Quick Warm-Up

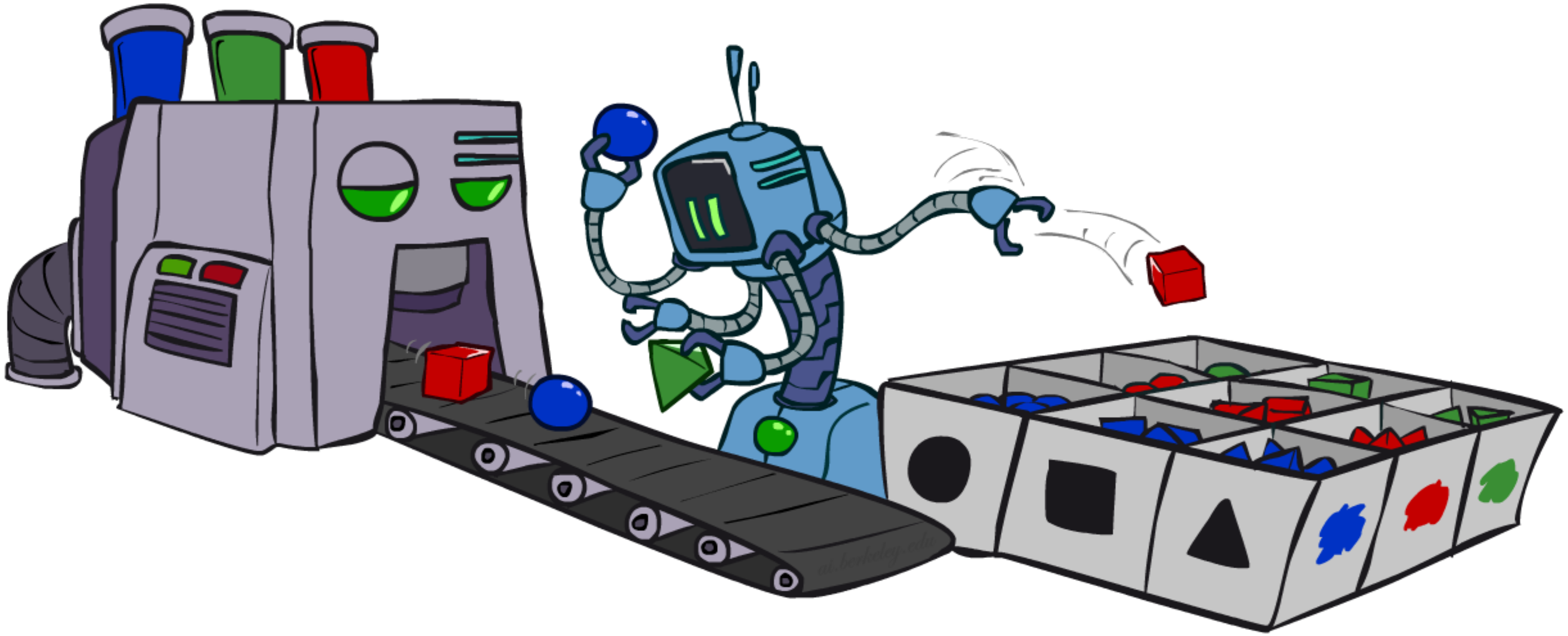
- Suppose we have a ***biased*** coin that comes up heads with some ***unknown*** probability p; how can we use it to produce random bits with probabilities of ***exactly*** 0.5 for 0 and 1?

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



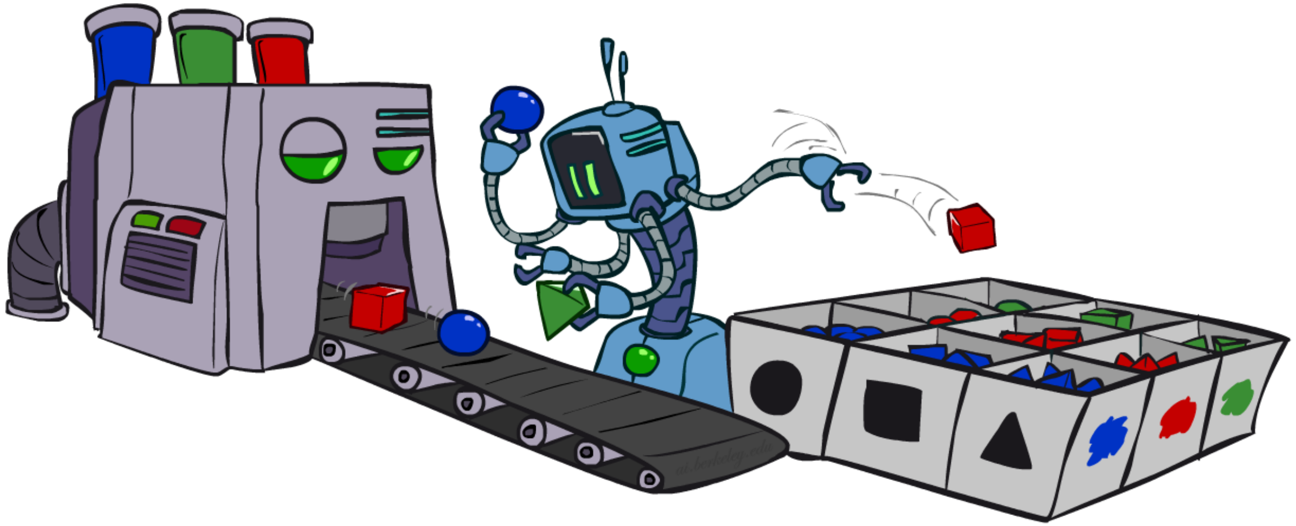
Instructors: Stuart Russell and Dawn Song

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

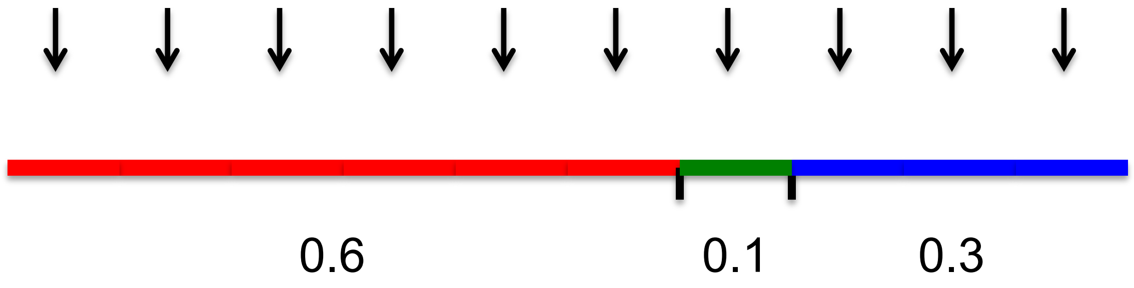


Example

- 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 $P(x)$:
  - 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 $[0,1)$

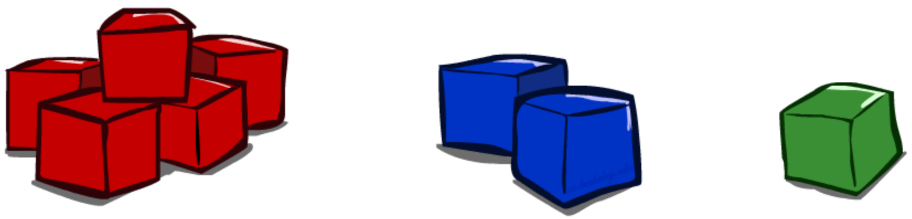


- Example

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

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

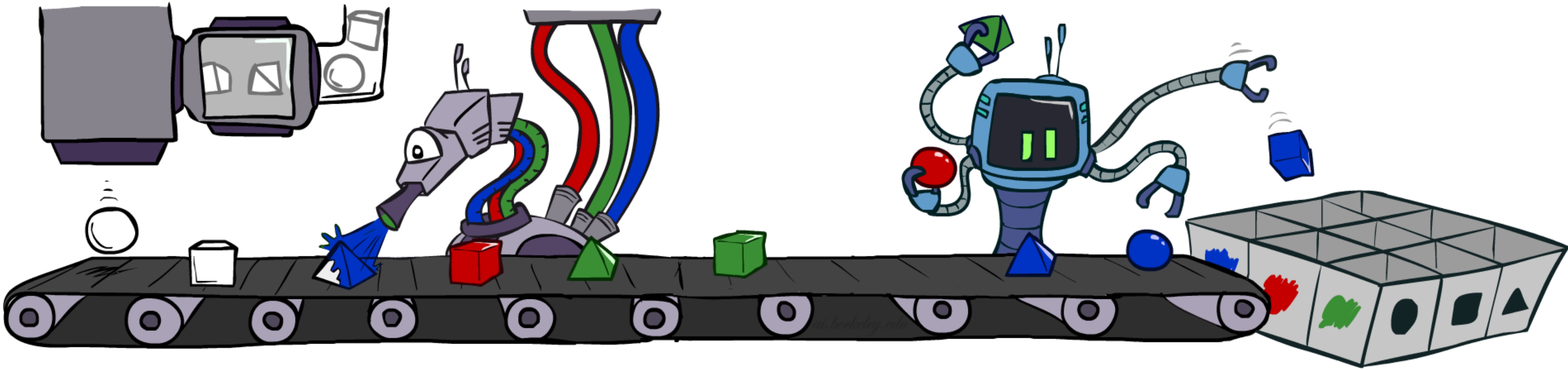
- If random() returns $u = 0.83$, then the sample is $C = blue$
- E.g, after sampling 8 times:



