Step 2 a): Choose A to eliminate

$P(A \mid B,E)$

$P(j \mid A)$

$P(m \mid A)$

Step 2 b) join factors and sum out A



$P(j,m \mid B,E)$

new factors:

$P(B)$ $P(E)$ $P(j,m \mid B,E)$

Var. Elim: Example

new factors:

$P(B)$ $P(E)$ $P(j,m | B,E)$

Step 2 a): Choose E to eliminate

Step 2 b) join factors and sum out E

$P(E)$
 $P(j,m | B,E)$ \times \rightarrow Σ $P(j,m | B)$

new factors:

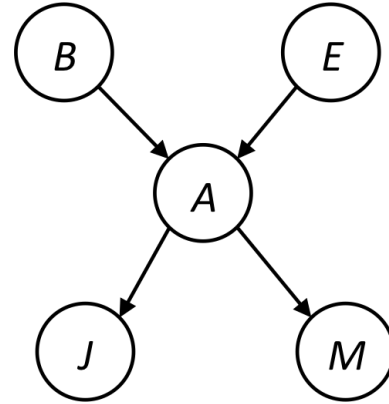
$P(B)$ $P(j,m | B)$

Step 3: Finish with B

Step 3 join factors and normalize

$P(B)$
 $P(j,m | B)$ \times \rightarrow $P(j,m, B)$ \rightarrow Normalize $P(B | j,m)$

Done!



Var. Elim: Order matters

- Order the terms Z, A, B C, D

- $$P(D) = \alpha \sum_{z,a,b,c} P(z) P(a|z) P(b|z) P(c|z) P(D|z)$$
$$= \alpha \sum_z P(z) \sum_a P(a|z) \sum_b P(b|z) \sum_c P(c|z) P(D|z)$$

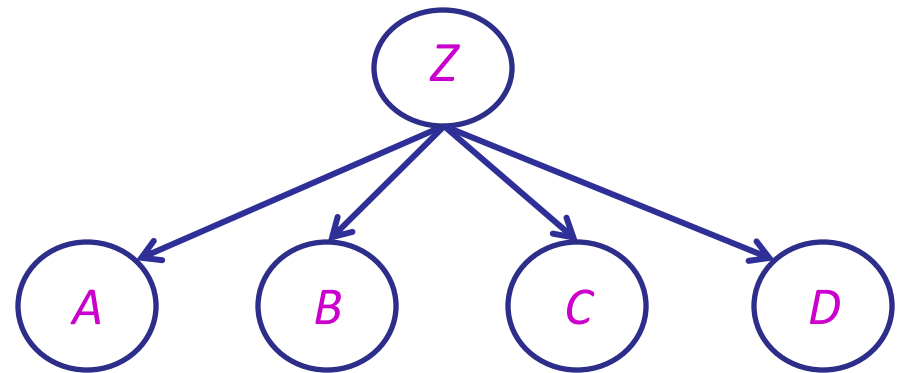
- Largest factor has 2 variables (D,Z)

- Order the terms A, B C, D, Z

- $$P(D) = \alpha \sum_{a,b,c,z} P(a|z) P(b|z) P(c|z) P(D|z) P(z)$$
$$= \alpha \sum_a \sum_b \sum_c \sum_z P(a|z) P(b|z) P(c|z) P(D|z) P(z)$$

- Largest factor has 4 variables (A,B,C,D)

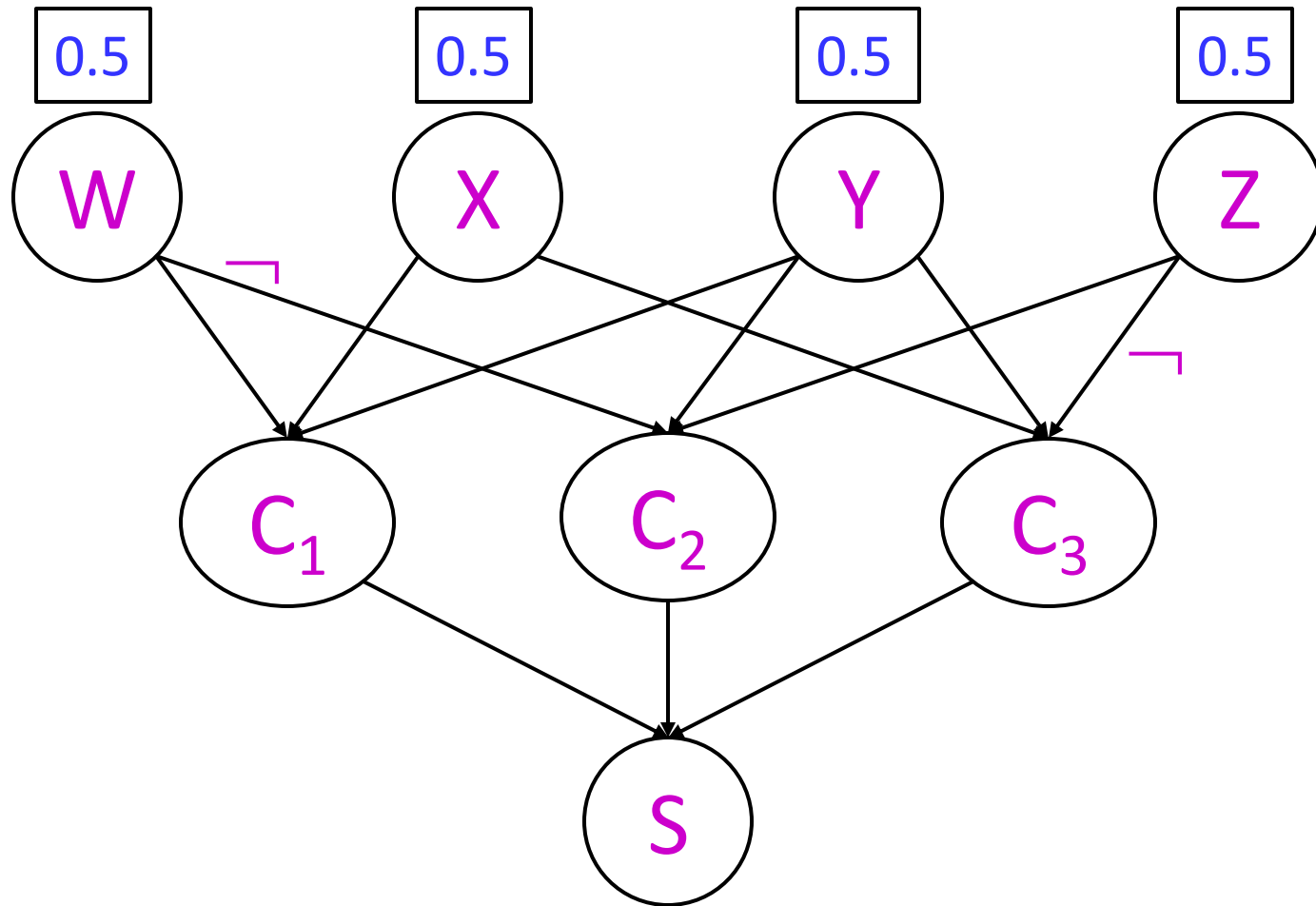
- In general, with n leaves, factor of size 2^n in general d^n



Var. Elim.: Computational and Space Complexity

- The computational and space complexity of variable elimination is determined by the largest factor
 - (Space is the difficult part.)
- The elimination ordering can greatly affect the size of the largest factor.
 - E.g., previous slide's example 2^n vs. 2
- Does there always exist an ordering that only results in small factors?
 - **No!**

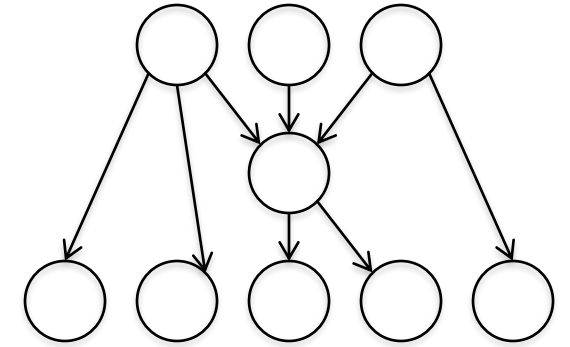
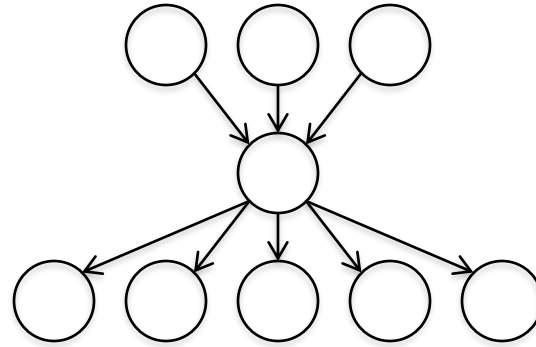
Bonus: Worst Case Complexity?



