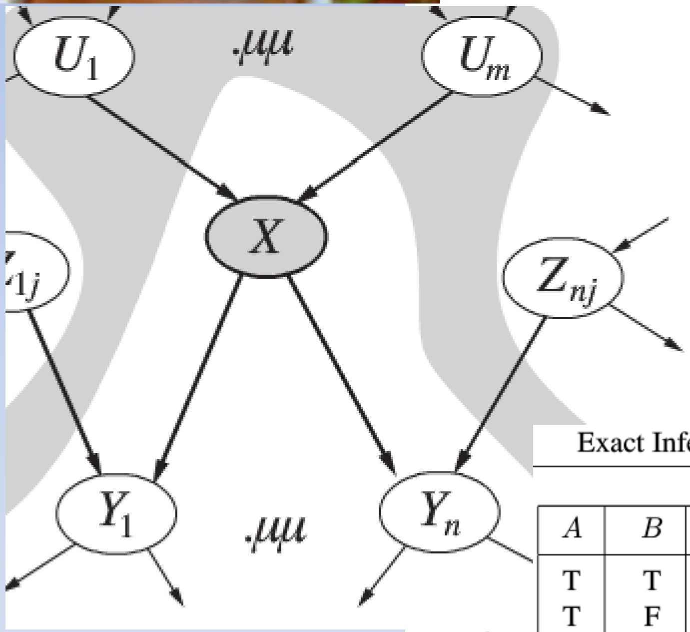
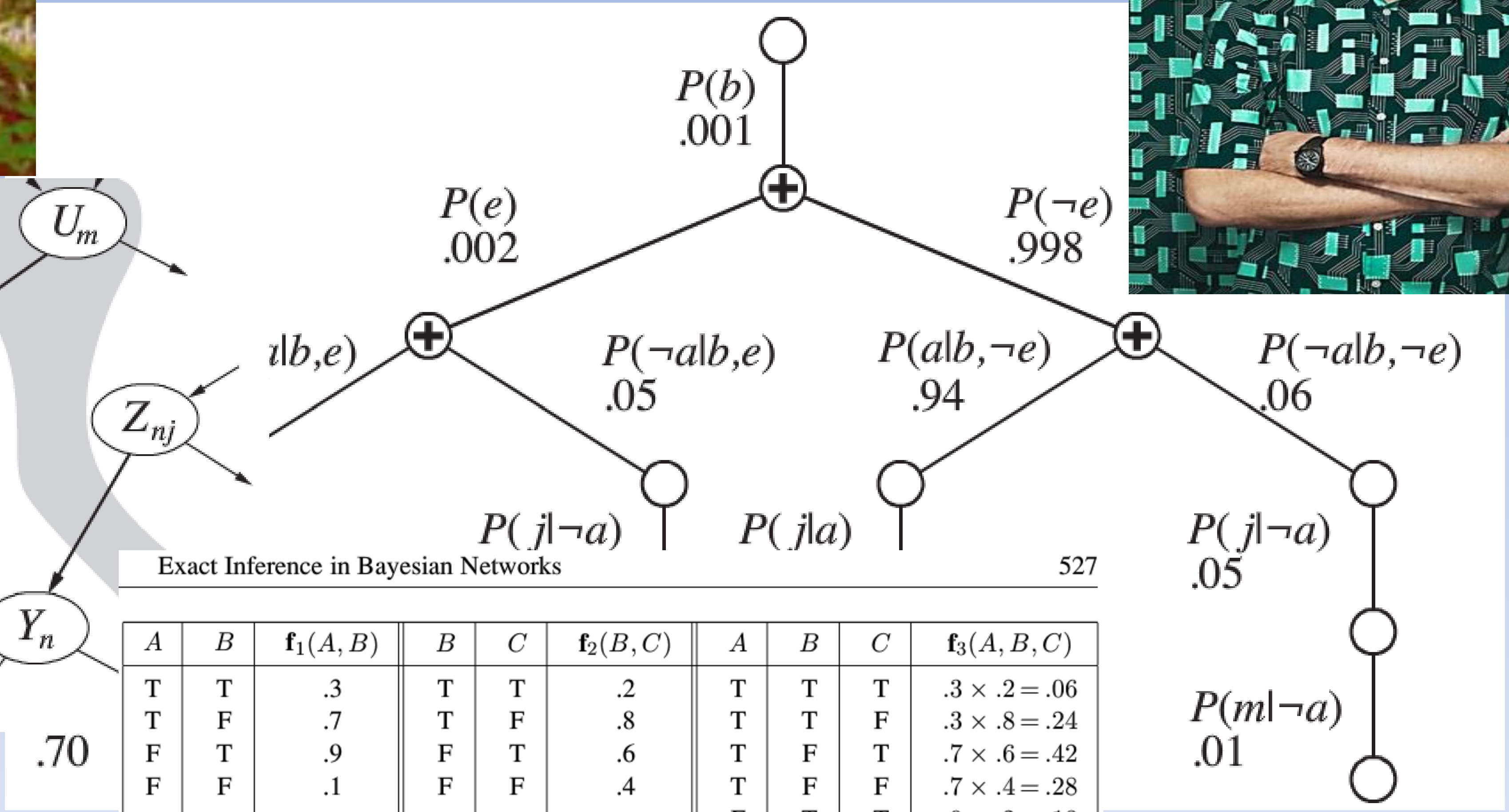
Principles of AI

Chapter 14: Probabilistic Reasoning





Exact Inference in Bayesian Networks 527

A	B	$\mathbf{f}_1(A,B)$	B	C	$\mathbf{f}_2(B,C)$	A	B	C	$\mathbf{f}_3(A,B,C)$
T	T	.3	T	T	.2	T	T	T	$.3 \times .2 = .06$
T	F	.7	T	F	.8	T	T	F	$.3 \times .8 = .24$
F	T	.9	F	T	.6	T	F	T	$.7 \times .6 = .42$
F	F	.1	F	F	.4	T	F	F	$.7 \times .4 = .28$
						F	T	T	$.9 \times .2 = .18$
						F	T	F	$.9 \times .8 = .72$
						F	F	T	$.1 \times .6 = .06$
						F	F	F	$.1 \times .4 = .04$

Figure 14.10 Illustrating pointwise multiplication: $\mathbf{f}_1(A,B) \times \mathbf{f}_2(B,C) = \mathbf{f}_3(A,B,C)$.

Exact Inference: Enumeration

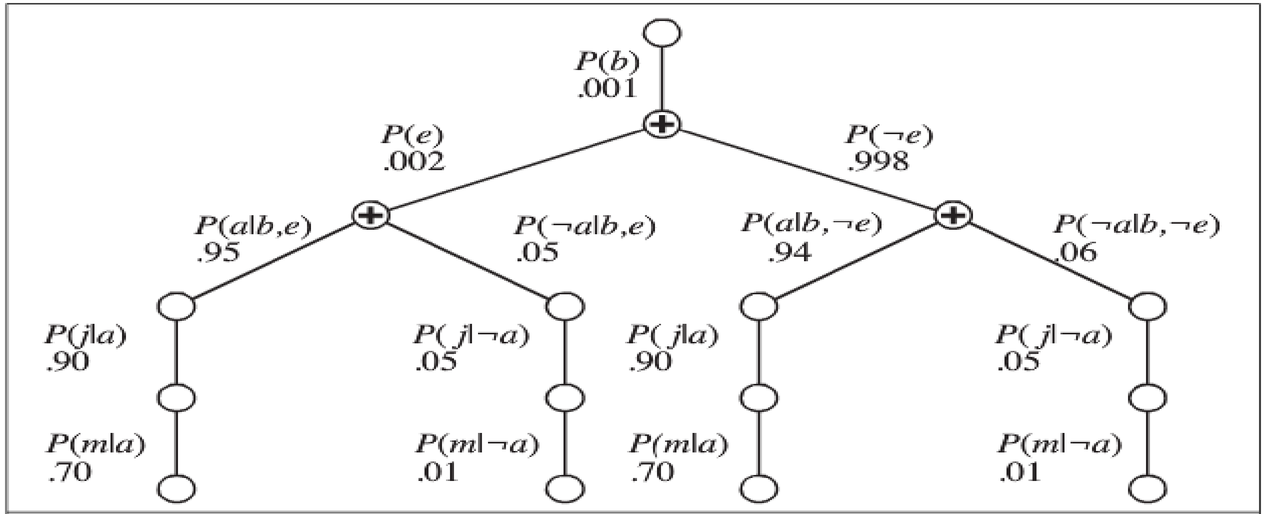


Figure 14.8 The structure of the expression shown in Equation (14.4). The evaluation proceeds top down, multiplying values along each path and summing at the "+" nodes. Notice the repetition of the paths for j and m.

- Recursive depth-first enumeration:
  - – O(n) space,
  - – $O(d^n)$ time

- Lots of repeated calculations
- Maybe Dynamic Programming!

Variable Elimination

- Variable elimination:
  - carry out summations right-to-left,
  - store intermediate results (factors) to avoid recomputation

function ELIMINATION-ASK(X, $\mathbf{e}$, bn) **returns** a distribution over X
  inputs: X, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $factors \leftarrow [\,]$
  for each var **in** ORDER(bn.VARS) **do**
    $factors \leftarrow$ [MAKE-FACTOR(var, $\mathbf{e}$)|$factors$]
    if var is a hidden variable **then** $factors \leftarrow$ SUM-OUT(var, $factors$)
  return NORMALIZE(POINTWISE-PRODUCT($factors$))

Procedure Sum-Product-VE (
  Φ, // Set of factors
  $\boldsymbol{Z}$, // Set of variables to be eliminated
  $\prec$ // Ordering on $\boldsymbol{Z}$
)
  Let $Z_1, \ldots, Z_k$ be an ordering of $\boldsymbol{Z}$ such that
    $Z_i \prec Z_j$ if and only if $i < j$
  for $i = 1, \ldots, k$
    $\Phi \leftarrow$ Sum-Product-Eliminate-Var(Φ, Z_i)
  $\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$
  return ϕ^*

Procedure Sum-Product-Eliminate-Var (
  Φ, // Set of factors
  Z // Variable to be eliminated
)
  $\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$
  $\Phi'' \leftarrow \Phi - \Phi'$
  $\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
  $\tau \leftarrow \sum_Z \psi$
  return $\Phi'' \cup \{\tau\}$

function ELIMINATION-ASK(X, **e**, bn) **returns** a distribution over X
inputs: X, the query variable
e, observed values for variables **E**
bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$


```
Procedure Sum-Product-VE (
   Φ,     // Set of factors
   Z,     // Set of variables to be eli
   ≺      // Ordering on Z
)
   Let Z_1, ..., Z_k be an orderin
     Z_i ≺ Z_j if and only if i < 
   for i = 1, ..., k
     Φ ← Sum-Product-Elimina
   φ* ← ∏_{φ∈Φ} φ
   return φ*

Procedure Sum-Product-Elimin
   Φ,     // Set of factors
   Z      // Variable to be eliminated
)
   Φ' ← {φ ∈ Φ : Z ∈ Scope[φ]}
   Φ'' ← Φ − Φ'
   ψ ← ∏_{φ∈Φ'} φ
   τ ← ∑_Z ψ
   return Φ'' ∪ {τ}
```