Sampling in Bayes nets

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

Prior sampling

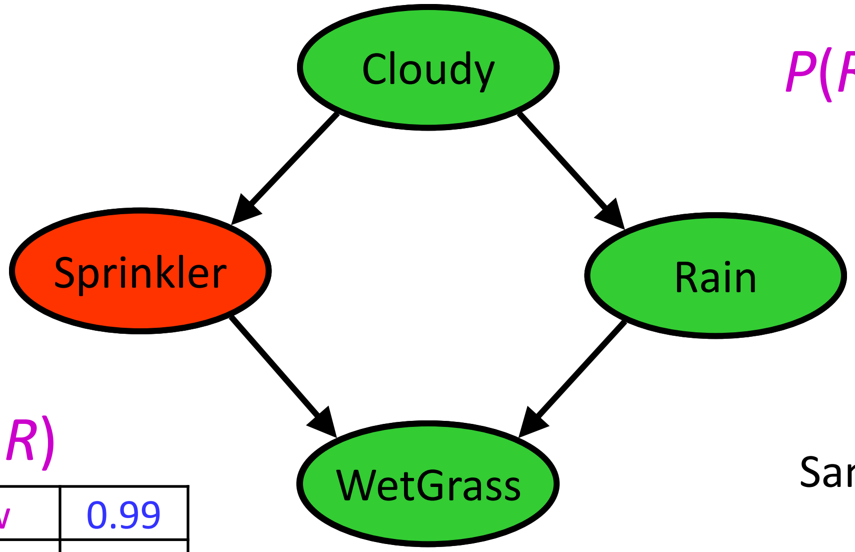


$$\text{Shape} \rightarrow \text{Color}$$

Prior sampling

P(*C*)

c	0.5
¬c	0.5



P(*S* | *C*)

c	s	0.1
	¬s	0.9
¬c	s	0.5
	¬s	0.5

P(*R* | *C*)

c	r	0.8
	¬r	0.2
¬c	r	0.2
	¬r	0.8

P(*W* | *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

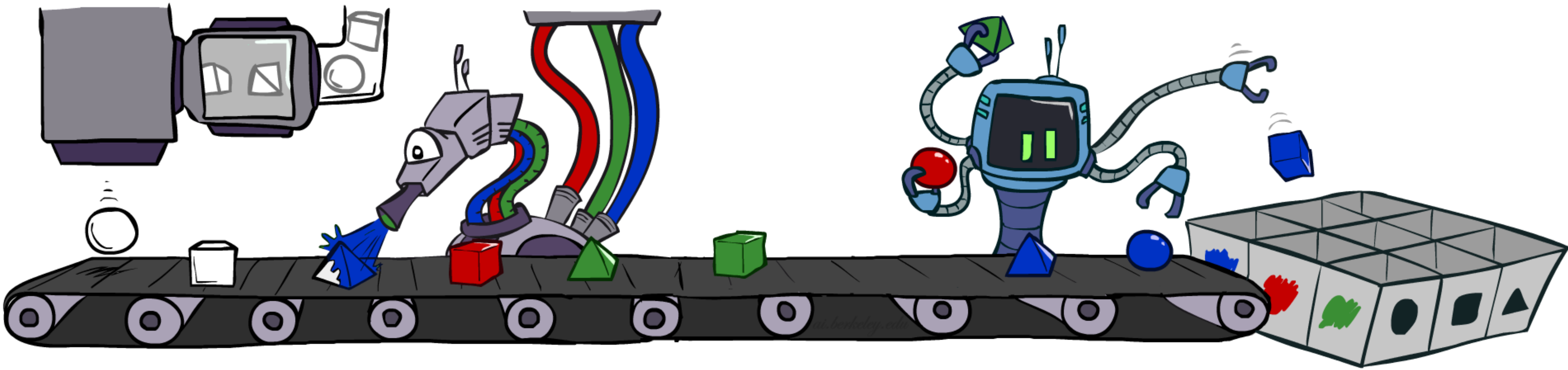
¬c, s, ¬r, w

…

S_{PS}(c,¬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:

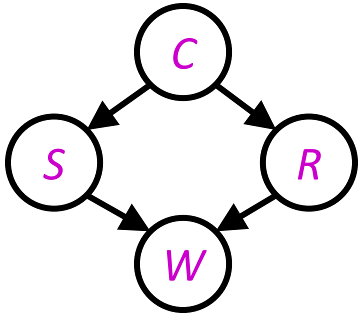
  c, ¬s, r, w

  c, s, r, w

  ¬c, s, r, ¬w

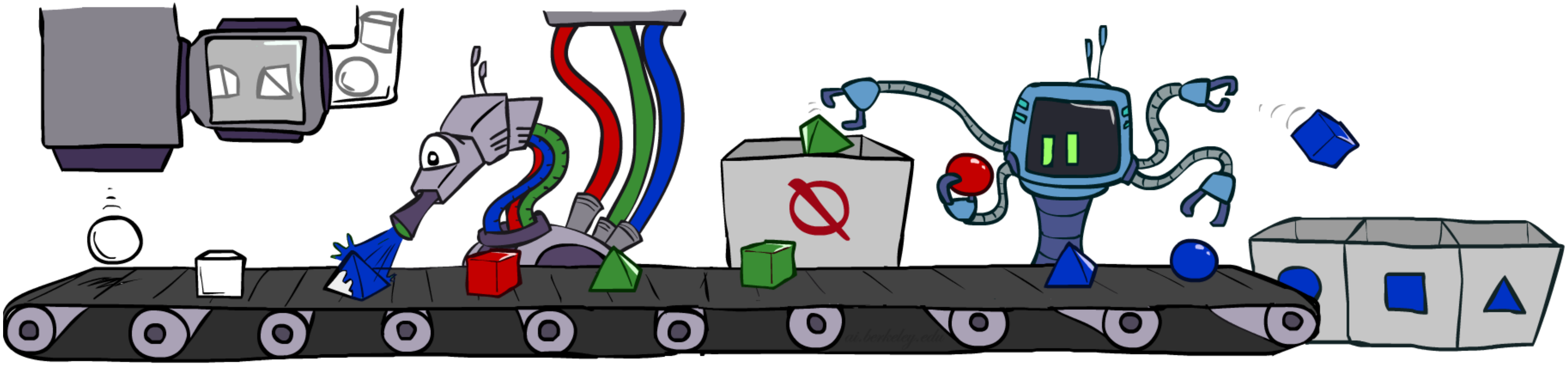
  c, ¬s, r, w

  ¬c, ¬s, ¬r, w



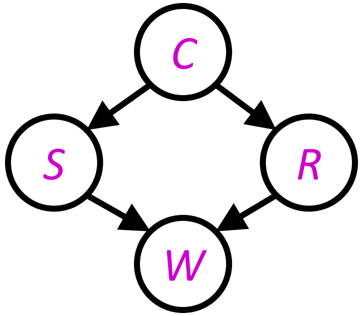
- If we want to know $P(W)$
  - We have counts <w:4, ¬w:1>
  - Normalize to get $P(W)$ = <w:0.8, ¬w:0.2>
  - This will get closer to the true distribution with more samples

Rejection sampling



Rejection sampling

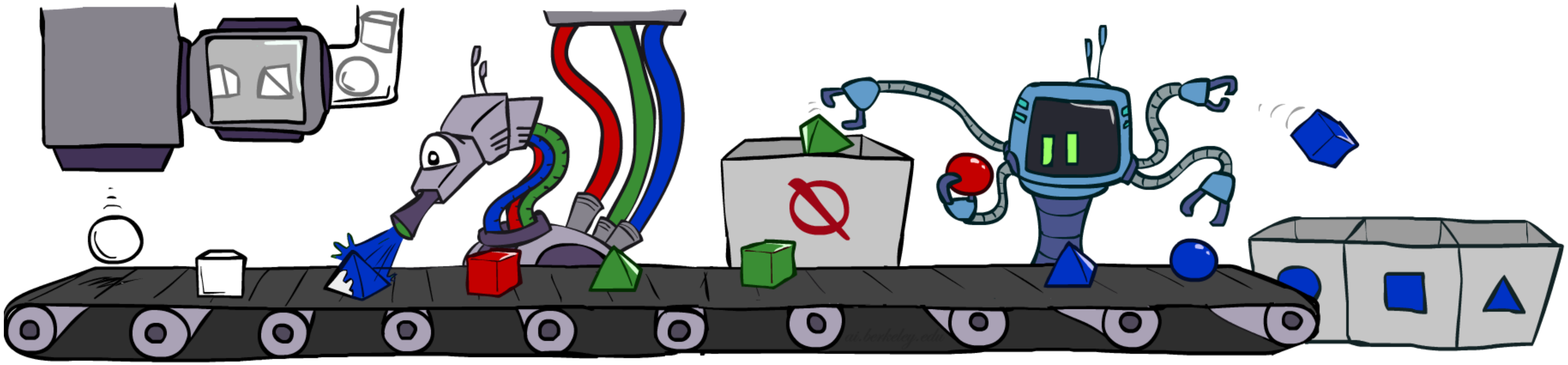
- A simple application of prior sampling for estimating conditional probabilities
  - Let’s say we want $P(C|\ r, w) = \alpha\ P(C, r, w)$
  - For these counts, samples with $\neg r$ or $\neg w$ ***are not relevant***
  - So count the C outcomes for samples with r, w and reject all other samples
- This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



