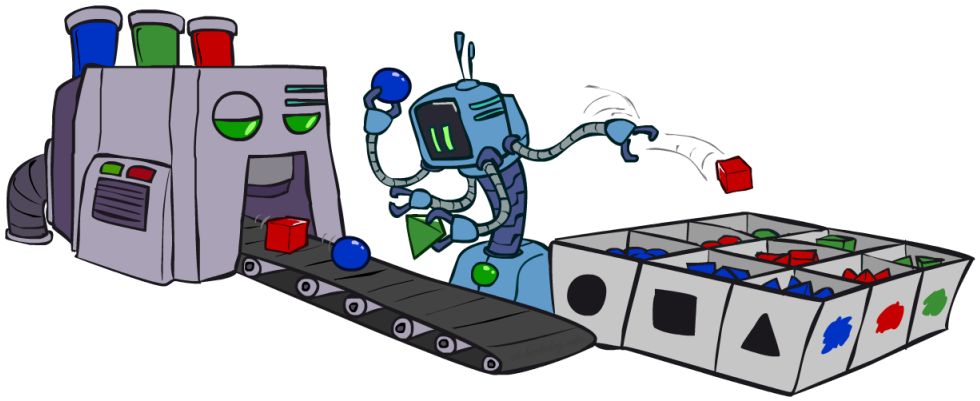
Artificial Intelligence - INFOF311

Bayes nets, approximate inference

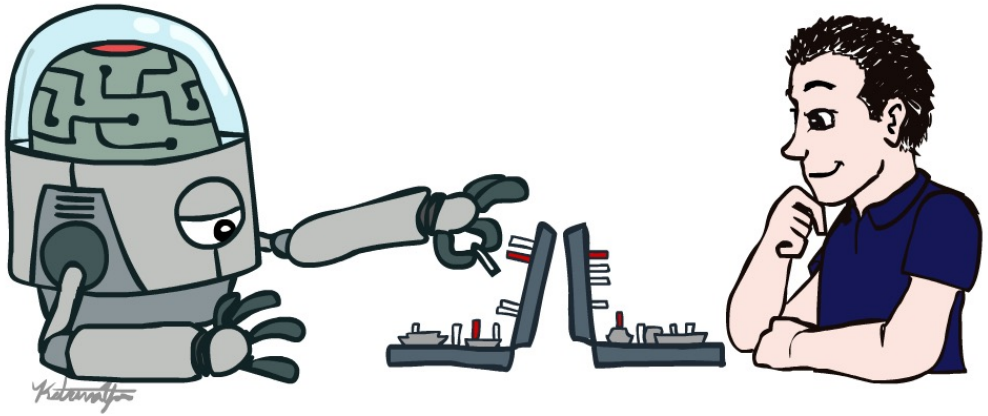


Instructor : Tom Lenaerts

Acknowledgement

We thank Stuart Russell for his generosity in allowing us to use the slide set of the UC Berkeley Course CS188, Introduction to Artificial Intelligence. These slides were created by Dan Klein, Pieter Abbeel and Anca Dragan for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.

Center for Human-Compatible Artificial Intelligence



The slides for INFOF311 are slightly modified versions of the slides of the spring and summer CS188 sessions in 2021 and 2022

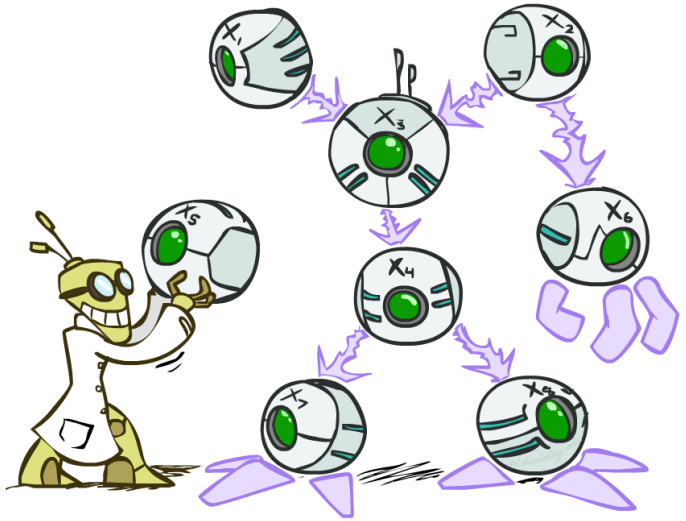
Bayes' Net Representation

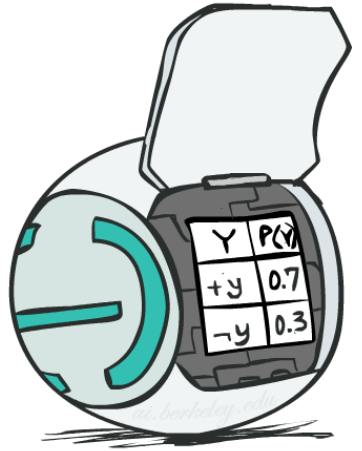
- A directed, acyclic graph, one node per random variable
- A conditional probability table (CPT) for each node
  - A collection of distributions over X, one for each combination of parents' values

$$P(X|a_1 \ldots a_n)$$

- Bayes' nets implicitly encode joint distributions
  - As a product of local conditional distributions
  - To see what probability a BN gives to a full assignment, multiply all the relevant conditionals together:

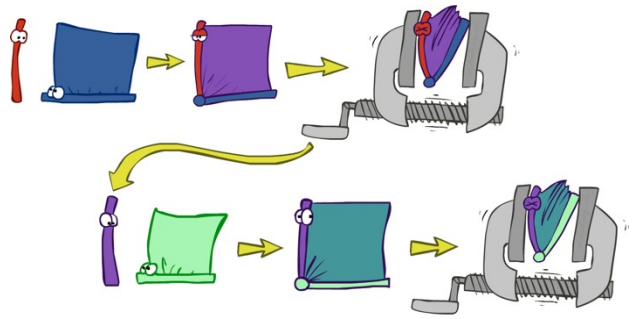
$$P(x_1, x_2, \ldots x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i))$$

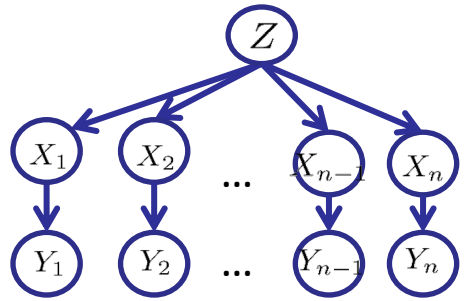


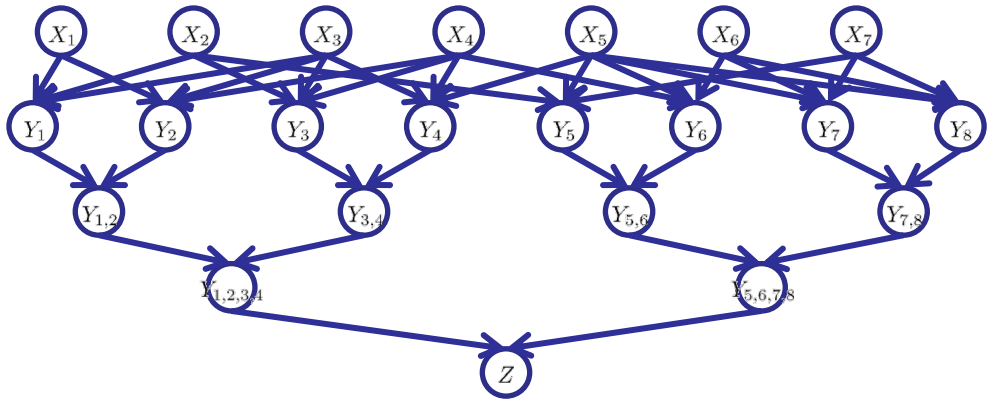


Variable Elimination

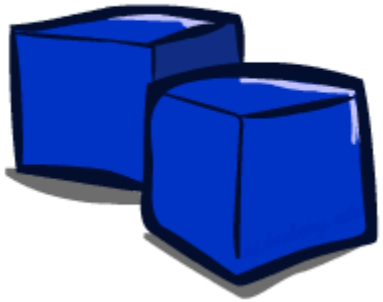
- Interleave joining and marginalizing
- d^k entries computed for a factor over k variables with domain sizes d
- Ordering of elimination of hidden variables can affect size of factors generated
- Worst case: running time exponential in the size of the Bayes' net







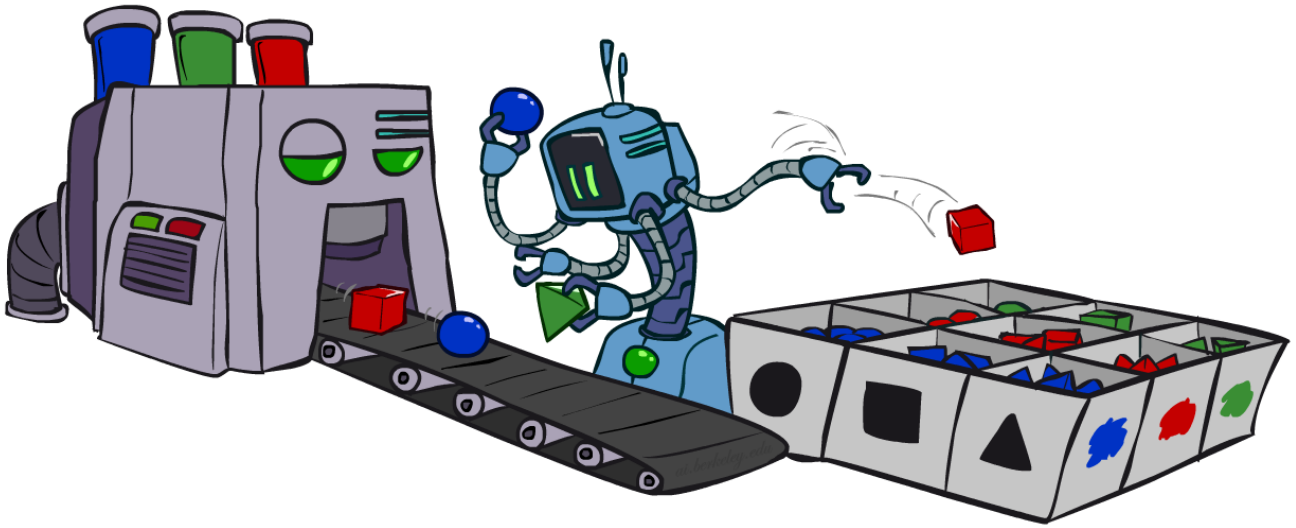
Approximate Inference: Sampling



Sampling

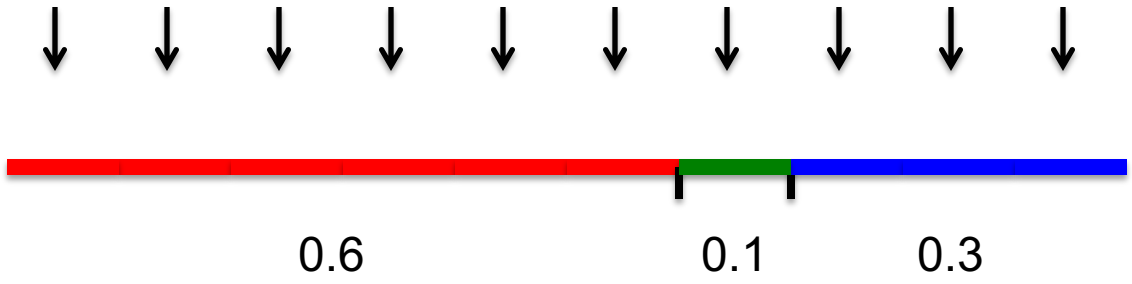
- Basic idea
  - Draw *N* samples from a ***sampling distribution*** *S*
- Sampling is a lot like repeated simulation
  - Predicting the weather, basketball games, …
  - Compute an approximate posterior probability
  - Show this converges to the true probability *P*