Algorithm 9.2 Using Sum-Product-VE **for computing conditional probabilities**

```
Procedure Cond-Prob-VE (
   K,      // A network over X
   Y,      // Set of query variables
   E = e   // Evidence
)
   Φ ← Factors parameterizing K
   Replace each φ ∈ Φ by φ[E = e]
   Select an elimination ordering ≺
   Z ← = X − Y − E
   φ* ← Sum-Product-VE(Φ, ≺, Z)
   α ← ∑_{y∈Val(Y)} φ*(y)
   return α, φ*
```

Eliminate One Hidden Variable

Procedure Sum-Product-Eliminate-Var (

Φ, // Set of factors

Z // Variable to be eliminated

)

$\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$

$\Phi'' \leftarrow \Phi - \Phi'$

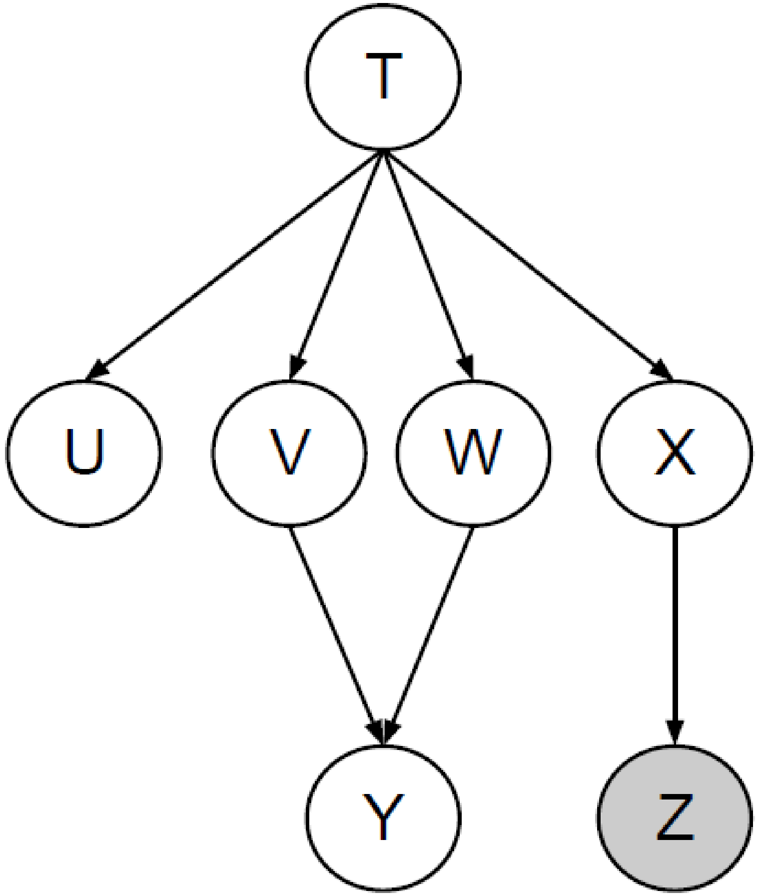
$\psi \leftarrow \prod_{\phi \in \Phi'} \phi$

$\tau \leftarrow \sum_{Z} \psi$

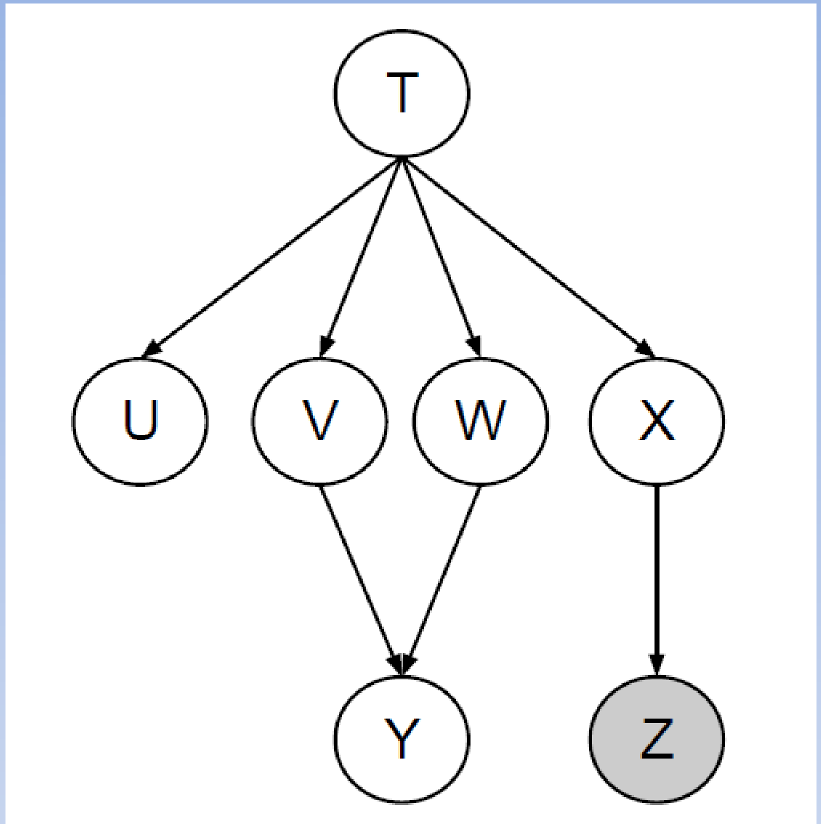
return $\Phi'' \cup \{\tau\}$

2 Variable Elimination

For the Bayes' net below, we are given the query $P(Y \mid +z)$. All variables have binary domains. Assume we run variable elimination to compute the answer to this query, with the following variable elimination ordering: X, T, U, V, W.



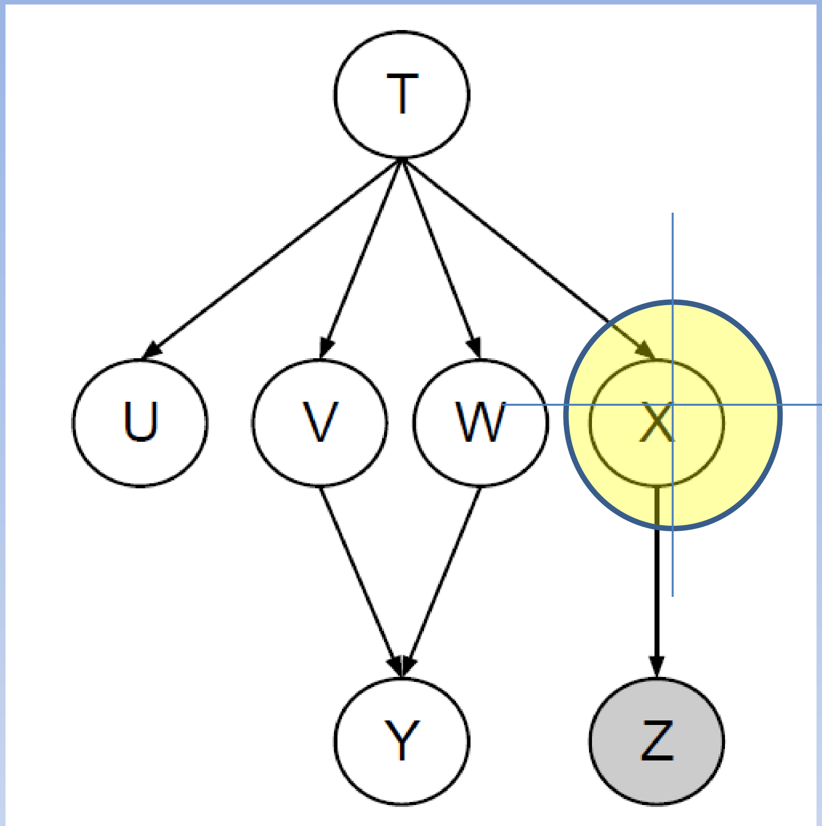
P(Y | +z):
Eliminate Hidden Variables



- Initial Factors after inserting evidence:
- P(T), P(U|T), P(V |T), P(W|T), P(X|T), P(Y |V,W), P(+z|X)

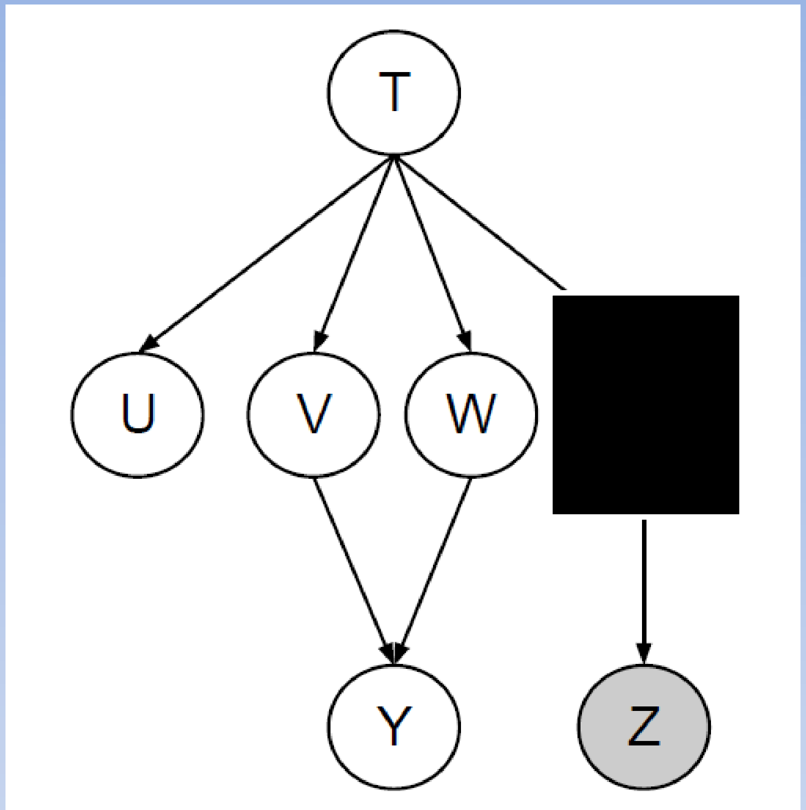
$P(Y \mid +z)$

Eliminate First Hidden Variable:
X



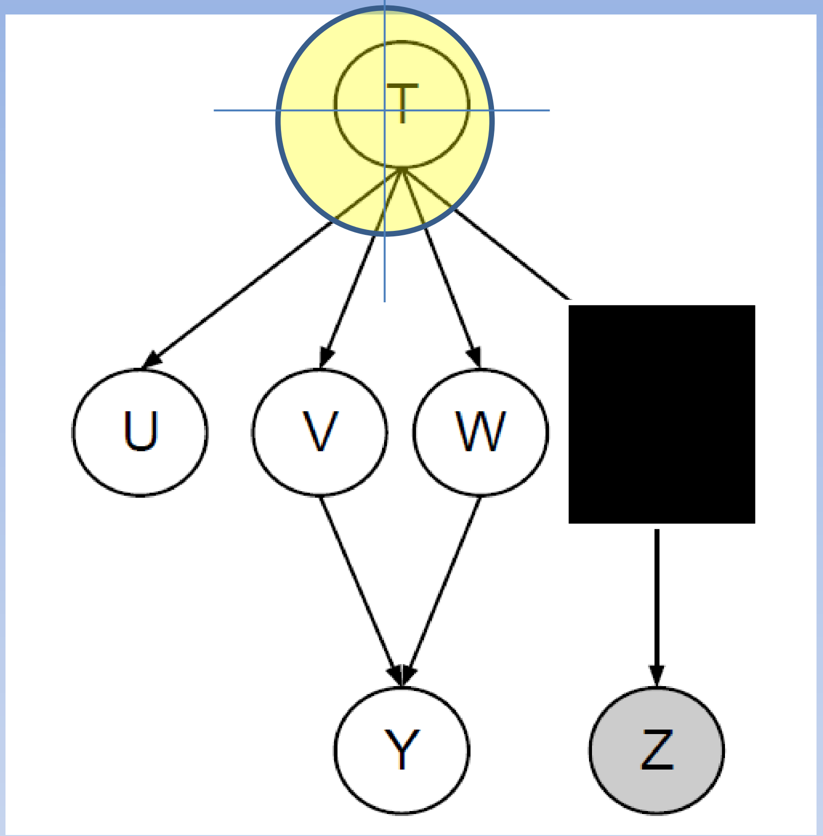
- P(T), P(U|T), P(V |T), P(W|T), P(X|T), P(Y |V,W), P(+z|X)
- (a) Now Eliminate X and generate a new factor f1:

P(Y | +z)



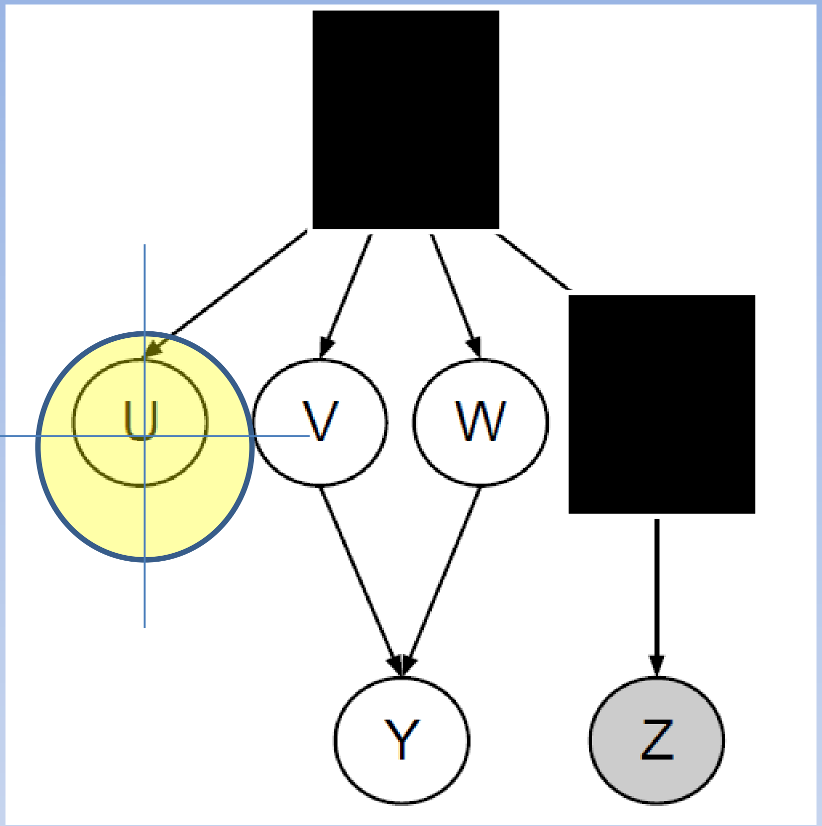
- P(T), P(U|T), P(V |T), P(W|T), P(X|T), P(Y|V,W), P(+z|X)
- (a) Now Eliminate X and generate a new factor f1:
  - Step 1: Collect/Combine Factors
    - P(X|T), P(+z|X)
  - Step 2: Marginalize out X
    - f1(T, +z)=$\sum_x P(x|T)P(+z|x)$
  - (b) Leaving us w/ Factors=P(T), P(U|T), P(V |T), P(W|T), P(Y |V,W), **f1(T,+z)**

P(Y | +z)
Eliminate Second Hidden Variable: **T**



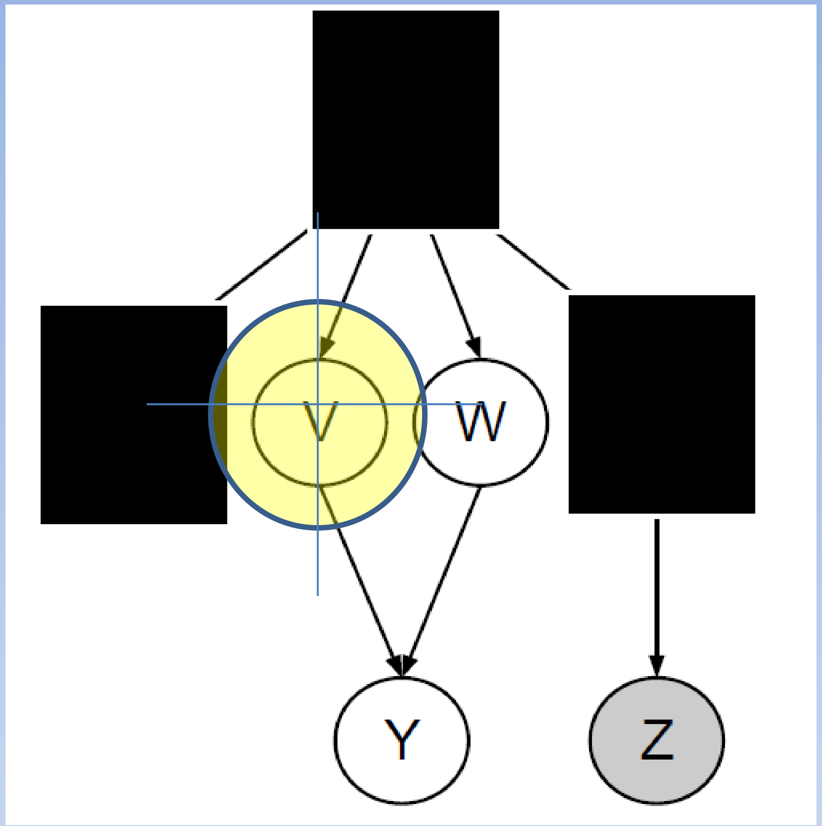
- P(T), P(U|T), P(V |T), P(W|T), P(Y |V,W), f1(T,+z)
- (c) Eliminate T -- generating a new factor f2:
  - Step 1: Collect/Combine Factors
    - P(T), P(U|T), P(V |T), P(W|T), f1(T,+z)
  - Step 2: Marginalize out T
    - f2(U, V, W, +z) =$\sum_t$ P(T), P(U|T), P(V |T), P(W|T), f1(T,+z)
  - (d) Leaving us w/ Factors= P(Y|V,W), f2(U,V,W,+z)

P(Y | +z)
Eliminate Third Hidden Variable: **U**



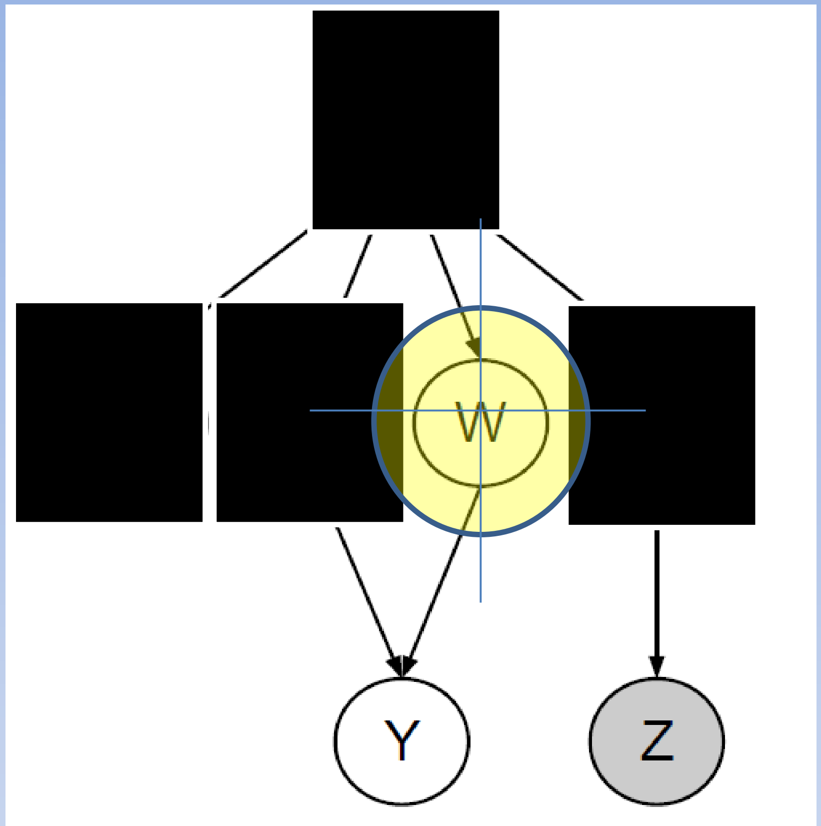
- P(Y |V,W), f2(U, V, W, +z)
- (e) Eliminate U, generate a new factor f3 :
  - Step 1: Collect/Combine Factors
    - f2(U, V, W, +z)
  - Step 2: Marginalize out T
    - f3(V, W, +z)=$\sum_u$ f2(u, V, W, +z)
  - (f) Leaving us w/ Factors= P(Y|V,W), f3(V,W,+z)

