

Quick Warm-Up

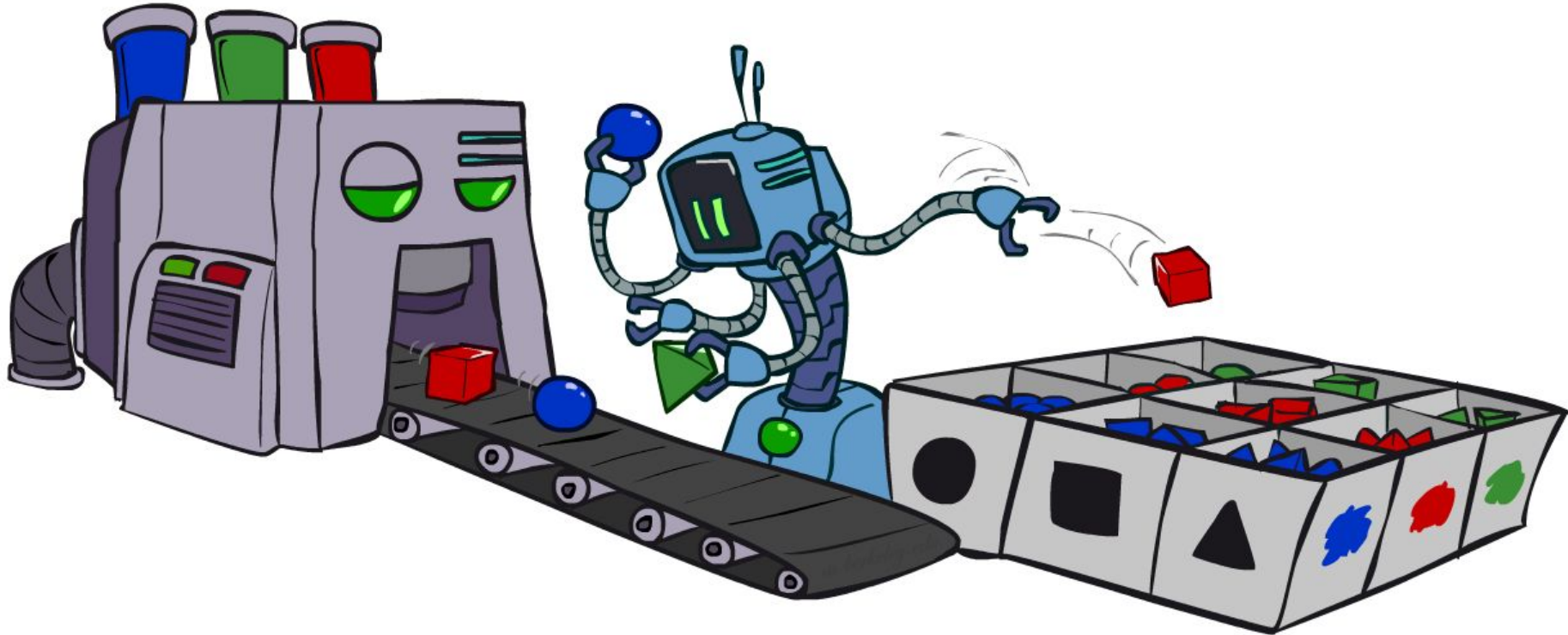
- Suppose we have a *biased* coin that comes up heads with some *unknown* probability p ; how can we use it to produce random bits with probabilities of *exactly* 0.5 for 0 and 1?

Quick Warm-Up

- Suppose we have a *biased* coin that comes up heads with some *unknown* probability p ; how can we use it to produce random bits with probabilities of *exactly* 0.5 for 0 and 1?
- Answer (von Neumann):
 - Flip coin twice, repeat until the outcomes are different
 - HT = 0, TH = 1, each has probability $p(1-p)$

CS 188: Artificial Intelligence

Bayes Nets: Approximate Inference



Instructors: Dan Klein and Stuart Russell

University of California, Berkeley

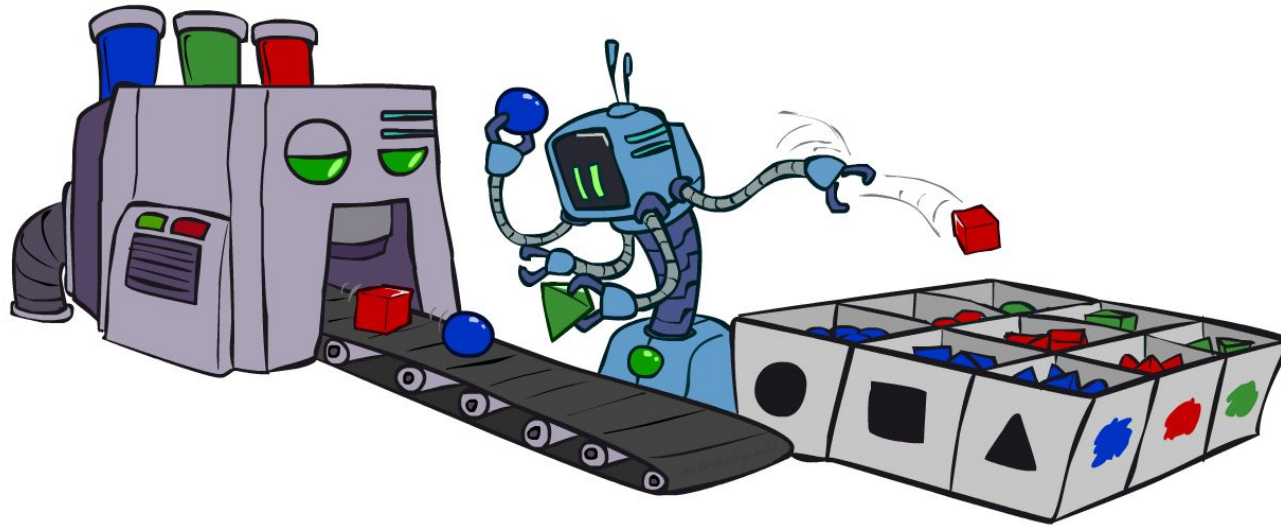
Sampling

- 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



Example

- Suppose you have two agent programs **A** and **B** for Monopoly
- What is the probability that **A** wins?
 - Method 1:
 - Let s be a sequence of dice rolls and Chance and Community Chest cards
 - Given s , the outcome $V(s)$ is determined (1 for a win, 0 for a loss)
 - Probability that **A** wins is $\sum_s P(s) V(s)$
 - Problem: infinitely many sequences s !
 - Method 2:
 - Sample N sequences from $P(s)$, play N games (maybe 100)
 - Probability that **A** wins is roughly $1/N \sum_i V(s_i)$, i.e., fraction of wins in the sample

Sampling basics: discrete (*categorical*) distribution

- To simulate a biased d-sided coin $P(x)$:
 - Example

- 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_i with a $P(x_i)$ -sized sub-interval of $[0,1)$

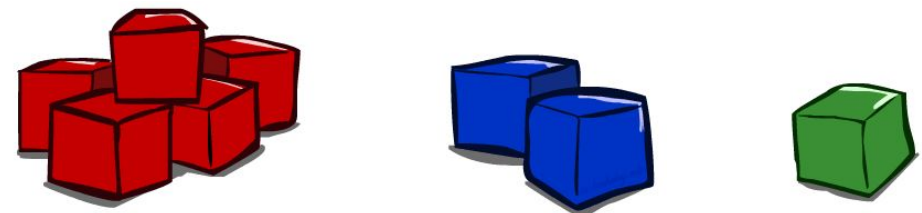
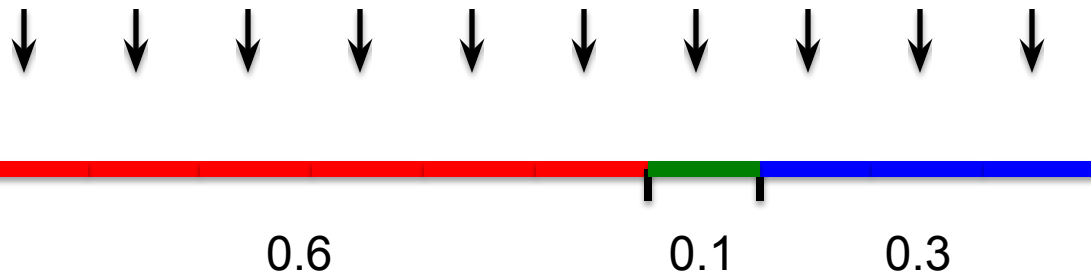
C	$P(C)$
red	0.6
green	0.1
blue	0.3