- 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)

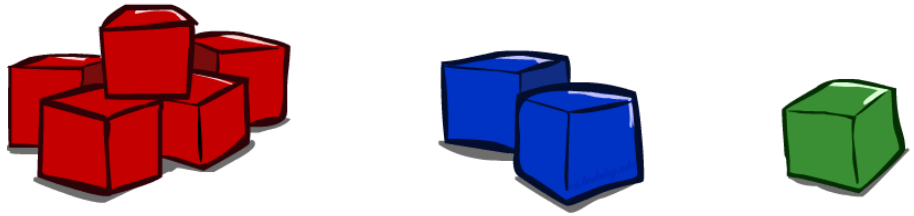


- Example

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

$0.0 \le u < 0.6, \rightarrow C=red$
$0.6 \le u < 0.7, \rightarrow C=green$
$0.7 \le u < 1.0, \rightarrow C=blue$

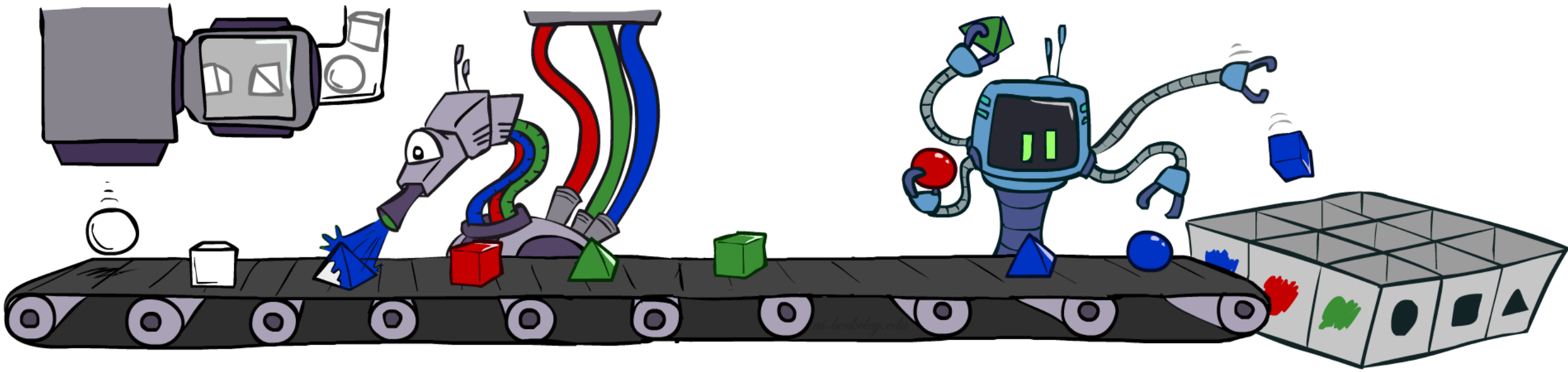
- If random() returns $u = 0.83$, then the sample is *C = blue*
- E.g, after sampling 8 times:



Sampling in Bayes Nets

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

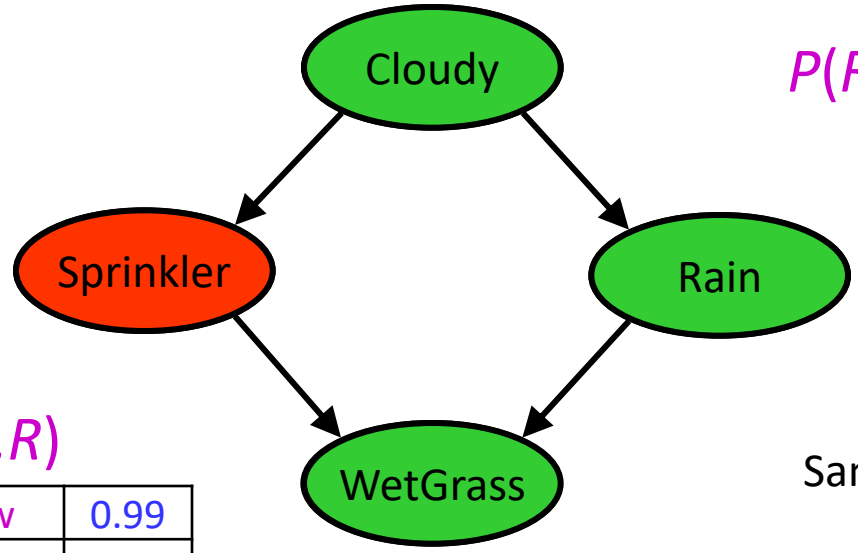
Prior Sampling



Prior Sampling

$P(C)$

c	0.5
¬c	0.5



$P(S \mid C)$

c	s	0.1
	¬s	0.9
¬c	s	0.5
	¬s	0.5

$P(R \mid C)$

c	r	0.8
	¬r	0.2
¬c	r	0.2
	¬r	0.8

$P(W \mid S,R)$

s	r	w	0.99
		¬w	0.01
	¬r	w	0.90
		¬w	0.10
¬s	r	w	0.90
		¬w	0.10
	¬r	w	0.01
		¬w	0.99

Samples:

c, ¬s, r, w

¬c, s, ¬r, w

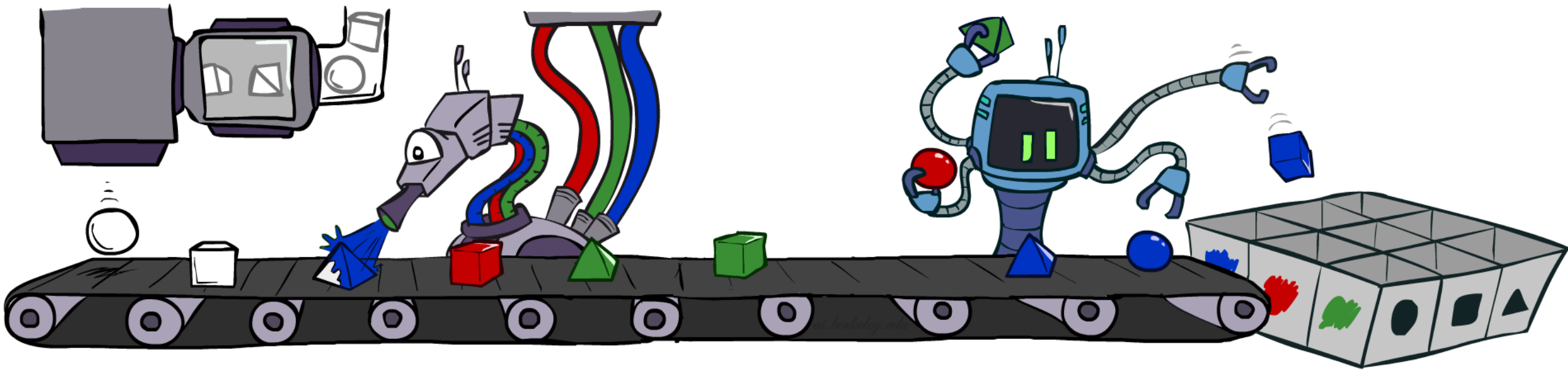
…

$S_{PS}(c, \neg s, r, w) = P(c)\, P(\neg s|c)\, P(r|c)\, P(w|s,r)$

$= P(c, \neg s, r, w)$

Prior Sampling

- For i=1, 2, …, n (in topological order)
  - Sample X_i from $P(X_i \mid parents(X_i))$
- Return $(x_1, x_2, \ldots, x_n)$



Prior Sampling

- This process generates samples with probability:

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

  …i.e. the BN's joint probability

- Let the number of samples of an event be $N_{PS}(x_1,\ldots,x_n)$
- Estimate from N samples is $Q_N(x_1,\ldots,x_n) = N_{PS}(x_1,\ldots,x_n)/N$
- Then $\lim_{N\to\infty} Q_N(x_1,\ldots,x_n) = \lim_{N\to\infty} N_{PS}(x_1,\ldots,x_n)/N$

  $= S_{PS}(x_1,\ldots,x_n)$

  $= P(x_1,\ldots,x_n)$

- I.e., the sampling procedure is ***consistent***

Example

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

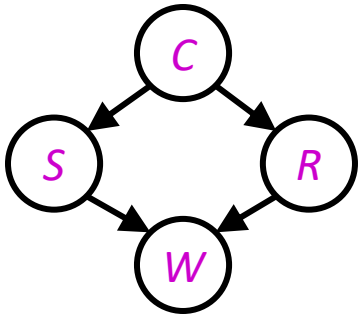
  c, ¬s, r, w

  c, s, r, w

  ¬c, s, r, ¬w

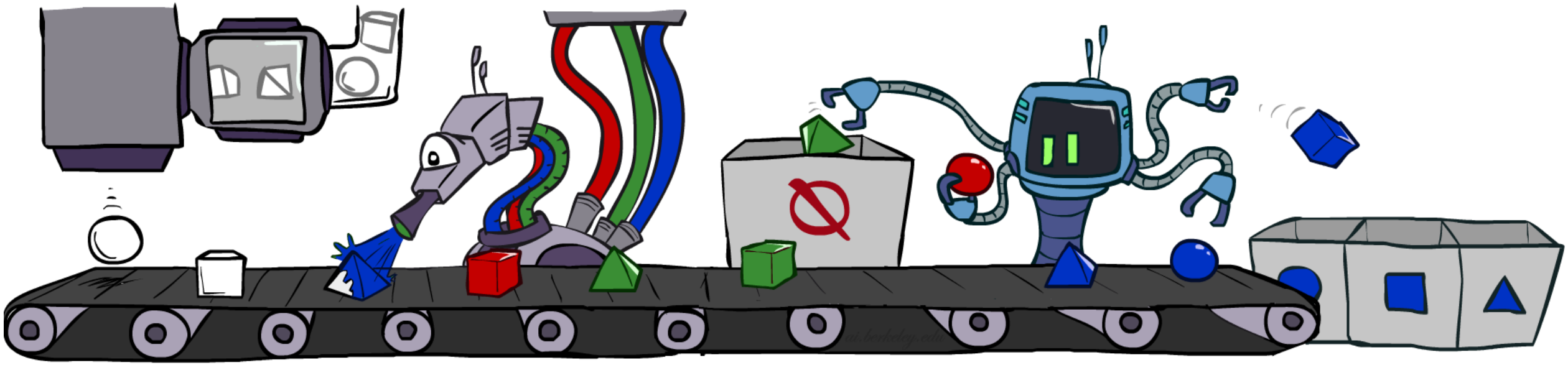
  c, ¬s, r, w

  ¬c, ¬s, ¬r, w



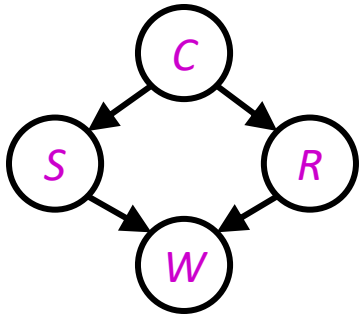
- If we want to know *P*(*W*)
  - We have counts <w:4, ¬w:1>
  - Normalize to get *P*(*W*) = <w:0.8, ¬w:0.2>
  - This will get closer to the true distribution with more samples
  - Can estimate anything else, too
    - P(C | w)? P(C | w,r)? P(C | ¬ w, ¬ r)?