P(Y | +z)
Eliminate Fourth Hidden Variable: **V**



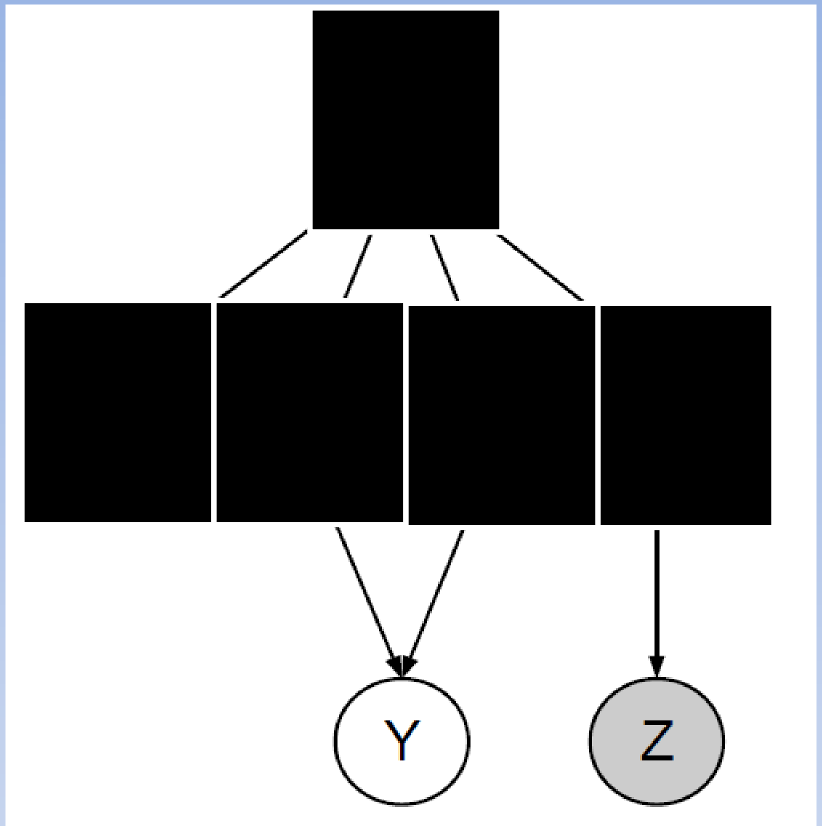
- P(Y |V,W), f3(V, W, +z)
- (g) Eliminating V, generate a new factor f4:
  - Step 1: Collect/Combine Factors
    - P(Y |V,W), f3(V, W, +z)
  - Step 2: Marginalize out V
    - f4(Y, W, +z)=$\sum_{v}$ P(Y |v,W), f3(v, W, +z)
  - (h) Leaving us w/ Factors= f4(Y, W, +z)

P(Y | +z)
Eliminate Fifth Hidden Variable: **W**



- f4(Y, W, +z)
- (i) Eliminating W, generate a new factor f5:
  - Step 1: Collect/Combine Factors
    - f4(Y, W, +z)
  - Step 2: Marginalize out W
    - f5(Y, +z)=$\sum_w$ f4(Y, W, +z)
  - (j) Leaving us w/ Factors= f5(Y, +z)

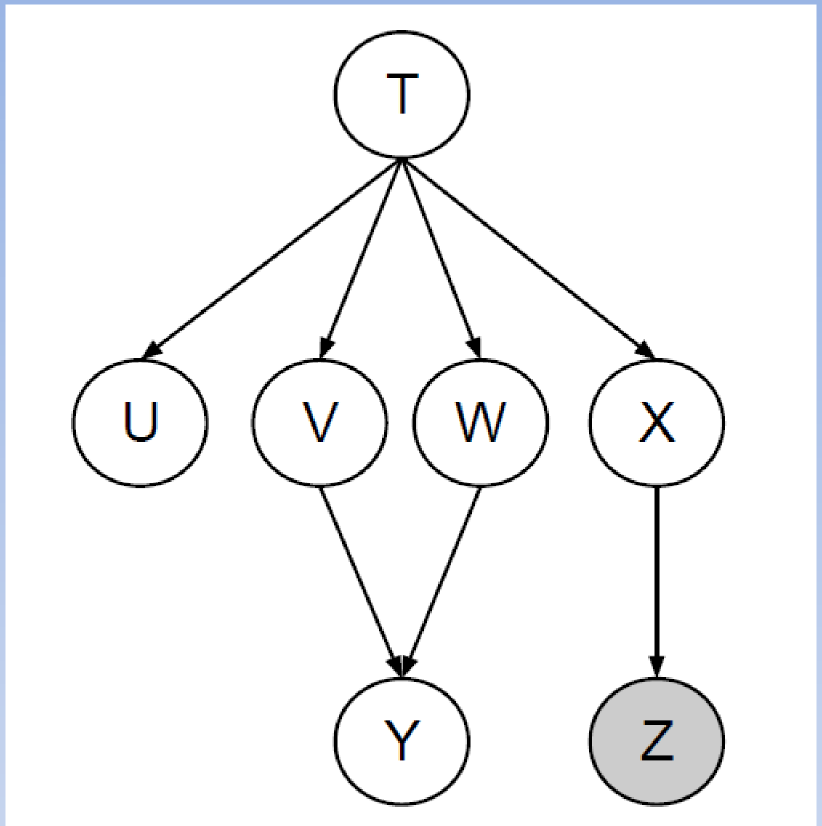
P(Y | +z)



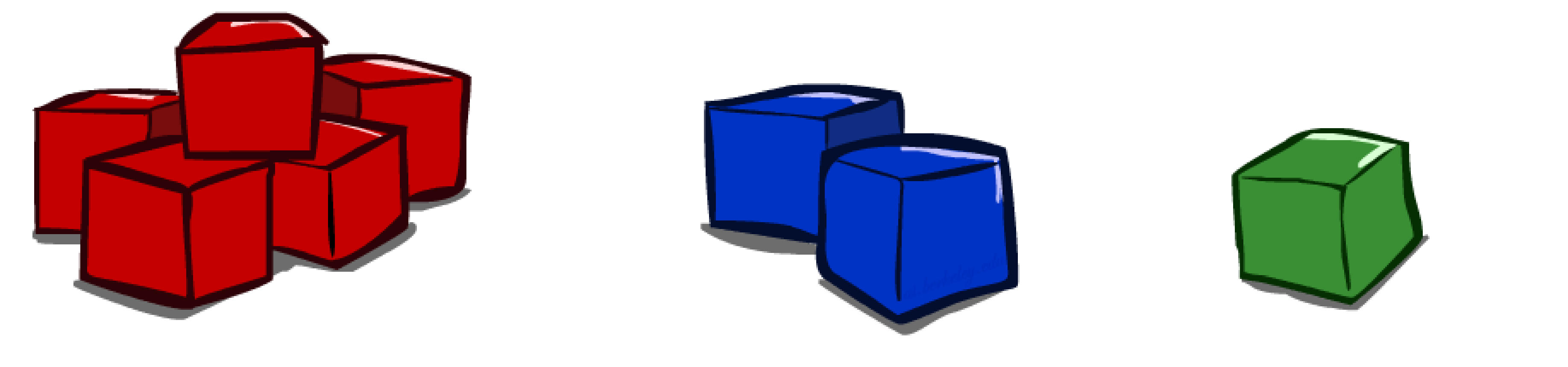
- f5(Y, +z)
- (k) How would you obtain P(Y|+z) from the factors left above:
  - Simply renormalize f5(Y,+z) to obtain P(Y|+z):
    - P(Y|+z)=$\frac{\text{f5(y,+z)}}{\sum_{y'} \text{f5}(y', +z)}$
- (l) What is the size of the largest factor generated?
- (m) Does there exist a better elimination ordering (one which generates smaller largest factors)?

P(Y | +z)

- f5(Y, +z)
- (k) How would you obtain P(Y j +z) from the factors left above:
  - – Simply renormalize f5(Y,+z) to obtain P(Y|+z):
    - $P(Y|+z)=\frac{f5(y,+z)}{\sum_{y'} f5(y', +z)}$
- (l) What is the size of the largest factor generated?
- (m) Does there exist a better elimination ordering (one which generates smaller largest factors)? **[X, U, V, T, W]**



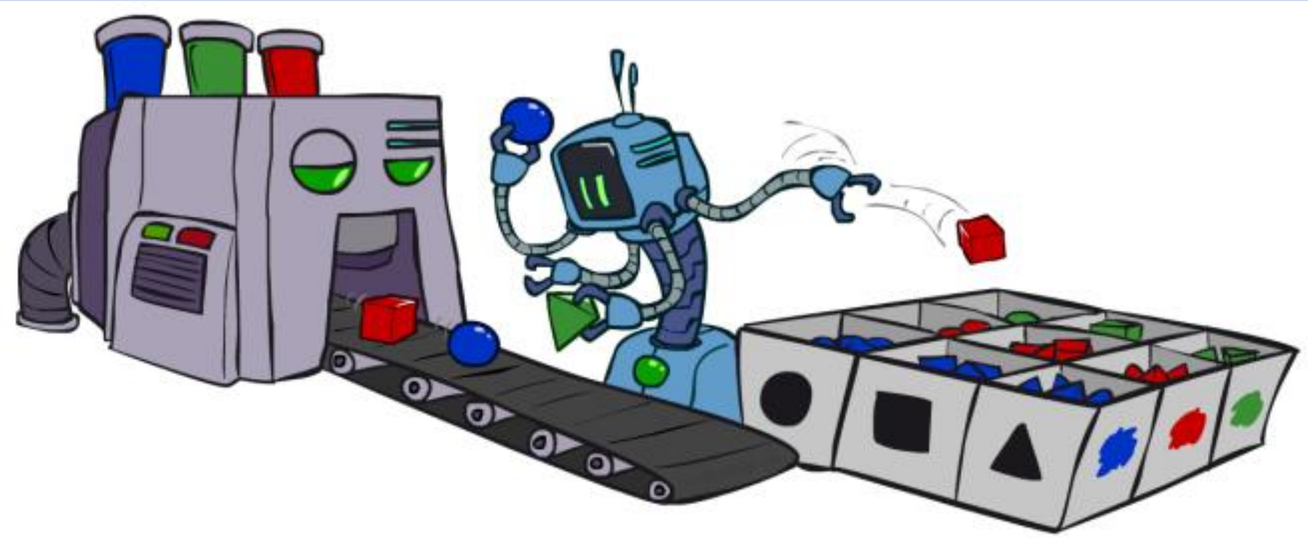
Approximate Inference: Sampling



Sampling

- Sampling is a lot like repeated simulation
  - Predicting the weather, basketball games, …
- 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?
  - Learning: get samples from a distribution you don't know
  - Inference: getting a sample is faster than computing the right answer (e.g. with variable elimination)



Sampling

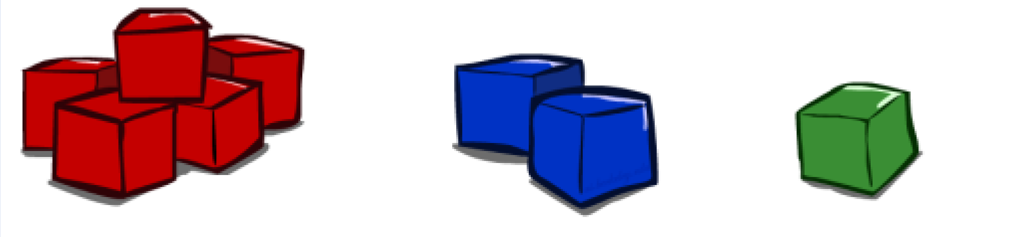
- Sampling from given distribution
  - Step 1: Get sample u from uniform distribution over [0, 1)
    - E.g. random() in python
  - Step 2: Convert this sample u into an outcome for the given distribution by having each outcome associated with a sub-interval of [0,1) with sub-interval size equal to probability of the outcome

- Example

C	P(C)
red	0.6
green	0.1
blue	0.3

$$0 \leq u < 0.6, \rightarrow C = red$$
$$0.6 \leq u < 0.7, \rightarrow C = green$$
$$0.7 \leq u < 1, \rightarrow C = blue$$