$0.0 \leq u < 0.6, \rightarrow C = \text{red}$

$0.6 \leq u < 0.7, \rightarrow C = \text{green}$

$0.7 \leq u < 1.0, \rightarrow C = \text{blue}$

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

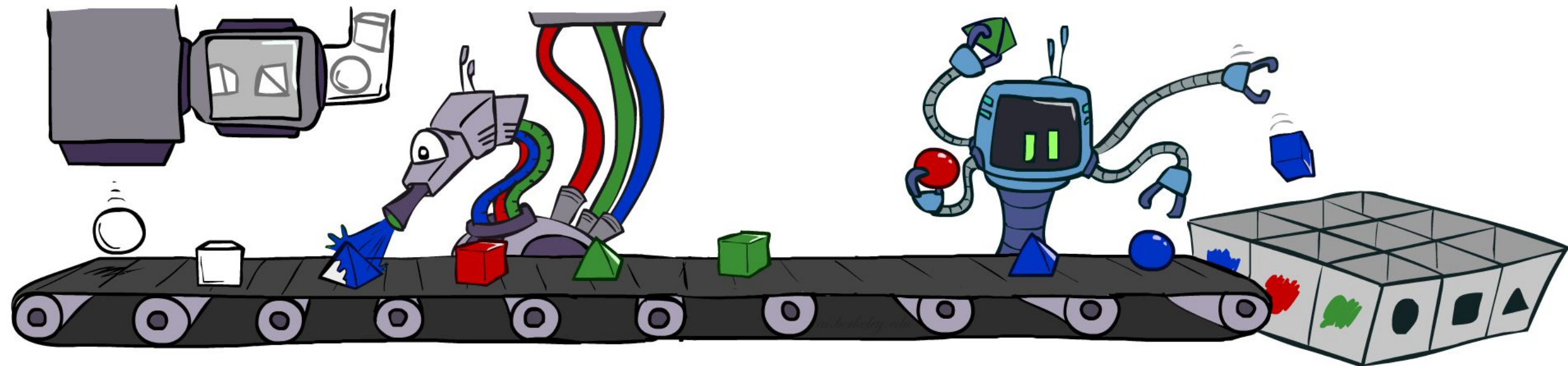


Sampling in Bayes nets

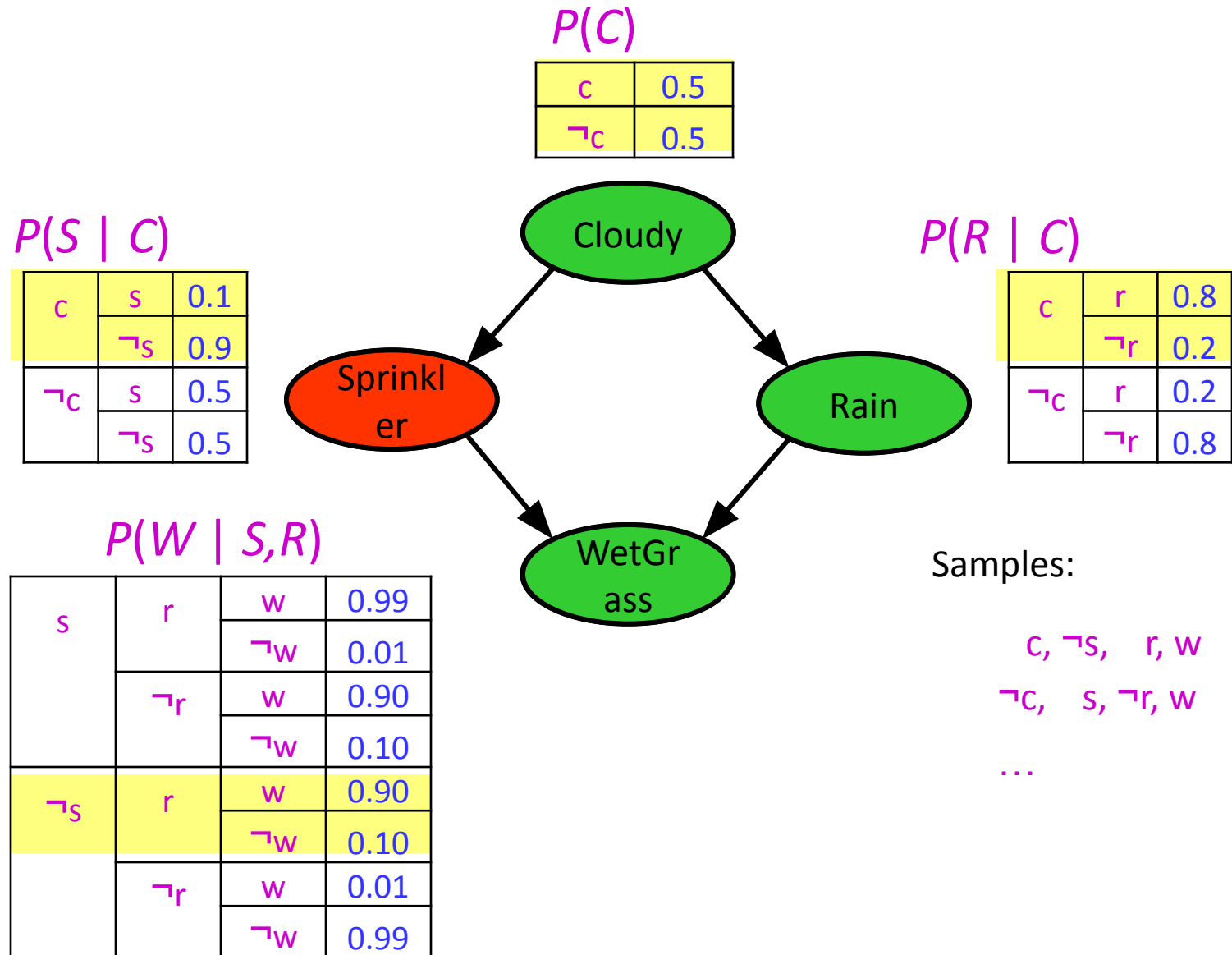
- Prior sampling
- Rejection sampling
- Likelihood weighting
- Gibbs sampling

Prior sampling (one sample)

- 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



Prior Sampling

- This process generates samples with probability:

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

...i.e. the BN'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** (correct in the limit of infinitely many samples)

Example

- Collect samples from the BN:

$C, \neg S, r, W$

C, S, r, W

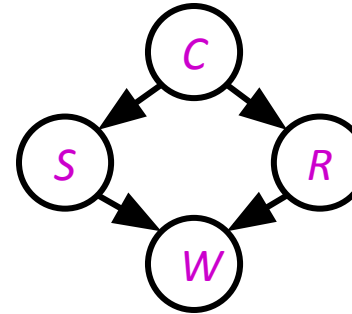
$\neg C, S, r, \neg W$

$C, \neg S, r, W$

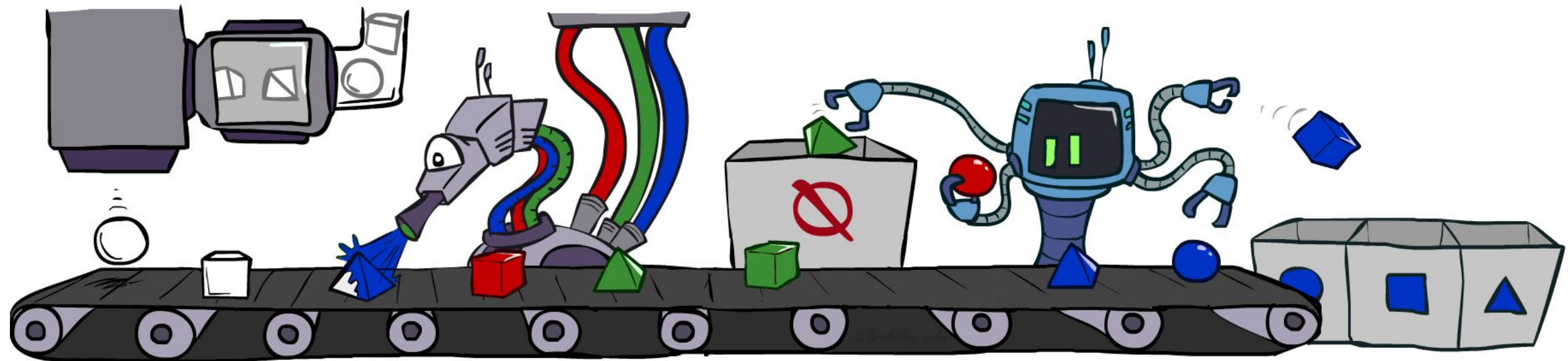
$\neg C, \neg S, \neg r, W$

- 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

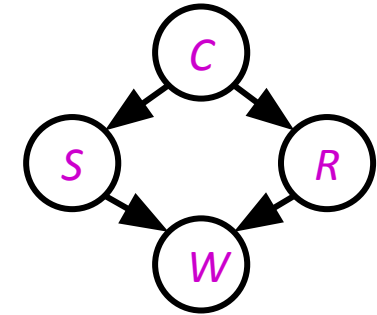


Rejection sampling



Rejection sampling

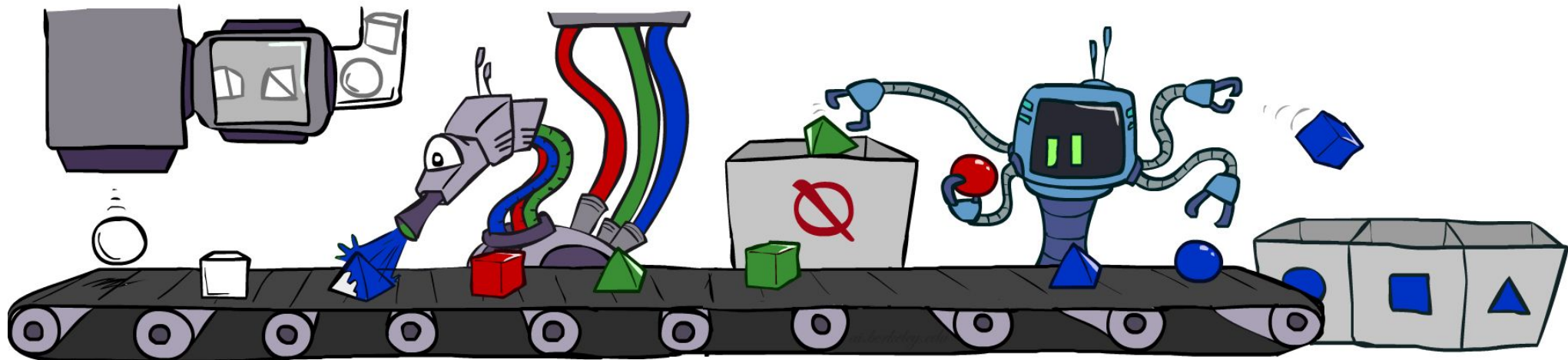
- A simple application of prior sampling for estimating conditional probabilities
 - Let's say we want $P(C | r, w) = \alpha P(C, r, w)$
 - For these counts, samples with $\neg r$ or $\neg w$ **are not relevant**
 - So count the C outcomes for samples with r, w and reject all other samples
- This is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



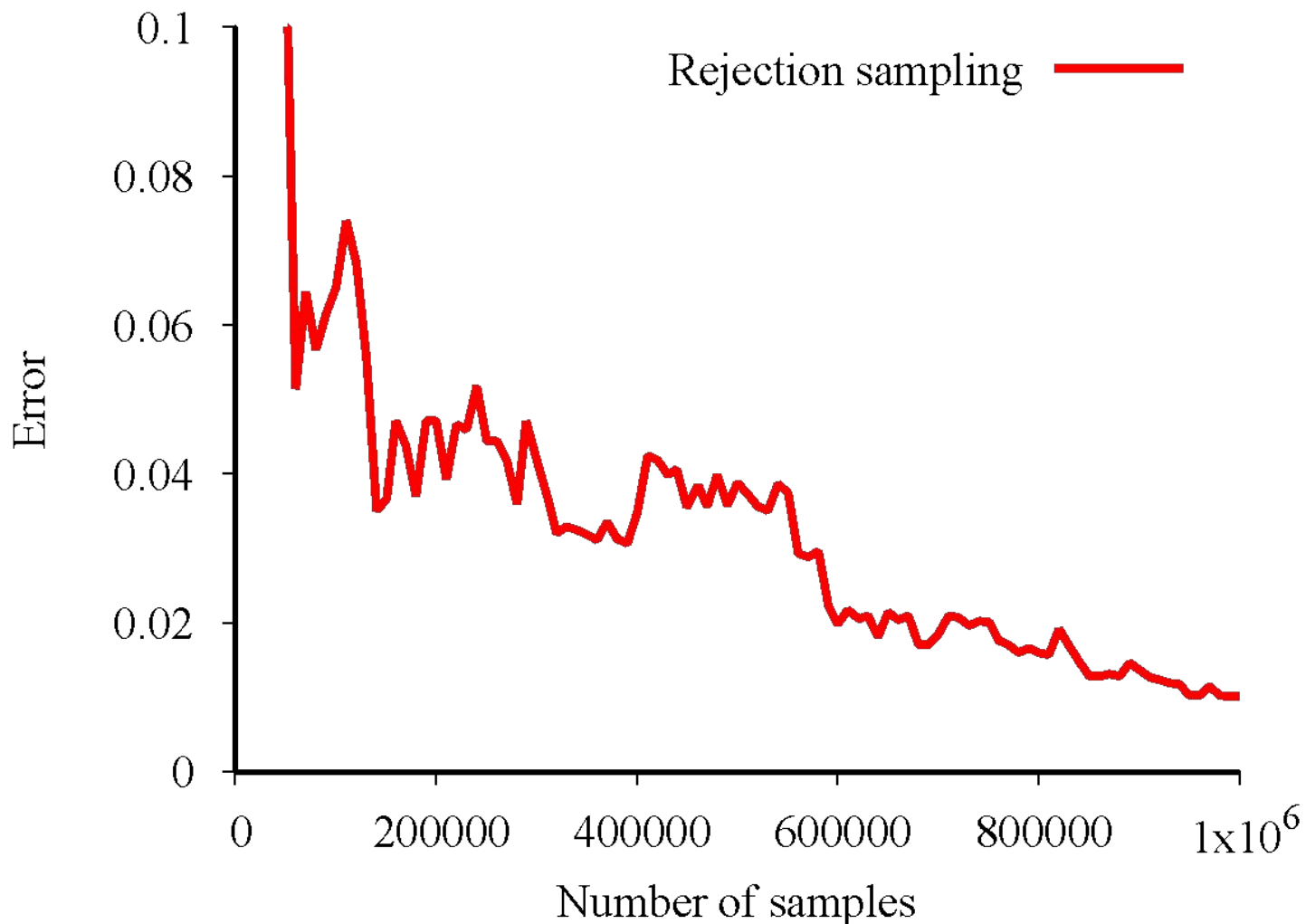
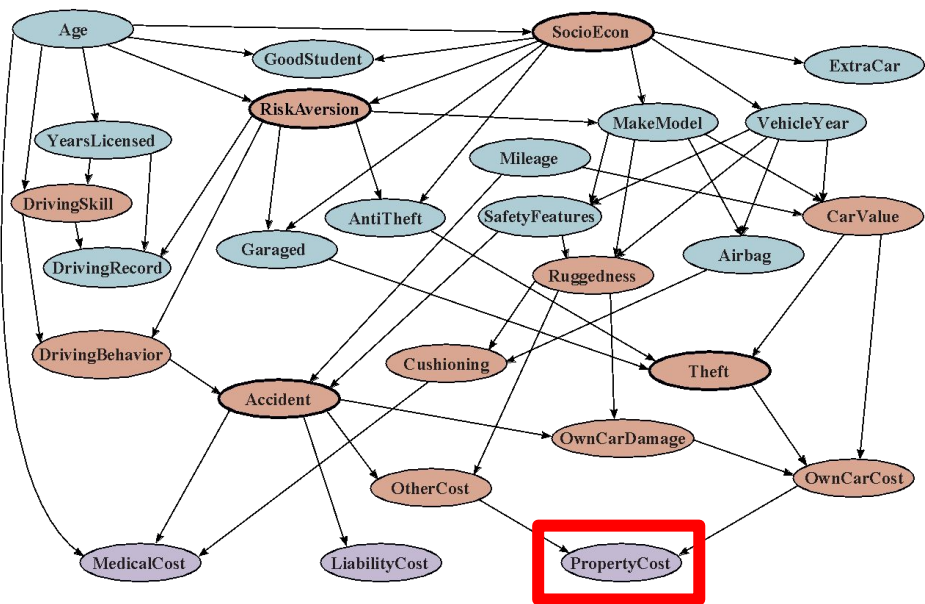
$c, \neg s, r, w$
 ~~$c, s, \neg r$~~
 ~~$\neg c, s, r, \neg w$~~
 ~~$c, \neg s, \neg r$~~
 $\neg c, \neg s, r, w$

Rejection sampling (one sample attempt)