Rejection Sampling



Rejection Sampling

- A simple application of prior sampling for estimating conditional probabilities
  - Let's say we want $P(C \mid r, w) = \alpha P(C, r, w)$
  - For these counts, samples with $\neg r$ or $\neg w$ ***are not relevant***
  - So count the C outcomes for samples with r, w and reject all other samples
- This is called ***rejection sampling***
  - It is also consistent for conditional probabilities (i.e., correct in the limit)



c, ¬s, r, w

~~c, s, ¬r~~

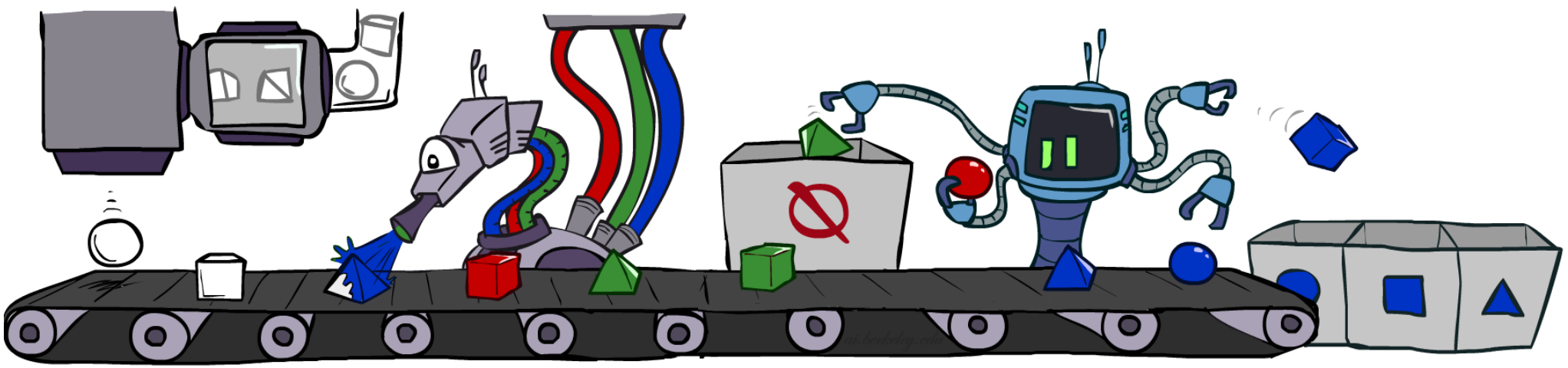
~~¬c, s, r, ¬w~~

~~c, ¬s, ¬r~~

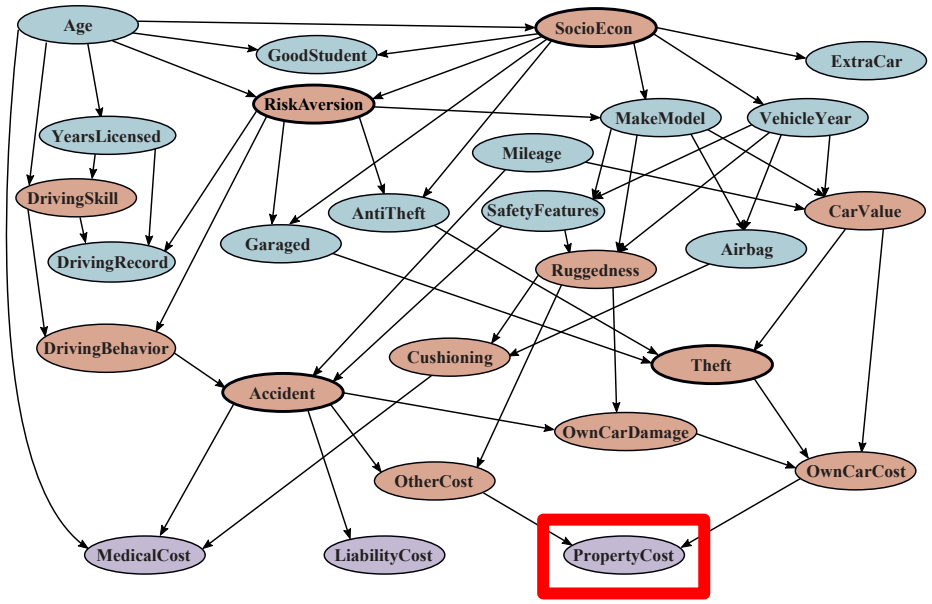
¬c, ¬s, r, w

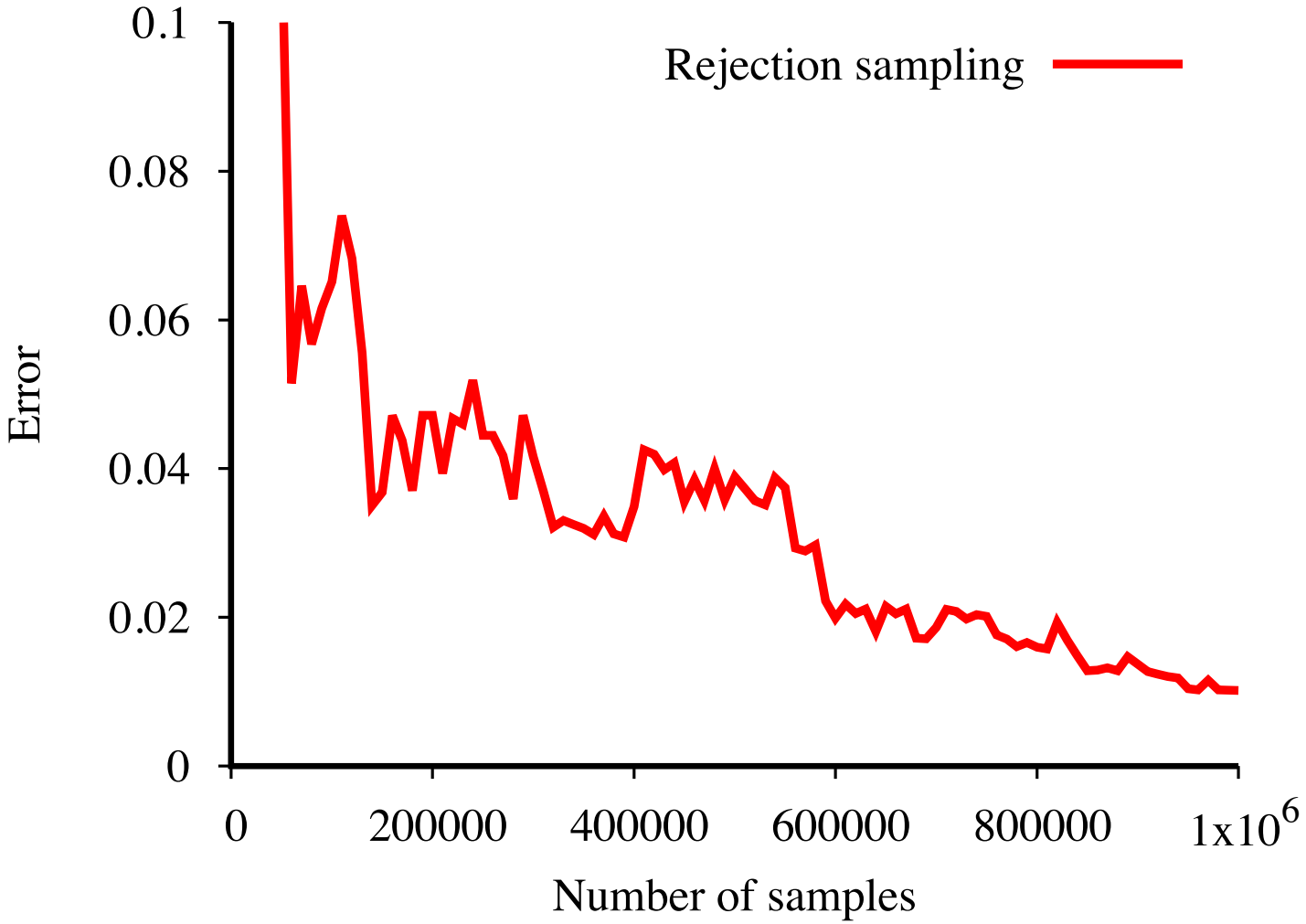
Rejection Sampling

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

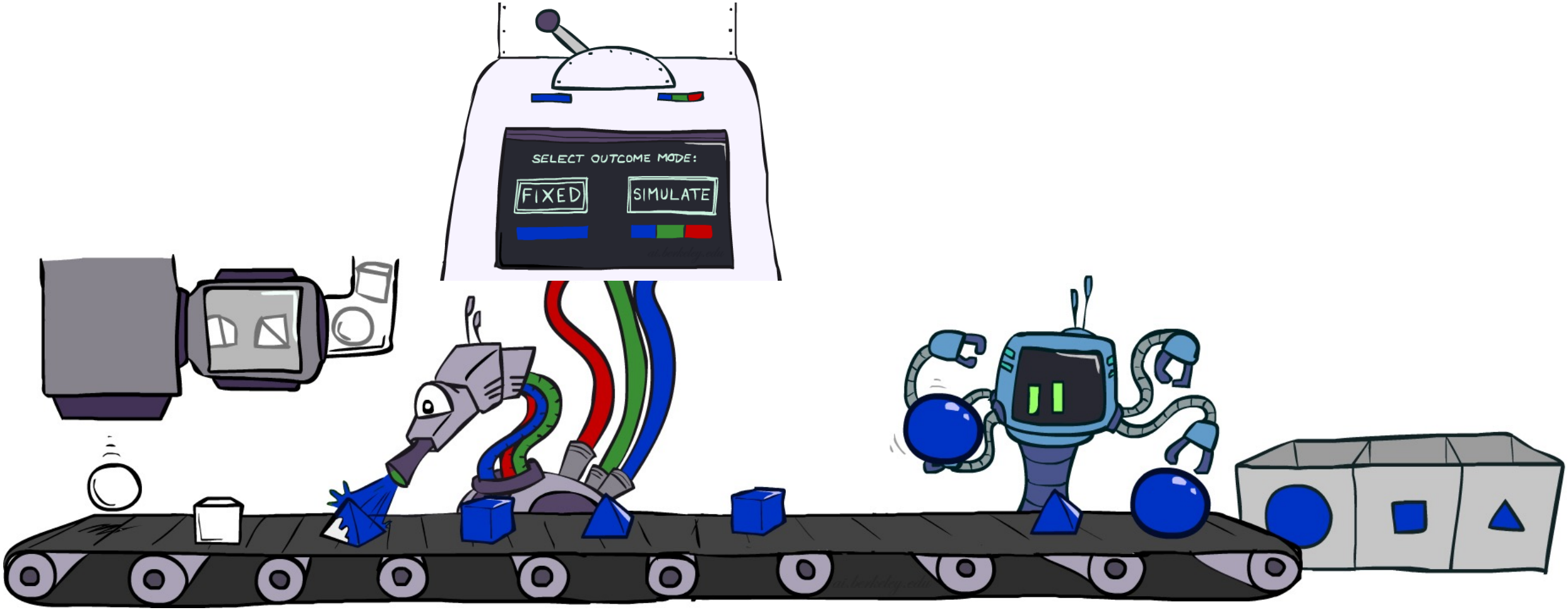


Car Insurance: *P(PropertyCost | **e**)*



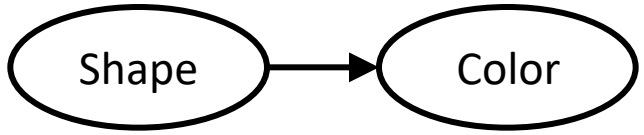


Likelihood Weighting

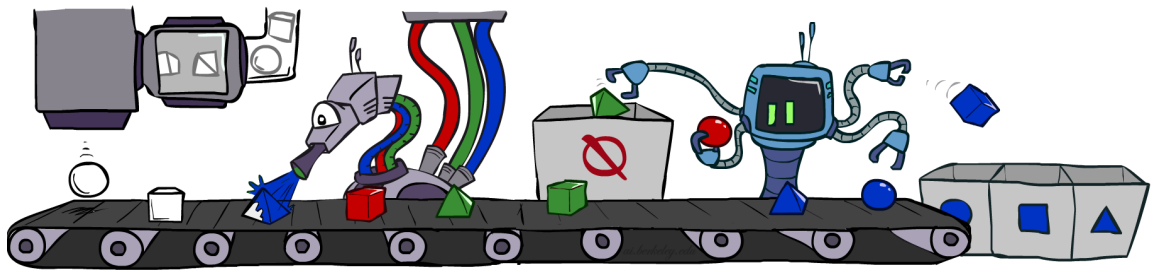


Likelihood Weighting

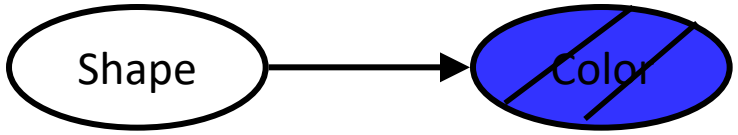
- Problem with rejection sampling:
  - If evidence is unlikely, rejects lots of samples
  - Evidence not exploited as you sample
  - Consider P(*Shape*|*Color=blue*)