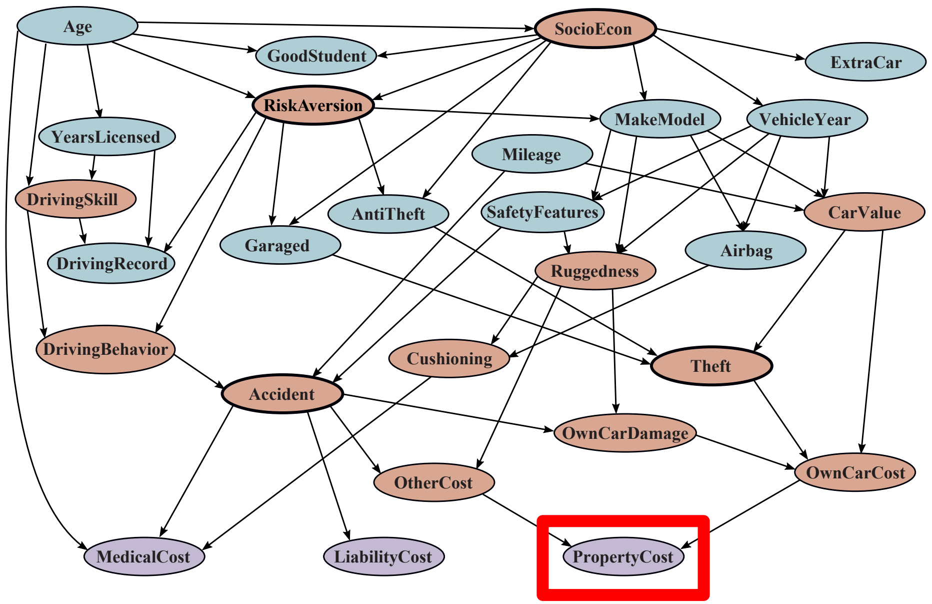
c, ¬s, r, w
~~c, s, ¬r~~
~~¬c, s, r, ¬w~~
~~c, ¬s, ¬r~~
¬c, ¬s, r, w

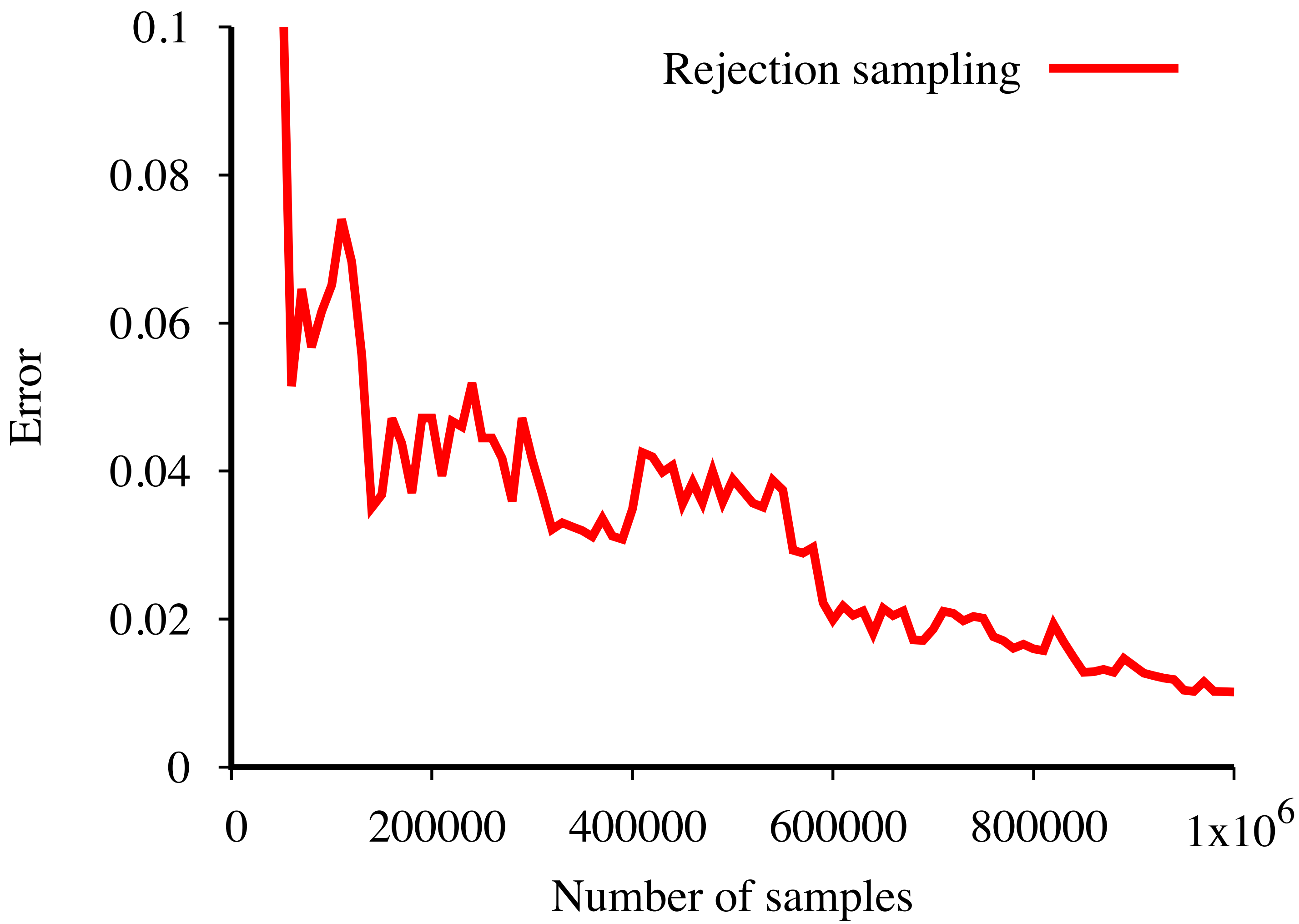
Rejection sampling

- Input: evidence $e_1,..,e_k$
- For i=1, 2, …, n
  - 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 $(x_1, x_2, …, x_n)$

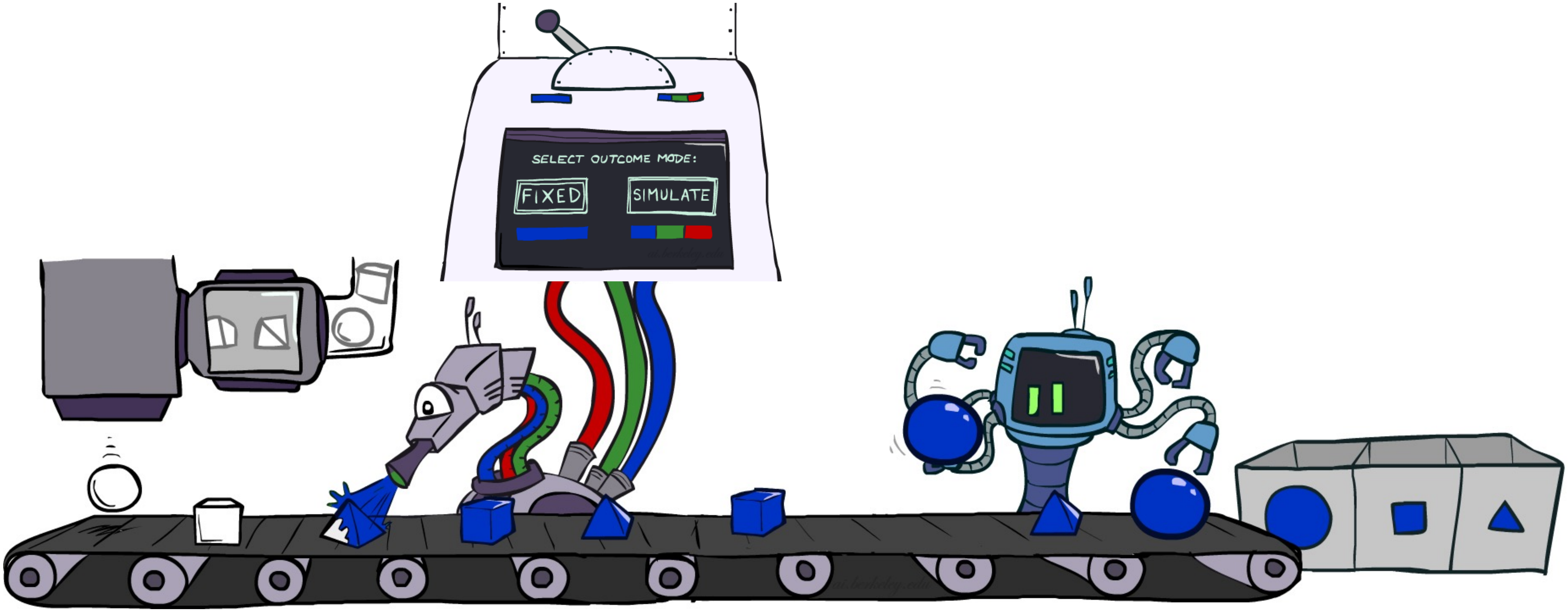


Car Insurance: *P*(*PropertyCost* | ***e***)