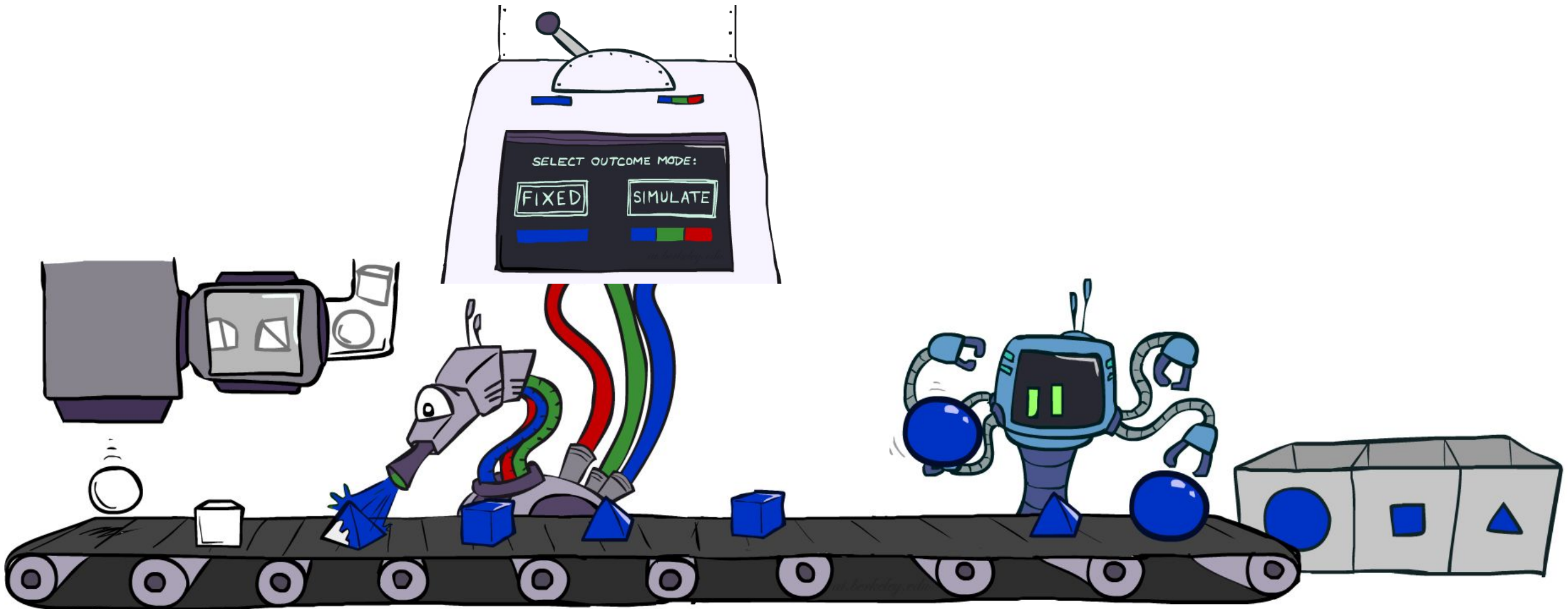
- Input: evidence e_1, \dots, e_k
- For $i=1, 2, \dots, n$
 - Sample x_i from $P(x_i \mid \text{parents}(x_i))$
 - If x_i not consistent with evidence
 - Reject: Return, and no sample is generated in this cycle
- Return (x_1, x_2, \dots, x_n)



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



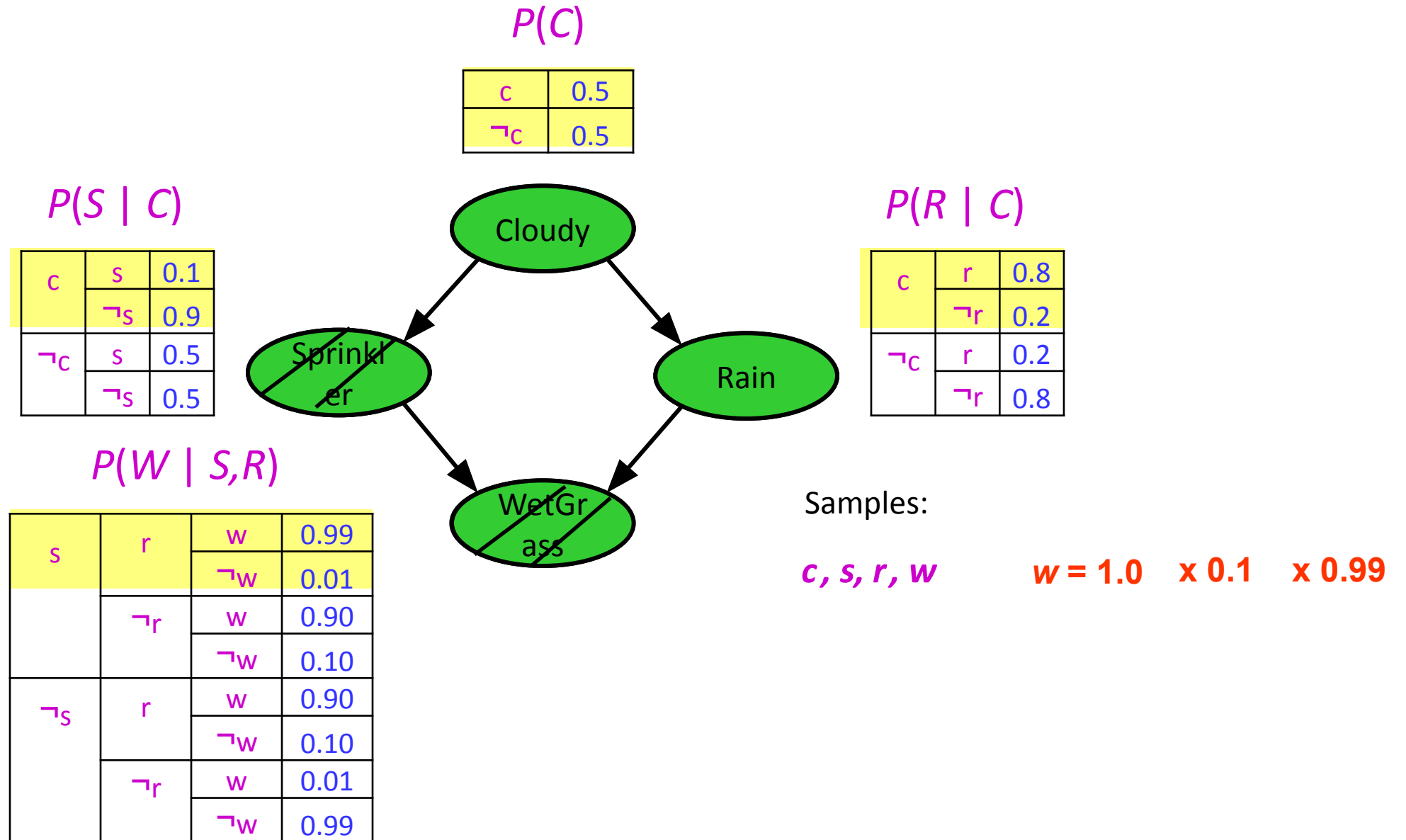
Likelihood weighting



Likelihood weighting

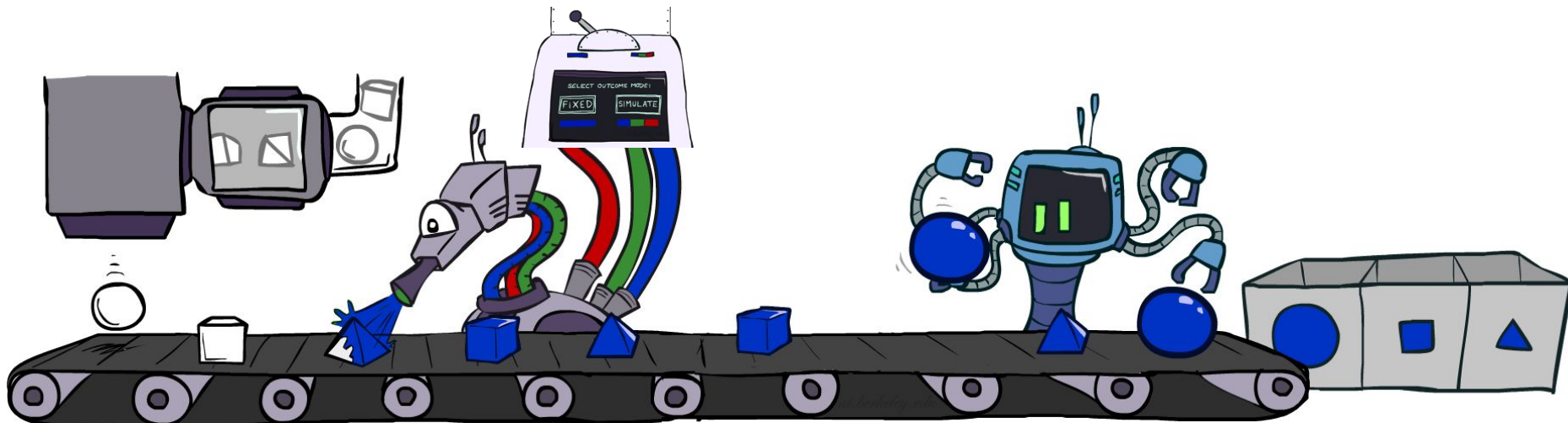
- Problems with rejection sampling:
 - If evidence is unlikely, rejects lots of samples
 - With multiple evidence variables, the evidence is usually very unlikely!
 - The sampling process (prior sampling) ignores the evidence when generating samples
- Idea: fix evidence variables, sample the rest
 - Problem: sample distribution not consistent!
 - Solution: *weight* each sample by probability of evidence variables given parents