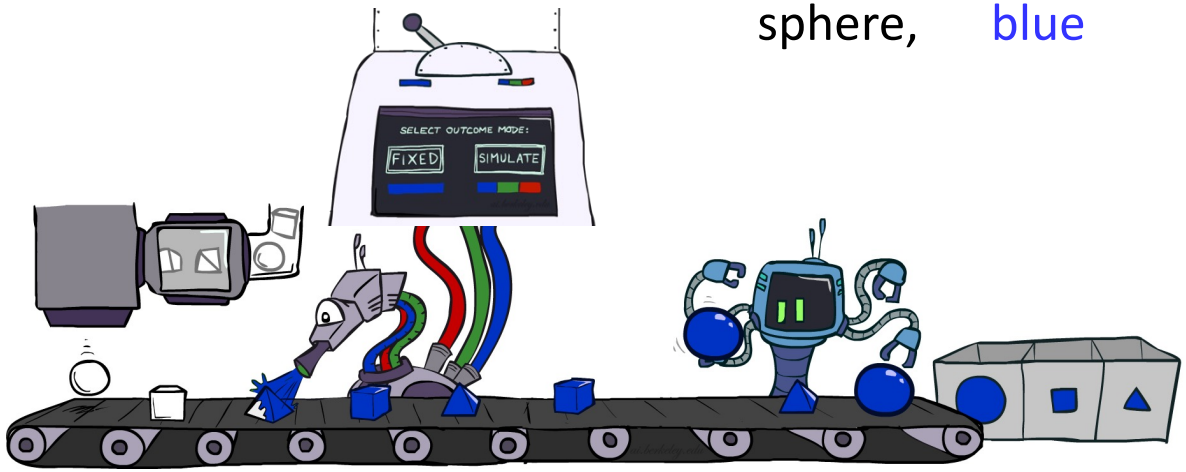
~~pyramid, green~~
~~pyramid, red~~
sphere, blue
~~cube, red~~
~~sphere, green~~



- Idea: fix evidence variables, sample the rest
  - Problem: sample distribution **not** consistent!
  - Solution: ***weight*** each sample by probability of evidence variables given parents



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



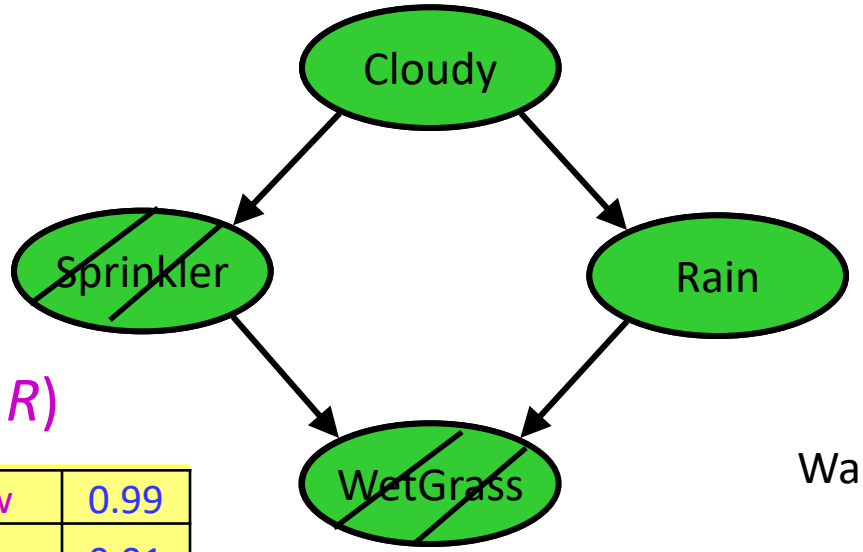
Likelihood Weighting

$P(C)$

c	0.5
¬c	0.5

$P(S \mid C)$

c	s	0.1
	¬s	0.9
¬c	s	0.5
	¬s	0.5



$P(R \mid C)$

c	r	0.8
	¬r	0.2
¬c	r	0.2
	¬r	0.8

$P(W \mid S,R)$

s	r	w	0.99
		¬w	0.01
	¬r	w	0.90
		¬w	0.10
¬s	r	w	0.90
		¬w	0.10
	¬r	w	0.01
		¬w	0.99

Want to estimate P (C | s, w)

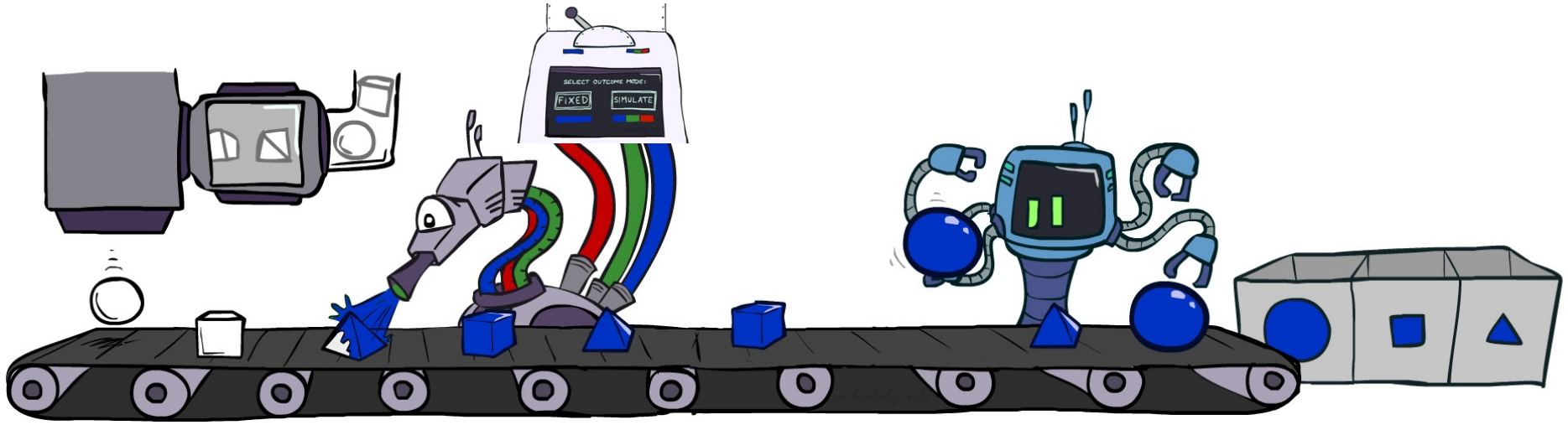
Samples:

c , s, r, w ***w* = 1.0 x 0.1 x 0.99**

¬c , s,¬r, w ***w* = 1.0 x 0.5 x 0.9**

Likelihood Weighting

- Input: evidence $e_1,..,e_k$
- $w = 1.0$
- for i=1, 2, …, n in topological order
  - if X_i is an evidence variable
    - x_i = observed value$_i$ for X_i
    - Set $w = w * P(x_i \mid parents(X_i))$
  - else
    - Sample x_i from $P(X_i \mid parents(X_i))$
- return $(x_1, x_2, …, x_n)$, w