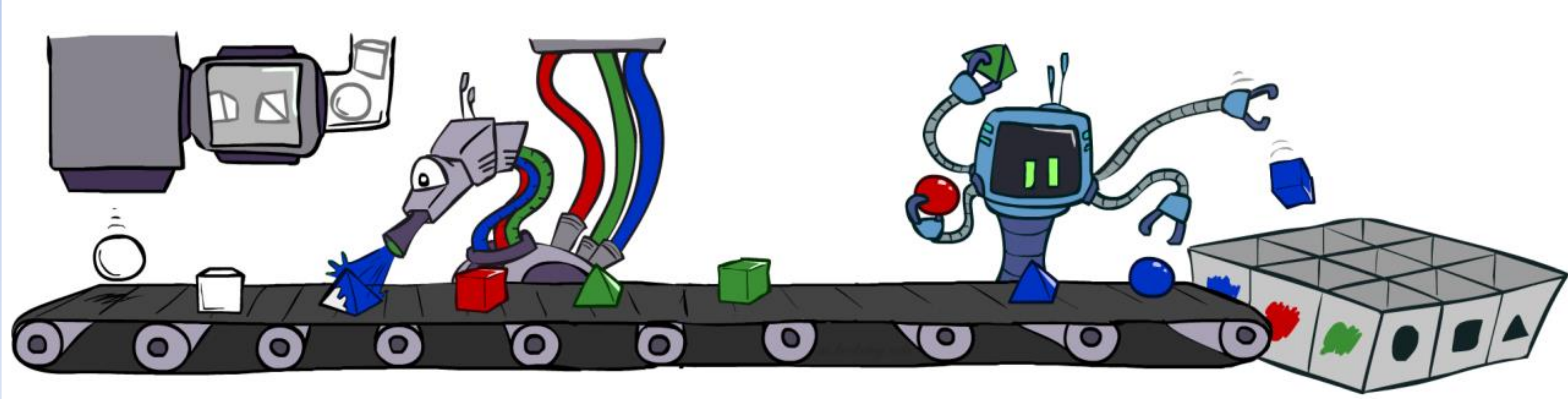
- If random() returns u = 0.83, then our sample is C = blue
- E.g, after sampling 8 times:



Sampling in Bayes’ Nets

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

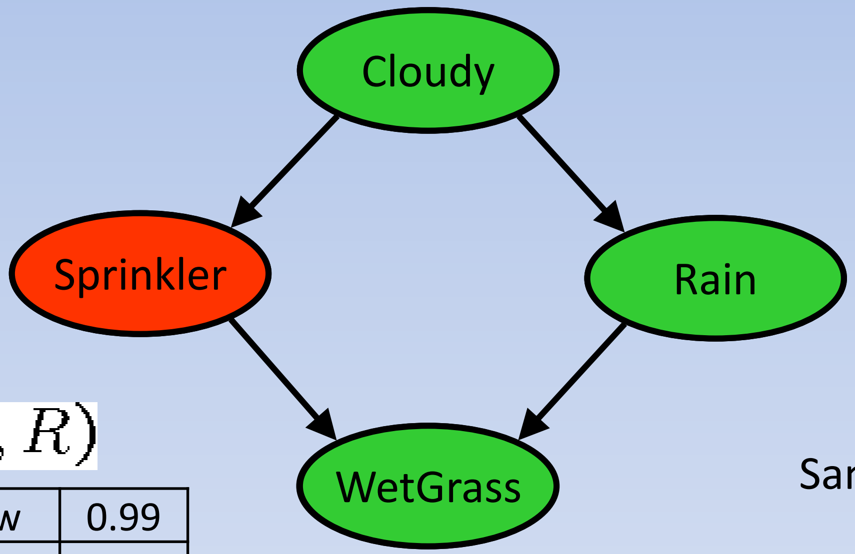
Prior Sampling



Prior Sampling

$P(C)$

+c	0.5
-c	0.5



$P(S|C)$

+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:

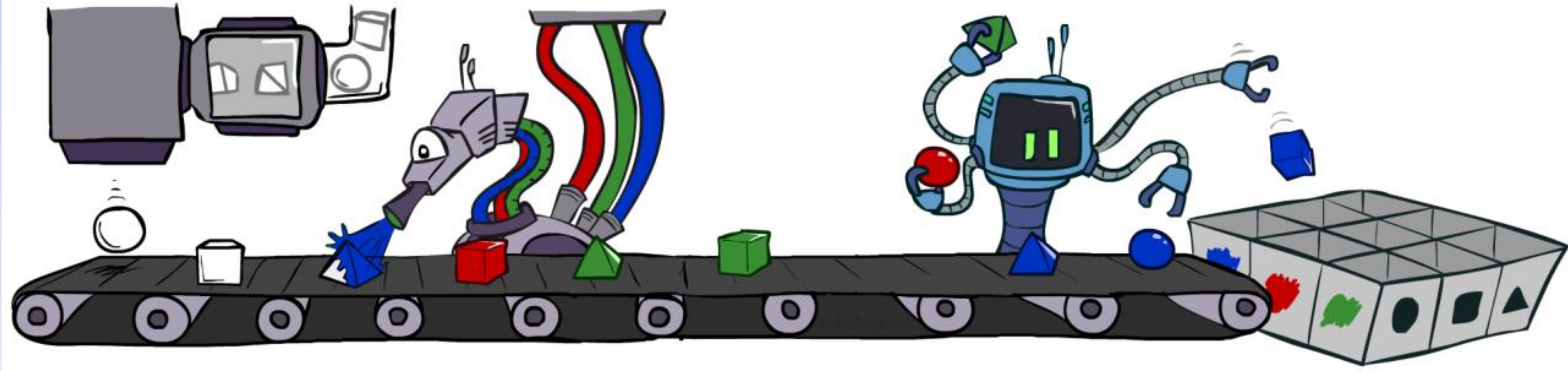
+c, -s, +r, +w

-c, +s, -r, +w

…

Prior Sampling

- For i=1, 2, …, n
  - Sample x_i from $P(X_i \mid Parents(X_i))$
- Return $(x_1, x_2, \ldots, x_n)$



function PRIOR-SAMPLE(bn) **returns** an event sampled from the prior specified by bn
  inputs: bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $\mathbf{x} \leftarrow$ an event with n elements
  foreach variable X_i **in** $X_1, \ldots, X_n$ **do**
    $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return x

Figure 14.13 A sampling algorithm that generates events from a Bayesian network. Each variable is sampled according to the conditional distribution given the values already sampled for the variable's parents.

Sampling from an empty network

function Prior-Sample(bn) **returns** an event sampled from bn
    inputs: bn, a belief network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

    $\mathbf{x} \leftarrow$ an event with n elements
    for $i = 1$ **to** n **do**
        $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
            given the values of $Parents(X_i)$ in $\mathbf{x}$
    return $\mathbf{x}$

Prior Sampling

- This process generates samples with probability:

$$S_{PS}(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i|\text{Parents}(X_i)) = P(x_1 \ldots x_n)$$

...i.e. the BN's joint probability

- Let the number of samples of an event be $N_{PS}(x_1 \ldots x_n)$
- Then

$$\begin{aligned}\lim_{N\to\infty} \hat{P}(x_1,\ldots,x_n) &= \lim_{N\to\infty} N_{PS}(x_1,\ldots,x_n)/N \\ &= S_{PS}(x_1,\ldots,x_n) \\ &= P(x_1 \ldots x_n)\end{aligned}$$

- I.e., the sampling procedure is consistent

Example

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

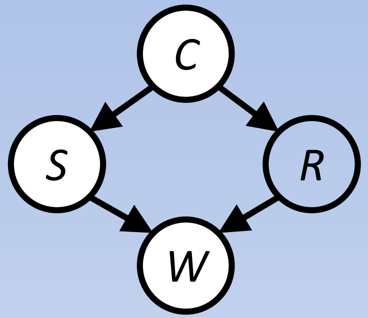
  +c, -s, +r, +w

  +c, +s, +r, +w

  -c, +s, +r, -w

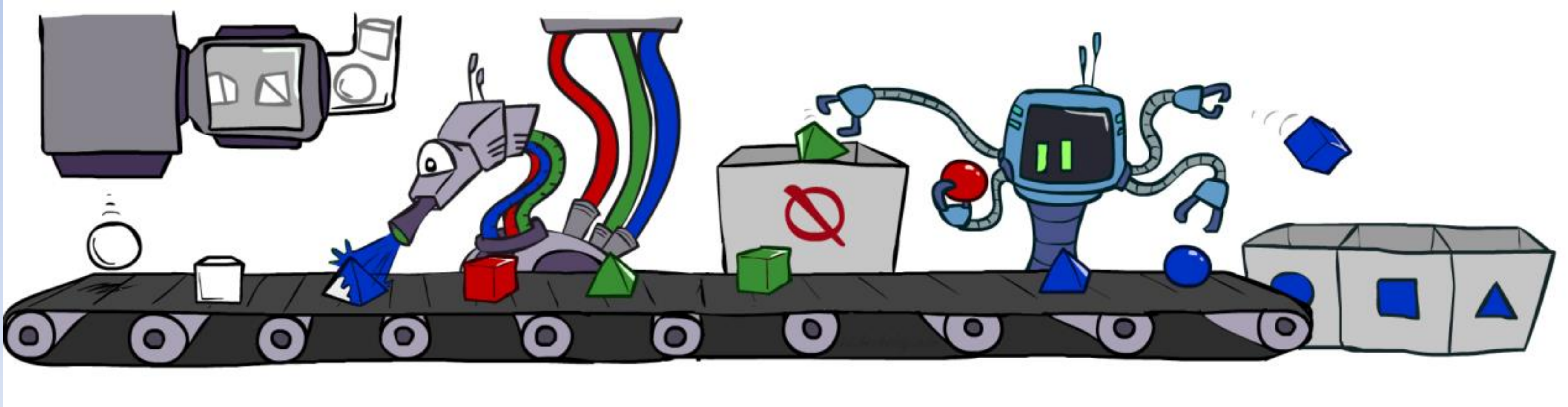
  +c, -s, +r, +w

  -c, -s, -r, +w



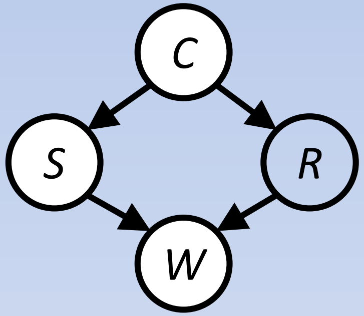
- If we want to know P(W)
  - We have counts <+w:4, -w:1>
  - Normalize to get P(W) = <+w:0.8, -w:0.2>
  - This will get closer to the true distribution with more samples
  - Can estimate anything else, too
  - What about P(C| +w)?   P(C| +r, +w)? P(C| -r, -w)?
  - Fast: can use fewer samples if less time (what's the drawback?)