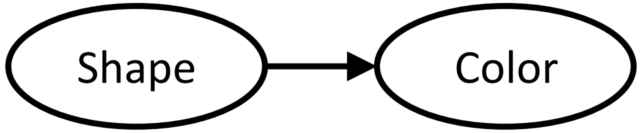


Likelihood weighting

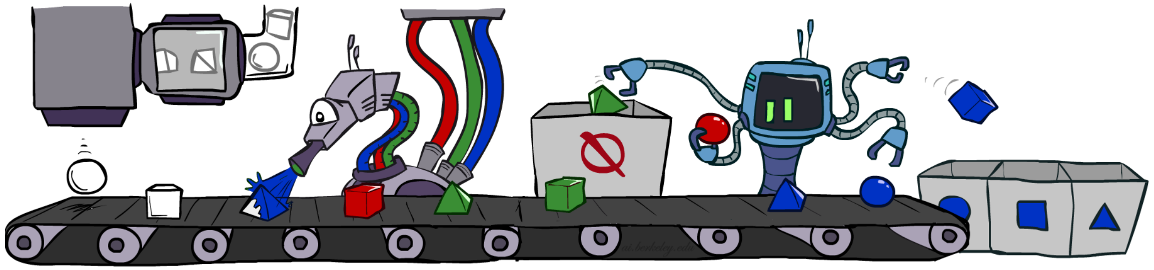


Likelihood weighting

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

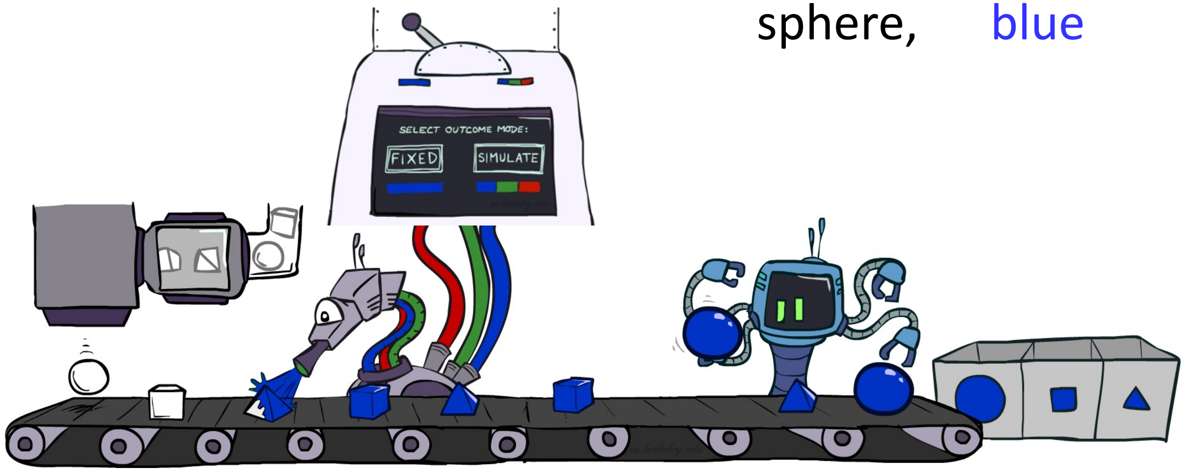


~~pyramid, green~~
~~pyramid, red~~
sphere, blue
~~cube, red~~
~~sphere, green~~



- 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

$P(C)$

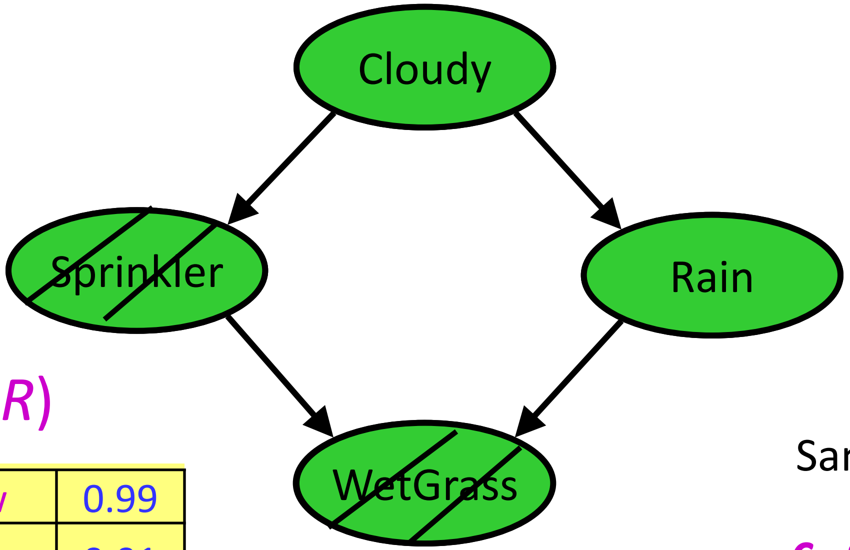
c	0.5
¬c	0.5

$P(S \mid C)$

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



$P(W \mid S,R)$

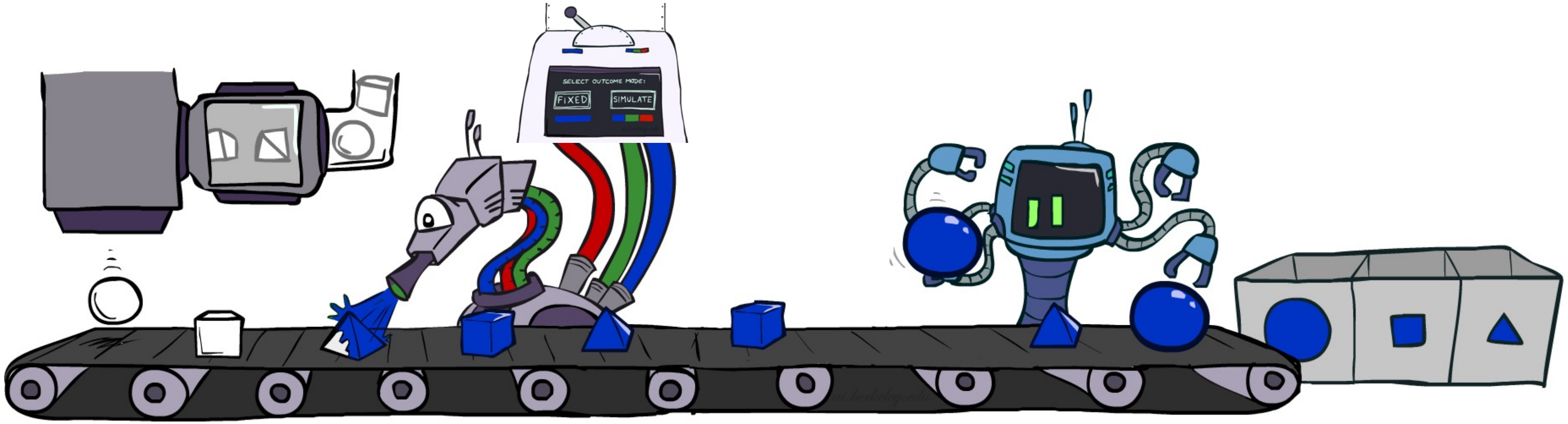
s	r	w	0.99
		¬w	0.01
	¬r	w	0.90
		¬w	0.10
¬s	r	w	0.90
		¬w	0.10
	¬r	w	0.01
		¬w	0.99

Samples:

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

Likelihood weighting

- Input: evidence $e_1,..,e_k$
- w = 1.0
- for i=1, 2, …, n
  - 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 $(x_1, x_2, \ldots, x_n)$, w