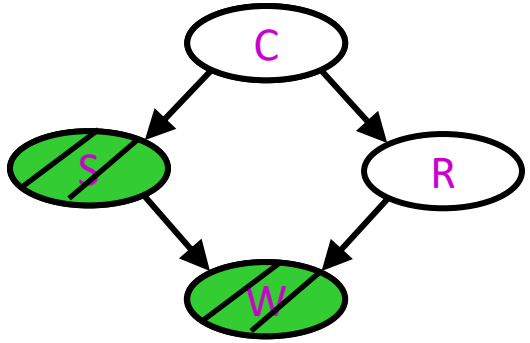
Likelihood Weighting

- Sampling distribution if **Z** sampled and **e** fixed evidence

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

- Now, samples have weights

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

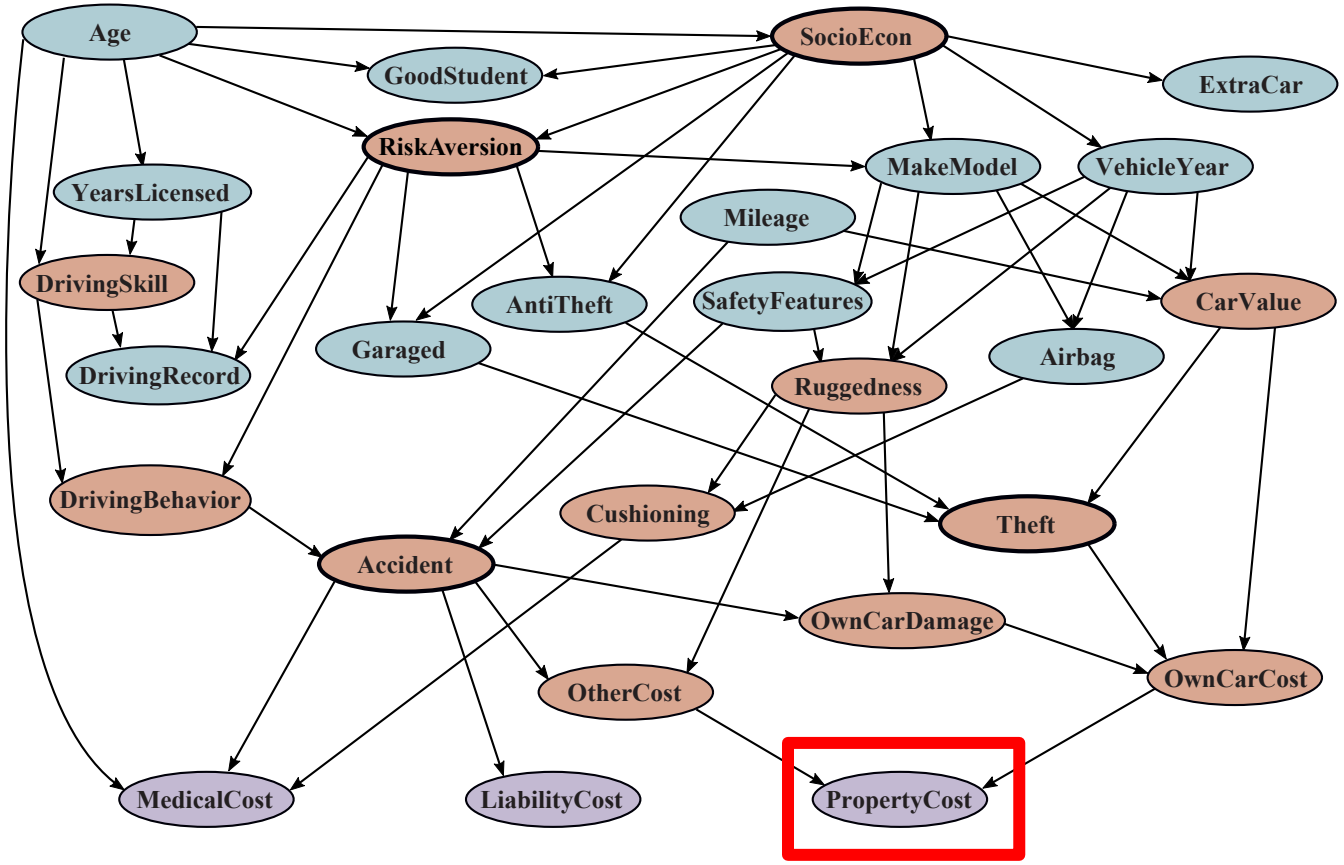


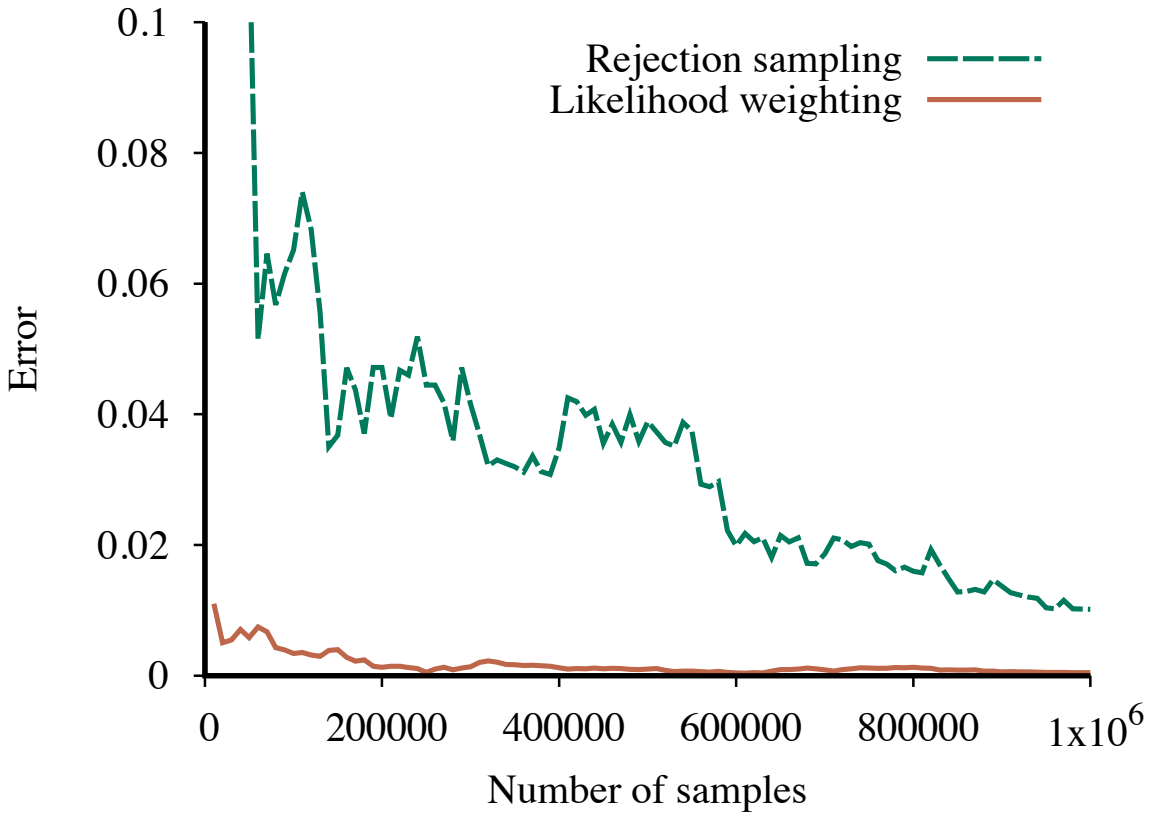
- Together, weighted sampling distribution is consistent

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

- 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 $P(x)/Q(x)$

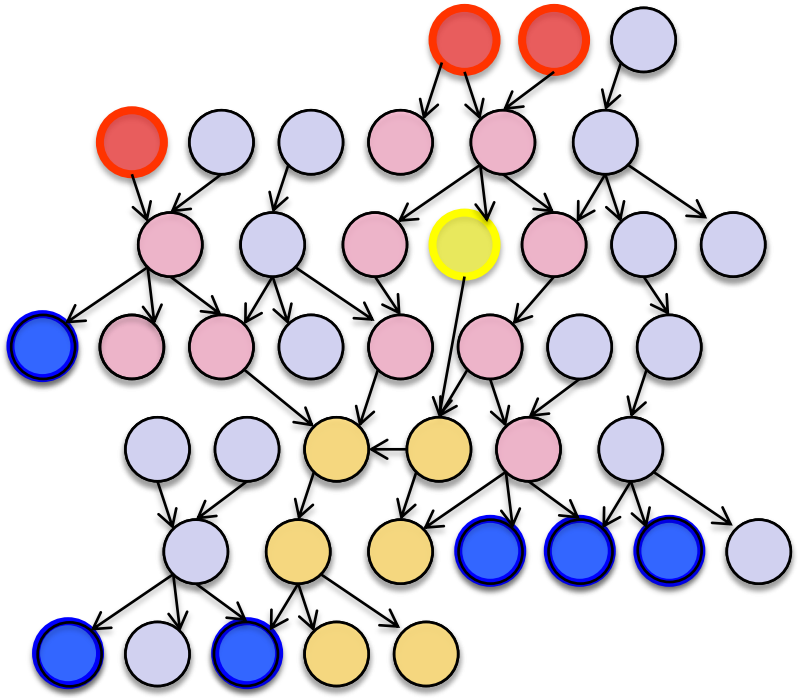
Car Insurance: $P(PropertyCost \mid \mathbf{e})$





Likelihood Weighting

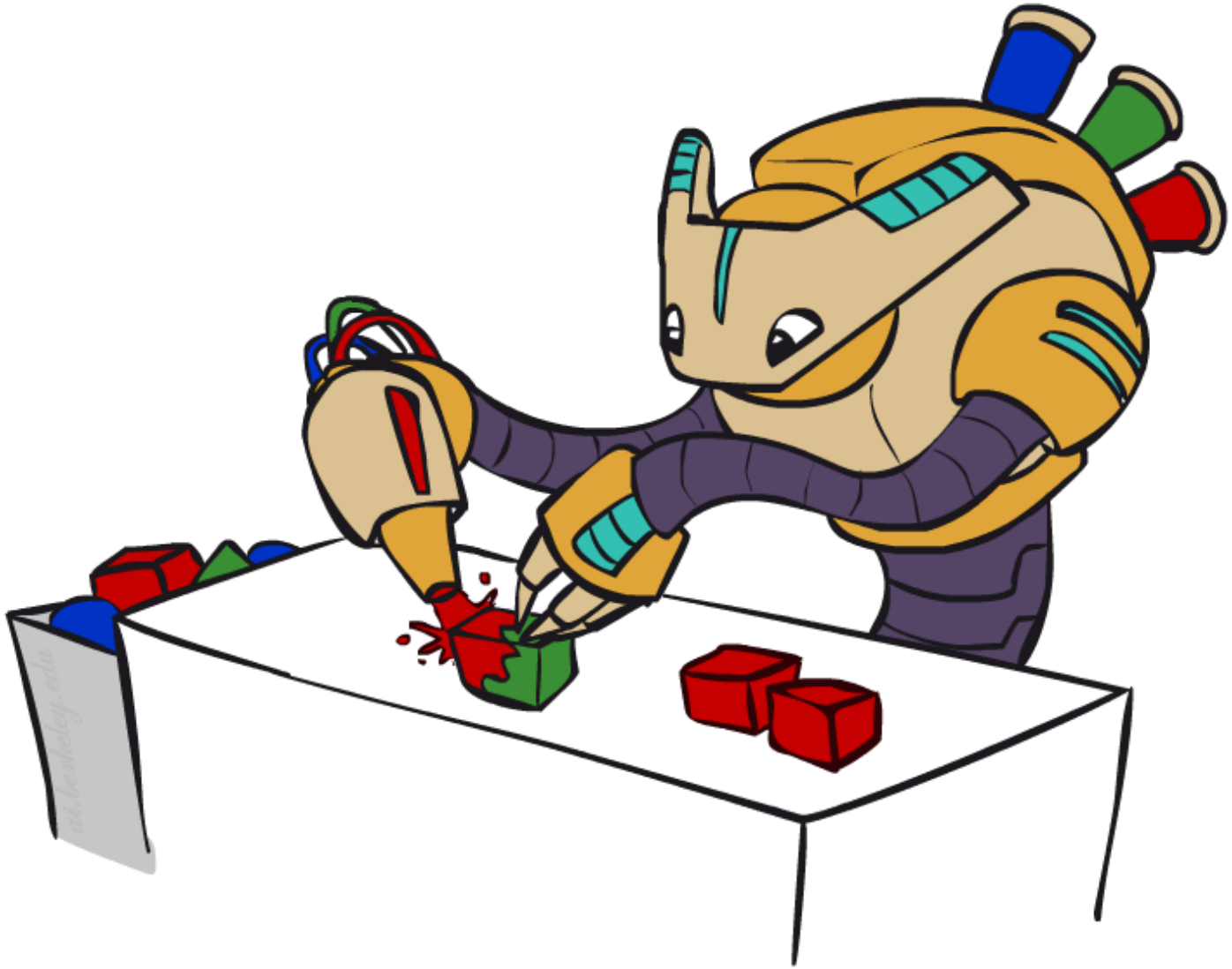
- Likelihood weighting is good
  - All samples are used (with a focus on the evidence)
    - Samples will reflect state of the world
  - The values of ***downstream*** variables are influenced by ***upstream*** evidence
    - E.g. W's value is picked based on the evidence



- 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
  - 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

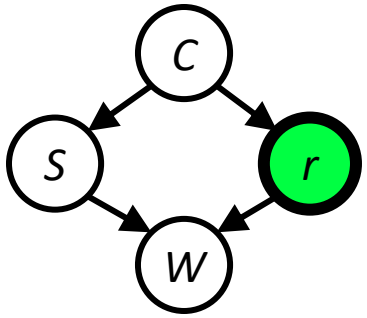
- We would like each variable to "see" ***all*** the evidence!

