Inference in Bayesian Networks

CE417: Introduction to Artificial Intelligence
Sharif University of Technology
Fall 2023

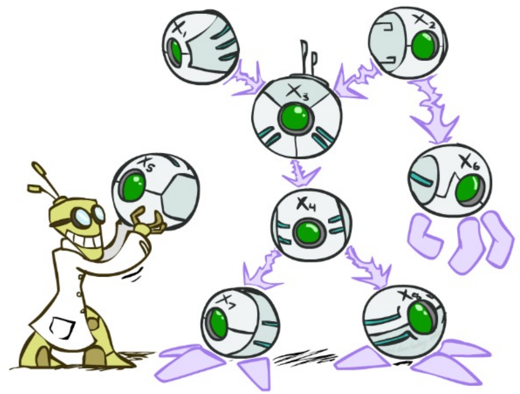
Soleymani

Slides are based on Klein and Abdeel, CS188, UC Berkeley.

Recap: Bayes' Net Representation

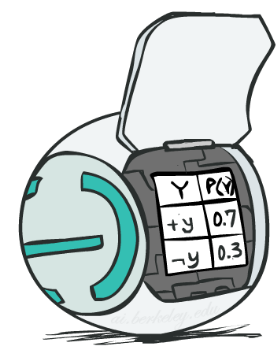
- A directed, acyclic graph, one node per random variable
- A conditional probability table (CPT) for each node
  - A collection of distributions over X, one for each combination parents' values

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



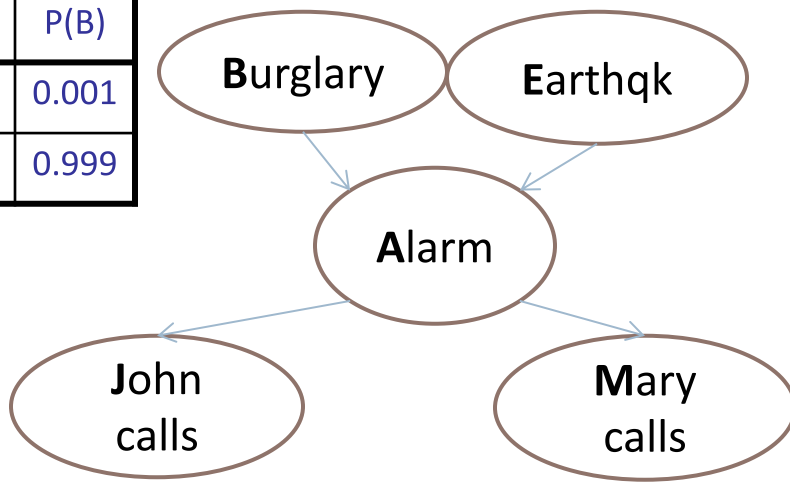
- Bayes' nets implicitly encode joint distributions
  - As a product of local conditional distributions
  - To see what probability a BN gives to a full assignment, multiply all the relevant conditionals together:

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

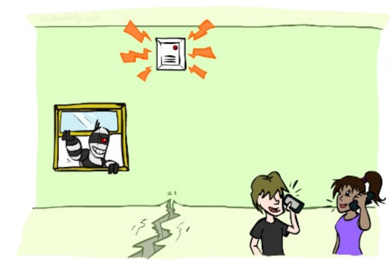


Example: Alarm Network

B	P(B)
+b	0.001
-b	0.999



E	P(E)
+e	0.002
-e	0.998



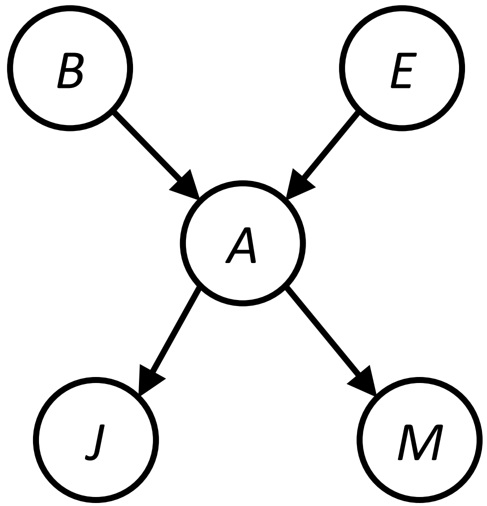
A	J	P(J|A)
+a	+j	0.9
+a	-j	0.1
-a	+j	0.05
-a	-j	0.95

A	M	P(M|A)
+a	+m	0.7
+a	-m	0.3
-a	+m	0.01
-a	-m	0.99

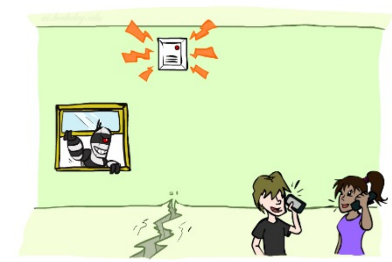
B	E	A	P(A|B,E)
+b	+e	+a	0.95
+b	+e	-a	0.05
+b	-e	+a	0.94
+b	-e	-a	0.06
-b	+e	+a	0.29
-b	+e	-a	0.71
-b	-e	+a	0.001
-b	-e	-a	0.999

Example: Alarm Network

B	P(B)
+b	0.001
-b	0.999



E	P(E)
+e	0.002
-e	0.998



A	J	P(J|A)
+a	+j	0.9
+a	-j	0.1
-a	+j	0.05
-a	-j	0.95

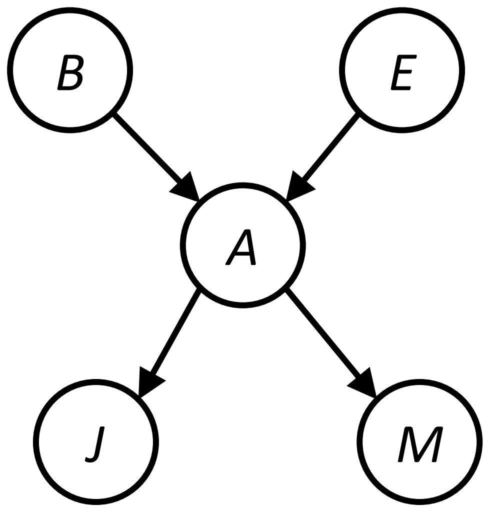
A	M	P(M|A)
+a	+m	0.7
+a	-m	0.3
-a	+m	0.01
-a	-m	0.99

B	E	A	P(A|B,E)
+b	+e	+a	0.95
+b	+e	-a	0.05
+b	-e	+a	0.94
+b	-e	-a	0.06
-b	+e	+a	0.29
-b	+e	-a	0.71
-b	-e	+a	0.001
-b	-e	-a	0.999