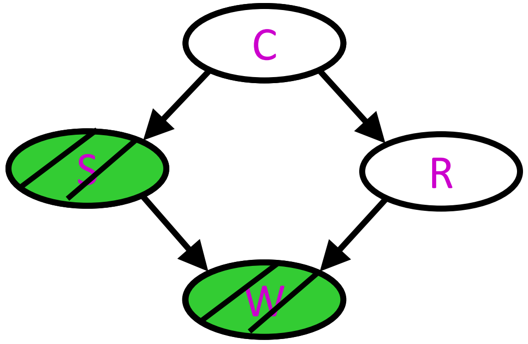
Likelihood weighting is consistent

- 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

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



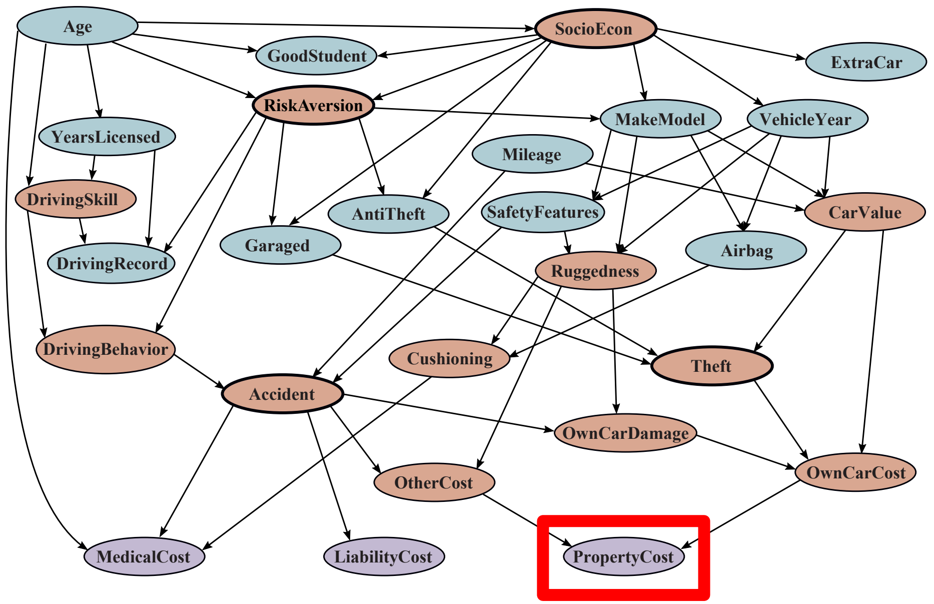
- Together, weighted sampling distribution is consistent

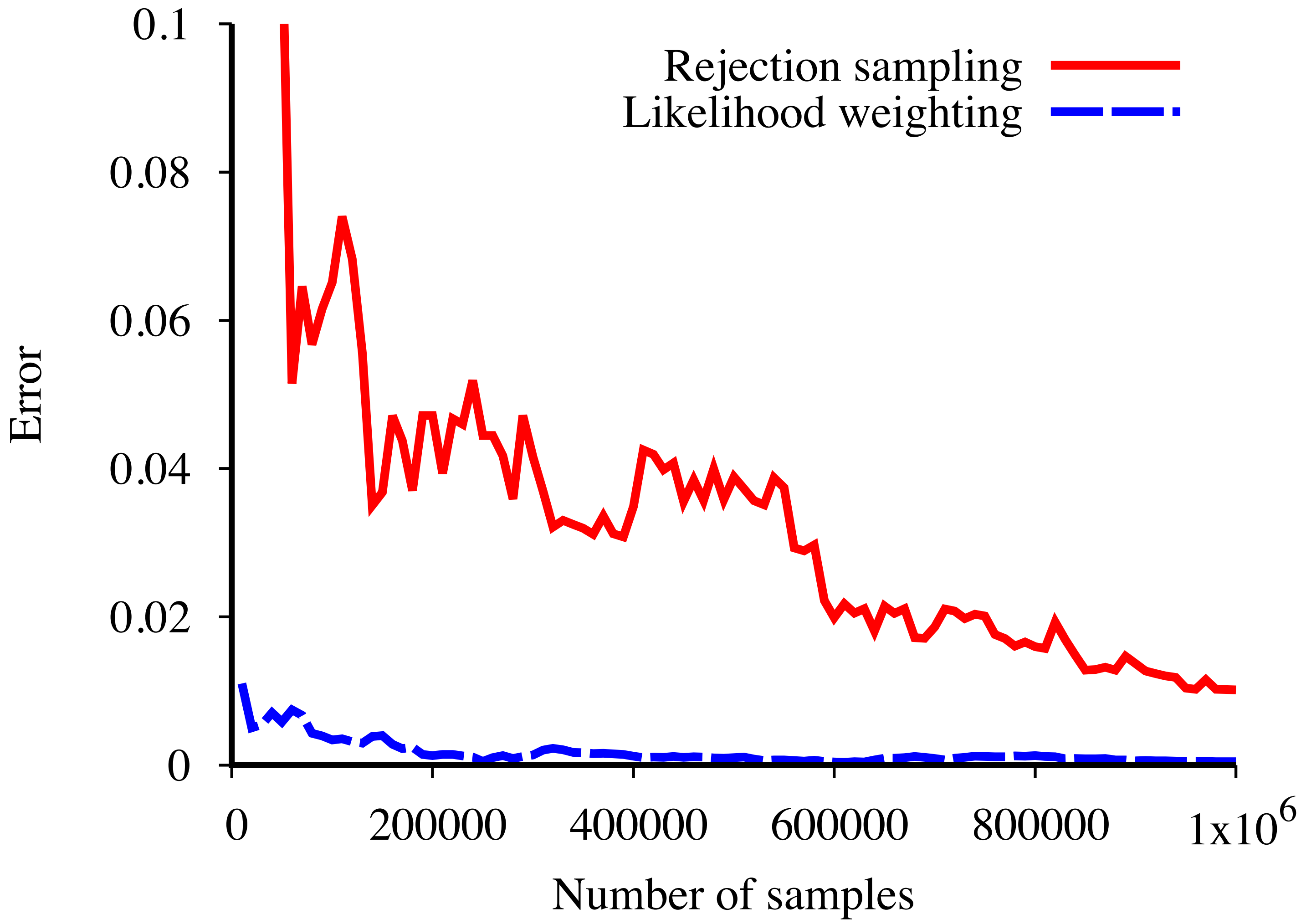
$$S_{WS}(\mathbf{z},\mathbf{e}) \cdot w(\mathbf{z},\mathbf{e}) = \prod_j P(z_j \mid parents(Z_j)) \prod_k P(e_k \mid parents(E_k))$$

$$= P(\mathbf{z},\mathbf{e})$$

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

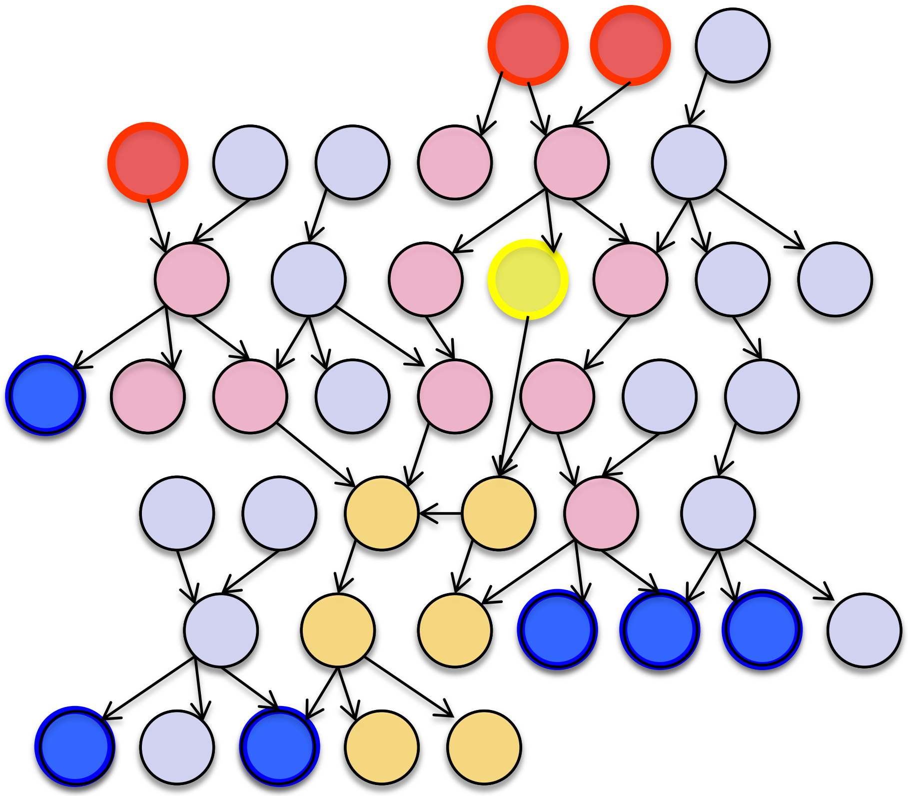
Car Insurance: *P*(*PropertyCost* | ***e***)





Likelihood weighting

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

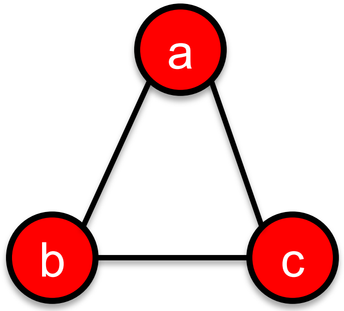


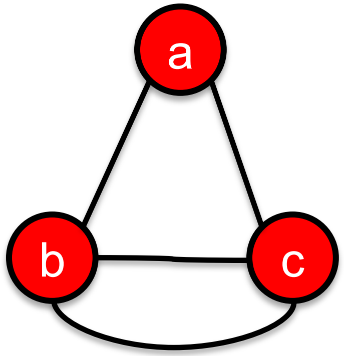
- Likelihood weighting still has weaknesses
  - The values of ***upstream*** variables are unaffected by ***downstream*** evidence
    - E.g., suppose evidence is a video of a traffic accident
  - With evidence in k leaf nodes, weights will be $O(2^{-k})$
  - With high probability, one lucky sample will 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:





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

Markov Chain Monte Carlo

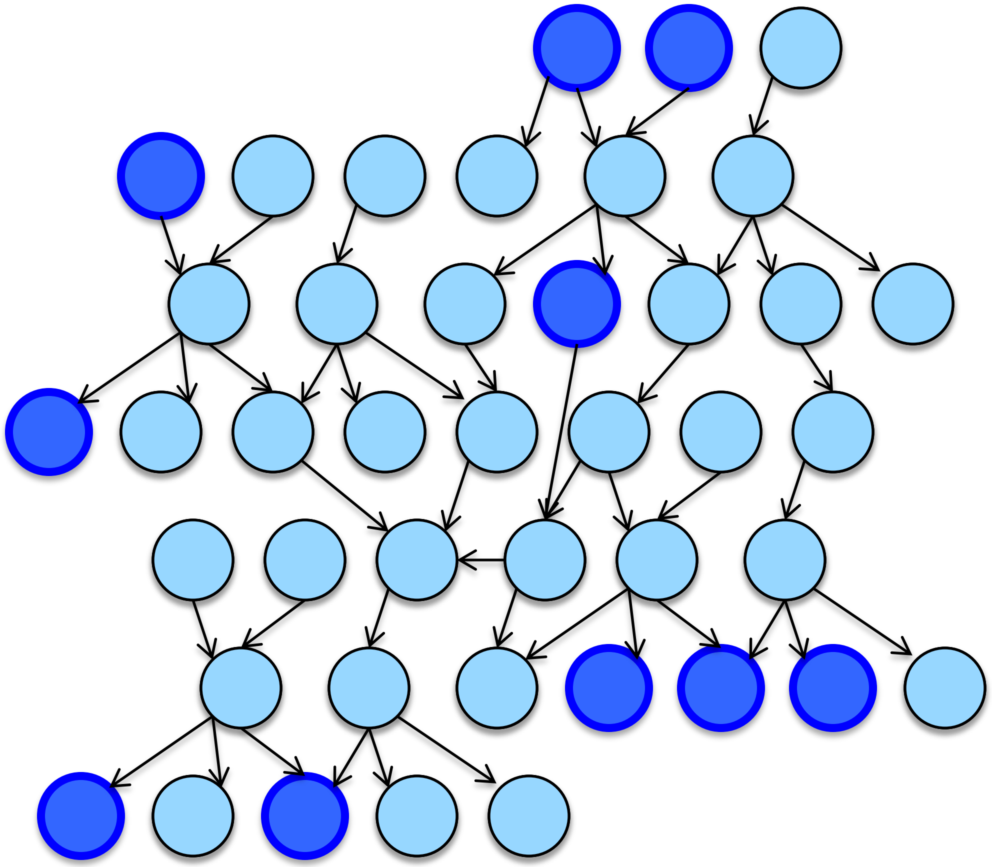
- MCMC (Markov chain Monte Carlo) is a family of randomized algorithms for approximating some quantity of interest over a very large state space
  - Markov chain = a sequence of randomly chosen states (“random walk”), where each state is chosen conditioned on the previous state
  - Monte Carlo = a very 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