Rejection Sampling



Rejection Sampling

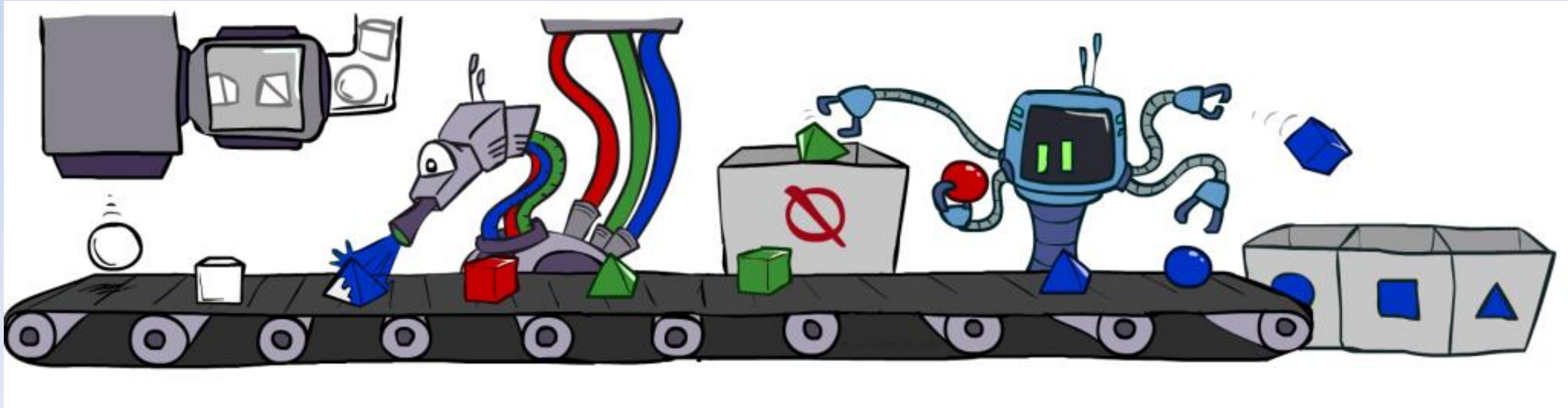
- Let' s say we want P(C)
  - No point keeping all samples around
  - Just tally counts of C as we go
- Let' s say we want P(C| +s)
  - Same thing: tally C outcomes, but ignore (reject) samples which don' t have S=+s
  - This is called rejection sampling
  - It is also consistent for conditional probabilities (i.e., correct in the limit)



+c, -s, +r, +w
+c, +s, +r, +w
-c, +s, +r, -w
+c, -s, +r, +w
-c, -s, -r, +w

Rejection Sampling

- IN: evidence instantiation
- 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 $(x_1, x_2, \dots, x_n)$

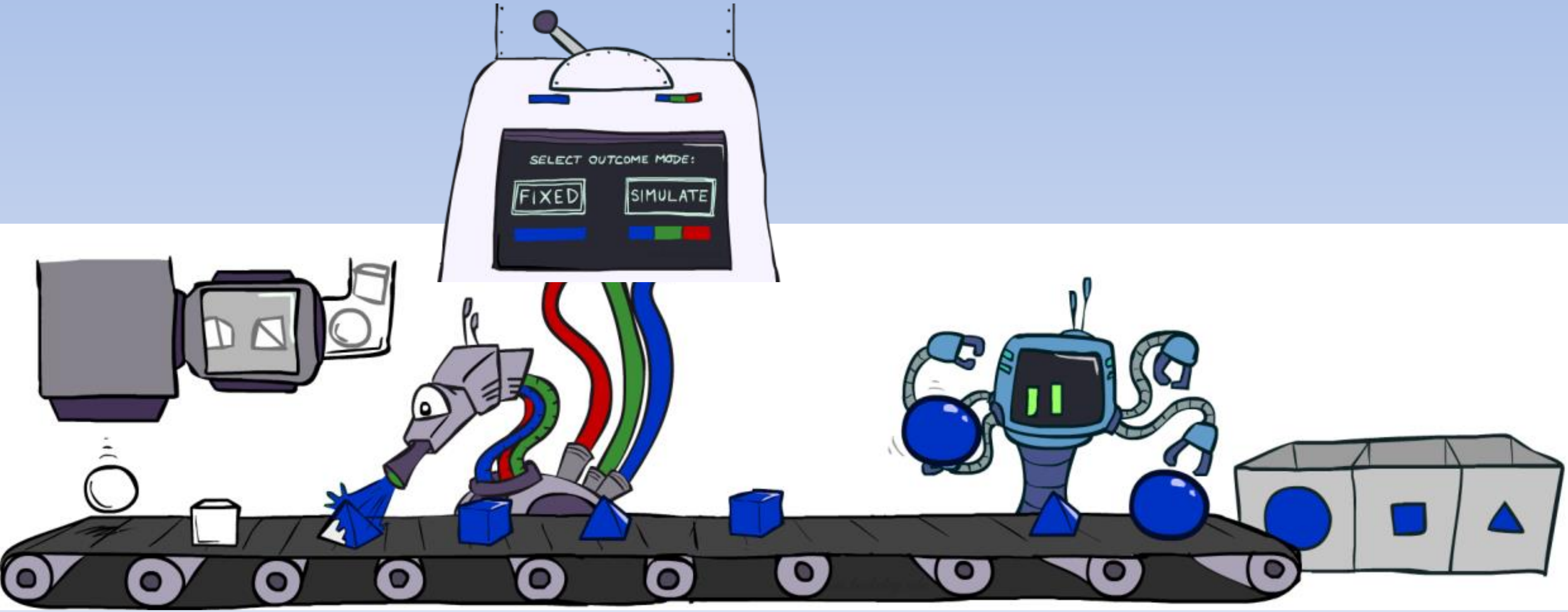


function REJECTION-SAMPLING(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
inputs: X, the query variable
e, observed values for variables **E**
bn, a Bayesian network
N, the total number of samples to be generated
local variables: **N**, a vector of counts for each value of X, initially zero

for $j = 1$ to N **do**
  x ← PRIOR-SAMPLE(bn)
  if x is consistent with **e then**
    $\mathbf{N}[x] \leftarrow \mathbf{N}[x]+1$ where x is the value of X in **x**
return NORMALIZE(**N**)

Figure 14.14 The rejection-sampling algorithm for answering queries given evidence in a Bayesian network.

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|blue)

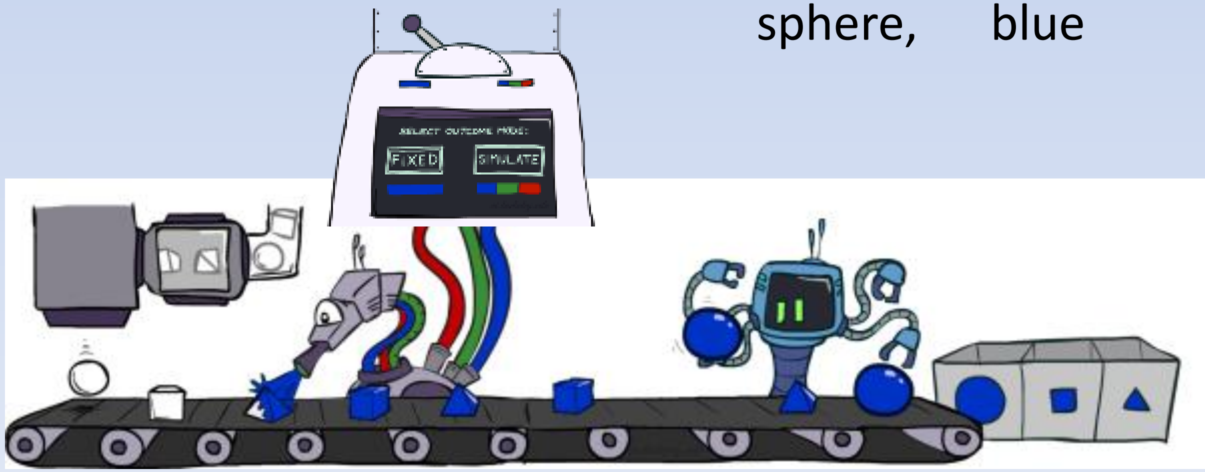
Shape → Color

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

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

Shape → ~~Color~~

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



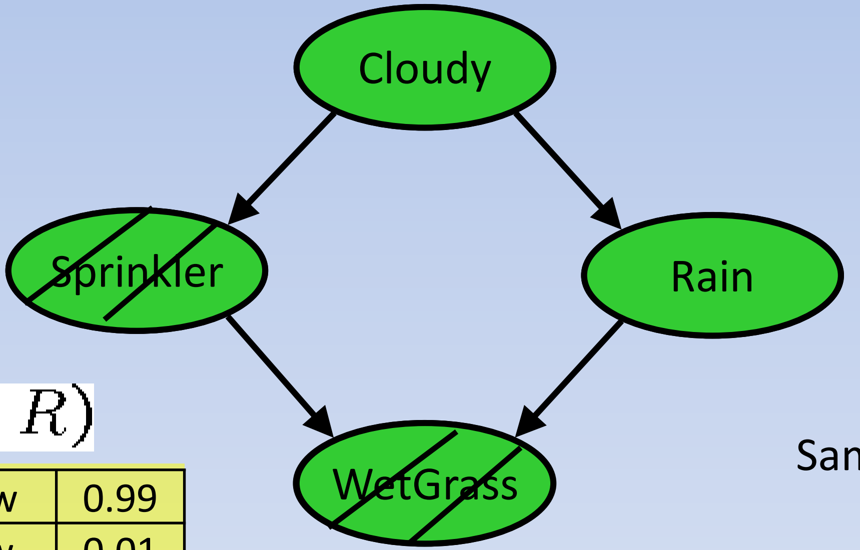
Likelihood Weighting

$P(C)$

+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:

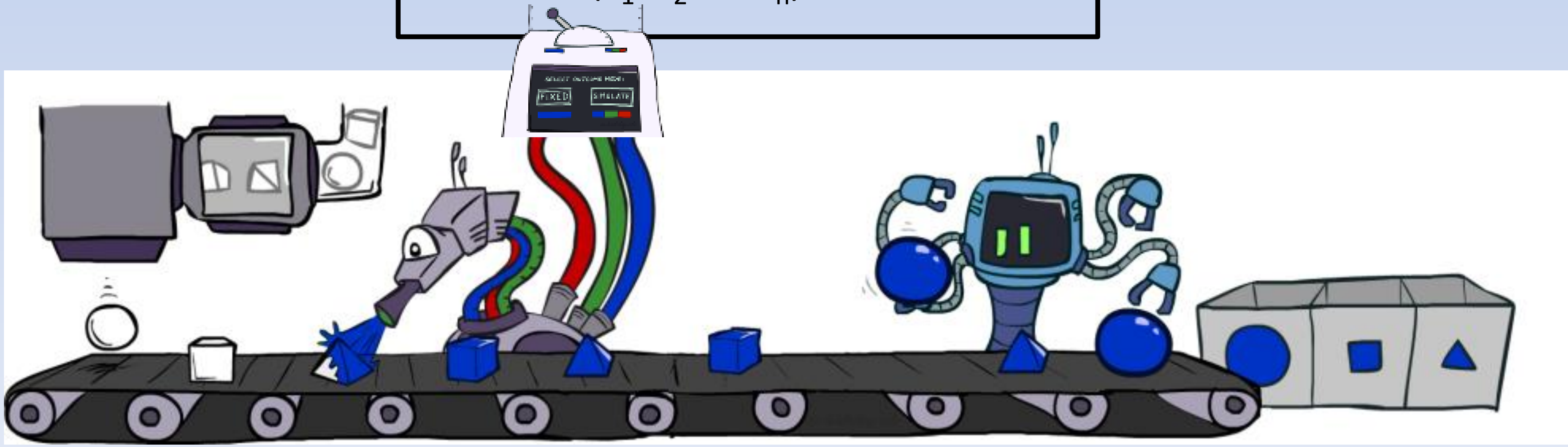
+c, +s, +r, +w

...