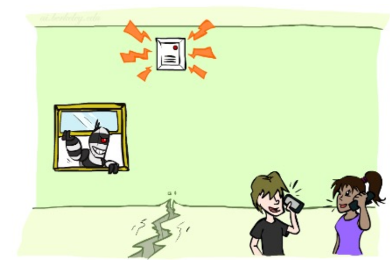
$$P(+b, -e, +a, -j, +m) =$$
$$P(+b)P(-e)P(+a|+b,-e)P(-j|+a)P(+m|+a) =$$

Example: Alarm Network

B	P(B)
+b	0.001
-b	0.999



E	P(E)
+e	0.002
-e	0.998



A	J	P(J|A)
+a	+j	0.9
+a	-j	0.1
-a	+j	0.05
-a	-j	0.95

A	M	P(M|A)
+a	+m	0.7
+a	-m	0.3
-a	+m	0.01
-a	-m	0.99

B	E	A	P(A|B,E)
+b	+e	+a	0.95
+b	+e	-a	0.05
+b	-e	+a	0.94
+b	-e	-a	0.06
-b	+e	+a	0.29
-b	+e	-a	0.71
-b	-e	+a	0.001
-b	-e	-a	0.999

$$P(+b, -e, +a, -j, +m) =$$
$$P(+b)P(-e)P(+a|+b, -e)P(-j|+a)P(+m|+a) =$$
$$0.001 \times 0.998 \times 0.94 \times 0.1 \times 0.7$$

Video of Demo BN Applet

Bayes' Nets

- ✓ Representation
- ✓ Conditional Independences
- Probabilistic Inference
  - Enumeration (exact, exponential complexity)
  - Variable elimination (exact, worst-case exponential complexity, often better)
  - Inference is NP-complete
  - Sampling (approximate)
- Learning Bayes' Nets from Data

Inference

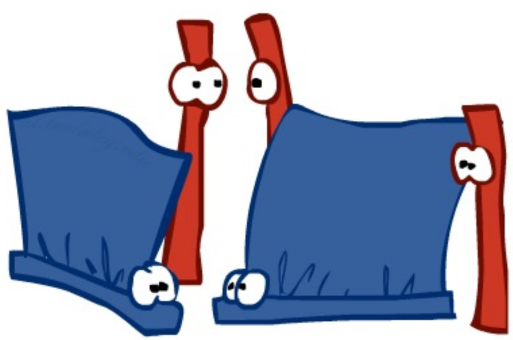
- Inference: calculating some useful quantity from a joint probability distribution

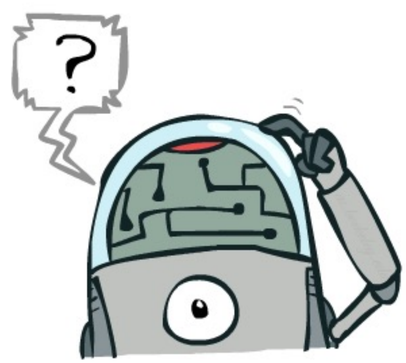
- Examples:
  - Posterior probability

    $$P(Q|E_1 = e_1, \ldots E_k = e_k)$$

  - Most likely explanation:

    $$\text{argmax}_q \; P(Q = q|E_1 = e_1 \ldots)$$





Inference by Enumeration

- General case:
  - Evidence variables: $E_1 \ldots E_k = e_1 \ldots e_k$
  - Query* variable: Q
  - Hidden variables: $H_1 \ldots H_r$

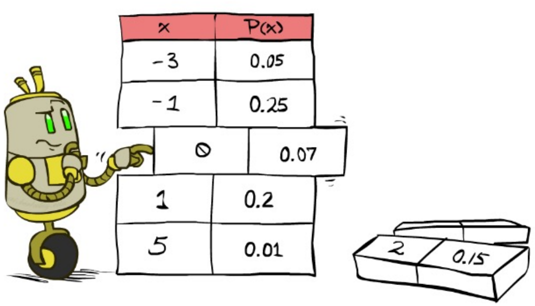
  } $X_1, X_2, \ldots X_n$ *All variables*

- We want:

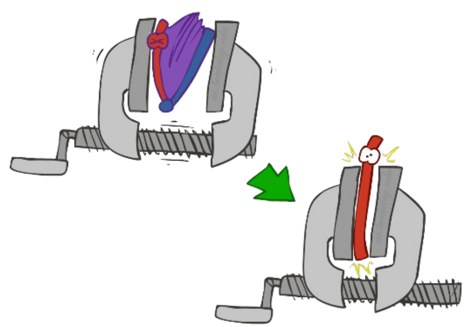
** Works fine with multiple query variables, too*

$$P(Q|e_1 \ldots e_k)$$

- Step 1: Select the entries consistent with the evidence



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



$$P(Q, e_1 \ldots e_k) = \sum_{h_1 \ldots h_r} \underbrace{P(Q, h_1 \ldots h_r, e_1 \ldots e_k)}_{X_1, X_2, \ldots X_n}$$

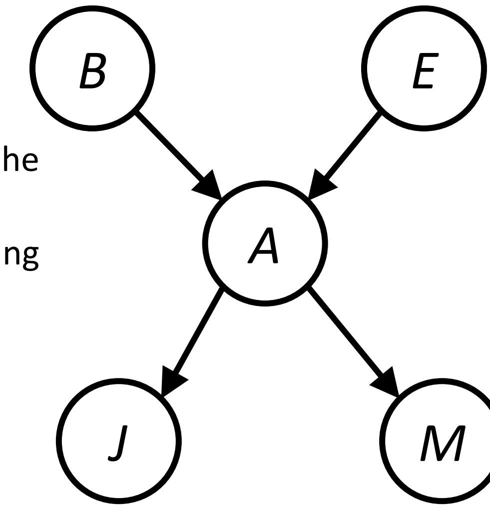
- Step 3: Normalize

$$\times \frac{1}{Z}$$

$$Z = \sum_q P(Q, e_1 \cdots e_k)$$

$$P(Q|e_1 \cdots e_k) = \frac{1}{Z} P(Q, e_1 \cdots e_k)$$

Inference by Enumeration in Bayes' Net



- Given unlimited time, inference in BNs is easy
- Reminder of inference by enumeration:
  - Any probability of interest can be computed by summing entries from the joint distribution: $P(\boldsymbol{Q} \mid \boldsymbol{e}) = \alpha \sum_h P(\boldsymbol{Q}, \boldsymbol{h}, \boldsymbol{e})$
  - Entries from the joint distribution can be obtained from a BN by multiplying the corresponding conditional probabilities