Gibbs Sampling

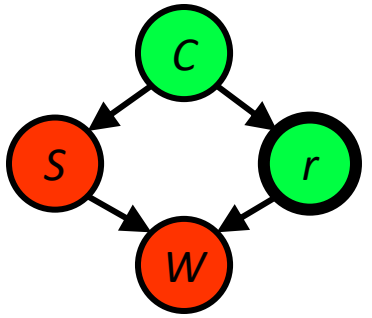


Gibbs Sampling Example: $P(S \mid r)$

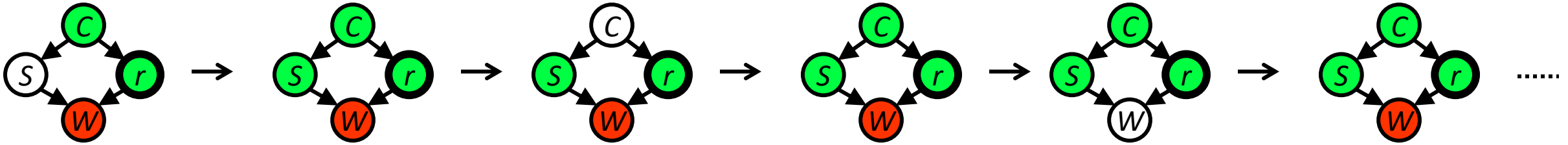
- Step 1: Fix evidence
  - R = true



- Step 2: Initialize other variables
  - Randomly



- Step 3: Repeat
  - Choose a non-evidence variable X
  - Resample X from $P(X \mid \text{all other variables})$



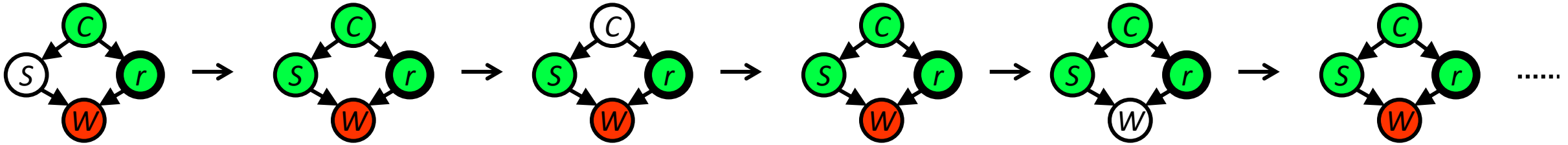
Sample from $P(S \mid c, r, \neg w)$

Sample from $P(C \mid s, r, \neg w)$

Sample from $P(W \mid s, r, c)$

Gibbs Sampling

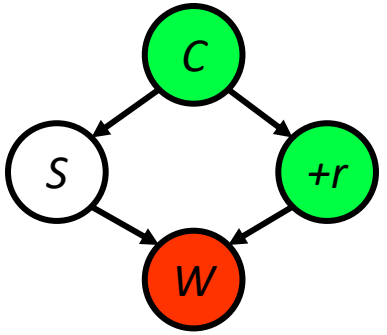
- *Procedure:* keep track of a full instantiation $x_1, x_2, \ldots, x_n$. Start with an arbitrary instantiation consistent with the evidence. Sample one variable at a time, conditioned on all the rest, but keep evidence fixed. Keep repeating this for a long time.
- *Property:* in the limit of repeating this infinitely many times the resulting samples come from the correct distribution (i.e. conditioned on evidence).
- *Rationale*: both upstream and downstream variables condition on evidence.
- In contrast: likelihood weighting only conditions on upstream evidence, and hence weights obtained in likelihood weighting can sometimes be very small. Sum of weights over all samples is indicative of how many "effective" samples were obtained, so we want high weight.



Resampling of One Variable

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

$$P(S|+c,+r,-w)$$



- Many things cancel out – only CPTs with S remain!
- More generally: only CPTs that have resampled variable need to be considered, and joined together
- **Note the Markov blanket of S !**

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

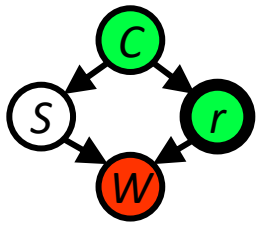
- A particular kind of MCMC
  - States are complete assignments to all variables
    - (Cf local search: closely related to simulated annealing!)
  - Evidence variables remain fixed, other variables change
  - To generate the next state, pick a variable and sample a value for it conditioned on all the other variables: $X_i' \sim P(X_i \mid x_1,..,x_{i-1},x_{i+1},..,x_n)$
    - Will tend to move towards states of higher probability, but can go down too
    - In a Bayes net, $P(X_i \mid x_1,..,x_{i-1},x_{i+1},..,x_n) = P(X_i \mid markov\_blanket(X_i))$
- Theorem: Gibbs sampling is consistent*
  - Provided all Gibbs distributions are bounded away from 0 and 1 and variable selection is fair

Resampling of One Variable

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

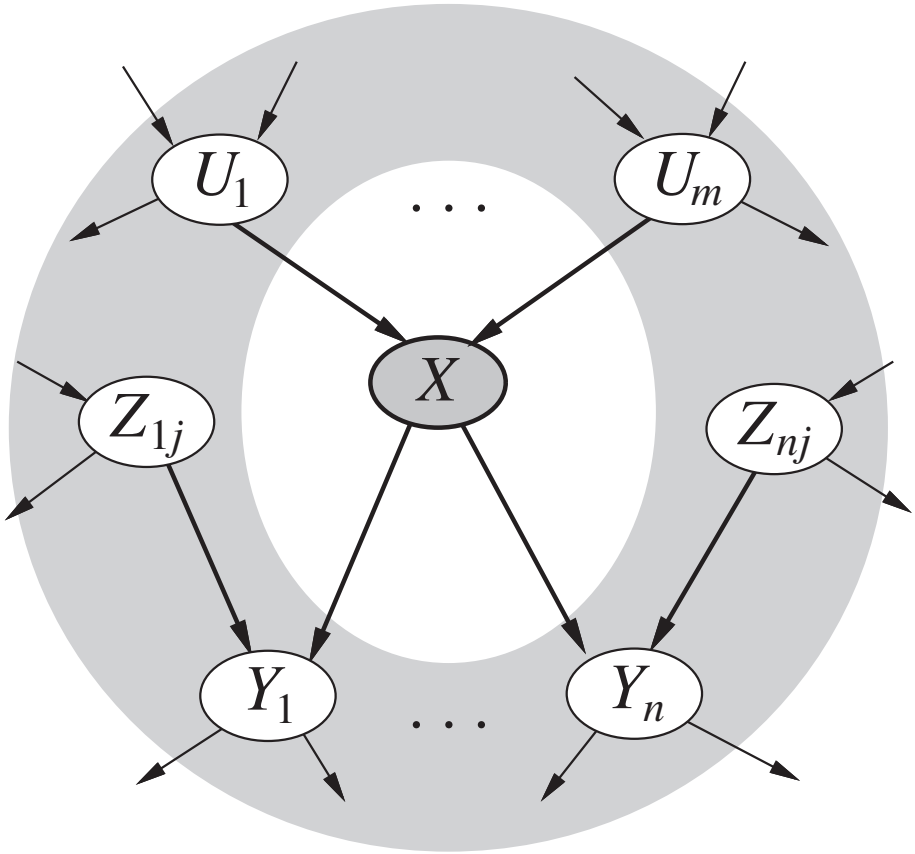
$P(X_i \mid x_1,..,x_{i-1},x_{i+1},..,x_n) = P(X_i \mid markov\_blanket(X_i))$

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

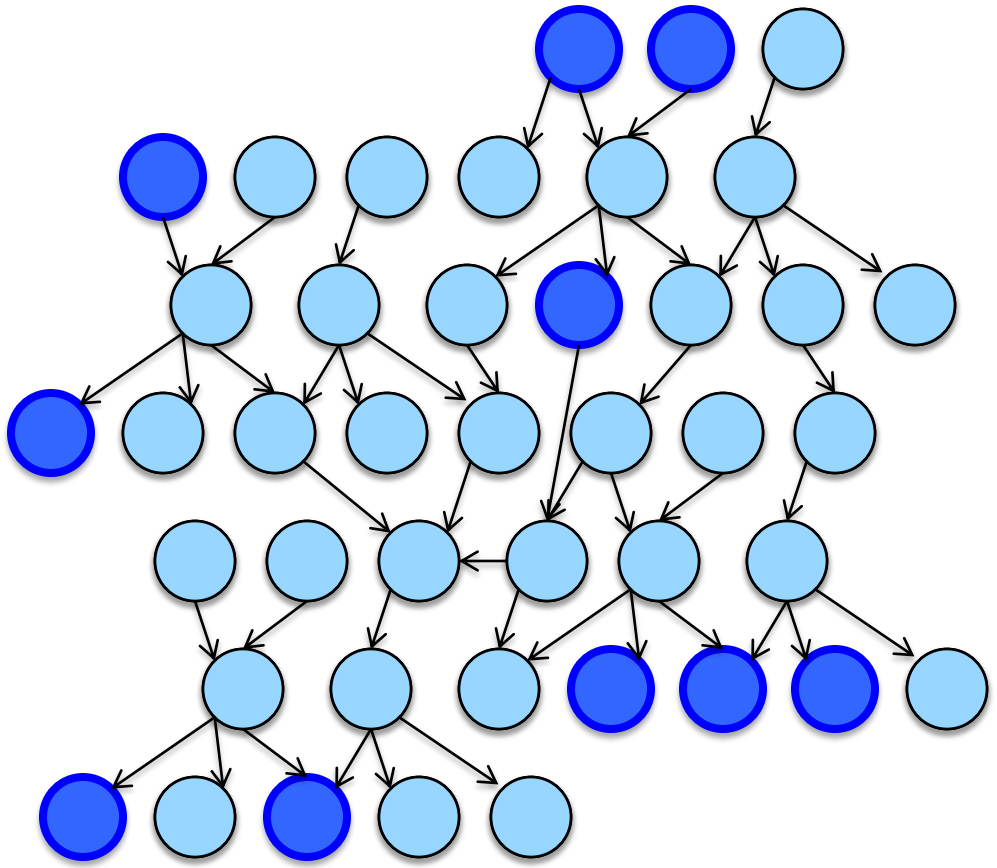


Remember the example:

$P(S \mid c, r, \neg w) = \alpha\ P(S \mid parents(S))\ \prod_w P(w \mid parents(w))$
$= \alpha\ P(S \mid c)\ P(\neg w \mid S,r)$



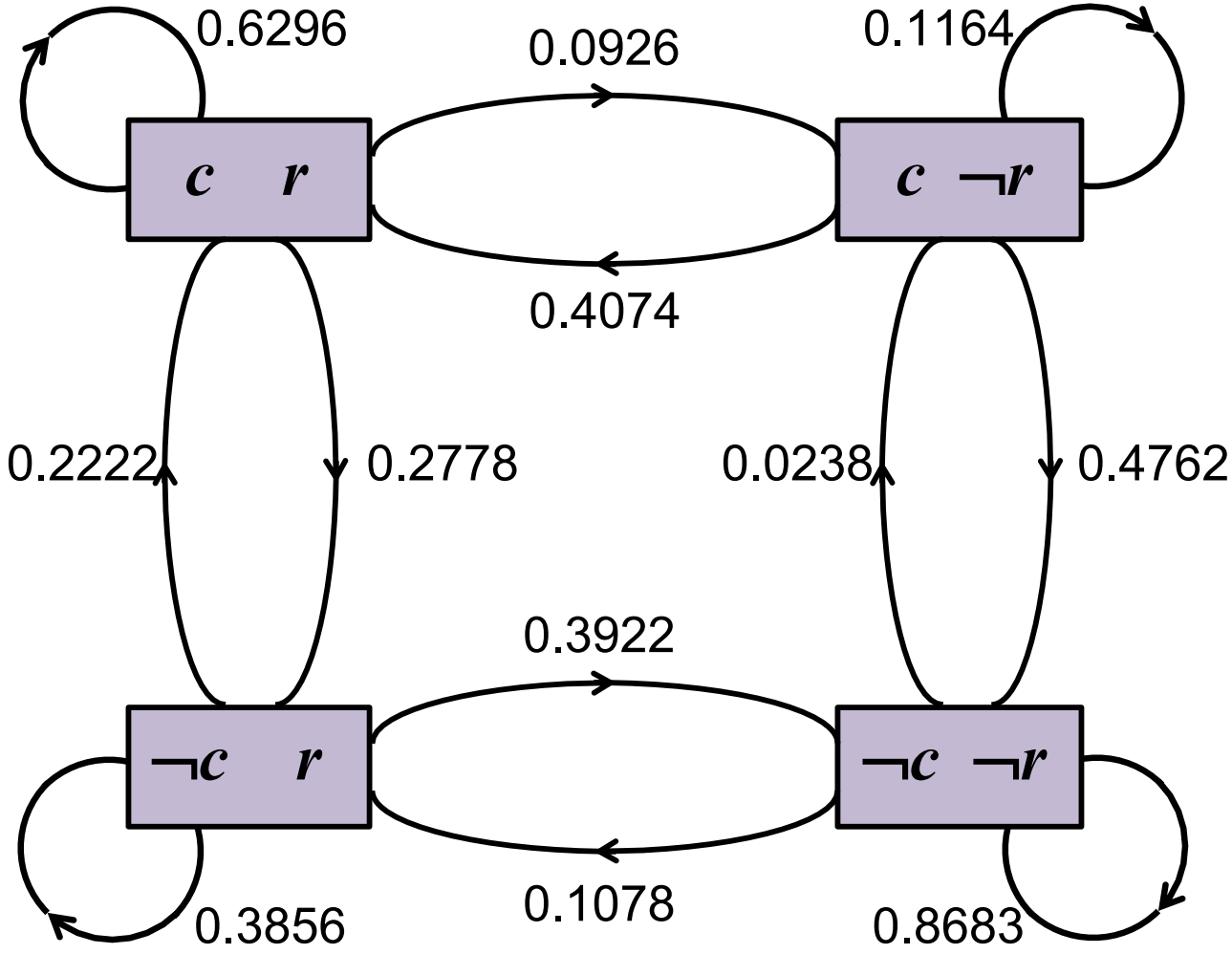
Advantages of MCMC



Samples soon begin to reflect all the evidence in the network

Eventually they are being drawn from the true posterior!