Likelihood Weighting



Likelihood weighting (one sample)

- Input: evidence e_1, \dots, e_k
- $w = 1.0$
- for $i=1, 2, \dots, n$
 - if X_i is an evidence variable
 - $x_i =$ 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 is consistent

- 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

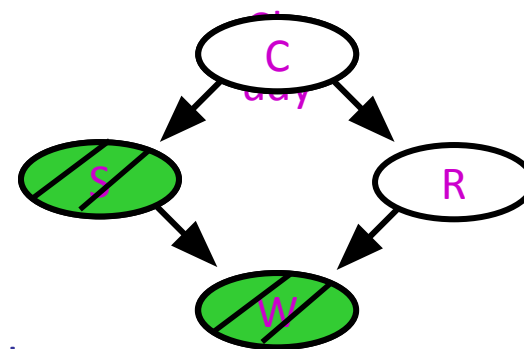
$$w(\mathbf{z}, \mathbf{e}) = \prod_k P(e_k \mid \text{parents}(E_k))$$

- Together, weighted sampling distribution is consistent

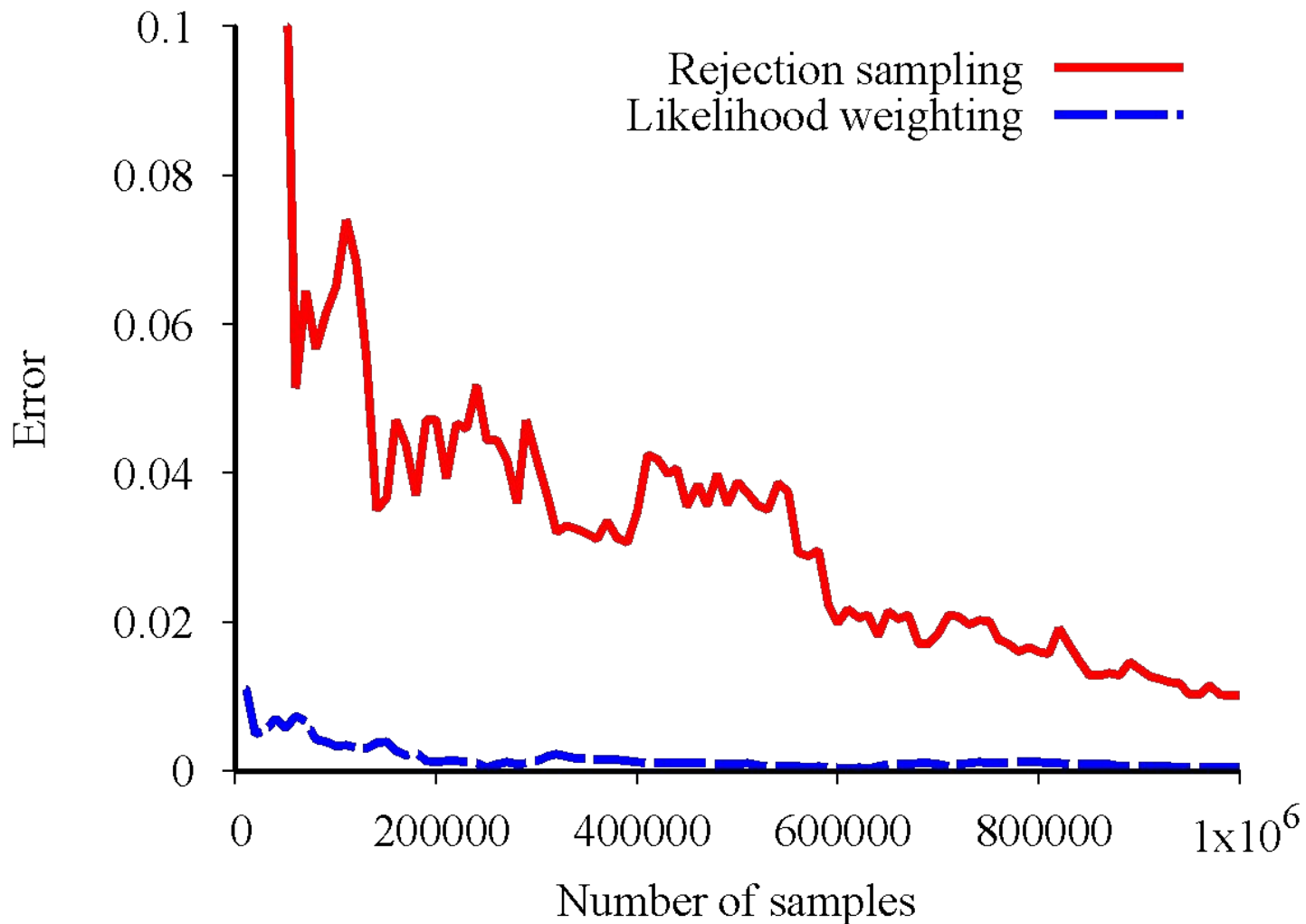
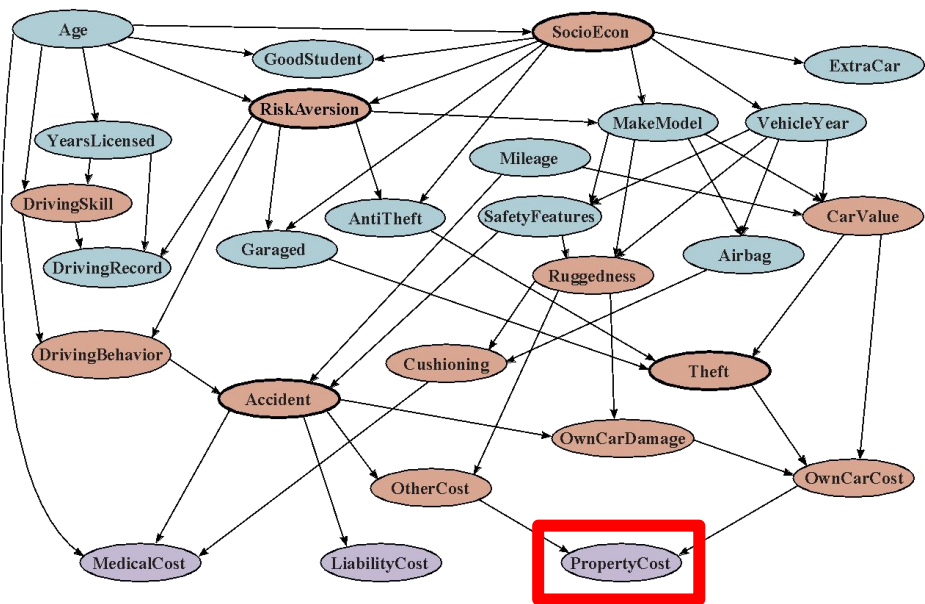
$$\begin{aligned} S_{WS}(\mathbf{z}, \mathbf{e}) \cdot w(\mathbf{z}, \mathbf{e}) &= \prod_j P(z_j \mid \text{parents}(Z_j)) \prod_k P(e_k \mid \text{parents}(E_k)) \\ &= P(\mathbf{z}, \mathbf{e}) \end{aligned}$$