$$P(B \mid +j, +m) \propto_B P(B, +j, +m)$$

$$= \sum_{e,a} P(B, e, a, +j, +m)$$

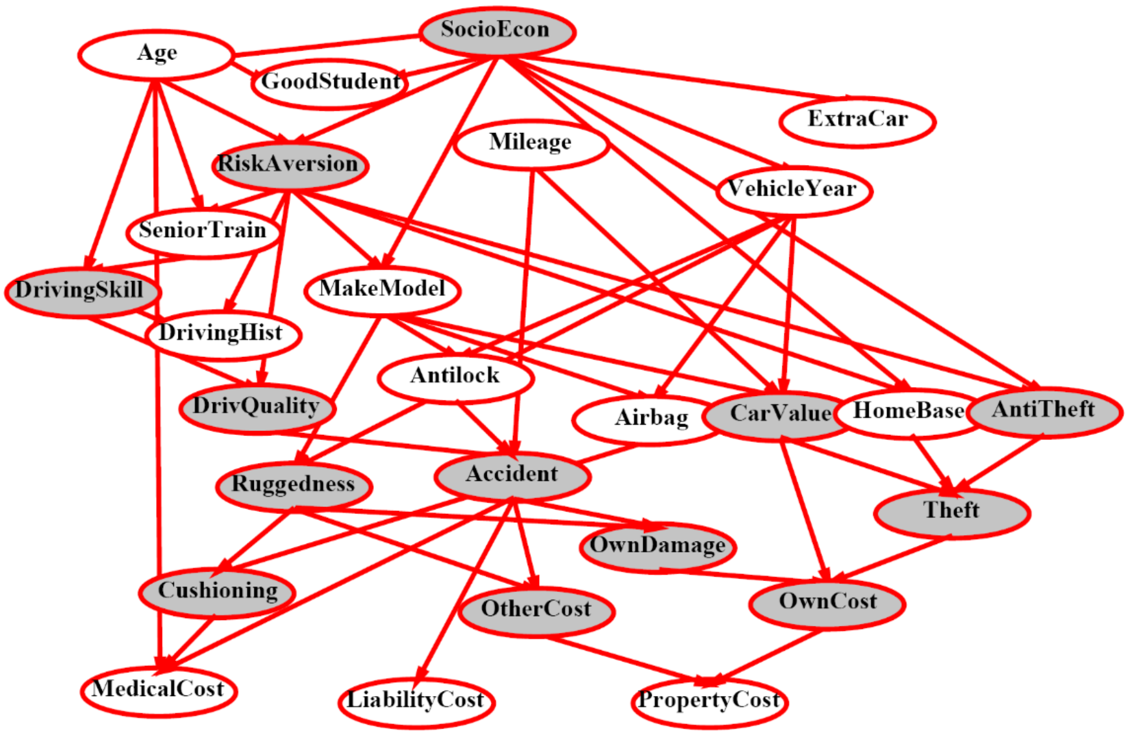
$$= \sum_{e,a} P(B)P(e)P(a|B,e)P(+j|a)P(+m|a)$$

$$= P(B)P(+e)P(+a|B,+e)P(+j|+a)P(+m|+a) + P(B)P(+e)P(-a|B,+e)P(+j|-a)P(+m|-a)$$

$$P(B)P(-e)P(+a|B,-e)P(+j|+a)P(+m|+a) + P(B)P(-e)P(-a|B,-e)P(+j|-a)P(+m|-a)$$

- So inference in Bayes nets means computing sums of products of numbers: sounds easy!!
- Problem: sums of exponentially many products!

Inference by Enumeration?



$$P(Antilock|observed\ variables) = ?$$

Distribution of Products on Sums

- Exploiting the factorization properties to allow sums and products to be interchanged
  - $a \times b + a \times c$ needs three operations while $a \times (b + c)$ requires two
  - $a \times (b_1 + \cdots + b_n)$
- `

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

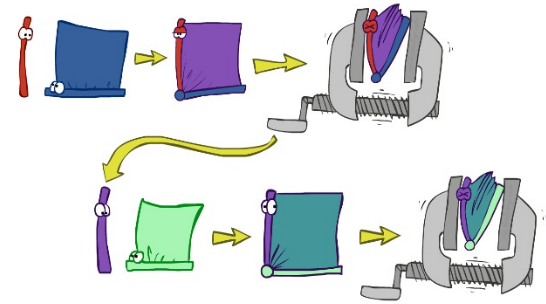
$\sum_{e,a} P(B)\ P(e)\ P(a|B,e)\ P(j|a)\ P(m|a)$

$= P(B)P(e)P(a|B,e)P(j|a)P(m|a) + P(B)P(\neg e)P(a|B,\neg e)P(j|a)P(m|a)$

$+ P(B)P(e)P(\neg a|B,e)P(j|\neg a)P(m|\neg a) + P(B)P(\neg e)P(\neg a|B,\neg e)P(j|\neg a)P(m|\neg a)$

Lots of repeated subexpressions!

Variable Elimination: The basic ideas



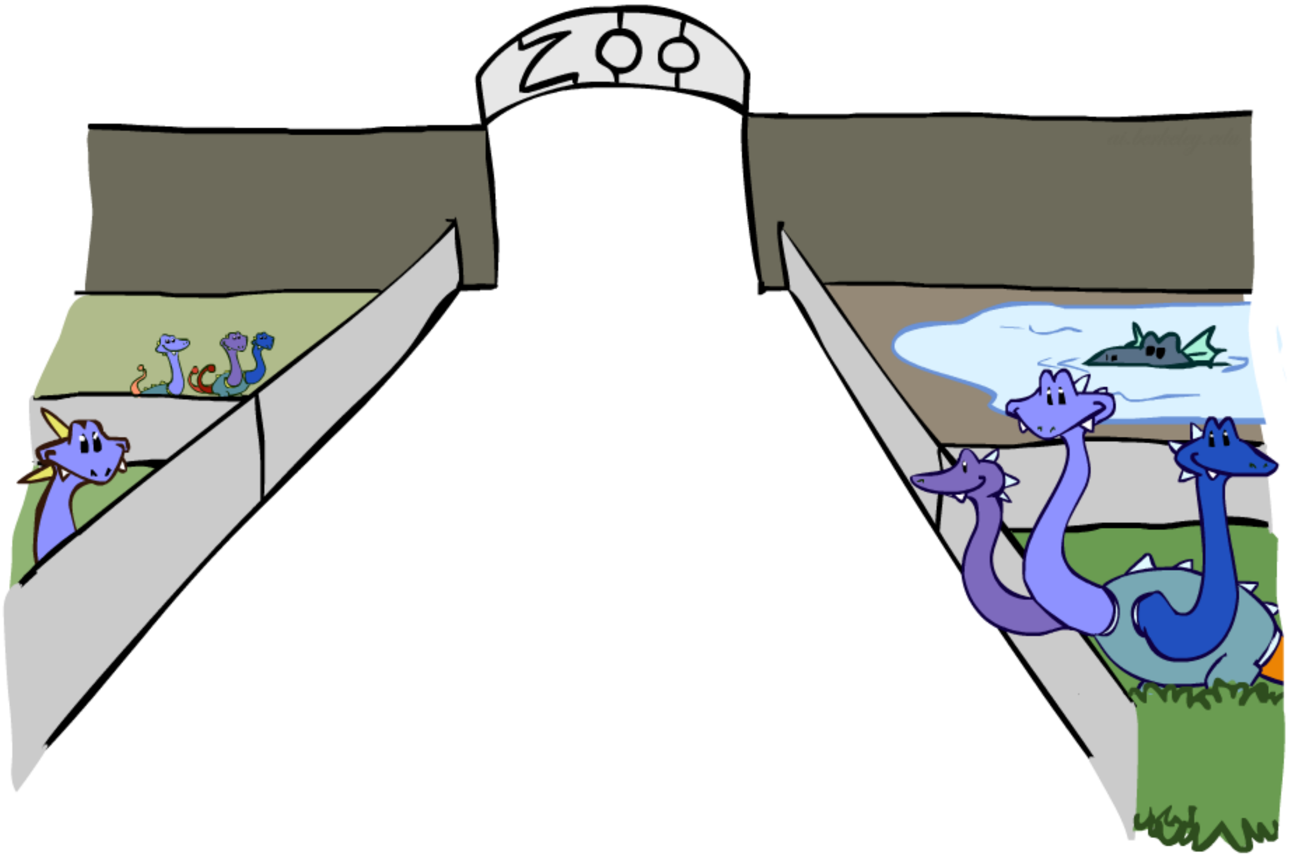
- Move summations inwards as far as possible