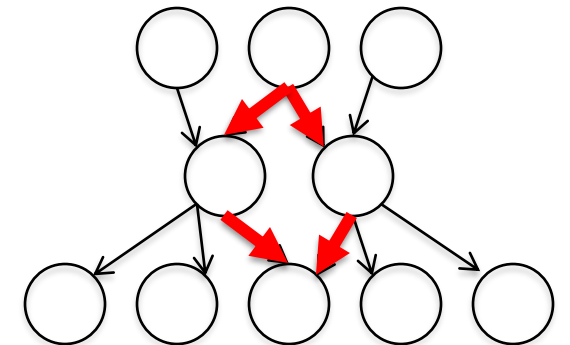
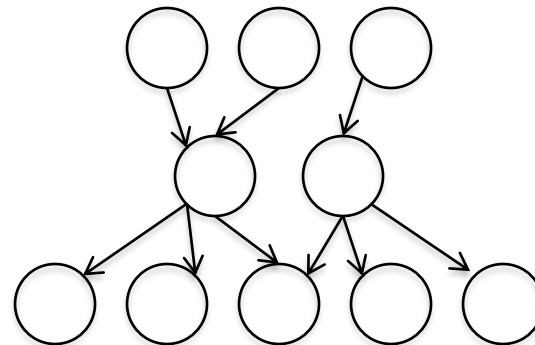
- Variables: W, X, Y, Z
- Clauses:
 1. $C_1 = W \vee X \vee Y$
 2. $C_2 = Y \vee Z \vee \neg W$
 3. $C_3 = X \vee Y \vee \neg Z$
- Sentence $S = C_1 \wedge C_2 \wedge C_3$
- $P(S) > 0$ iff S is satisfiable
 \Rightarrow **NP-hard**
- $P(S) = K \times 0.5^n$ where K is the number of satisfying assignments for clauses
 \Rightarrow **#P-hard**

Bonus: Polytrees

- A polytree is a directed graph with no undirected cycles
 - At most one undirected path b/w any two nodes
 - E.g. burglary network but not insurance network



- For poly-trees the complexity of variable elimination is **linear in the network size** if you eliminate from the leaf towards the roots



Bayes Nets

✓ Part I: Representation

✓ Part II: Independence

✓ Part III: Exact inference

Part IV: Approximate Inference

- Prior Sampling
- Rejection Sampling
- Likelihood Weighting
- Gibbs Sampling

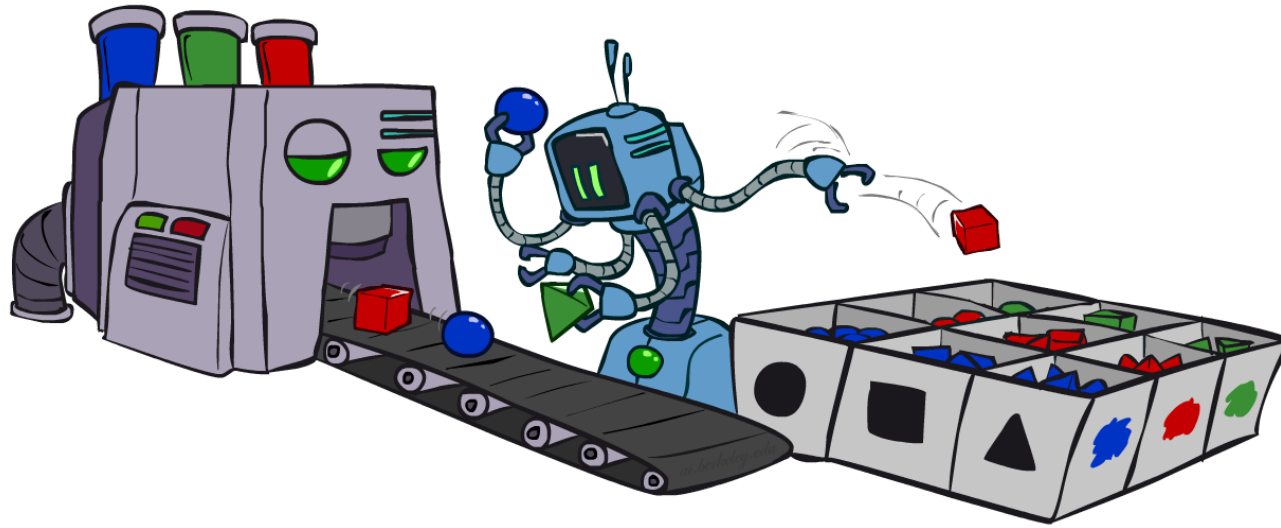
Sampling

- Basic idea

- Draw N samples from a *sampling distribution* S
- Compute an approximate posterior probability
- Show this converges to the true probability P

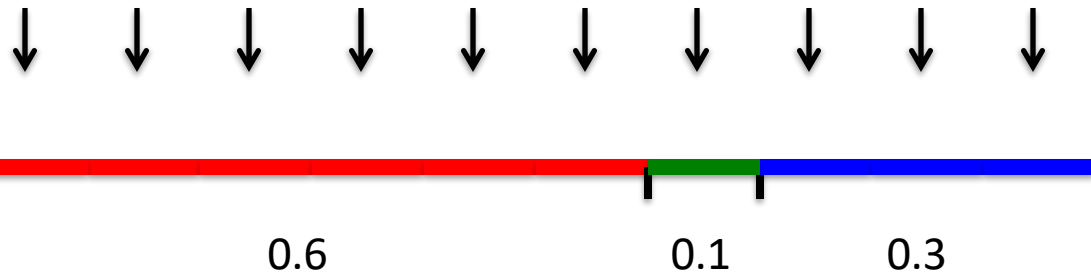
- Why sample?

- Often very fast to get a decent approximate answer
- The algorithms are very simple and general (easy to apply to fancy models)
- They require very little memory ($O(n)$)
- They can be applied to large models, whereas exact algorithms blow up



Sampling basics: discrete (*categorical*) distribution

- To simulate a biased d-sided coin:
 - Step 1: Get sample u from uniform distribution over $[0, 1)$
 - E.g. `random()` in python
 - Step 2: Convert this sample u into an outcome for the given distribution by associating each outcome x with a $P(x)$ -sized sub-interval of $[0,1)$

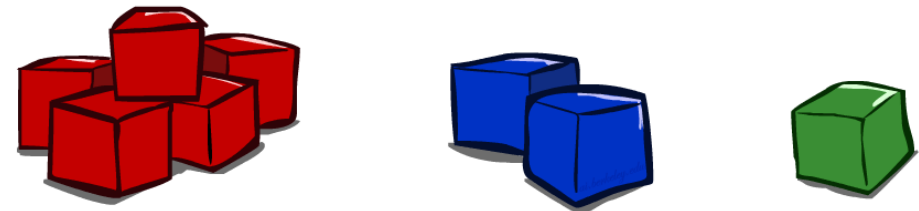


- Example

C	$P(C)$
red	0.6
green	0.1
blue	0.3

- $0.0 \leq u < 0.6, \rightarrow C = \text{red}$
- $0.6 \leq u < 0.7, \rightarrow C = \text{green}$
- $0.7 \leq u < 1.0, \rightarrow C = \text{blue}$

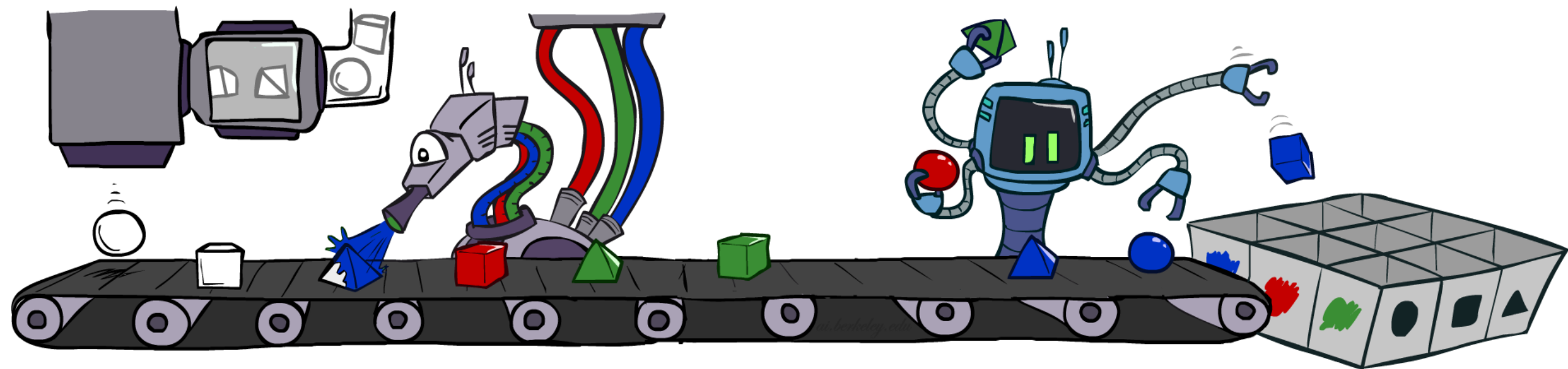
- If `random()` returns $u = 0.83$, then the sample is $C = \text{blue}$
- E.g, after sampling 8 times:



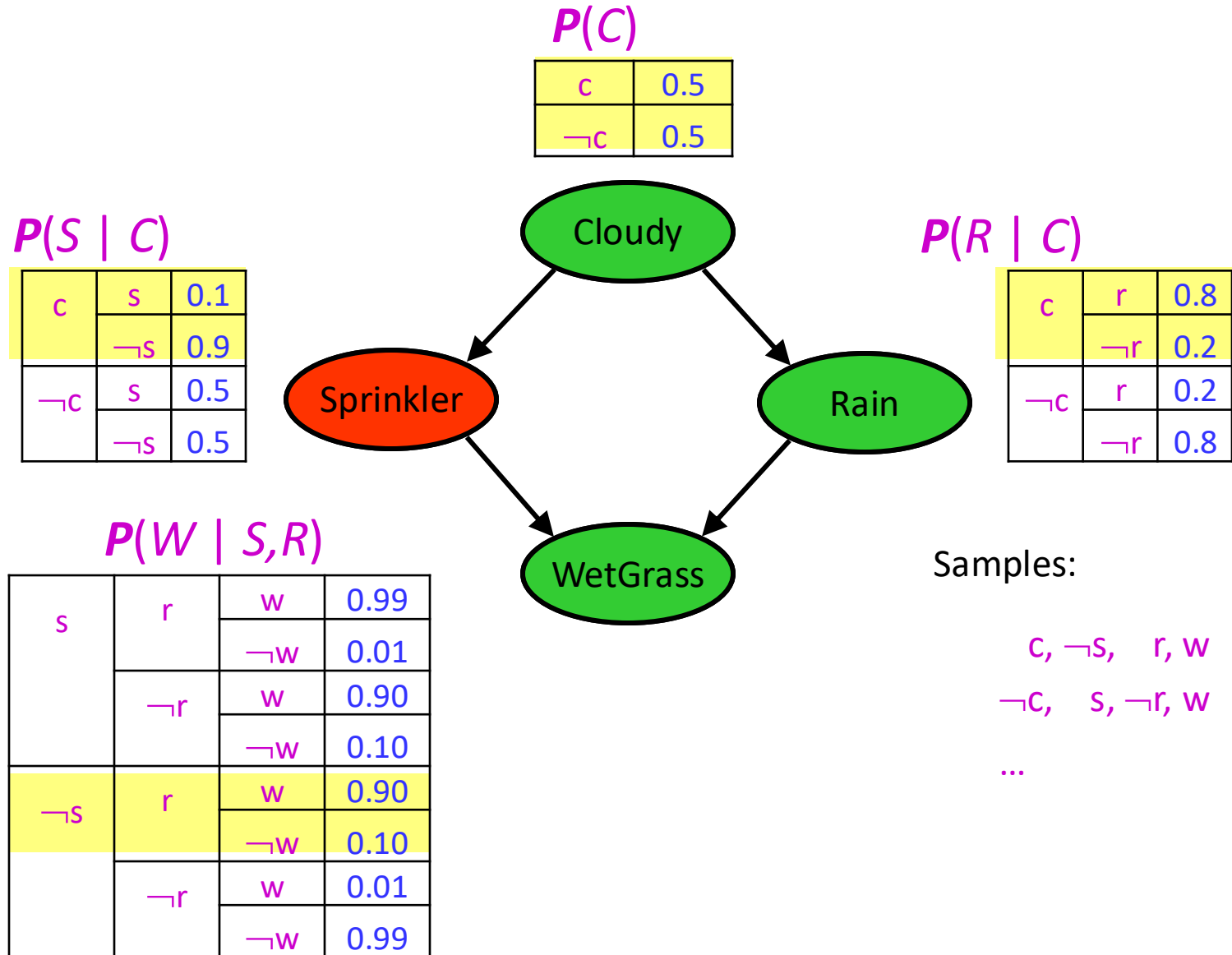
Sampling in Bayes Nets

- Prior Sampling
- Rejection Sampling
- Likelihood Weighting
- Gibbs Sampling

Prior Sampling

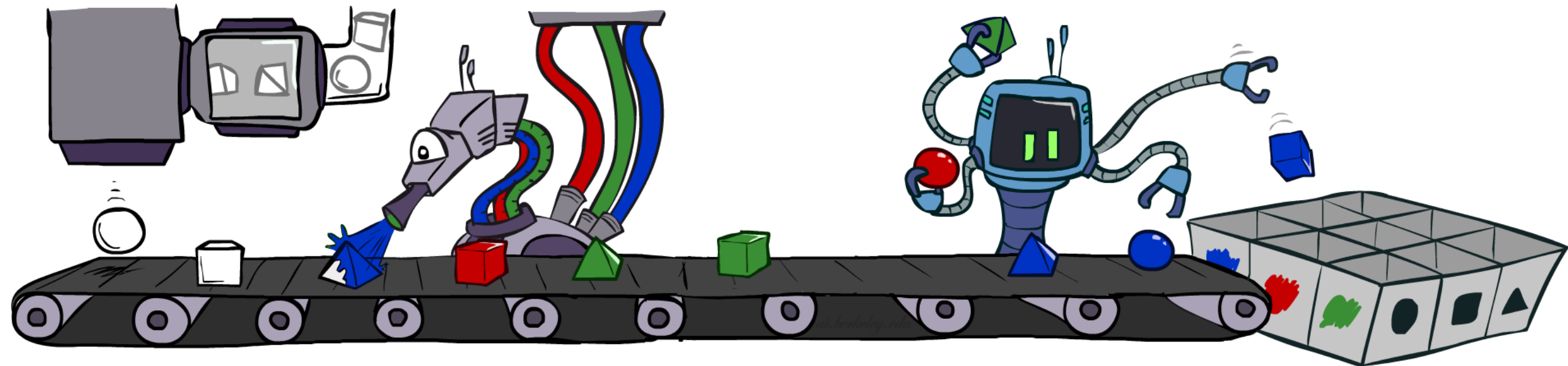


Prior Sampling



Prior Sampling

- For $i=1, 2, \dots, n$ (in topological order)
 - Sample X_i from $P(X_i \mid \text{parents}(X_i))$
- Return (X_1, X_2, \dots, X_n)



Prior Sampling

- This process generates samples with probability:

$$S_{PS}(x_1, \dots, x_n) = \prod_i P(x_i \mid \text{parents}(X_i)) = P(x_1, \dots, x_n)$$

$$S_{PS} = \text{Sample}_{\text{prior sampling}}$$

...i.e. the Bayes' net's joint probability

- Let the number of samples of an event be $N_{PS}(x_1, \dots, x_n)$
- Estimate from N samples is $Q_N(x_1, \dots, x_n) = N_{PS}(x_1, \dots, x_n)/N$
- Then $\lim_{N \rightarrow \infty} Q_N(x_1, \dots, x_n) = \lim_{N \rightarrow \infty} N_{PS}(x_1, \dots, x_n)/N$
 $= S_{PS}(x_1, \dots, x_n)$
 $= P(x_1, \dots, x_n)$
- I.e., the sampling procedure is **consistent**

Example

- We'll get a bunch of samples from the Bayes' net:

$c, \neg s, r, w$

c, s, r, w

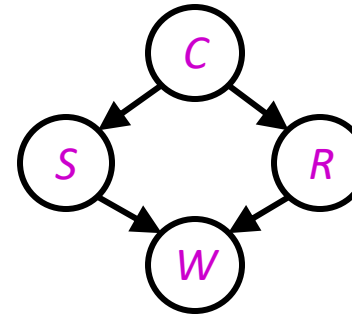
$\neg c, s, r, \neg w$

$c, \neg s, r, w$

$\neg c, \neg s, \neg r, w$

- If we want to know $P(W)$

- We have counts $\langle w:4, \neg w:1 \rangle$
- Normalize to get $P(W) = \langle w:0.8, \neg w:0.2 \rangle$
- This will get closer to the true distribution with more samples
- Can estimate anything else, too
 - E.g., for query $P(C | r, w)$ use $P(C | r, w) = \alpha P(C, r, w)$



Example

- Say we want to know $P(W \mid s, \neg r)$
- We'll get a bunch of samples from the Bayes' net:

$c, \neg s, r, w$

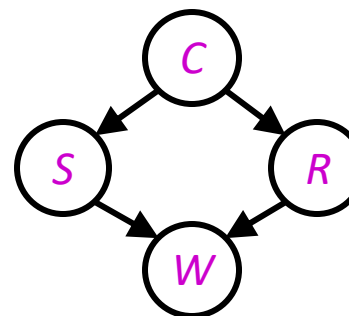
c, s, r, w

$\neg c, s, r, \neg w$

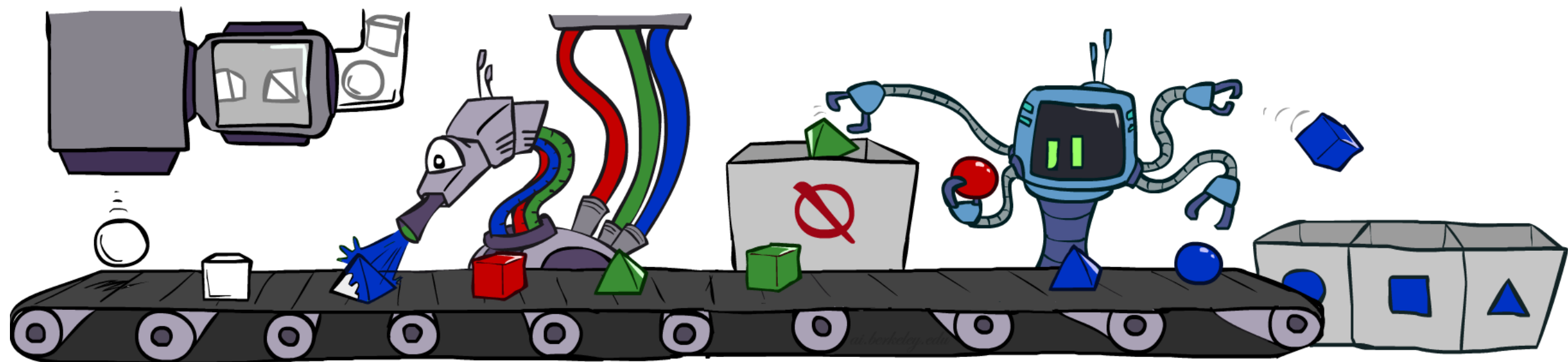
$c, \neg s, r, w$

$\neg c, \neg s, \neg r, w$

- What's wrong?

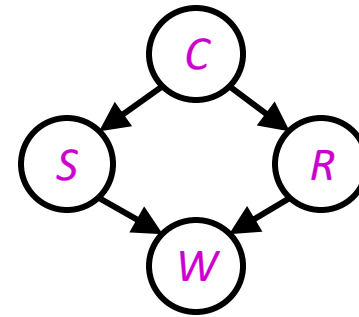


Rejection Sampling



Rejection Sampling

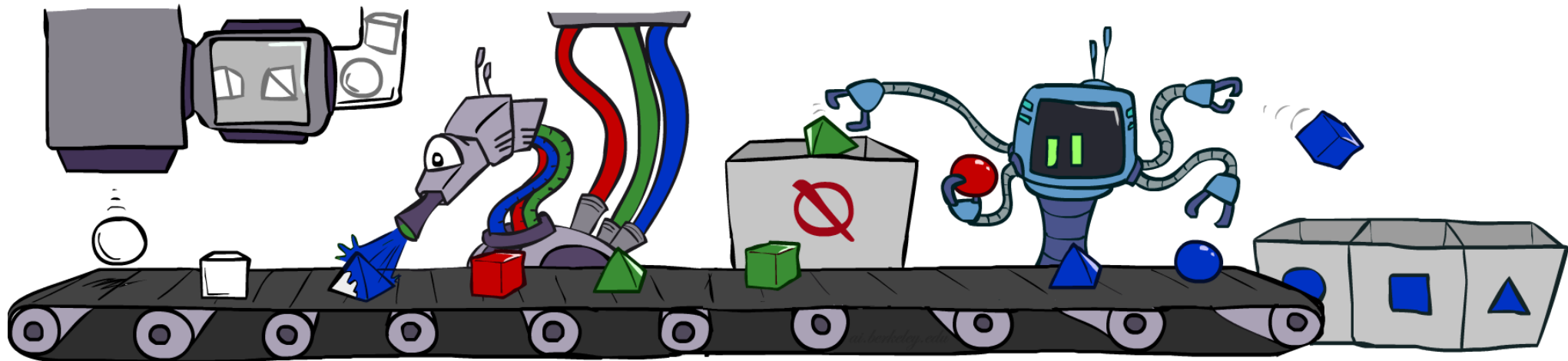
- A simple modification of prior sampling for conditional probabilities
- Let's say we want $P(C | r, w)$
- Count the C outcomes, but ignore (reject) samples that don't have $R=true$, $W=true$
 - This is called *rejection sampling*
 - It is also consistent for conditional probabilities (i.e., correct in the limit)