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

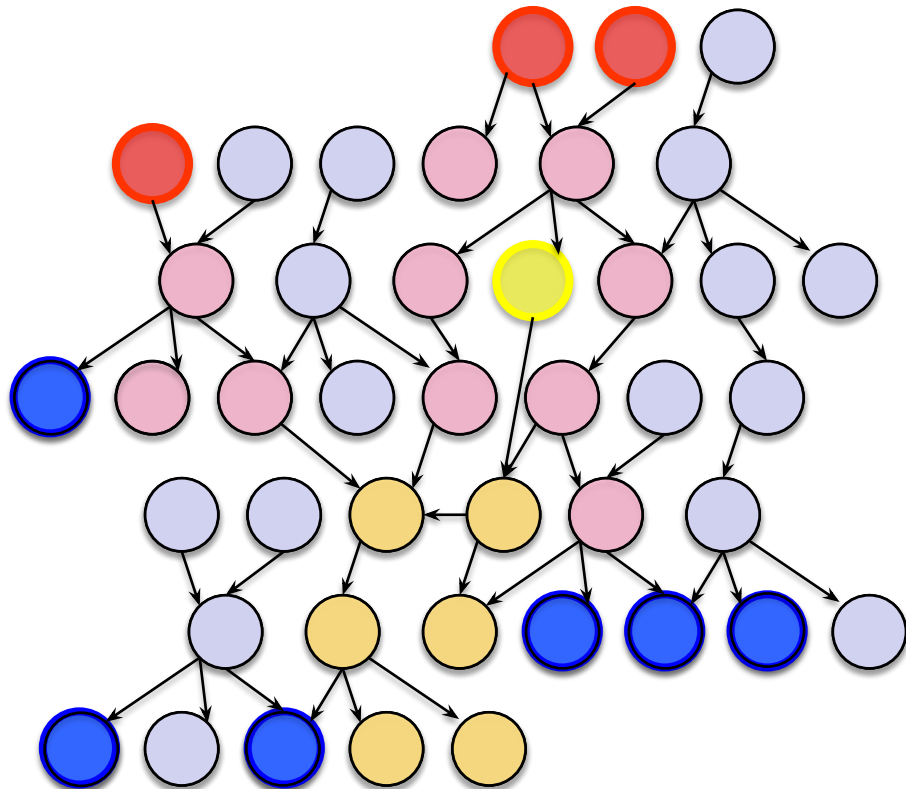


Car Insurance: $P(\text{PropertyCost} \mid e)$



Likelihood weighting

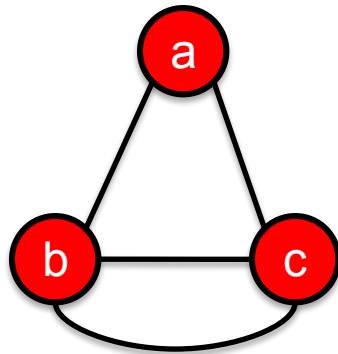
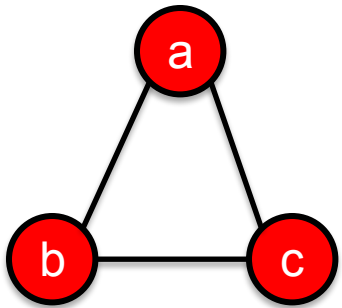
- Likelihood weighting is good
 - All samples are used
 - The values of *downstream* variables are influenced by *upstream* evidence



- 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
 - 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!

Quiz

- Suppose I perform a random walk on a graph, following the arcs out of a node *uniformly at random*. In the infinite limit, what fraction of time do I spend at each node?
 - Consider these two examples:



- MCM algo very

- M
- W
- M
- M
- pr

- MCM



d
a
walk”),
no
some

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: $X_i' \sim P(X_i | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$

- Will tend to move towards states of higher probability, but can go down too

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

- Theorem: Gibbs sampling is consistent*

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

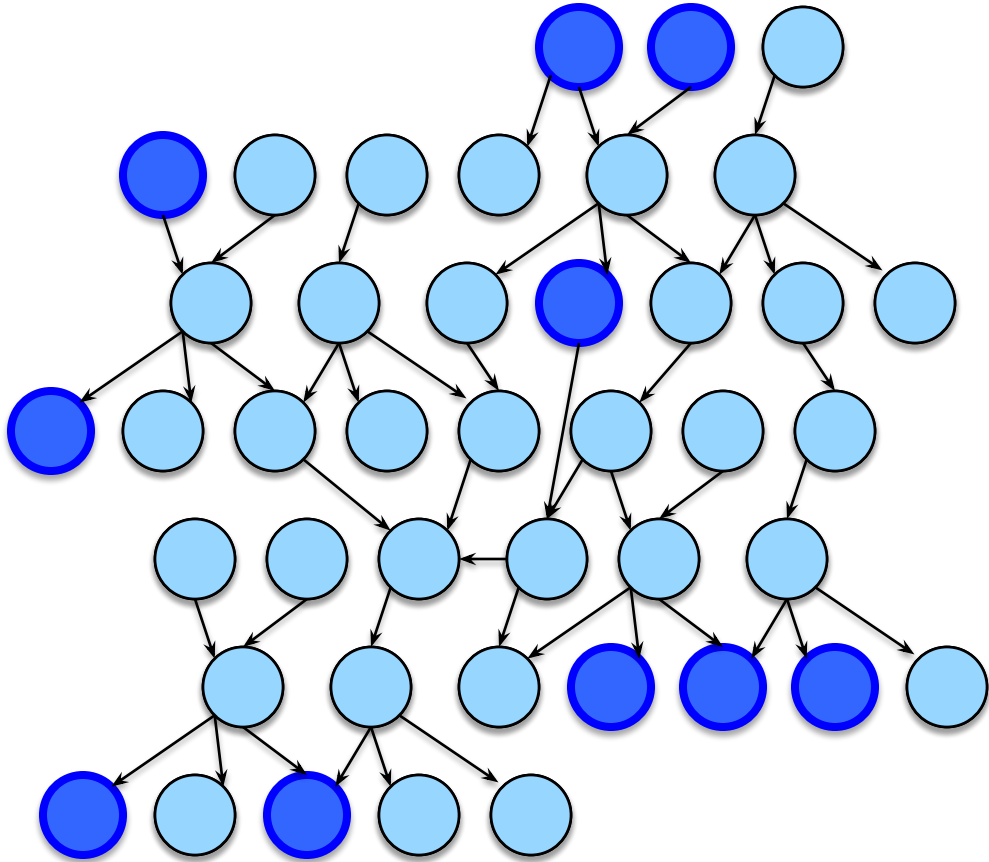
Markov blanket =

Parents +

Children +

Children's other parents

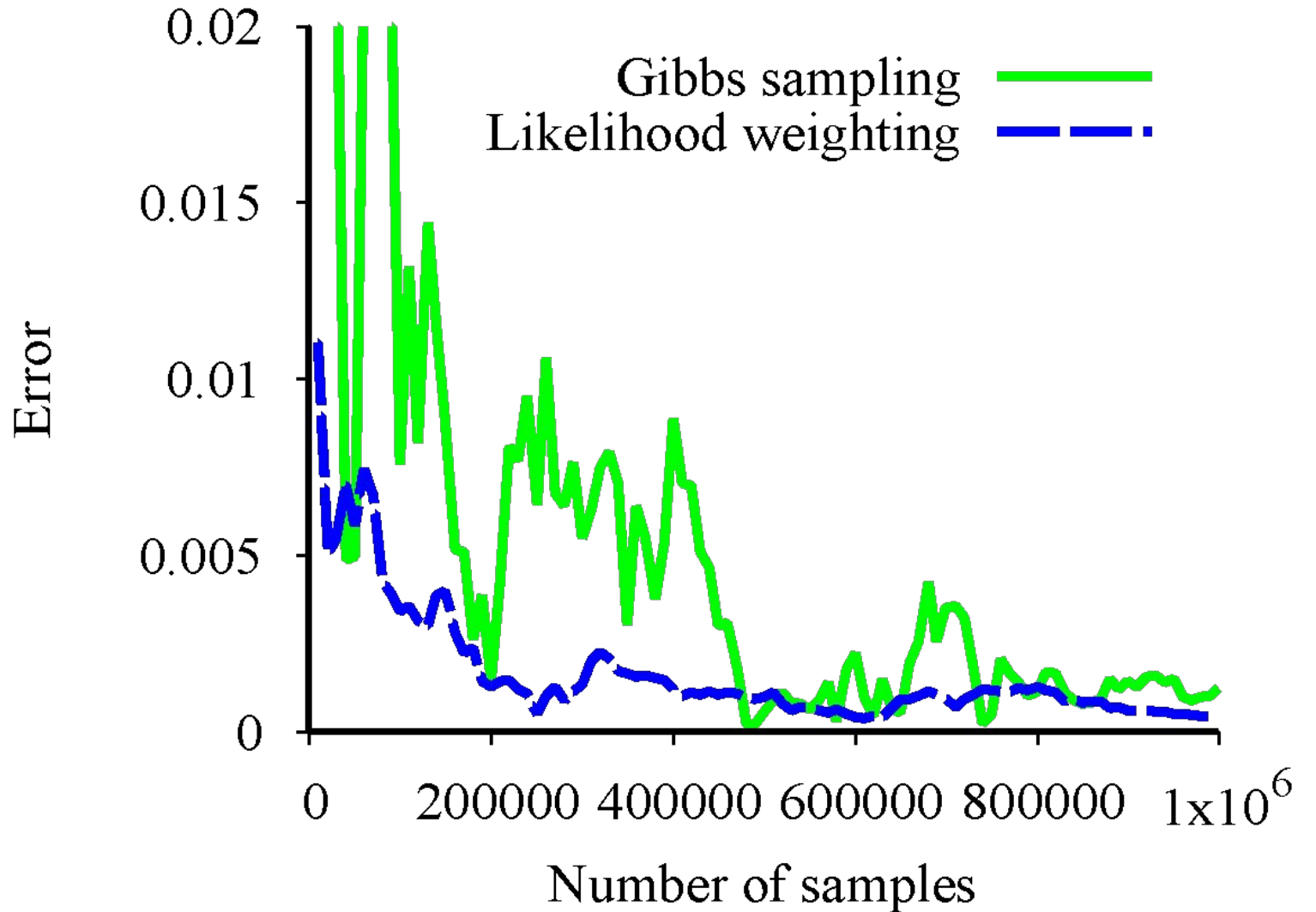
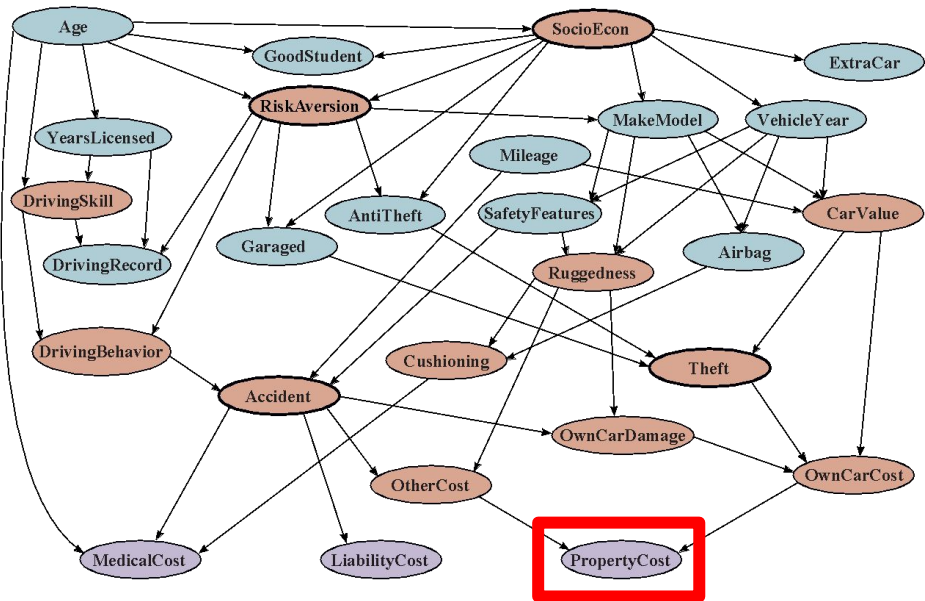
Advantages of MCMC