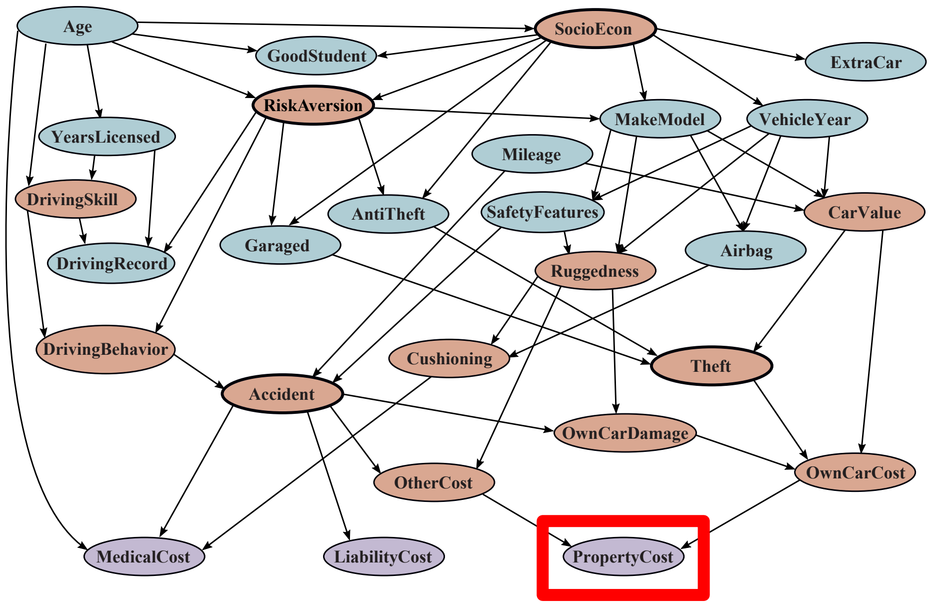
Advantages of MCMC

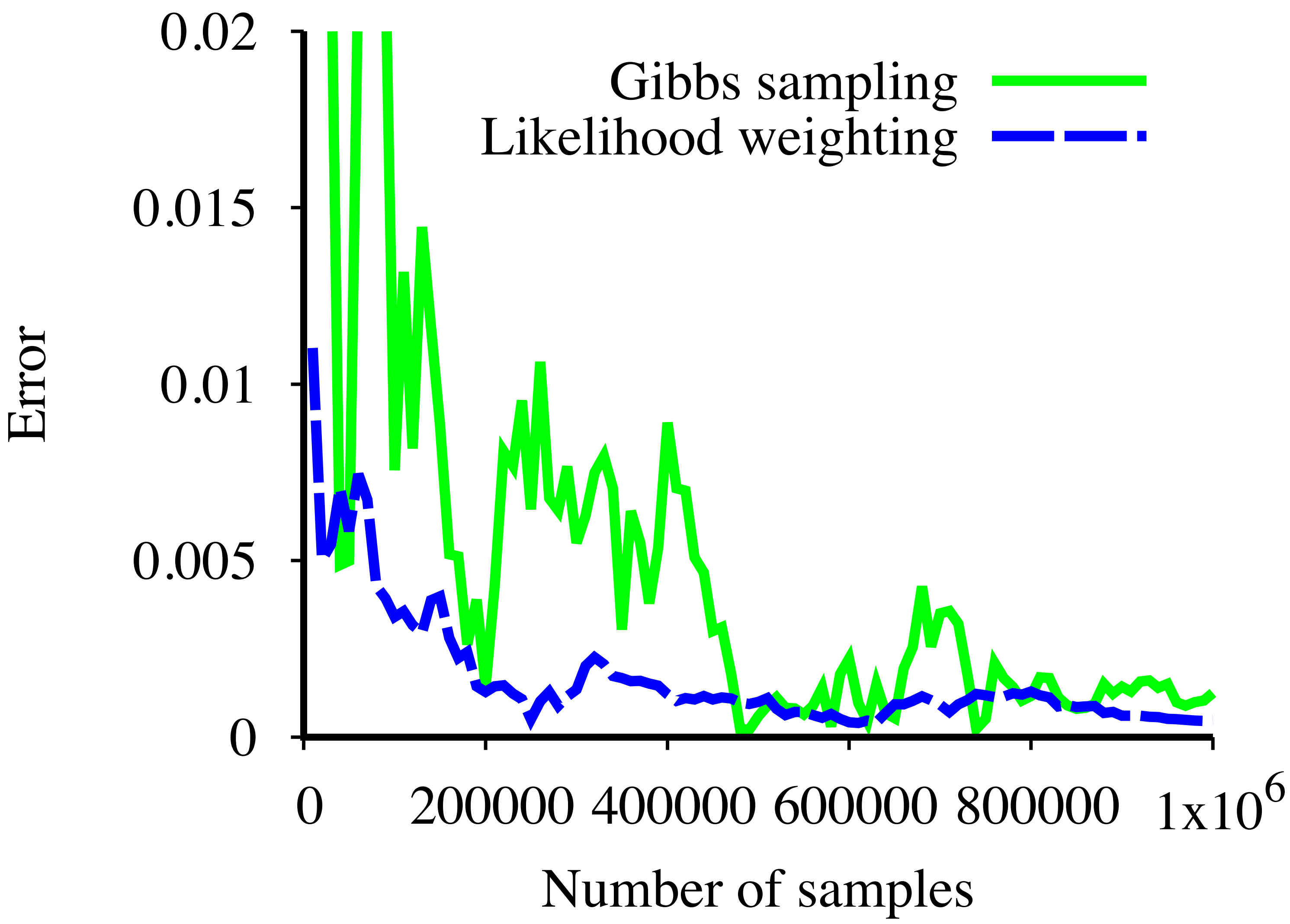


Samples soon begin to reflect all the evidence in the network

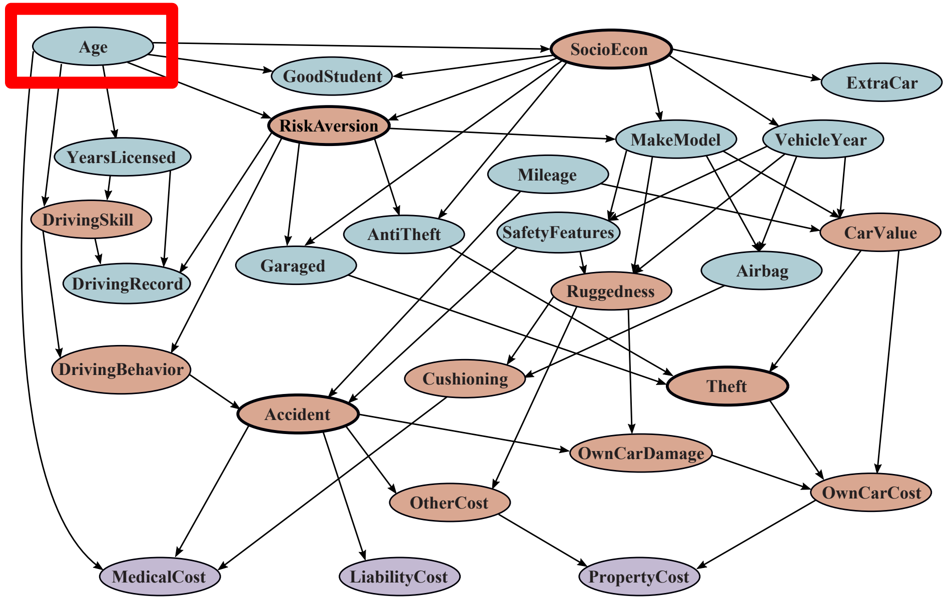
Eventually they are being drawn from the true posterior!

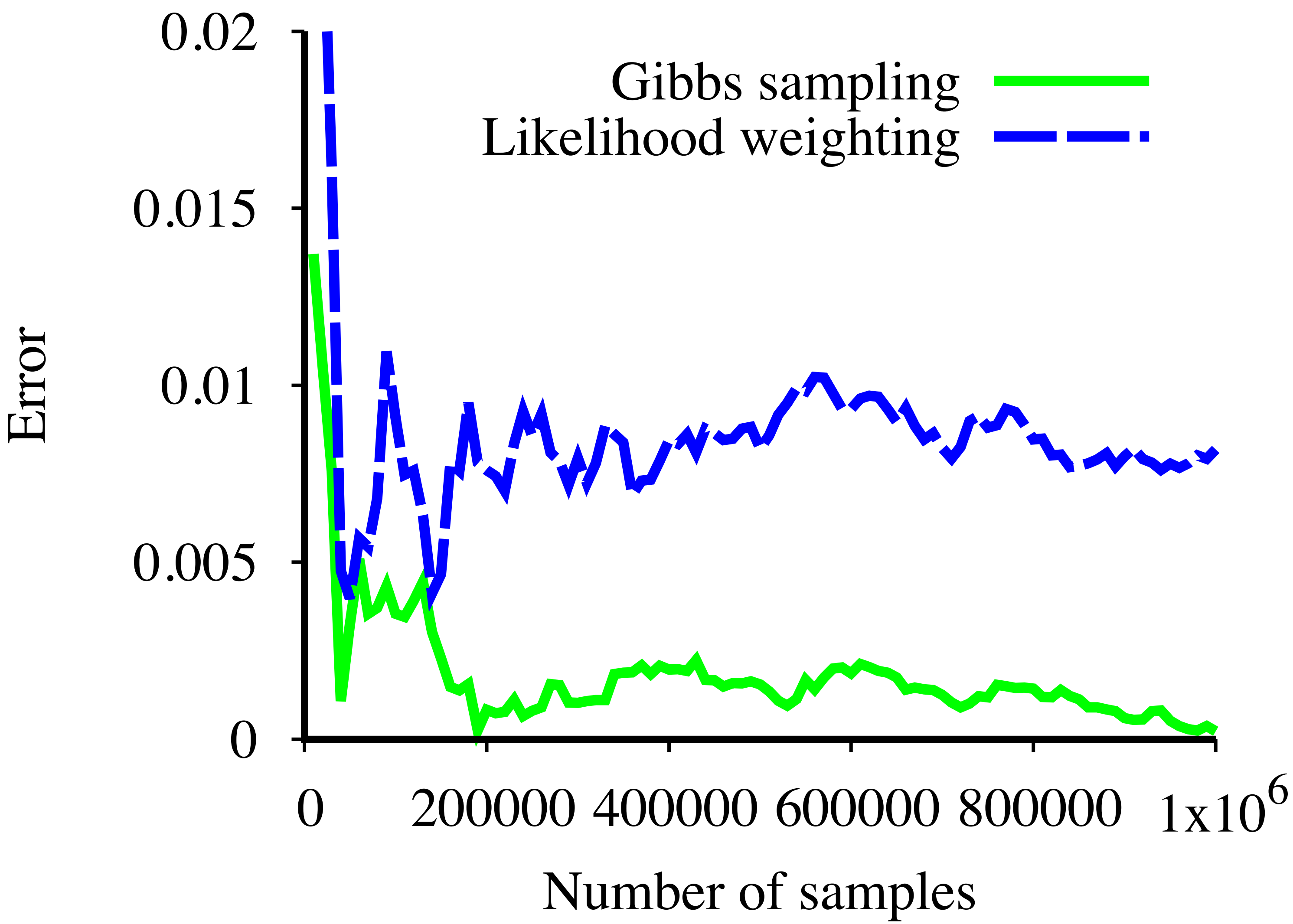
Car Insurance: *P*(*PropertyCost* | ***e***)





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



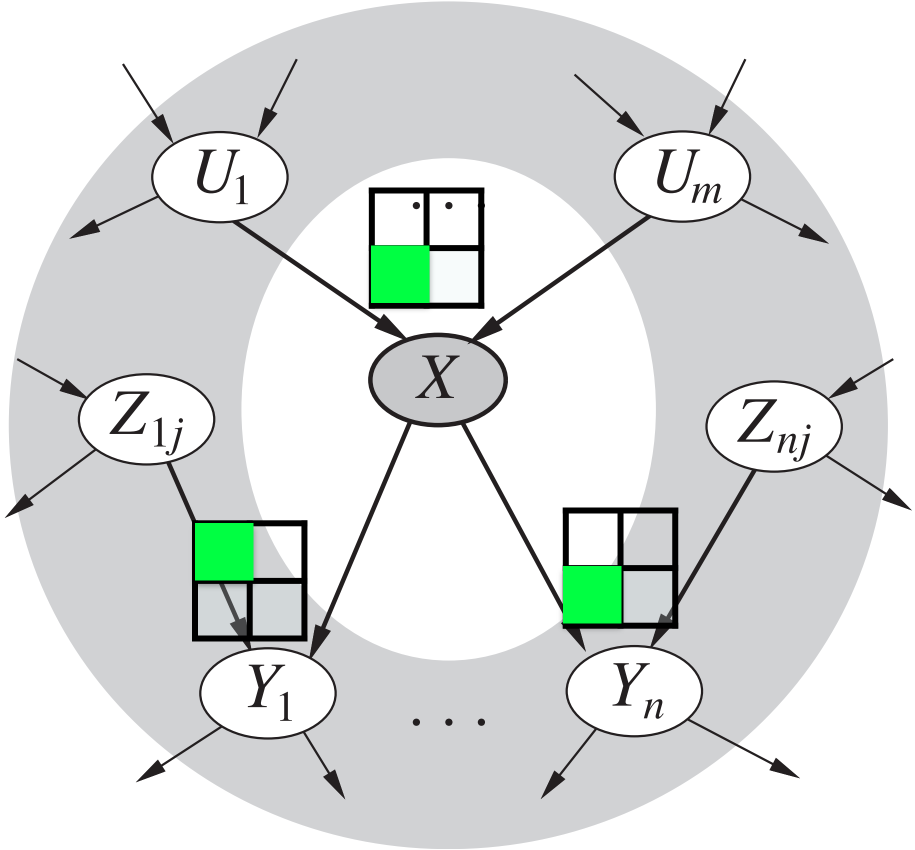


Gibbs sampling algorithm

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

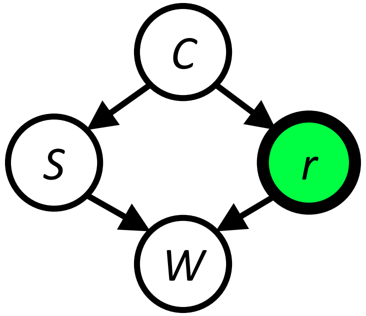
$P(X_i \mid x_1,..,x_{i-1},x_{i+1},..,x_n) = P(X_i \mid markov\_blanket(X_i))$

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

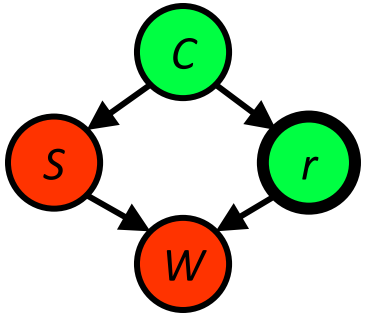


Gibbs Sampling Example: *P*(*S* | *r*)