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

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

- 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|x|Y| matrix
  - Sums to 1

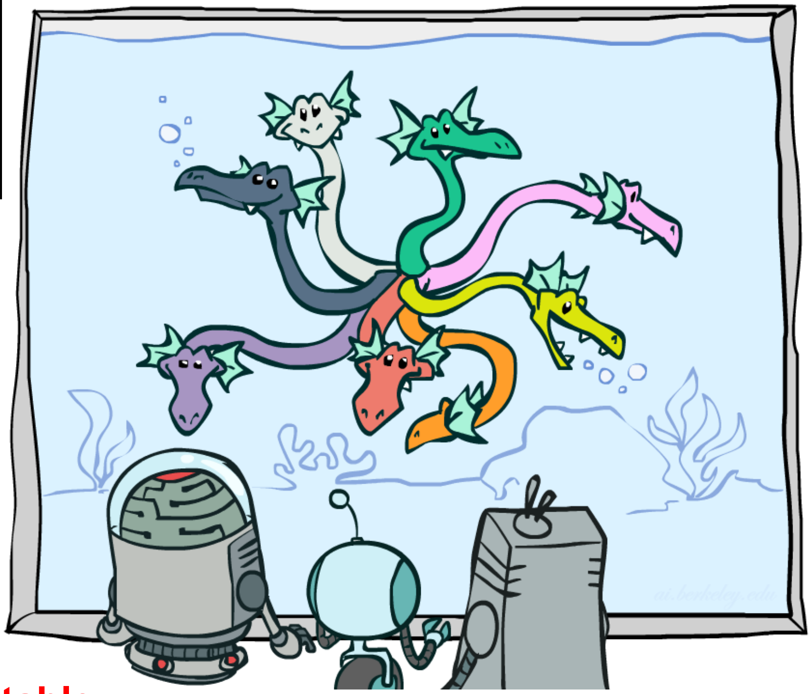
$P(A,J)$

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)

$P(a,J) = P_a(J)$

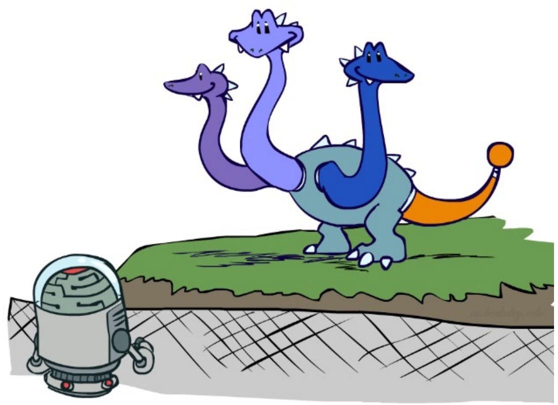
A \ J	true	false
true	0.09	0.01



Number of variables (capitals) = dimensionality of the table

Factor Zoo II

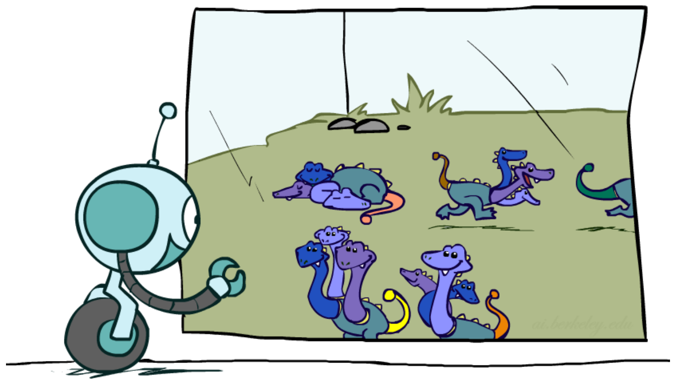
- Single conditional: P(Y | x)
  - Entries P(y | x) for fixed x, all y
  - Sums to 1



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)

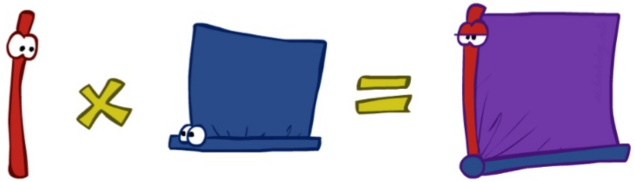
A \ J	true	false
true	0.9	0.1
false	0.05	0.95

} - *P(J|a)*

} - *P(J|¬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*(*J*|*A*) x *P*(*A*) = *P*(*A*,*J*)

P(*A*)

true	0.1
false	0.9

x

P(*J*|*A*)

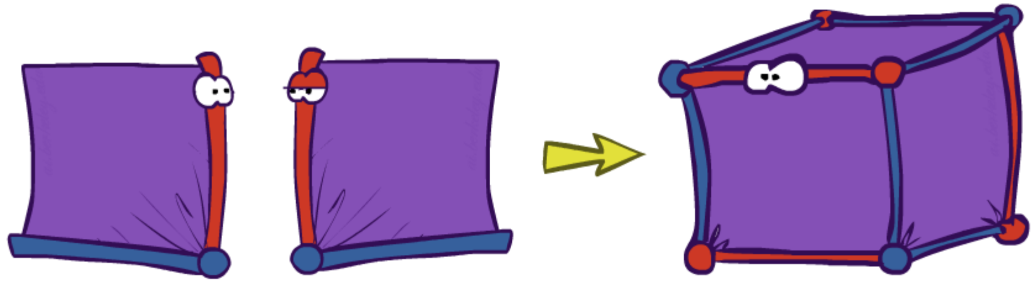
A \ J	true	false
true	0.9	0.1
false	0.05	0.95