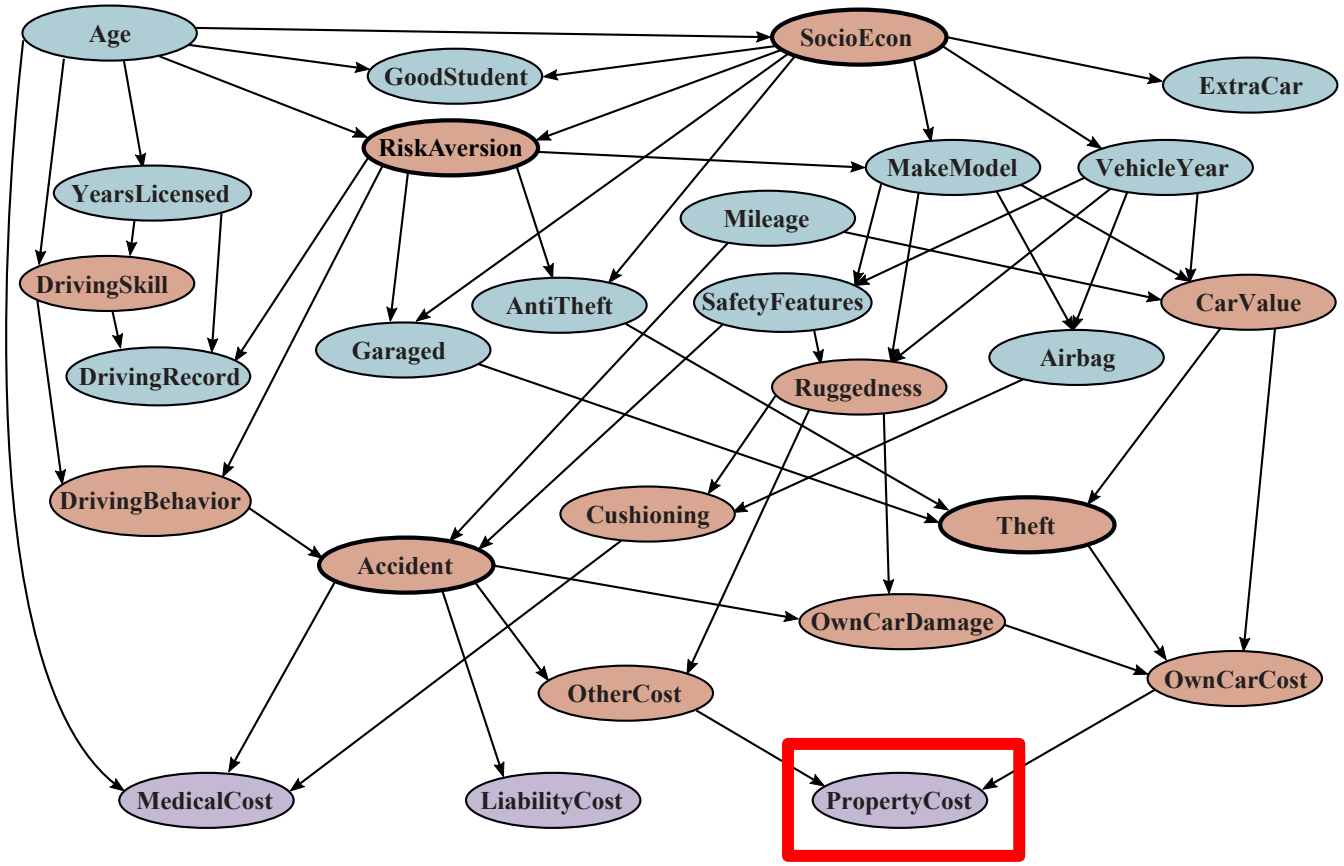
Markov chain given *s*, *w*

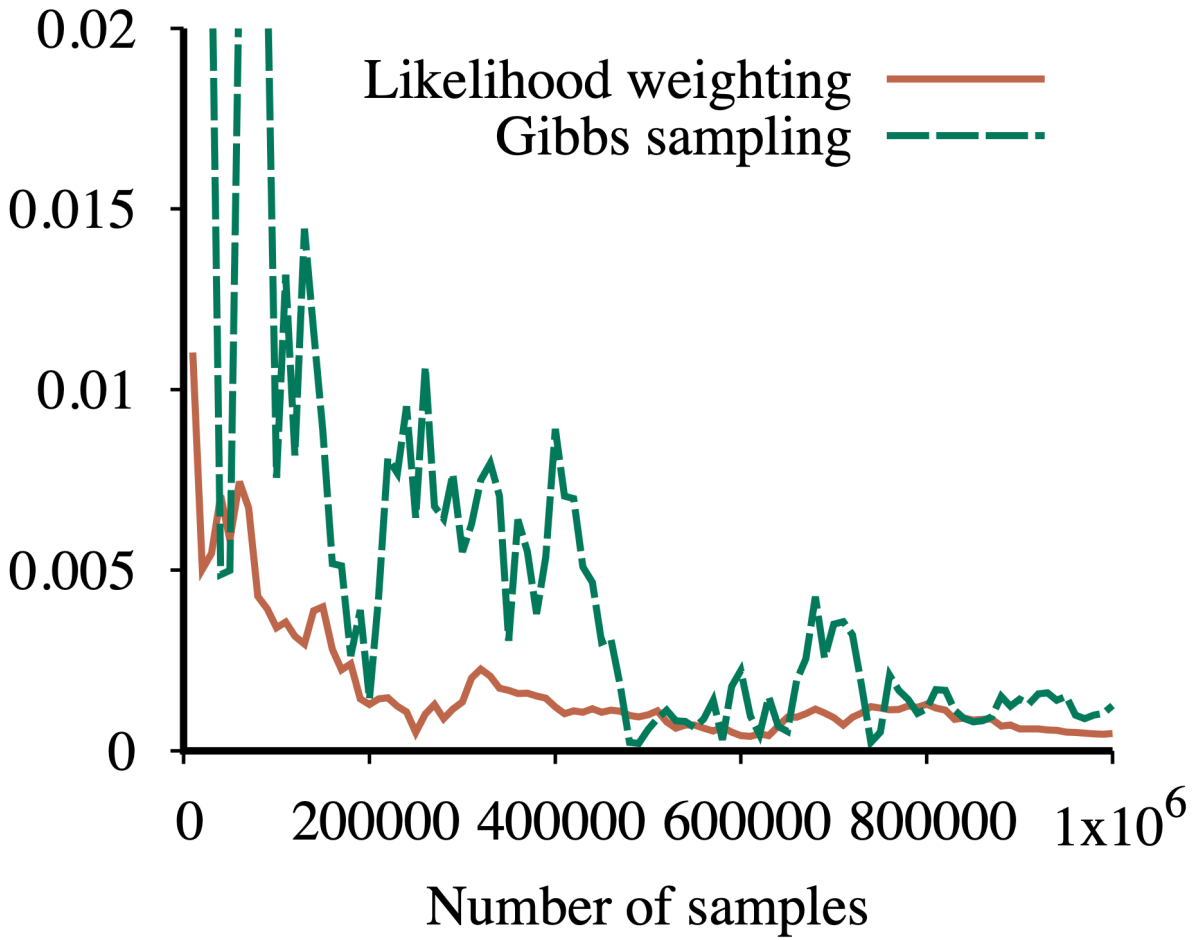


Gibbs sampling and MCMC in practice

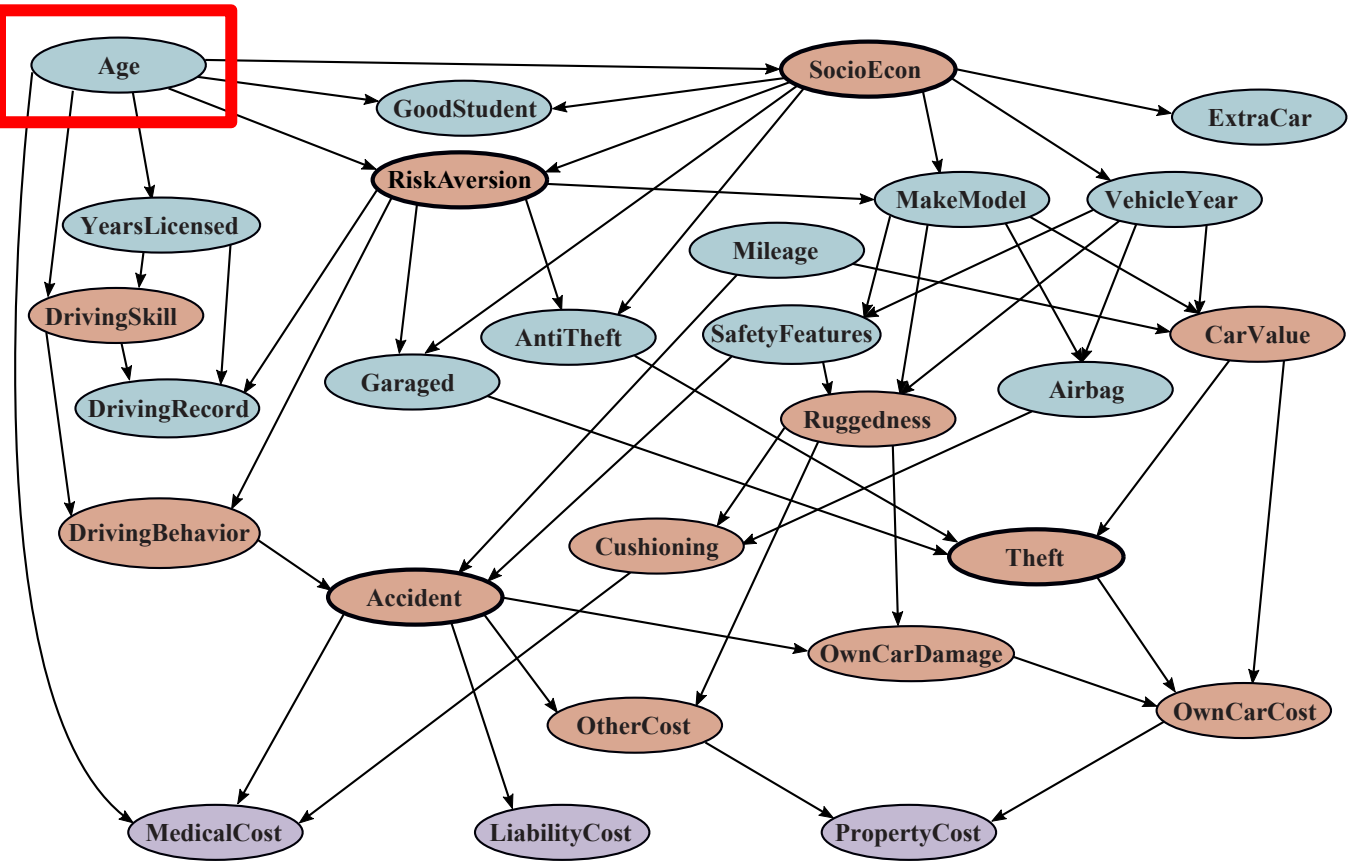
- The most commonly used method for large Bayes nets
  - See, e.g., BUGS, JAGS, STAN, infer.net, BLOG, etc.
- Can be *compiled* to run very fast
  - Eliminate all data structure references, just multiply and sample
  - ~100 million samples per second on a laptop
- Can run asynchronously in parallel (one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