$$w = 1.0 \times 0.1 \times 0.99$$

Likelihood Weighting

- IN: evidence instantiation
- w = 1.0
- for i=1, 2, …, n
  - if X_i is an evidence variable
    - X_i = observation x_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



```
function LIKELIHOOD-WEIGHTING(X, e, bn, N) returns an estimate of P(X|e)
  inputs: X, the query variable
          e, observed values for variables E
          bn, a Bayesian network specifying joint distribution P(X_1, ..., X_n)
          N, the total number of samples to be generated
  local variables: W, a vector of weighted counts for each value of X, initially zero

  for j = 1 to N do
     x, w ← WEIGHTED-SAMPLE(bn, e)
     W[x] ← W[x] + w where x is the value of X in x
  return NORMALIZE(W)
```

```
function WEIGHTED-SAMPLE(bn, e) returns an event and a weight

  w ← 1; x ← an event with n elements initialized from e
  foreach variable X_i in X_1, ..., X_n do
     if X_i is an evidence variable with value x_i in e
        then w ← w × P(X_i = x_i | parents(X_i))
        else x[i] ← a random sample from P(X_i | parents(X_i))
  return x, w
```

Figure 14.15 The likelihood-weighting algorithm for inference in Bayesian networks. In WEIGHTED-SAMPLE, each nonevidence variable is sampled according to the conditional distribution given the values already sampled for the variable's parents, while a weight is accumulated based on the likelihood for each evidence variable.

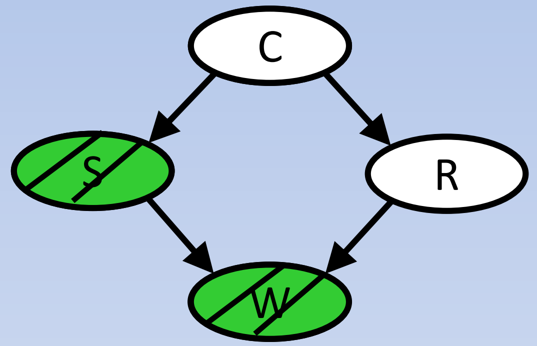
Likelihood Weighting

- Sampling distribution if z sampled and e fixed evidence

$$S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i | \text{Parents}(Z_i))$$

- Now, samples have weights

$$w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{m} P(e_i | \text{Parents}(E_i))$$



- Together, weighted sampling distribution is consistent

$$S_{\text{WS}}(z, e) \cdot w(z, e) = \prod_{i=1}^{l} P(z_i | \text{Parents}(z_i)) \prod_{i=1}^{m} P(e_i | \text{Parents}(e_i))$$

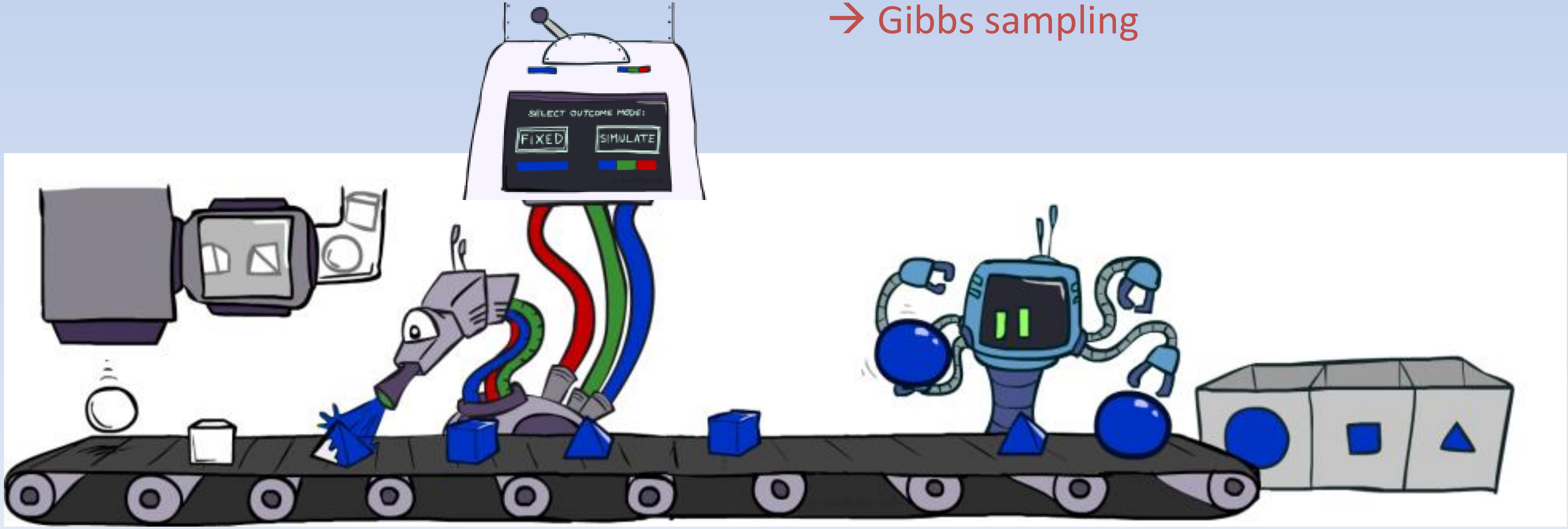
$$= P(\mathbf{z}, \mathbf{e})$$

Likelihood Weighting

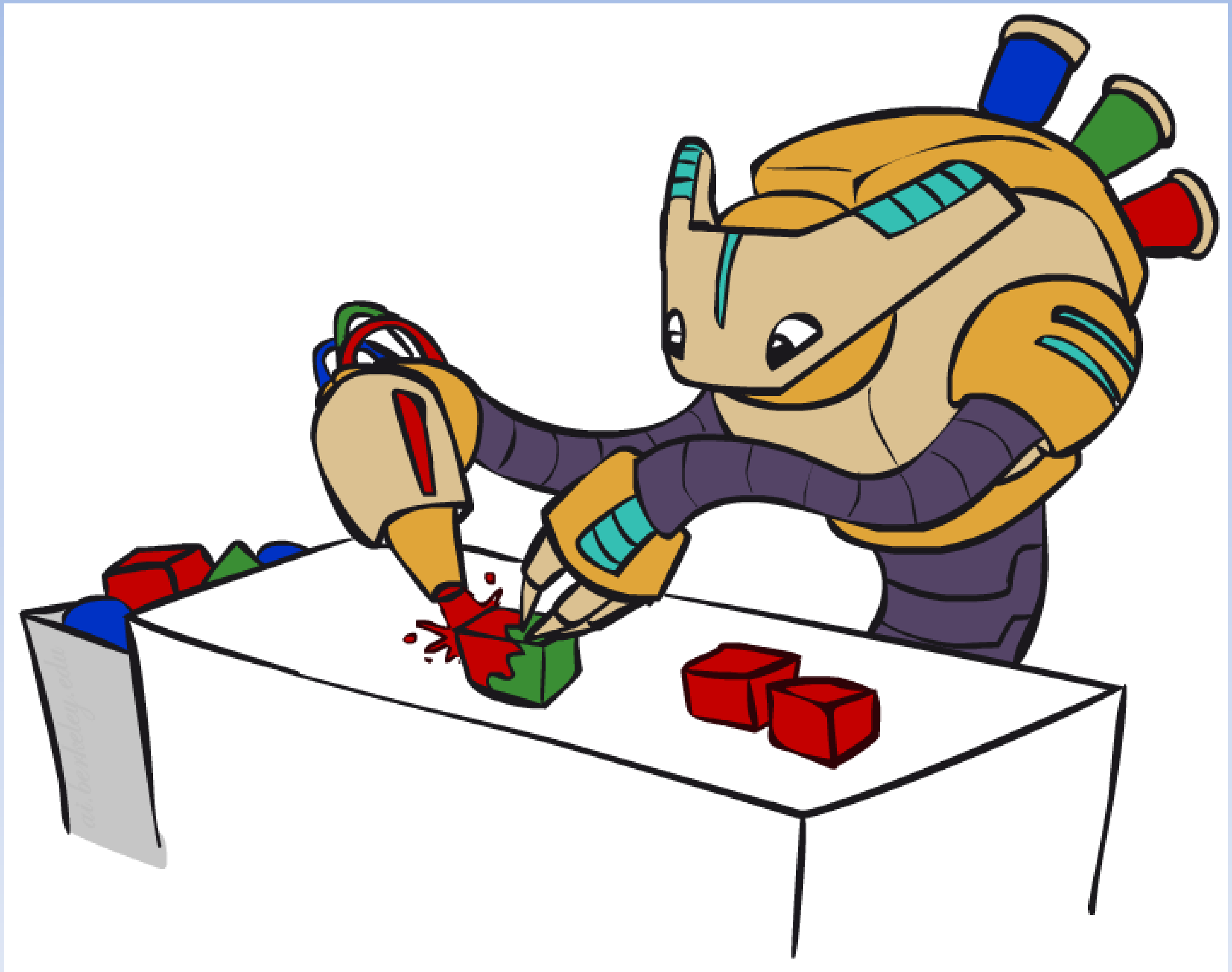
- Likelihood weighting is good
  - We have taken evidence into account as we generate the sample
  - E.g. here, W’s value will get picked based on the evidence values of S, R
  - More of our samples will reflect the state of the world suggested by the evidence

- Likelihood weighting doesn’t solve all our problems
  - Evidence influences the choice of downstream variables, but not upstream ones (C isn’t more likely to get a value matching the evidence)
- We would like to consider evidence when we sample every variable

→ Gibbs sampling



Gibbs Sampling

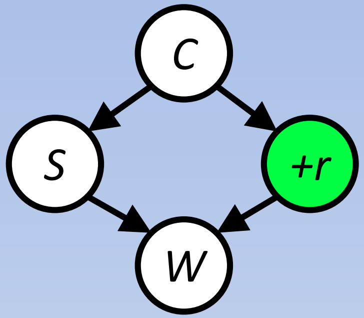


Gibbs Sampling

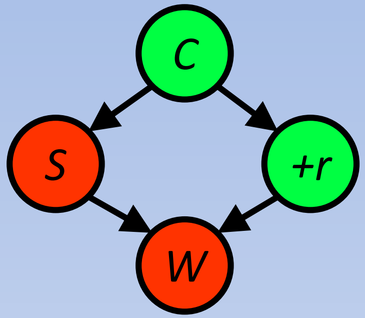
- *Procedure:* keep track of a full instantiation $x_1, x_2, \ldots, x_n$.
  - Start with an arbitrary instantiation consistent with the evidence.
  - Sample one variable at a time, conditioned on all the rest, but keep evidence fixed.
  - Keep repeating this for a long time.
- *Property:* in the limit of repeating this infinitely many times the resulting sample is coming from the correct distribution
- *Rationale*: both upstream and downstream variables condition on evidence.
- **In contrast:**
  - likelihood weighting only conditions on upstream evidence,
  - 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 want high weight.