=

P(*A*,*J*)

A \ J	true	false
true	0.09	0.01
false	0.045	0.855

Example: Making larger factors



- Example: *P(A,J)* x *P(A,M)* = *P(A,J,M)*

P(A,J)

A \ J	true	false
true	0.09	0.01
false	0.045	0.855

x

P(A,M)

A \ M	true	false
true	0.07	0.03
false	0.009	0.891

=

P(A,J,M)

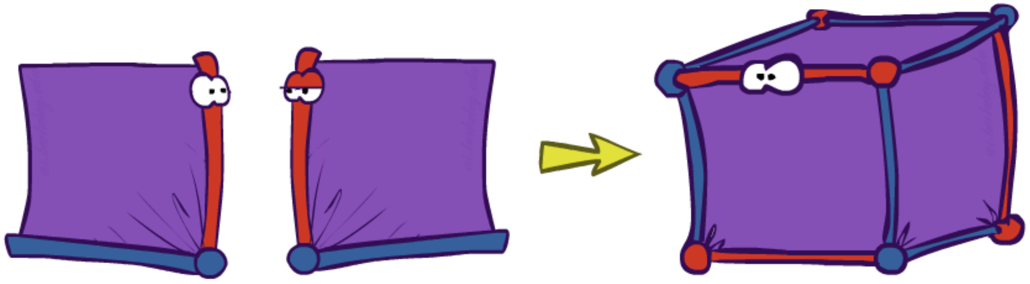
A=false

J \ M	true	false
		.7618

A=true

J \ M	true	false
true		
false		.0003

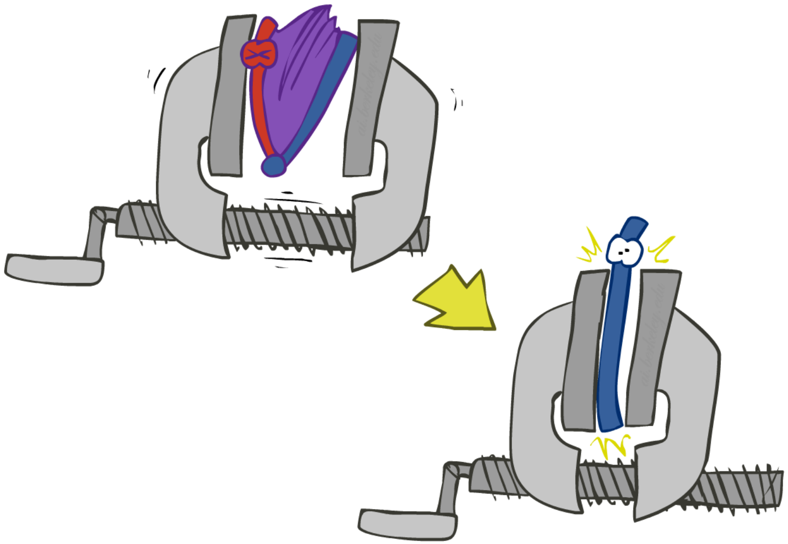
Example: Making larger factors



- Example: *P*(*U*,*V*) x *P*(*V*,*W*) x *P*(*W*,*X*) = *P*(*U*,*V*,*W*,*X*)
- Sizes: [10,10] x [10,10] x [10,10] = [10,10,10,10]
- I.e., 300 numbers blows up to 10,000 numbers!
- Factor blowup can make VE 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 →

$P(A)$

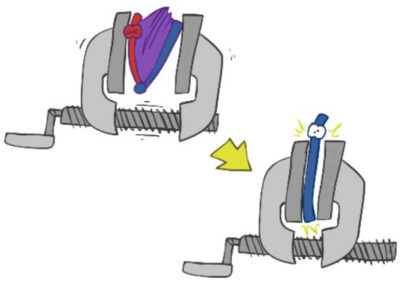
true	0.1
false	0.9

Summing out from a product of factors

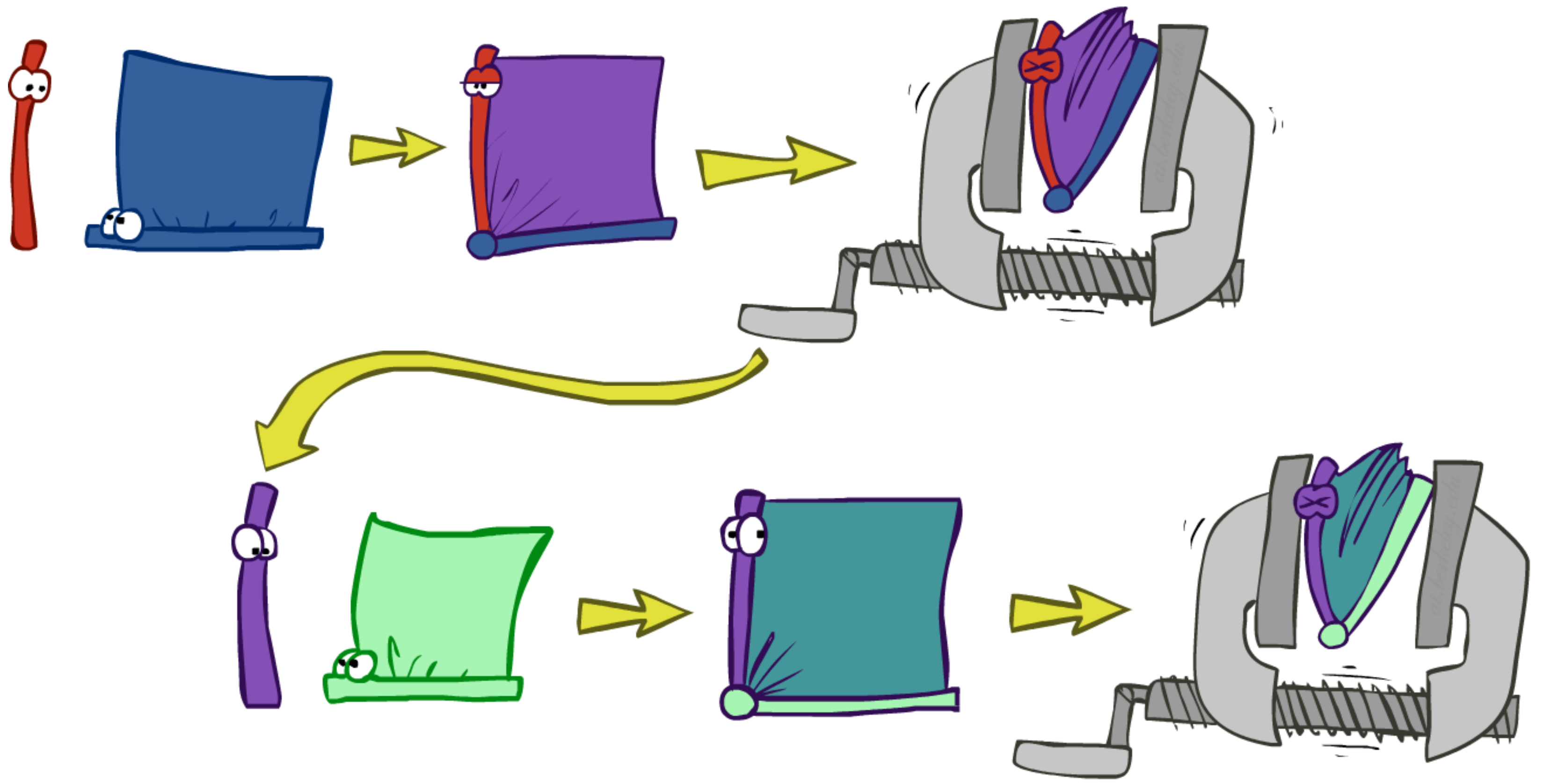
- Project the factors each way first, then sum the products
- Example: $\sum_a P(a|B,e) \times P(j|a) \times P(m|a)$