$C, \neg S, r, w$
 ~~$C, S, \neg r$~~
 ~~$\neg C, S, r, \neg w$~~
 ~~$C, \neg S, \neg r$~~
 $\neg C, \neg S, r, w$

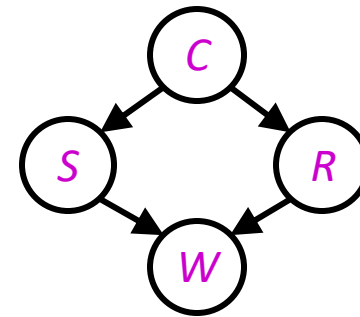
Rejection Sampling

- Input: evidence e_1, \dots, e_k
- For $i=1, 2, \dots, n$
 - Sample x_i from $P(x_i \mid \text{parents}(x_i))$
 - If x_i not consistent with evidence
 - Reject: Return, and no sample is generated in this cycle
- Return (x_1, x_2, \dots, x_n)



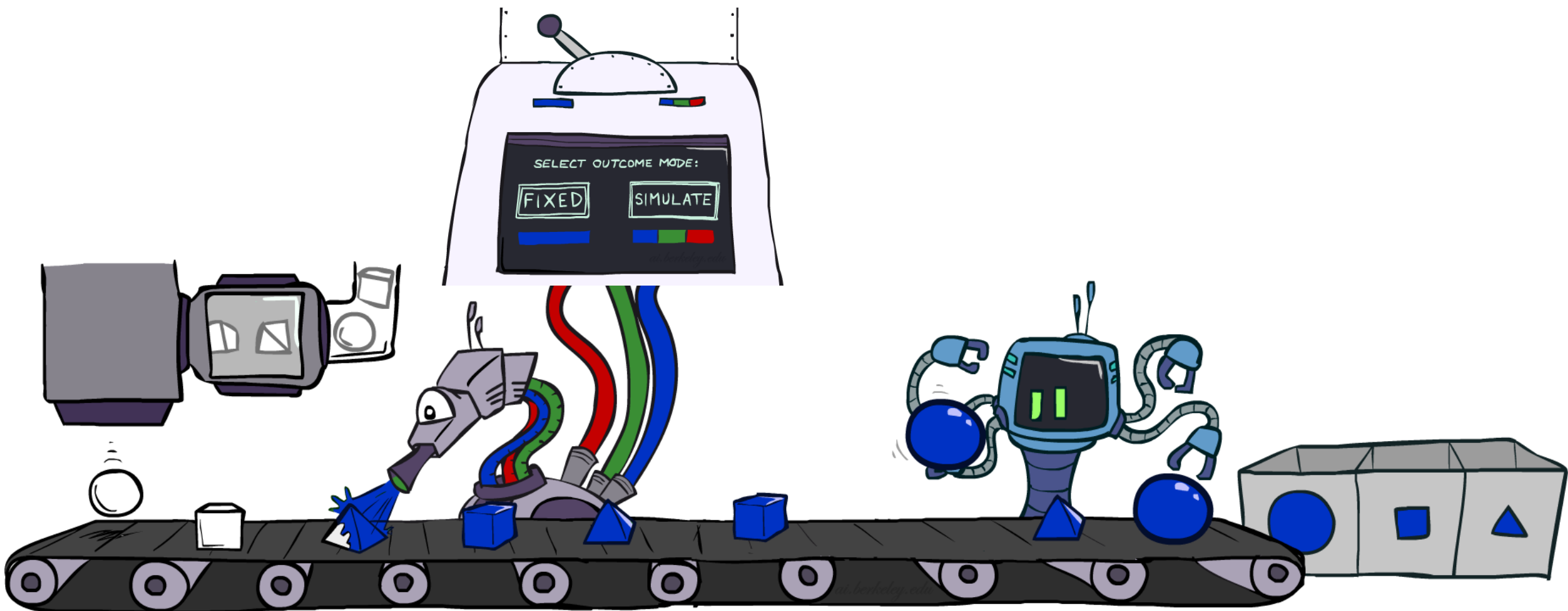
Rejection Sampling

- We want $P(C | r, w)$
- What if we have to reject a lot of samples?



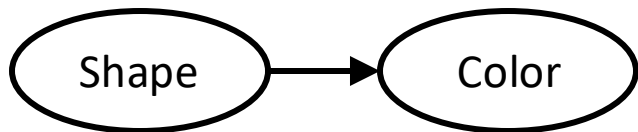
$C, \neg S, r, w$
 ~~$C, S, \neg r$~~
 ~~$\neg C, S, r, \neg w$~~
 ~~$C, \neg S, \neg r$~~
 $\neg C, \neg S, r, w$

Likelihood Weighting

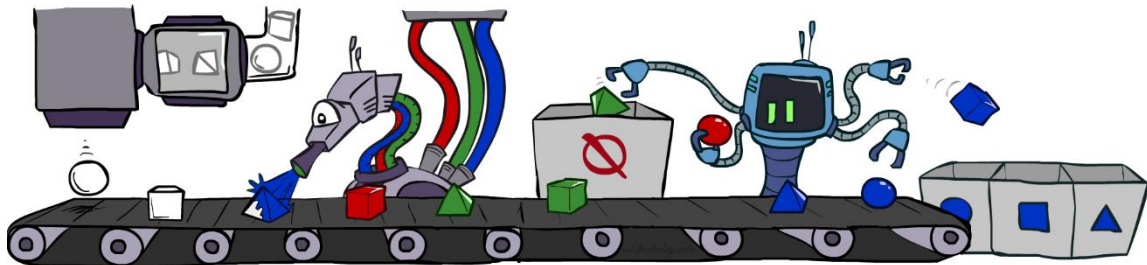


Likelihood Weighting

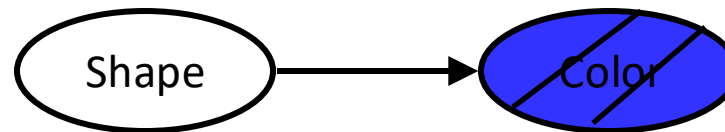
- Problem with rejection sampling:
 - If evidence is unlikely, rejects lots of samples
 - Evidence not exploited as you sample
 - Consider $P(\text{Shape} | \text{Color}=\text{blue})$



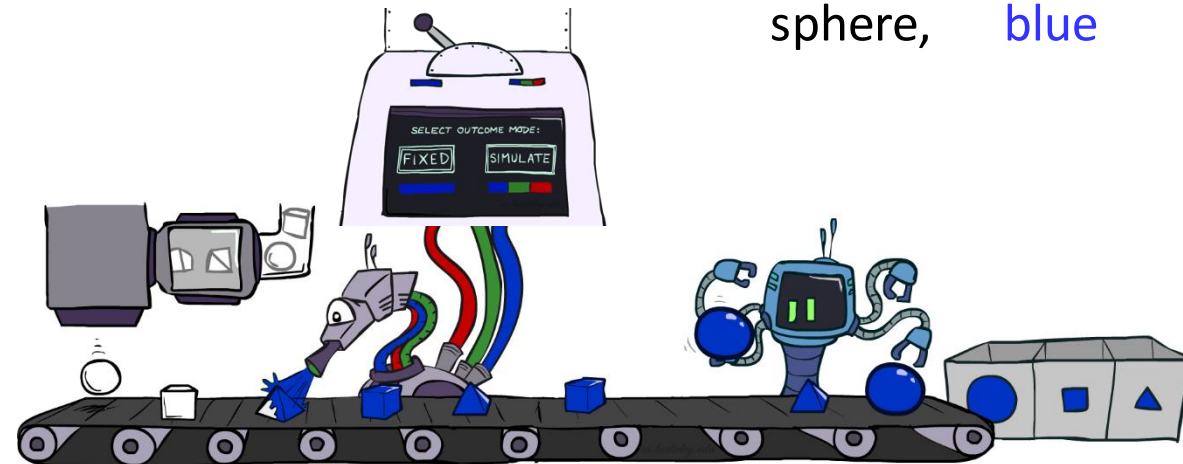
~~pyramid, green~~
~~pyramid, red~~
 sphere, blue
~~cube, red~~
~~sphere, green~~



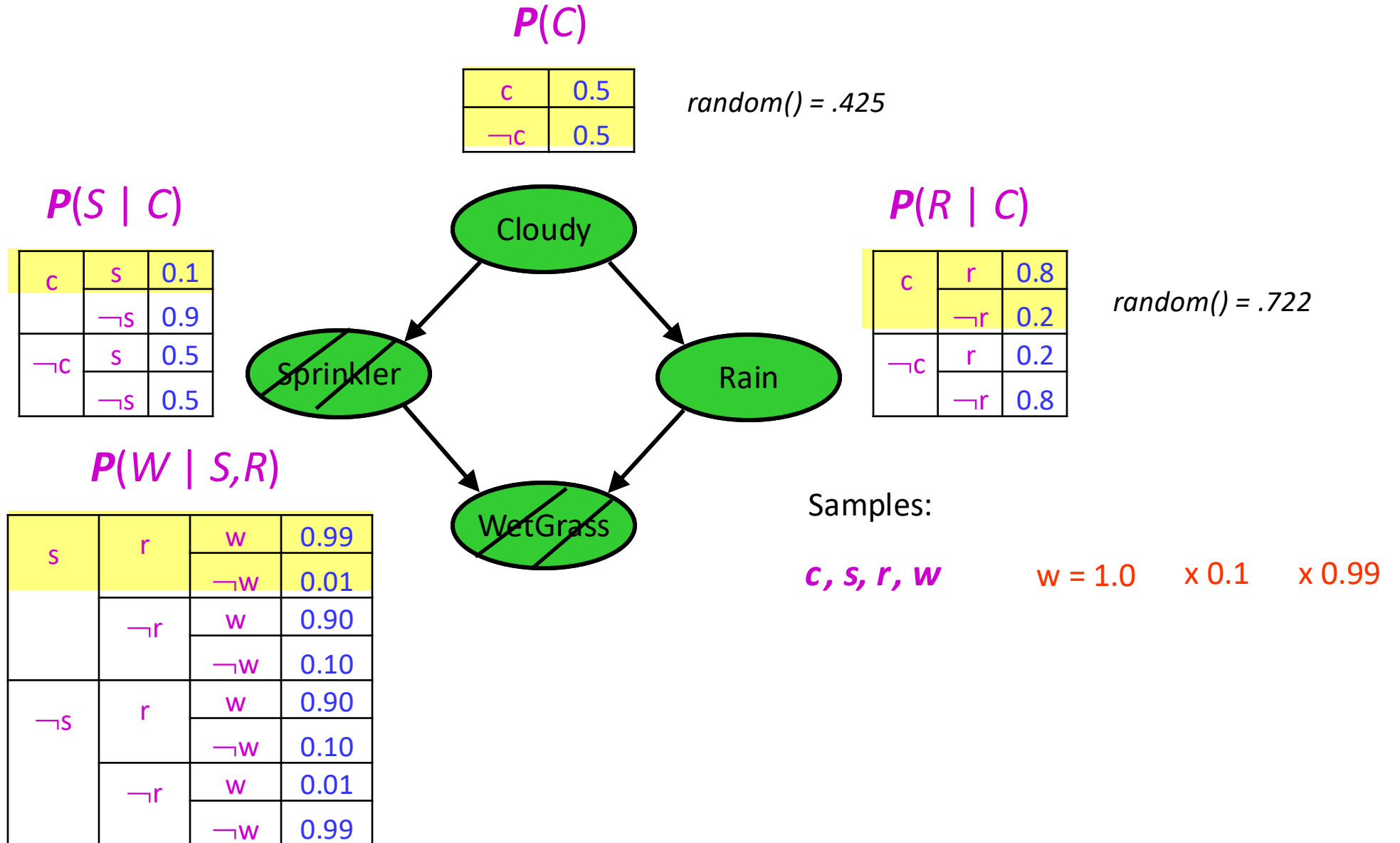
- Idea: fix evidence variables, sample the rest
 - Problem: sample distribution not consistent!
 - No longer sampling according to the CPT!
 - Solution: *weight* each sample by probability of evidence variables given parents