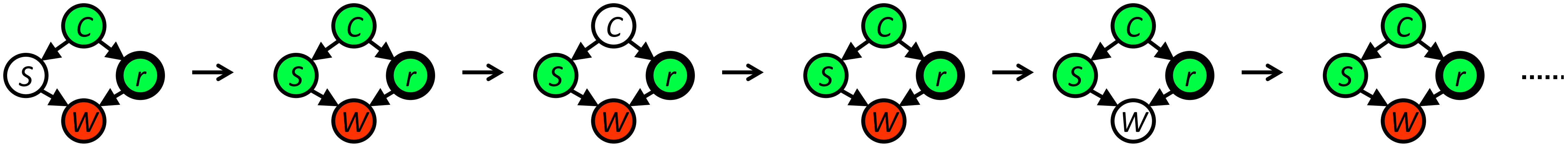
- Step 1: Fix evidence
  - *R* = true



- Step 2: Initialize other variables
  - Randomly

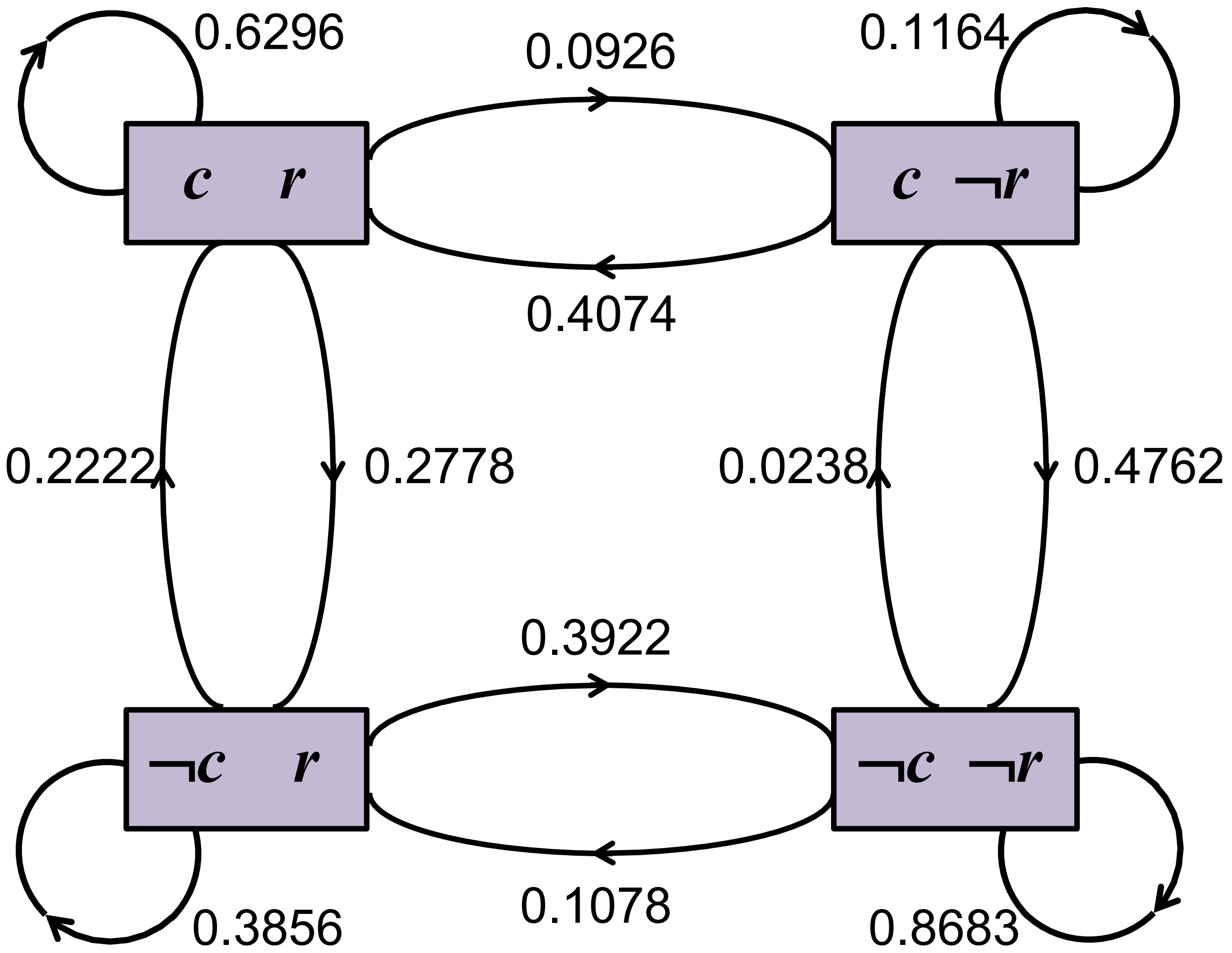


- Step 3: Repeat
  - Choose a non-evidence variable *X*
  - Resample *X* from *P*(*X* | *markov_blanket*(*X*))



Sample $S \sim P(S \mid c, r, \neg w)$ Sample $C \sim P(C \mid s, r)$ Sample $W \sim P(W \mid s, r)$

Markov chain given *s*, *w*



Gibbs sampling and MCMC in practice

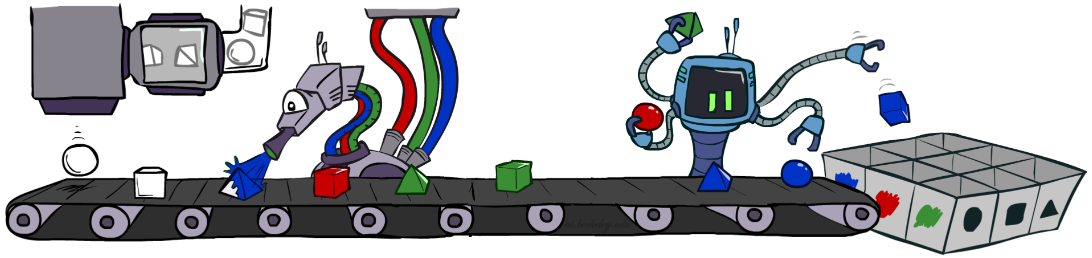
- The most commonly used method for large Bayes nets
  - See, e.g., BUGS, JAGS, STAN, infer.net, BLOG, etc.
- Can be *compiled* to run very fast
  - Eliminate all data structure references, just multiply and sample
  - ~100 million samples per second on a laptop
- Can run asynchronously in parallel (one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

Consistency of Gibbs (see AIMA 13.4.2 for details)

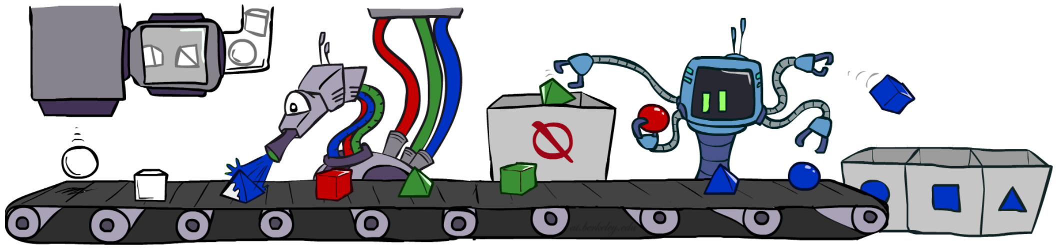
- Suppose we run it for a long time and predict the probability of reaching any given state at time t: $\pi_t(x_1,\ldots,x_n)$ or $\pi_t(\underline{\mathbf{x}})$
- Each Gibbs sampling step (pick a variable, resample its value) applied to a state $\underline{\mathbf{x}}$ has a probability $k(\underline{\mathbf{x}}' \mid \underline{\mathbf{x}})$ of reaching a next state $\underline{\mathbf{x}}'$
- So $\pi_{t+1}(\underline{\mathbf{x}}') = \sum_{\underline{\mathbf{x}}} k(\underline{\mathbf{x}}' \mid \underline{\mathbf{x}})\, \pi_t(\underline{\mathbf{x}})$ 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 $\pi = \mathrm{P}(x_1,\ldots,x_n \mid e_1,\ldots,e_k)$
    - * 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

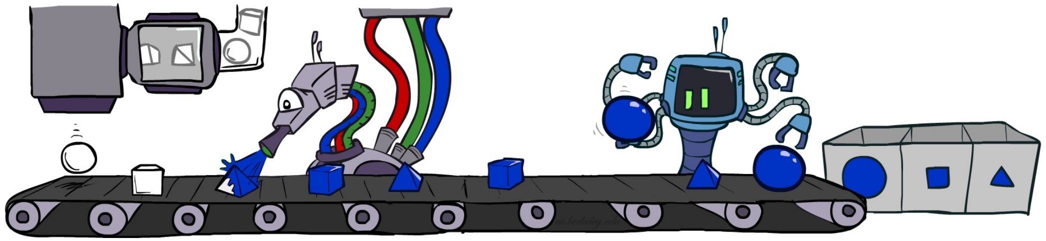
- Prior Sampling P :
  - Generate complete samples from $P(x_1,\ldots,x_n)$



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



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



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