  $= P(a|B,e) \times P(j|a) \times P(m|a) +$

  $P(\neg a|B,e) \times P(j|\neg a) \times P(m|\neg a)$



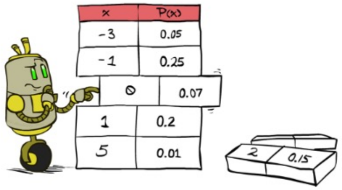
Variable Elimination



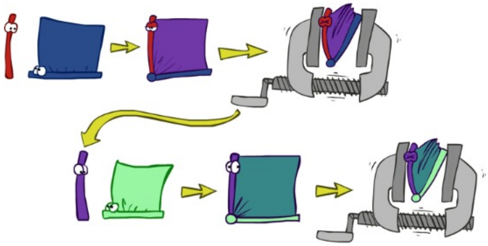
Variable Elimination

- Query: $P(Q|E_1=e_1,.., E_k=e_k)$

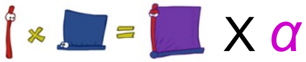
- Start with initial factors:
  - Local CPTs (but instantiated by evidence)



- 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

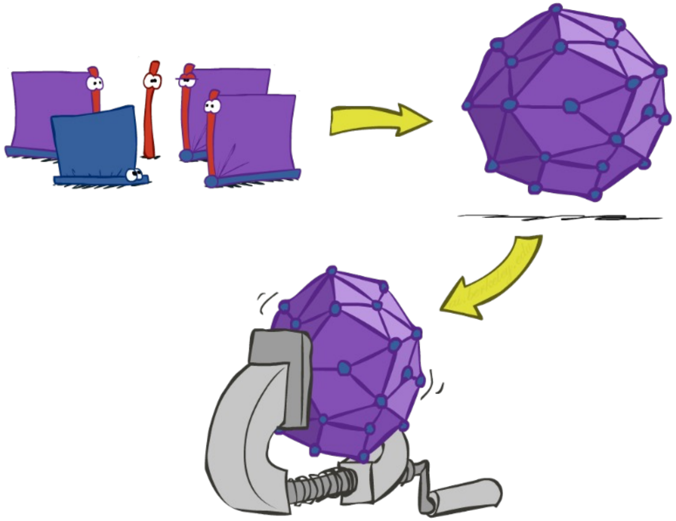


- Join all remaining factors and normalize

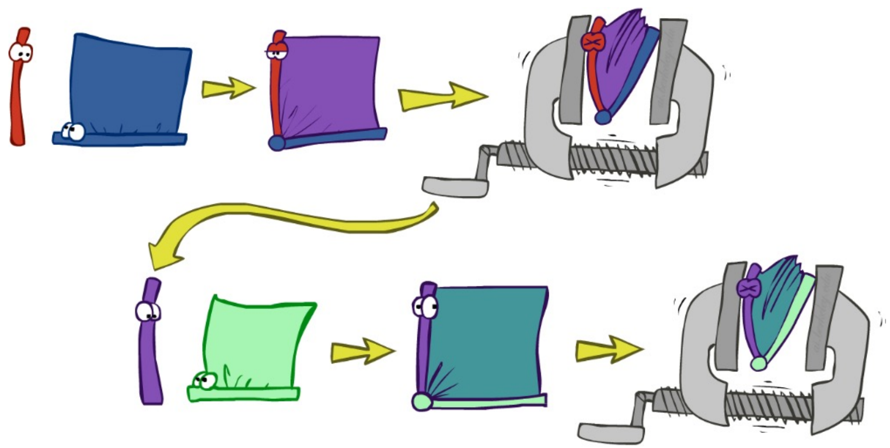


Inference by Enumeration vs. Variable Elimination

- Why is inference by enumeration so slow?
  - You join up the whole joint distribution before you sum out the hidden variables

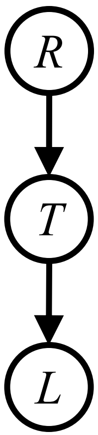


- Idea: interleave joining and marginalizing!
  - Called "Variable Elimination"
  - Still NP-hard, but usually much faster than inference by enumeration



First we'll need some new notation: factors

Traffic Domain



$$P(L) = ?$$

- Inference by Enumeration

$$= \sum_t \sum_r P(L|t)P(r)P(t|r)$$

Join on r

Join on t

Eliminate r

Eliminate t

- Variable Elimination

$$= \sum_t P(L|t) \sum_r P(r)P(t|r)$$

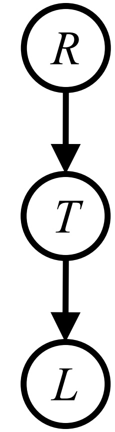
Join on r

Eliminate r

Join on t

Eliminate t

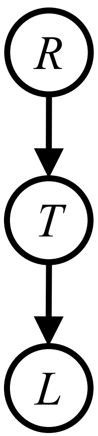
$P(L) = ?$



Marginalizing Early! (aka VE)

$P(R)$

+r	0.1
-r	0.9



$P(T|R)$

+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

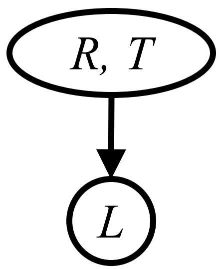
$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