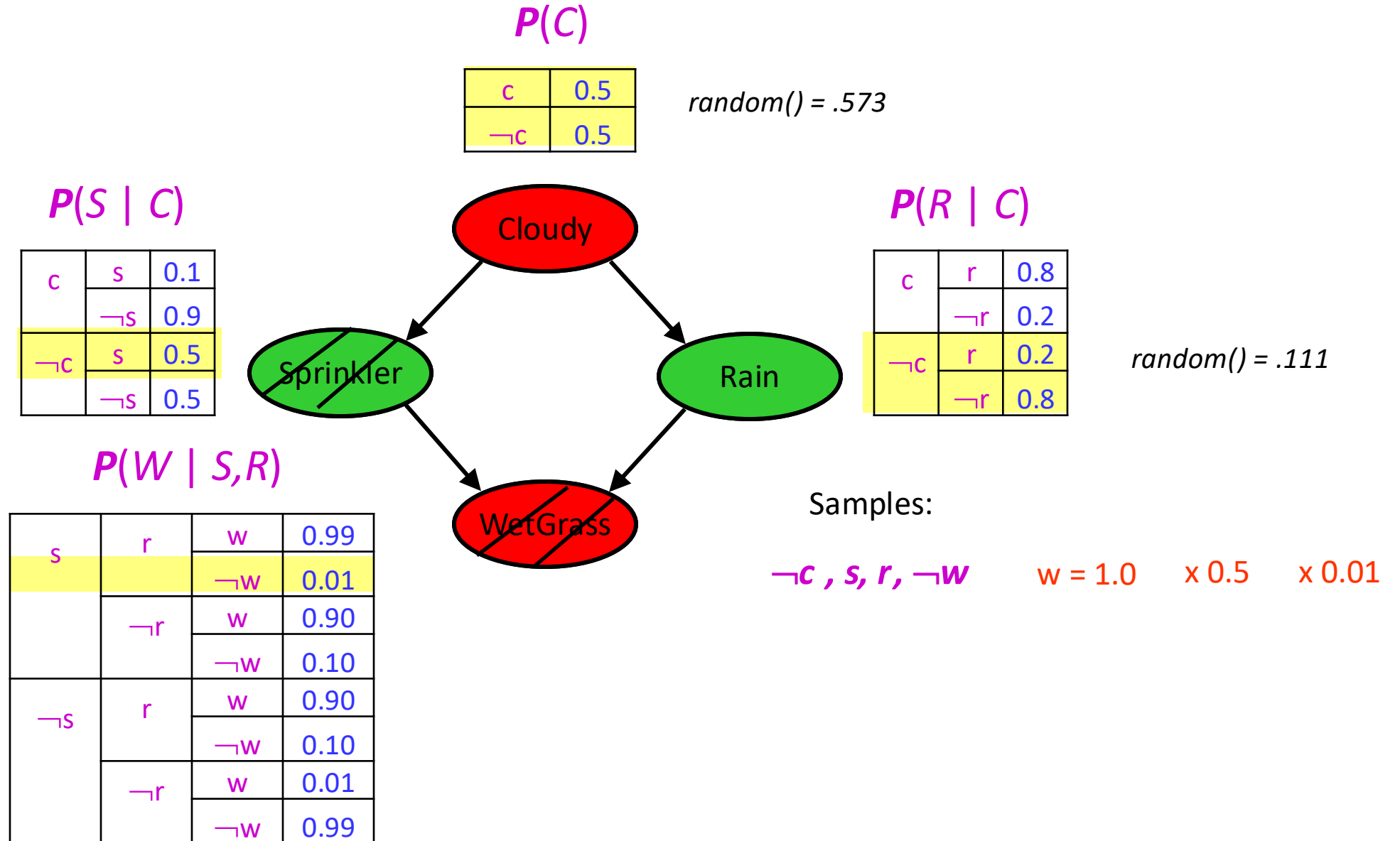
pyramid, blue
 pyramid, blue
 sphere, blue
 cube, blue
 sphere, blue



Likelihood Weighting: $P(C, R | s, w)$

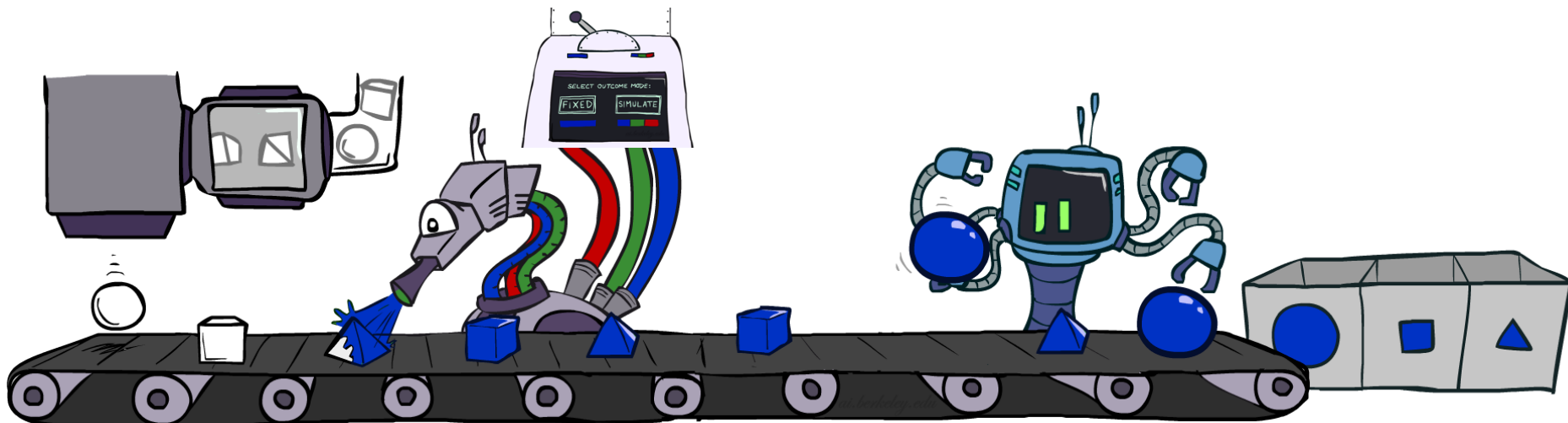


Likelihood Weighting: $P(C, R | s, \neg w)$



Likelihood Weighting

- Input: evidence e_1, \dots, e_k
- $w = 1.0$
- for $i=1, 2, \dots, n$
 - if X_i is an evidence variable
 - $x_i = \text{observed value}_i$ for X_i
 - Set $w = w * P(x_i | \text{parents}(X_i))$
 - else
 - Sample x_i from $P(X_i | \text{parents}(X_i))$
- return $(x_1, x_2, \dots, x_n), w$



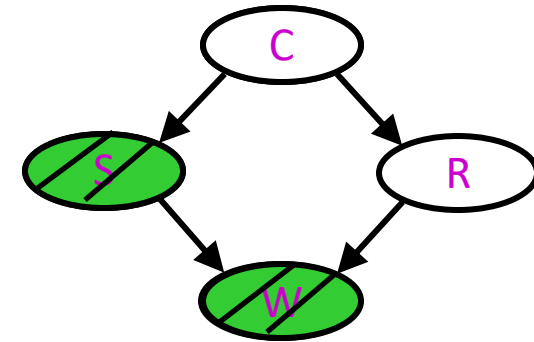
Likelihood Weighting

- Sampling distribution if \mathbf{z} sampled and \mathbf{e} fixed evidence

$$S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_j P(z_j \mid \text{parents}(Z_j))$$