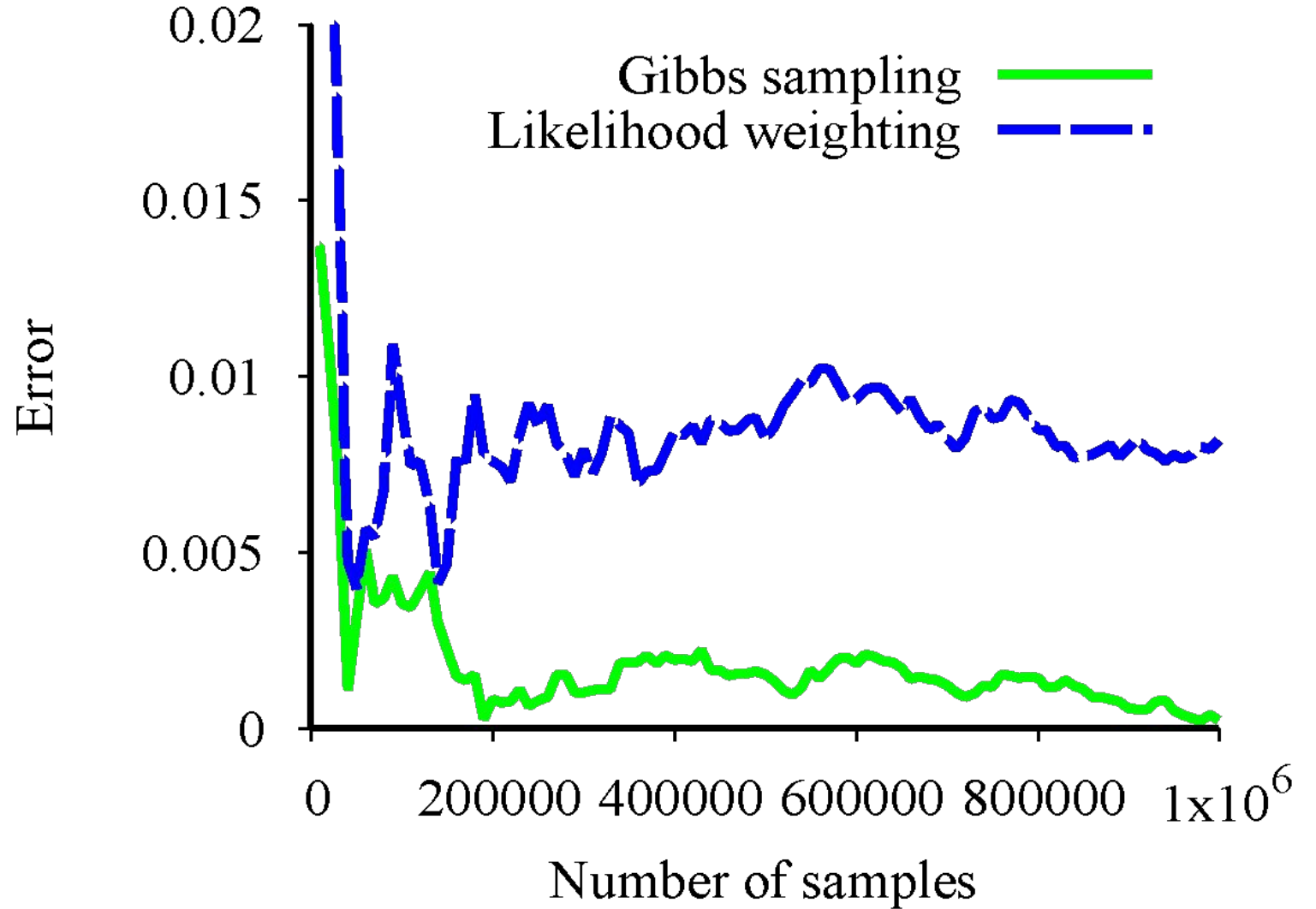
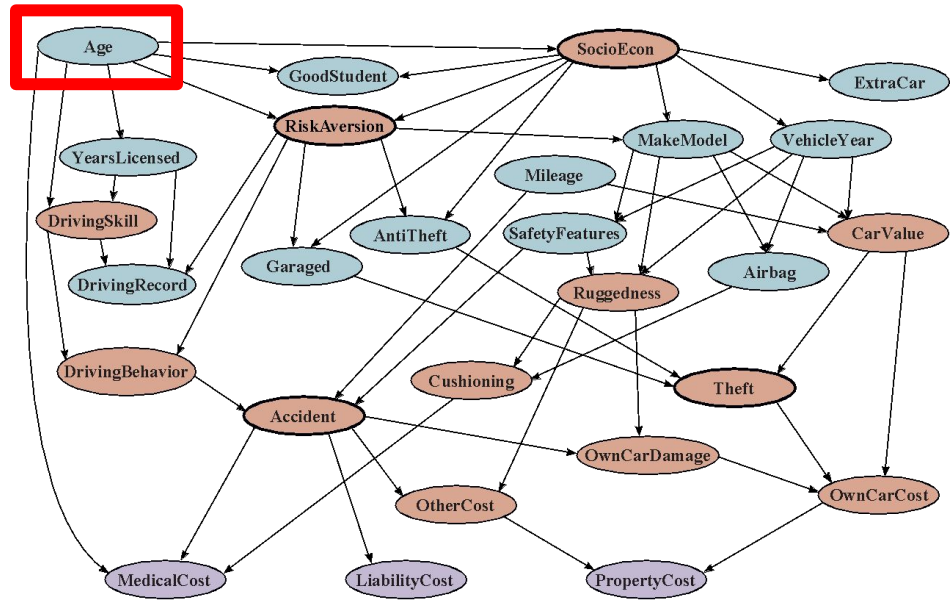
Samples soon begin to reflect all the evidence in the network

Eventually they are being drawn from the true posterior!