Gibbs Sampling Example: P(S | +r)

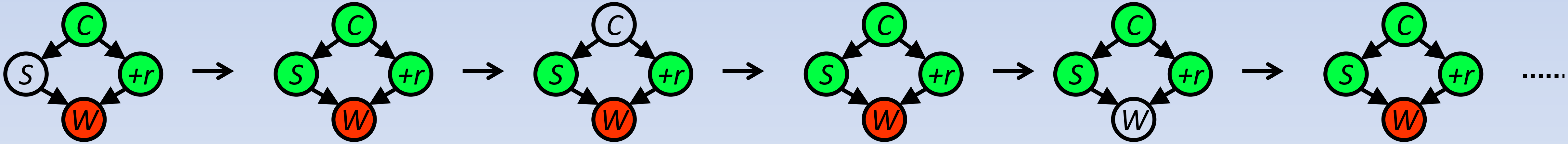
- Step 1: Fix evidence
  - R = +r



- Step 2: Initialize other variables
  - Randomly



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



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

function GIBBS-ASK(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  local variables: **N**, a vector of counts for each value of X, initially zero
    Z, the nonevidence variables in bn
    x, the current state of the network, initially copied from **e**

  initialize **x** with random values for the variables in **Z**
  for j = 1 to N **do**
    for each Z_i in **Z do**
      set the value of Z_i in **x** by sampling from $\mathbf{P}(Z_i|mb(Z_i))$
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in **x**
  return NORMALIZE(**N**)

Figure 14.16 The Gibbs sampling algorithm for approximate inference in Bayesian networks; this version cycles through the variables, but choosing variables at random also works.

Gibbs Sampling

- How is this better than sampling from the full joint?
  - In a Bayes' Net, sampling a variable given all the other variables (e.g. P(R|S,C,W)) is usually much easier than sampling from the full joint distribution
    - Only requires a join on the variable to be sampled (in this case, a join on R)
    - The resulting factor only depends on the variable's parents, its children, and its children's parents (this is often referred to as its Markov blanket)

Markov Blanket

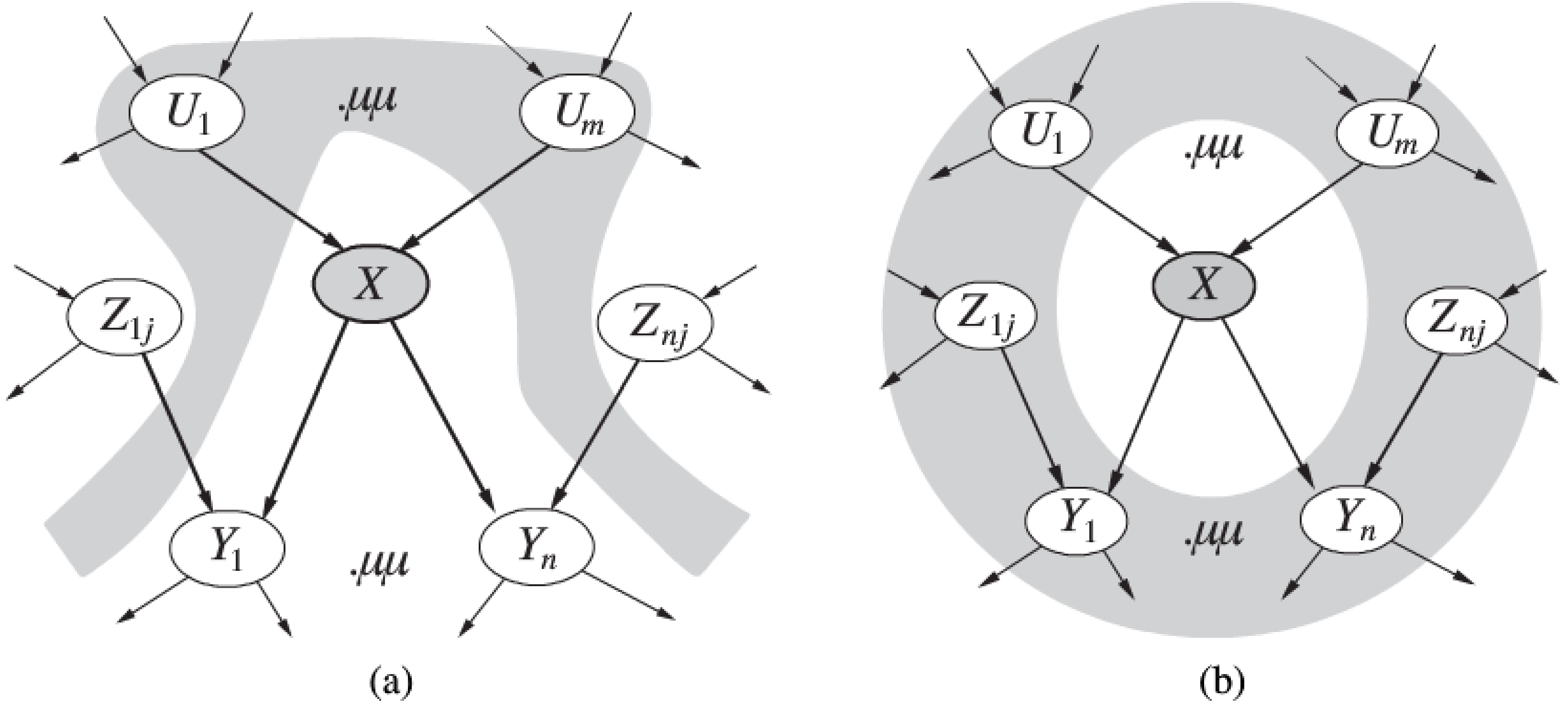
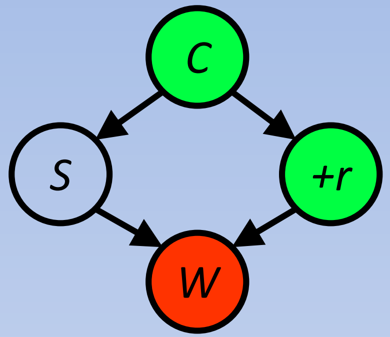


Figure 14.4 (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

Efficient Resampling of One Variable

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

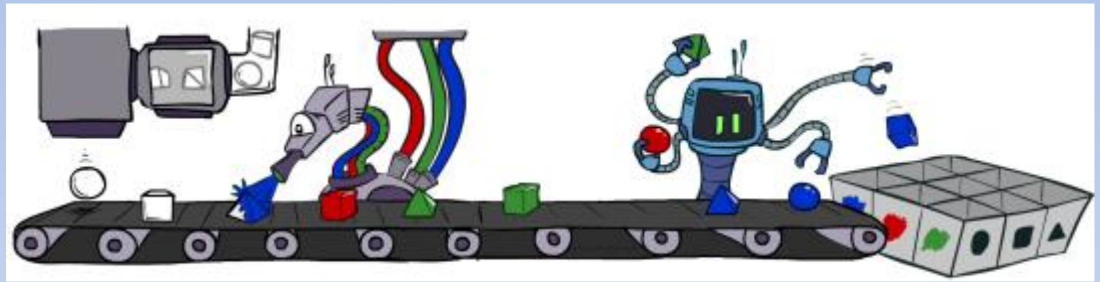
$$
\begin{aligned}
P(S|+c,+r,-w) &= \frac{P(S,+c,+r,-w)}{P(+c,+r,-w)} \\
&= \frac{P(S,+c,+r,-w)}{\sum_s P(s,+c,+r,-w)} \\
&= \frac{P(+c)P(S|+c)P(+r|+c)P(-w|S,+r)}{\sum_s P(+c)P(s|+c)P(+r|+c)P(-w|s,+r)} \\
&= \frac{P(+c)P(S|+c)P(+r|+c)P(-w|S,+r)}{P(+c)P(+r|+c)\sum_s P(s|+c)P(-w|s,+r)} \\
&= \frac{P(S|+c)P(-w|S,+r)}{\sum_s P(s|+c)P(-w|s,+r)}
\end{aligned}
$$



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

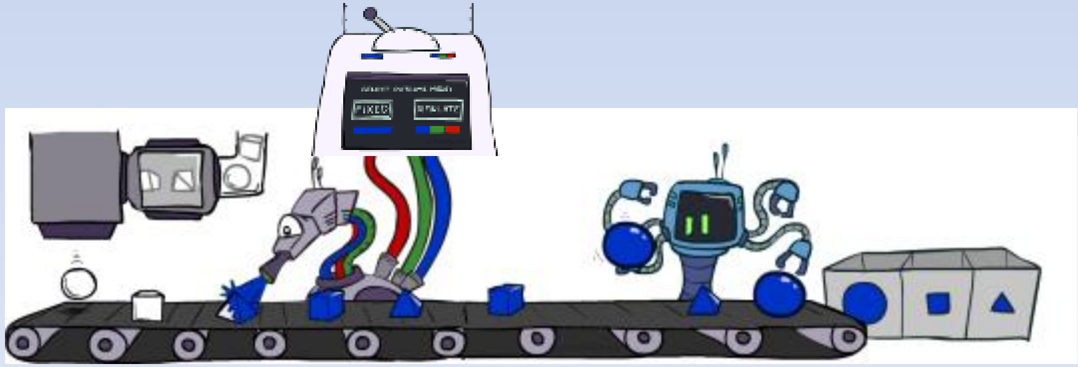
Bayes' Net Sampling Summary

- Prior Sampling P



- Rejection Sampling P(Q | e)

- Likelihood Weighting P(Q | e)



- Gibbs Sampling P(Q | e)

Further Reading on Gibbs Sampling*

- Gibbs sampling produces sample from the query distribution P(Q | e) in limit of re-sampling infinitely often

- Gibbs sampling is a special case of more general methods called Markov chain Monte Carlo (MCMC) methods

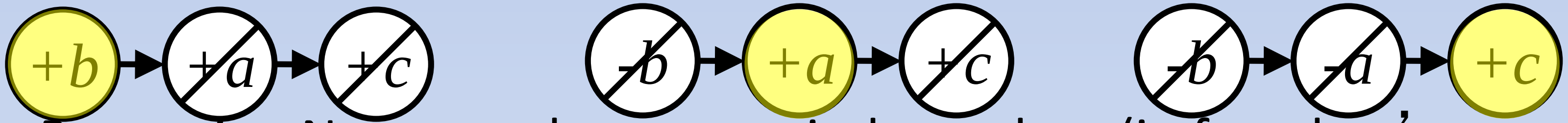
  – Metropolis-Hastings is one of the more famous MCMC methods (in fact, Gibbs sampling is a special case of Metropolis-Hastings)

- You may read about Monte Carlo methods – they’re just sampling

Markov Chain Monte Carlo*

- *Idea*: instead of sampling from scratch, create samples that are each like the last one.

- *Procedure*: resample one variable at a time, conditioned on all the rest, but keep evidence fixed. E.g., for P(b|c):



- *Properties*: Now samples are not independent (in fact they're nearly identical), but sample averages are still consistent estimators!

- *What's the point*: both upstream and downstream variables condition on evidence.