- Now, samples have weights

$$w(\mathbf{z}, \mathbf{e}) = \prod_k P(e_k \mid \text{parents}(E_k))$$



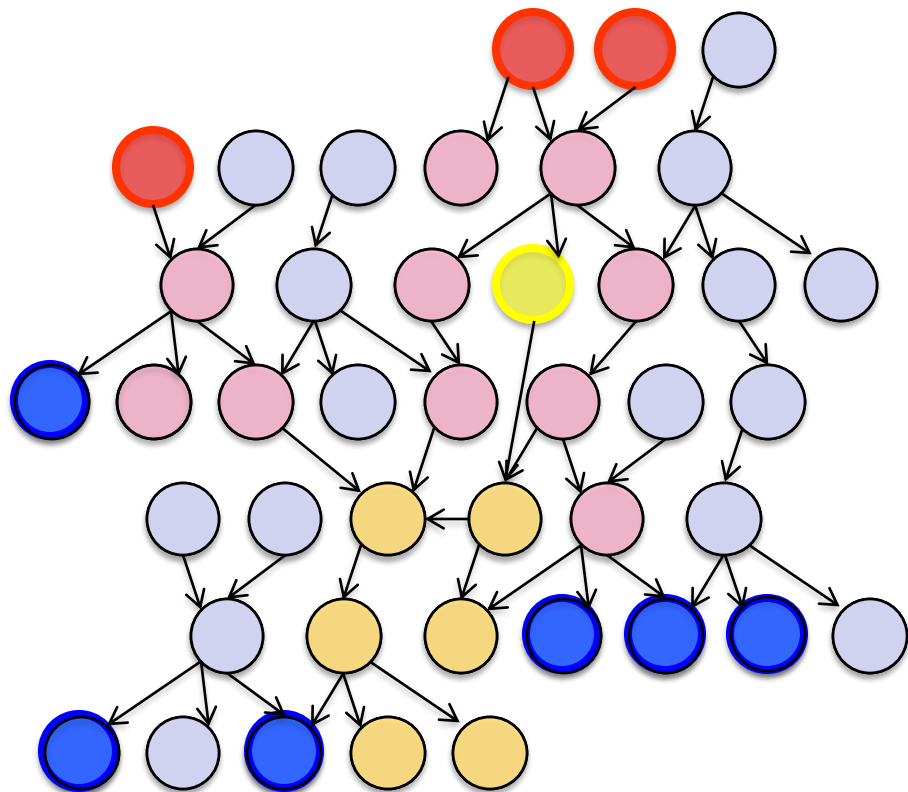
- Together, weighted sampling distribution is consistent

$$\begin{aligned} S_{WS}(\mathbf{z}, \mathbf{e}) \cdot w(\mathbf{z}, \mathbf{e}) &= \prod_j P(z_j \mid \text{parents}(Z_j)) \prod_k P(e_k \mid \text{parents}(E_k)) \\ &= P(\mathbf{z}, \mathbf{e}) \end{aligned}$$

- Likelihood weighting is an example of **importance sampling**
 - Would like to estimate some quantity based on samples from P
 - P is hard to sample from, so use Q instead
 - Weight each sample x by $P(x)/Q(x)$

Likelihood Weighting

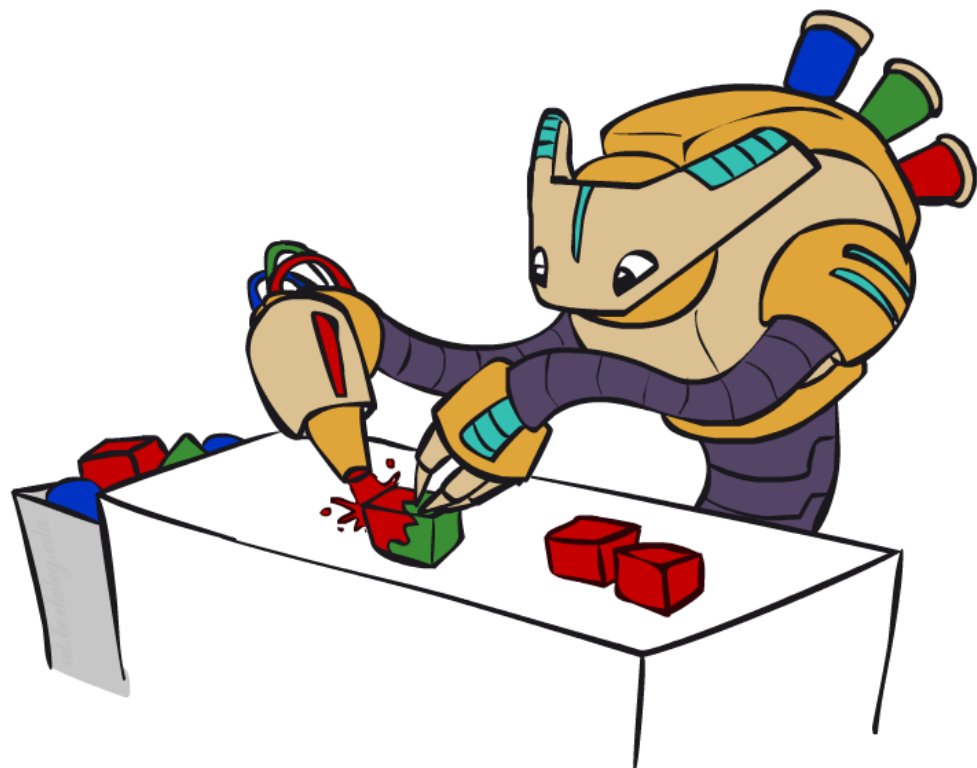
- Likelihood weighting is good
 - All samples are used
 - The values of **downstream** variables are influenced by **upstream** evidence



- Likelihood weighting still has weaknesses
 - The values of **upstream** variables are unaffected by **downstream** evidence
 - E.g., suppose evidence is a video of a traffic accident, query is likelihood of accident
 - With evidence in k leaf nodes, weights will be $O(2^{-k})$
 - Each will be very small!
 - 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!

**dark red nodes are upstream evidence, weighted by probability
light pink nodes are hidden variables conditioned on evidence
lavender nodes are hidden variables conditioned on the sample
circled yellow node is the query variable
gold nodes are hidden variables weighted by query and evidence
dark blue nodes are downstream evidence*

Gibbs Sampling



Markov Chain Monte Carlo

- MCMC (Markov chain Monte Carlo) is a family of randomized algorithms for approximating some quantity of interest over a very large state space
 - Markov chain = a sequence of randomly chosen states (“random walk”), where each state is chosen conditioned on the previous state
 - Monte Carlo = a very expensive city in Monaco with a famous casino
 - Monte Carlo = an algorithm (usually based on sampling) that has some probability of producing an incorrect answer
- MCMC = wander around for a bit, average what you see



Gibbs sampling

- A particular kind of MCMC
 - States are complete assignments to all variables
 - Evidence variables remain fixed, other variables change
 - To generate the next state, pick a variable and sample a value for it conditioned on all the other variables: $X_i' \sim P(X_i \mid x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$
 - Will tend to move towards states of higher probability, but can go down too
 - In a Bayes net, $P(X_i \mid x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(X_i \mid \text{markov_blanket}(X_i))$
- Theorem: Gibbs sampling is consistent*

Provided all Gibbs distributions are bounded away from 0 and 1 and variable selection is fair

$X \sim P(\dots)$ means X is drawn from distribution $P(\dots)$

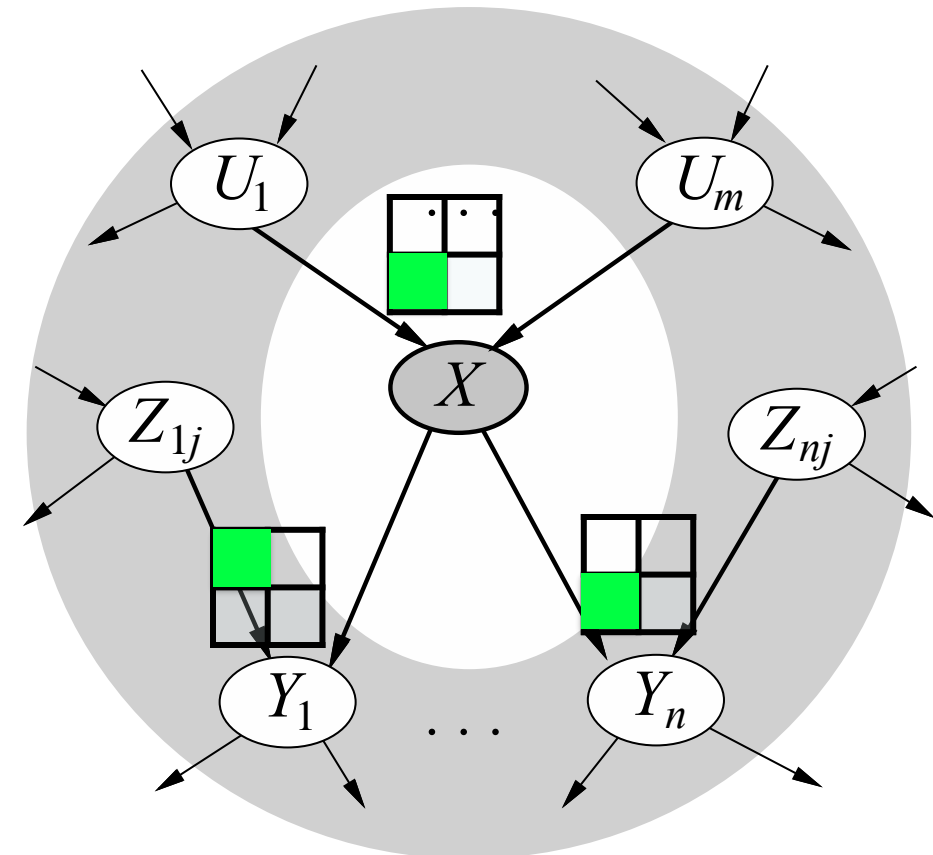
How to do this?