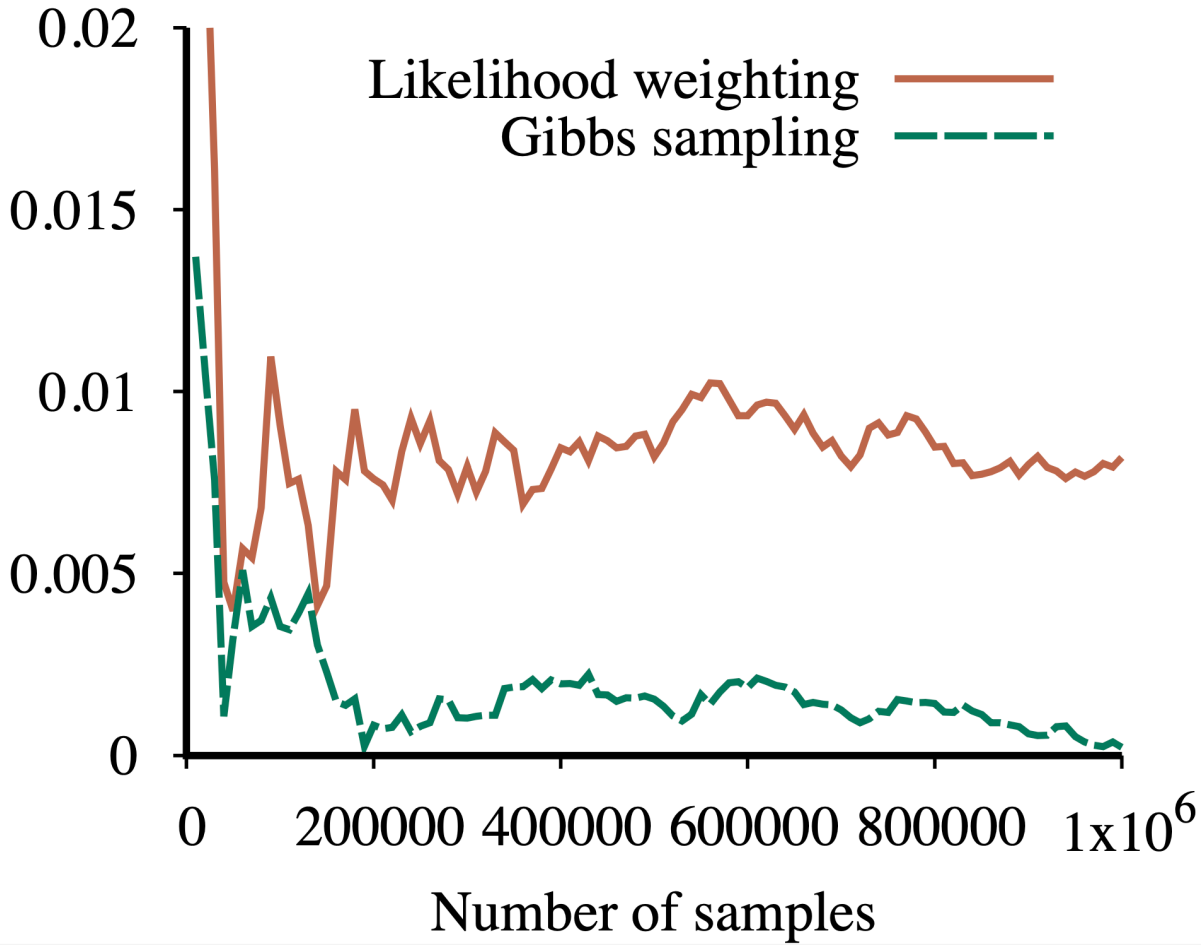
Car Insurance: $P(PropertyCost \mid \boldsymbol{e})$





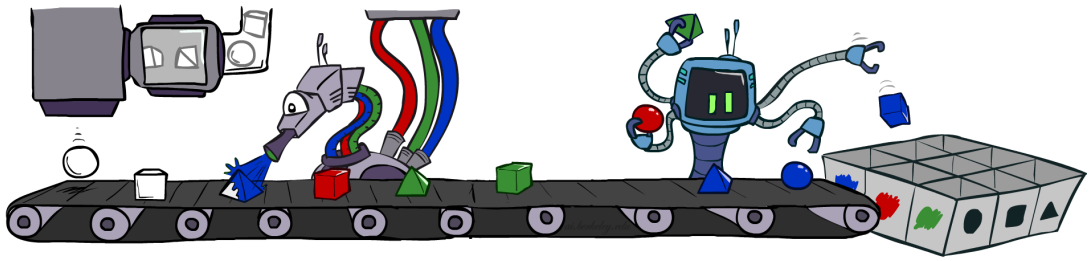
Car Insurance: *P(Age | **costs**)*



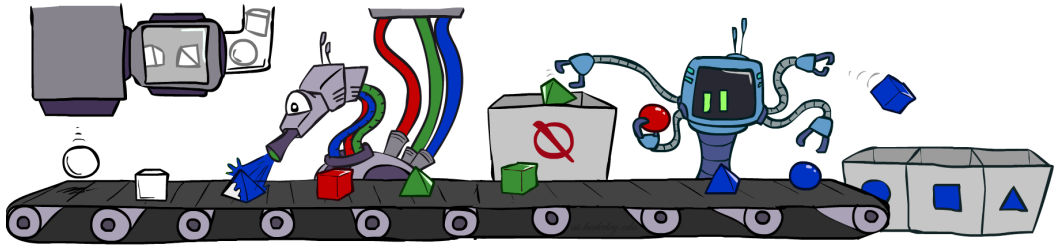


Bayes Net Sampling Summary

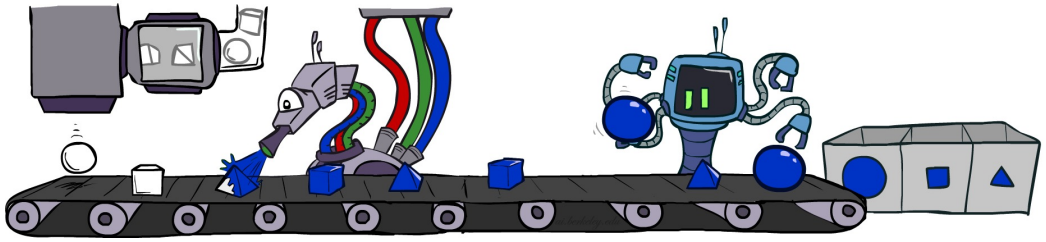
- Prior Sampling P :
  - Generate complete samples from $P(x_1,\ldots,x_n)$



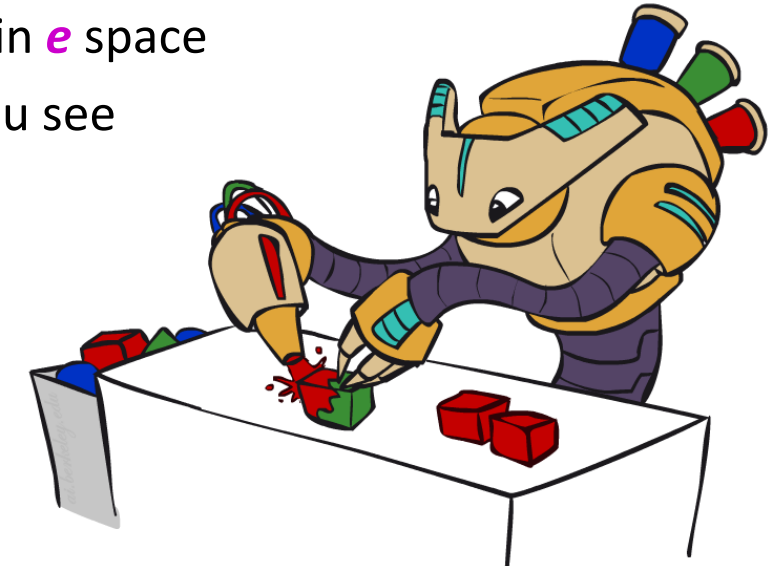
- Rejection Sampling $P(Q \mid \boldsymbol{e})$:
  - Reject samples that don't match $\boldsymbol{e}$



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



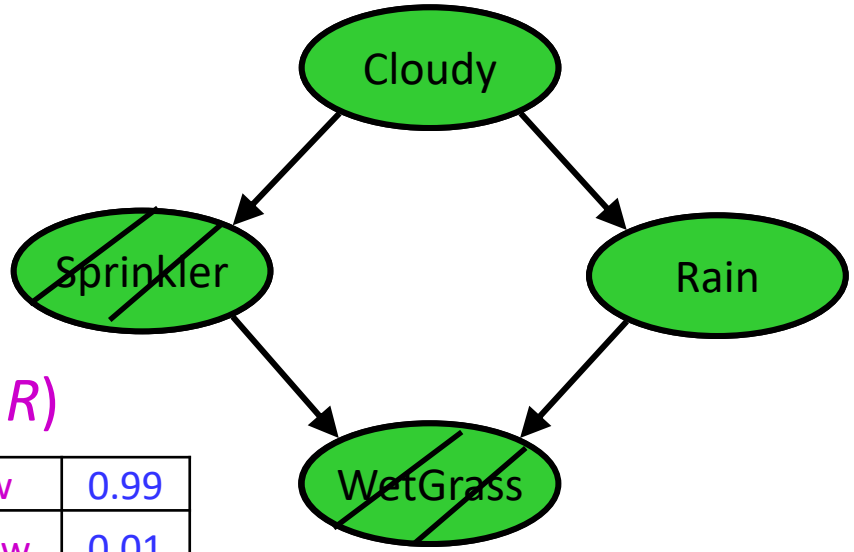
- Gibbs sampling $P(Q \mid \boldsymbol{e})$:
  - Wander around in $\boldsymbol{e}$ space
  - Average what you see



Gibbs Sampling

$P(C)$

c	0.5
¬c	0.5



$P(S \mid C)$

c	s	0.1
	¬s	0.9
¬c	s	0.5
	¬s	0.5

$P(R \mid C)$

c	r	0.8
	¬r	0.2
¬c	r	0.2
	¬r	0.8

$P(W \mid S,R)$

s	r	w	0.99
		¬w	0.01
	¬r	w	0.90
		¬w	0.10
¬s	r	w	0.90
		¬w	0.10
	¬r	w	0.01
		¬w	0.99

Want to estimate P (C | s, w)

(Arbitrarily) Pick R to resample

P (R | c, s, w) = α P(R | c) P (w | s,R)

(Arbitrarily) Pick C to resample

P (C | r, s, w) = α P(C) P(s | C) P (r | C)

Samples:

c, s, r, w

c, s, r, w

¬ c, s, r, w