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



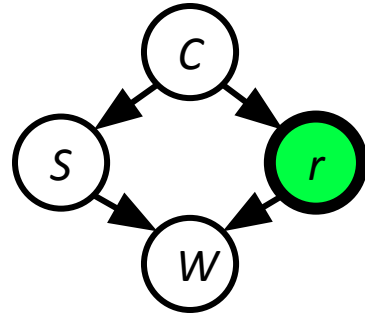
Car Insurance: $P(\text{Age} \mid e)$



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

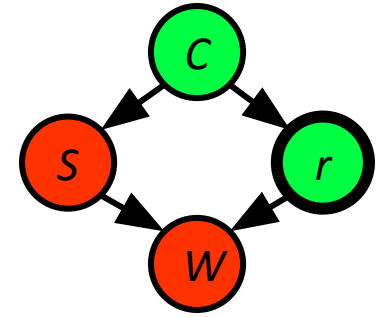
- Step 1: Fix evidence

- $R = \text{true}$



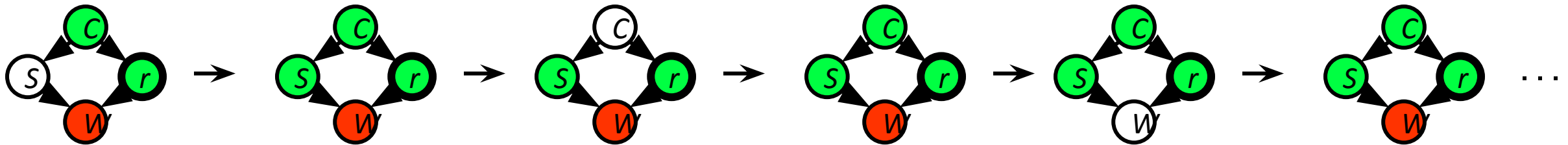
- Step 2: Initialize other variables

- Randomly



- Step 3: Repeat

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

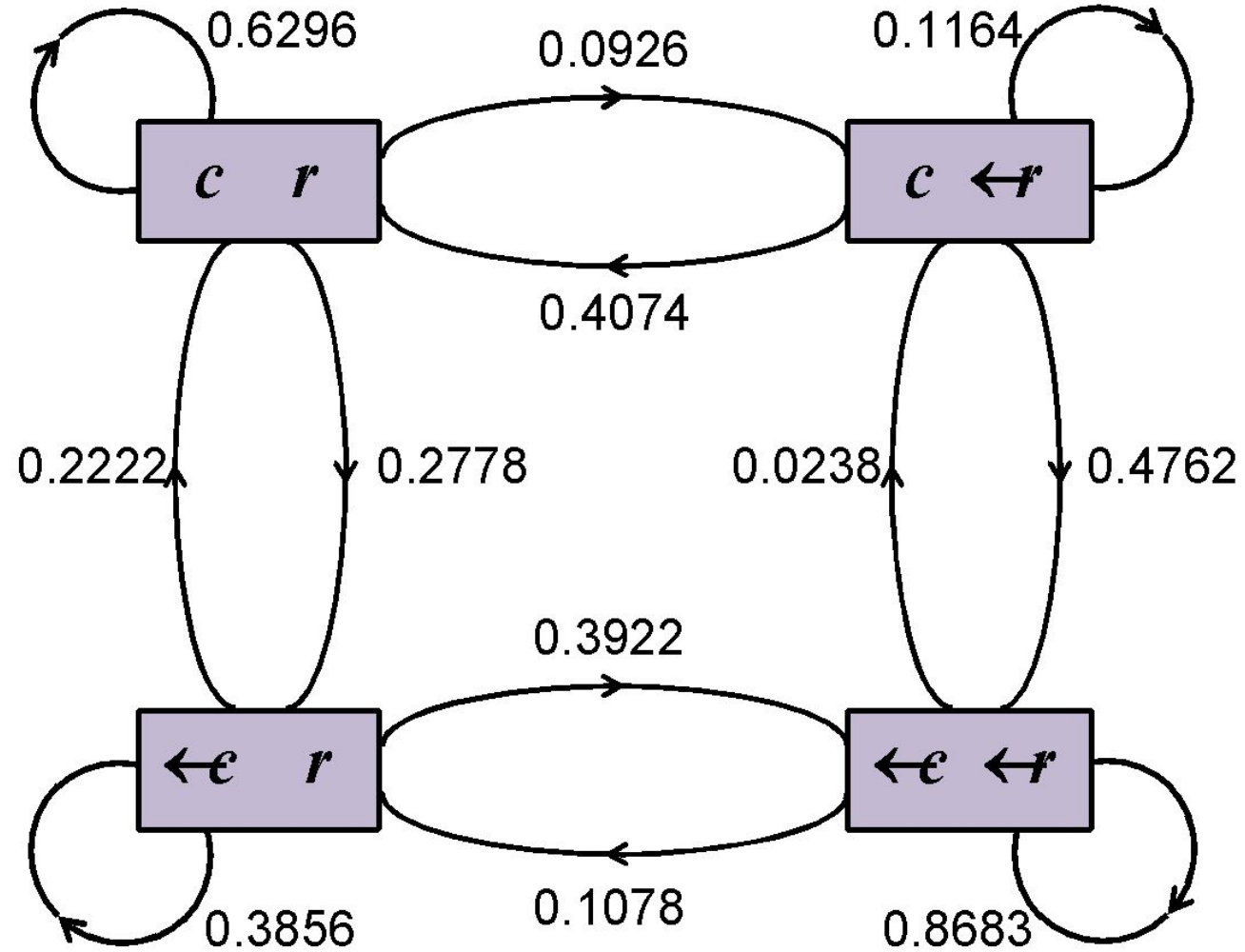


Sample $S \sim P(S | c, r, \neg w)$

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

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

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

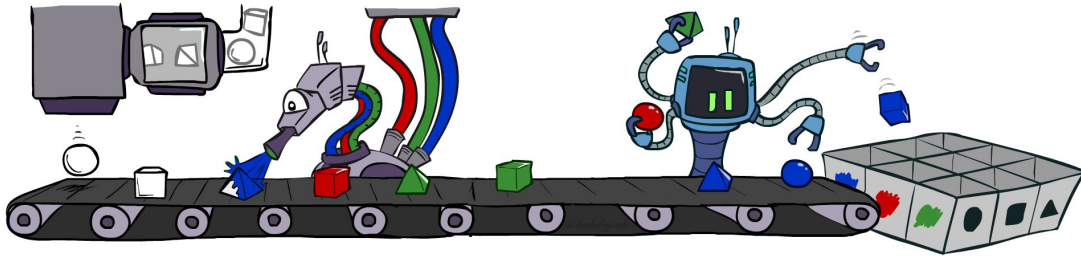
Consistency of Gibbs (see AIMA 13.4.2 for details)

- Suppose we run it 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 $\pi = P(x_1, \dots, x_n | e_1, \dots, e_k)$
 - * Markov chain must be *ergodic*, i.e., completely connected and aperiodic
 - Satisfied if all probabilities are bounded away from 0 and 1
- 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

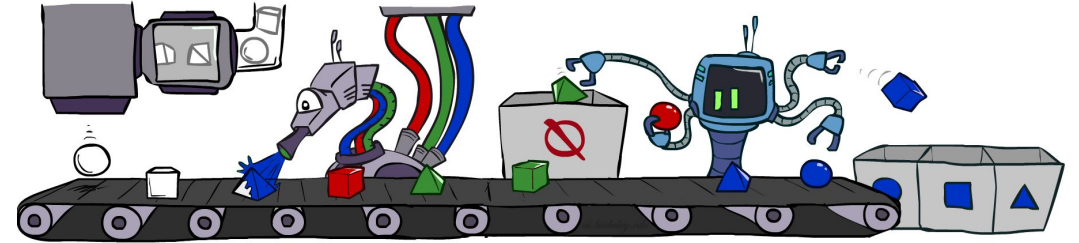
- Prior Sampling P :

- Generate complete samples from $P(x_1, \dots, x_n)$



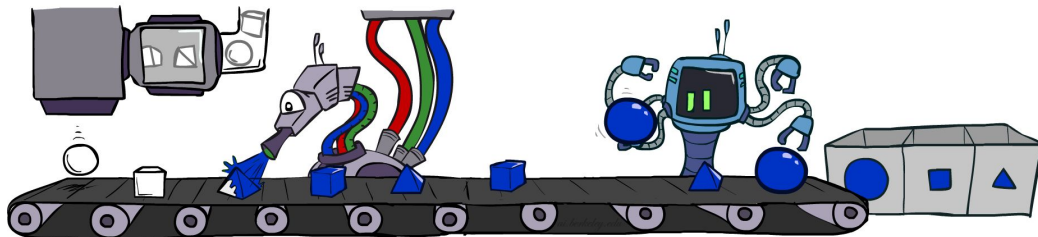
- Rejection Sampling $P(Q | e)$:

- Reject samples that don't match e



- Likelihood Weighting $P(Q | e)$:

- Weight samples by how well they predict e



- Gibbs sampling $P(Q | e)$:

- Wander around in e space
- Average what you see