Join R

$P(R, T)$

+r	+t	0.08
+r	-t	0.02
-r	+t	0.09
-r	-t	0.81



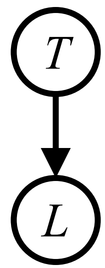
$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

Sum out R

$P(T)$

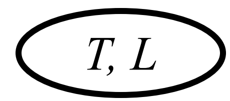
+t	0.17
-t	0.83



$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

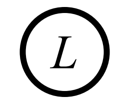
Join T



$P(T, L)$

+t	+l	0.051
+t	-l	0.119
-t	+l	0.083
-t	-l	0.747

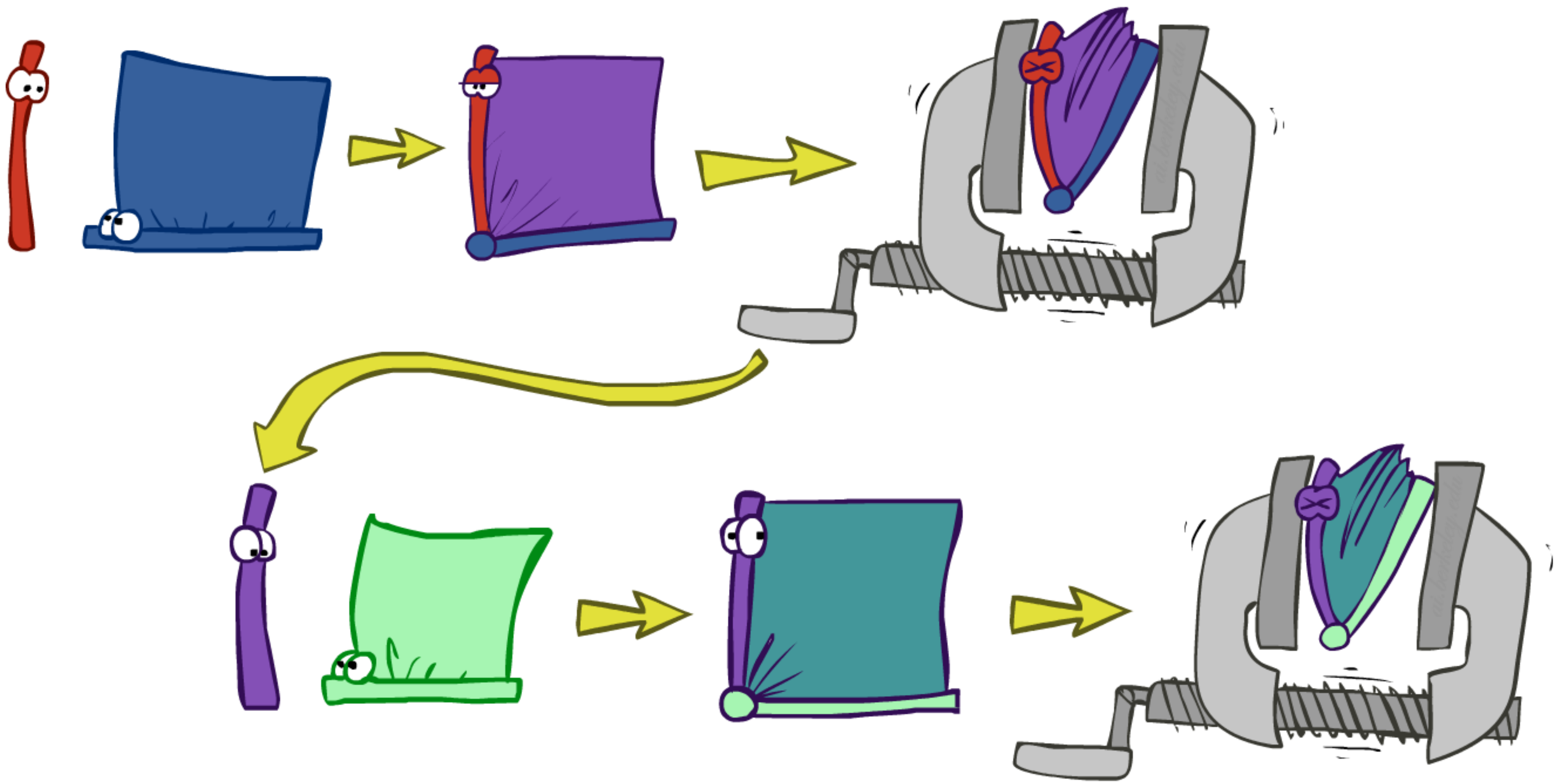
Sum out T



$P(L)$

+l	0.134
-l	0.866

Marginalizing Early (= Variable Elimination)



Improvement Reasons

- Computing an expression of the form (sum-product inference):

$$\sum_{\boldsymbol{H}} \prod_{\phi \in \boldsymbol{\Phi}} \phi$$

$\boldsymbol{\Phi}$: the set of factors

- We used the structure of BN to factorize the joint distribution and thus the scope of the resulted factors will be limited.

- Distributive law: If $h \notin \text{Scope}(\phi_1)$ then $\sum_h \phi_1 . \phi_2 = \phi_1 . \sum_h \phi_2$
  - Performing the summations over the product of only a subset of factors

- We find sub-expressions that can be computed once and then we save and reuse them in later computations
  - Instead of computing them exponentially many times

Variable Elimination Algorithm

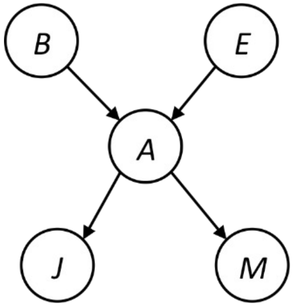
- Given: BN, evidence e, a query $P(\boldsymbol{Q}|\boldsymbol{x_e})$
- Choose an **ordering** on variables, e.g., X_1, …, X_n
- For i = 1 to n, If $X_i \notin \{\boldsymbol{Q}, \boldsymbol{X_e}\}$
  - Collect factors $\boldsymbol{f}_1, \dots, \boldsymbol{f}_k$ that include X_i
  - Generate a new factor by eliminating X_i from these factors:

$$\boldsymbol{g} = \sum_{X_i} \prod_{j=1}^{k} \boldsymbol{f}_j$$

- Multiply all remaining factors
- Normalize $P(\boldsymbol{Q}, \boldsymbol{x_e})$ to obtain $P(\boldsymbol{Q}|\boldsymbol{x_e})$

After this summation, X_i is eliminated

Example



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

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

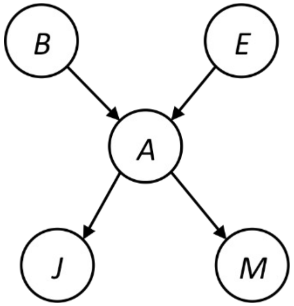
Choose A

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

$\times$ ⇒ $P(j,m,A|B,E)$ $\sum$ ⇒ $P(j,m|B,E)$

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

Example (Cont.)