- Repeat many times

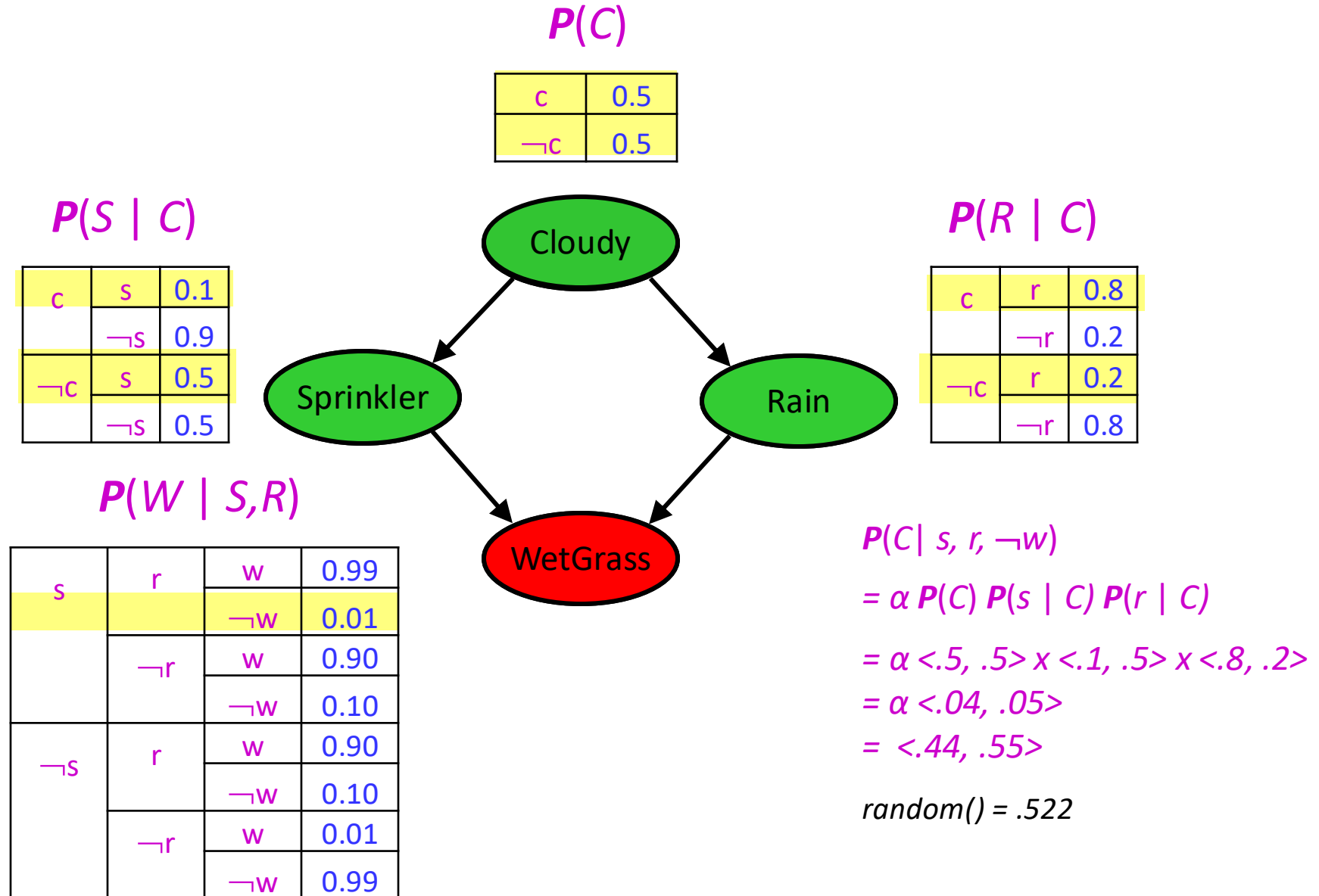
- Sample a non-evidence variable X_i from

$$P(X_i | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(X_i | \text{markov_blanket}(X_i))$$

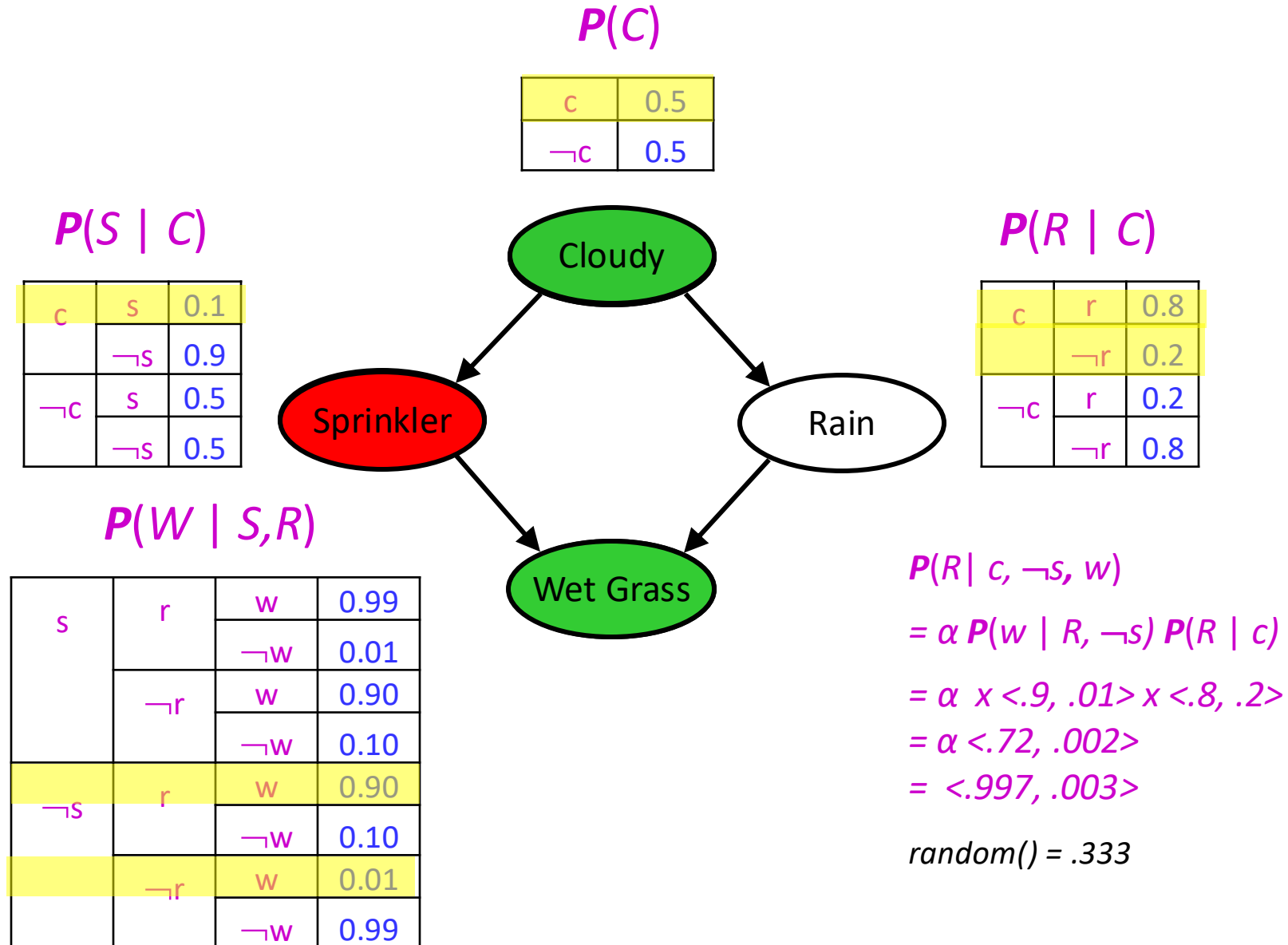
$$= \alpha P(X_i | \text{parents}(X_i)) \prod_j P(y_j | \text{parents}(Y_j))$$



Sample $C \sim P(C | s, r, \neg w)$



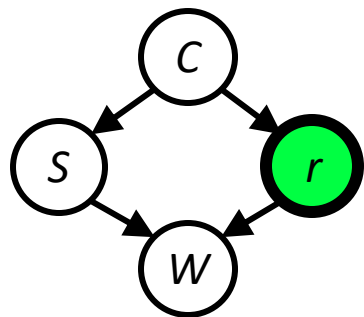
Your Turn: Sample $R \sim P(R | c, \neg s, w)$



Gibbs Sampling Example: $P(S | r)$

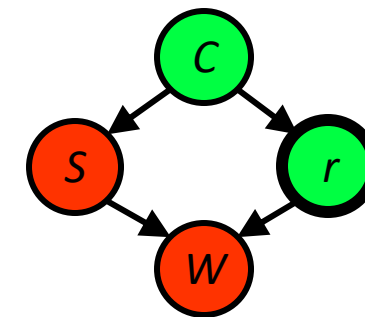
- Step 1: Fix evidence

- $R = \text{true}$



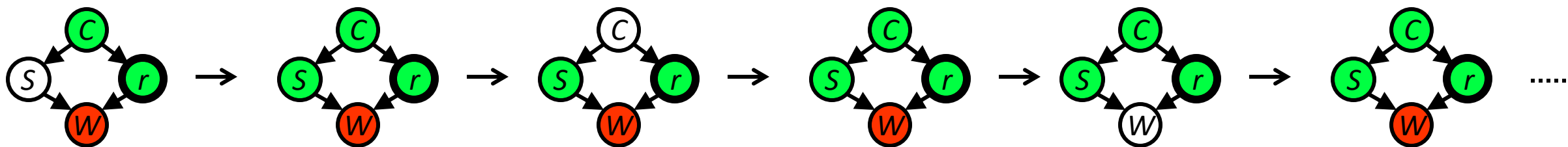
- Step 2: Initialize other variables

- Randomly



- Step 3: Repeat

- Choose a non-evidence variable X
- Resample X from $P(X | \text{markov_blanket}(X))$

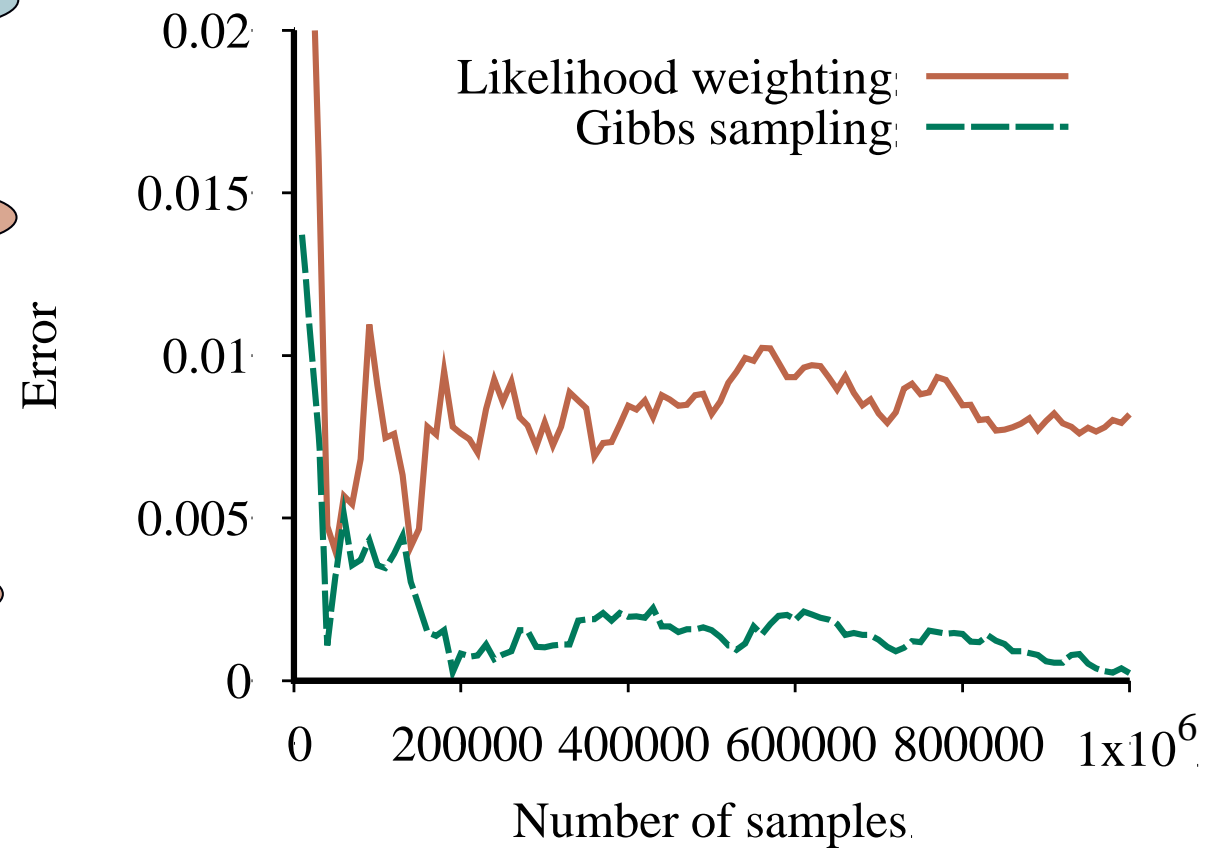
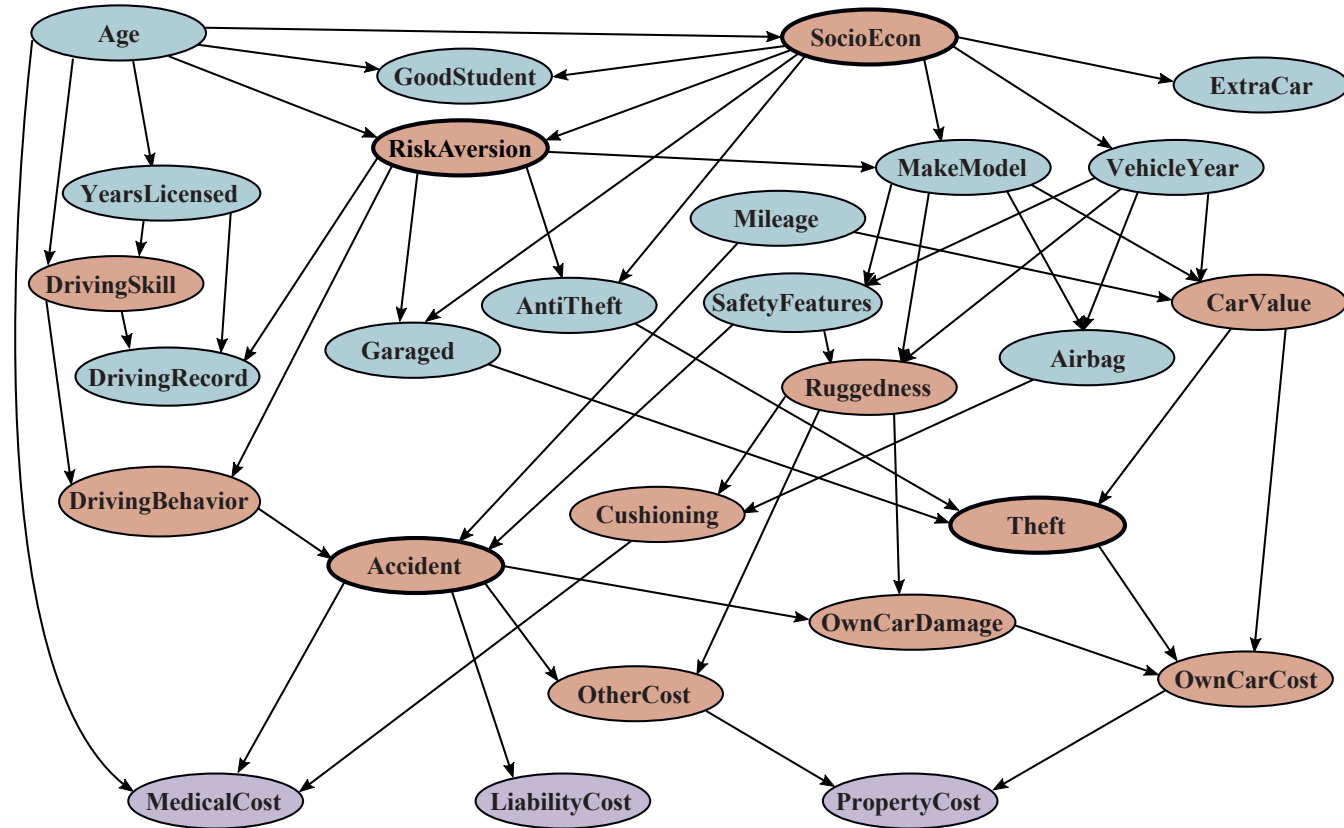


Sample $S \sim P(S | c, r, \neg w)$

Sample $C \sim P(C | s, r)$

Sample $W \sim P(W | s, r)$

Car Insurance: $P(\text{Age} \mid mc, lc, pc)$



Gibbs sampling and MCMC in practice

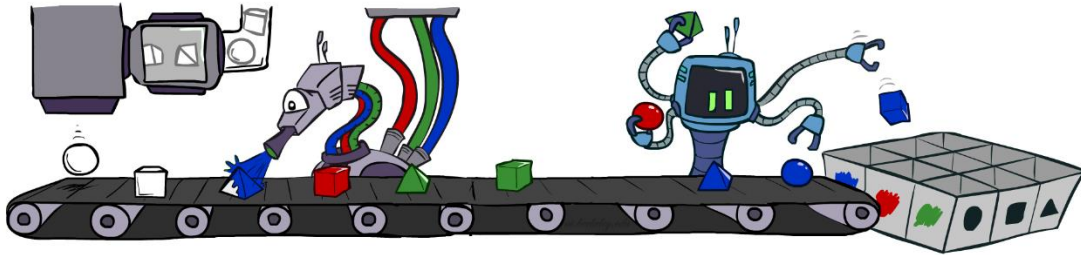
- The most commonly used method for large Bayes' nets
 - See, e.g., BUGS, JAGS, STAN, infer.net, BLOG, etc.
- Can be compiled to run very fast
 - Eliminate all data structure references, just multiply and sample
 - ~100 million samples per second on a laptop
- Can run asynchronously in parallel (one processor per variable)

Why does it work? (see R+N 13.4.2 for details)

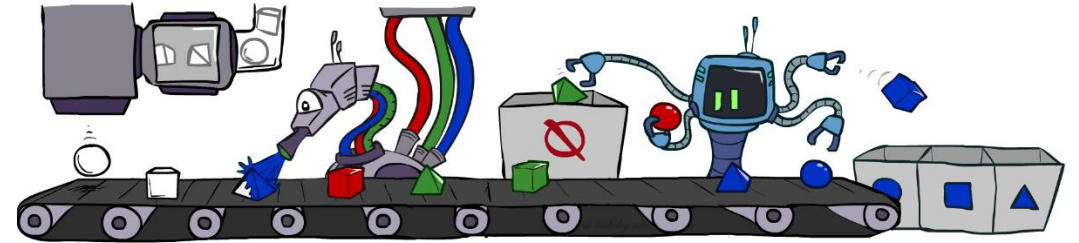
- Suppose we run the Markov chain for a long time and predict the probability of reaching any given state at time t : $\pi_t(x_1, \dots, x_n)$ or $\pi_t(\underline{x})$
- Each Gibbs sampling step (pick a variable, resample its value) applied to a state \underline{x} has a probability $k(\underline{x}' | \underline{x})$ of reaching a next state \underline{x}'
- So $\pi_{t+1}(\underline{x}') = \sum_{\underline{x}} k(\underline{x}' | \underline{x}) \pi_t(\underline{x})$ or, in matrix/vector form $\pi_{t+1} = \mathbf{K}\pi_t$
- When the process is in equilibrium $\pi_{t+1} = \pi_t = \pi$ so $\mathbf{K}\pi = \pi$
- This has a unique* solution $\pi = P(x_1, \dots, x_n | e_1, \dots, e_k)$
 - If the transition model is ergodic (every state is reachable). Otherwise gets stuck
- So for large enough t the next sample will be drawn from the true posterior
 - “Large enough” depends on CPTs in the Bayes net; takes longer if nearly deterministic

Bayes Net Sampling Summary

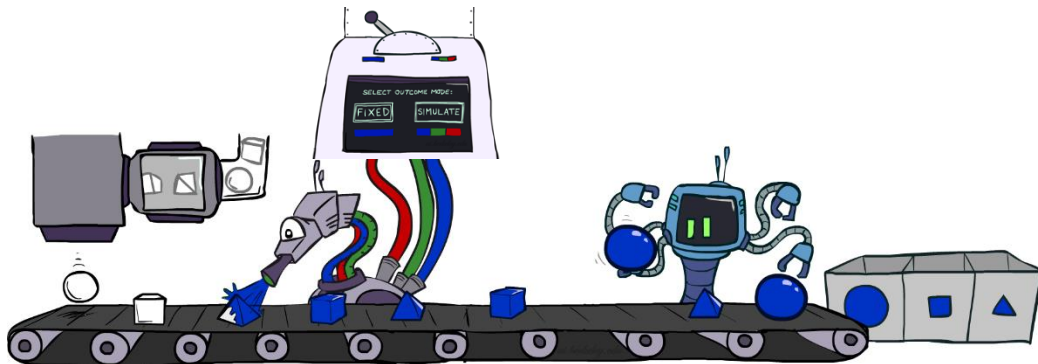
- Prior Sampling P



- Rejection Sampling $P(Q | e)$



- Likelihood Weighting $P(Q | e)$



- Gibbs Sampling $P(Q | e)$