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

Choose E

$P(E)$

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

$\times \Rightarrow P(j,m,E \mid B) \quad \sum \Rightarrow P(j,m \mid B)$

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

Finish with B

$P(B)$

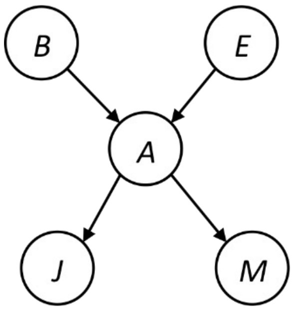
$P(j,m \mid B)$

$\times \Rightarrow P(j,m,B)$ Normalize $\Rightarrow P(B \mid j,m)$

Same Example in Equations

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

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



$$
\begin{aligned}
P(B|j,m) &\propto P(B,j,m) \\
&= \sum_{e,a} P(B,j,m,e,a) \\
&= \sum_{e,a} P(B)P(e)P(a|B,e)P(j|a)P(m|a) \\
&= \sum_{e} P(B)P(e)\sum_{a} P(a|B,e)P(j|a)P(m|a) \\
&= \sum_{e} P(B)P(e)f_1(B,e,j,m) \\
&= P(B)\sum_{e} P(e)f_1(B,e,j,m) \\
&= P(B)f_2(B,j,m)
\end{aligned}
$$

marginal can be obtained from joint by summing out

use Bayes' net joint distribution expression

use x*(y+z) = xy + xz

joining on a, and then summing out gives f_1

use x*(y+z) = xy + xz

joining on e, and then summing out gives f_2

All we are doing is exploiting uwy + uwz + uxy + uxz + vwy + vwz + vxy +vxz = (u+v)(w+x)(y+z) to improve computational efficiency!

Variable Elimination Algorithm

- Sum out each variable one at a time
  - all factors containing that variable are (removed from the set of factors and) multiplied to generate a product factor
  - The variable is summed out from the generated product factor and a new factor is obtained
  - The new factor is added to the set of the available factors

↓

The resulted factor does not necessarily correspond to any probability or conditional probability in the network

Variable Elimination Algorithm

- Given: BN, evidence e, a query $P(\boldsymbol{Q}|\boldsymbol{x_e})$
- Choose an **ordering** on variables, e.g., $X_1, \ldots, X_n$
- For i = 1 to n, If $X_i \notin \{\boldsymbol{Q}, \boldsymbol{X_e}\}$
  - Collect factors $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_k$ that include X_i
  - Generate a new factor by eliminating X_i from these factors:
  - $\boldsymbol{g} = \sum_{X_i} \prod_{j=1}^{k} \boldsymbol{f}_j$
- Multiply all remaining factors
- Normalize $P(\boldsymbol{Q}, \boldsymbol{x_e})$ to obtain $P(\boldsymbol{Q}|\boldsymbol{x_e})$

- Evaluating expressions in a proper order
- Storing intermediate results
- Summation only for those portions of the expression that depend on that variable

Complexity of Variable Elimination Algorithm

- In each elimination step, the following computations are required:
  - $f(x, x_1, \dots, x_k) = \prod_{i=1}^{M} g_i(x, \boldsymbol{x}_{c_i})$
  - $\sum_x f(x, x_1, \dots, x_k)$
- We need:
  - $(M-1) \times |Val(X)| \times \prod_{i=1}^{k} |Val(X_i)|$ multiplications
    - For each tuple $x, x_1, \dots, x_k$, we need $M-1$ multiplications
  - $|Val(X)| \times \prod_{i=1}^{k} |Val(X_i)|$ additions
    - For each tuple $x_1, \dots, x_k$, we need $Val(X)$ additions

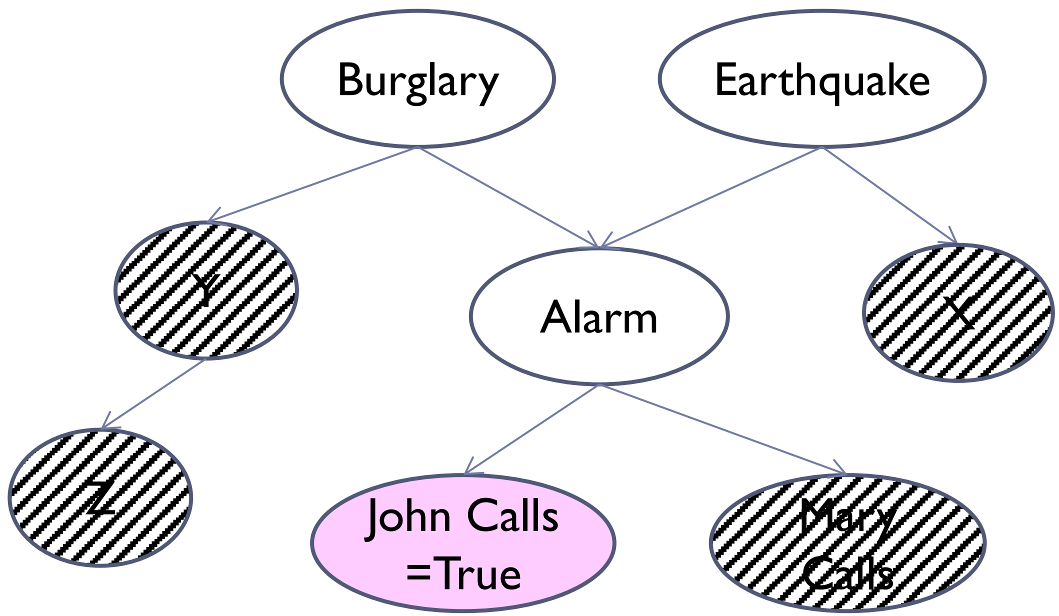
Complexity is exponential in number of variables in the intermediate factor
Size of the created factors is the dominant quantity in the complexity of VE

Variable Elimination: Pruning Irrelevant Variables

- Any variable that is not an ancestor of a query variable or evidence variable is irrelevant to the query.

- Prune all non-ancestors of query or evidence variables:

$P(b, j)$



Variable Elimination Algorithm

- Given: BN, evidence e, a query $P(\boldsymbol{Q}|\boldsymbol{x_e})$
- Prune non-ancestors of $\{\boldsymbol{Q}, \boldsymbol{X_e}\}$
- Choose an **ordering** on variables, e.g., X_1, …, X_n
- For i = 1 to n, If $X_i \notin \{\boldsymbol{Q}, \boldsymbol{X_e}\}$
  - Collect factors $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_k$ that include X_i
  - Generate a new factor by eliminating X_i from these factors:

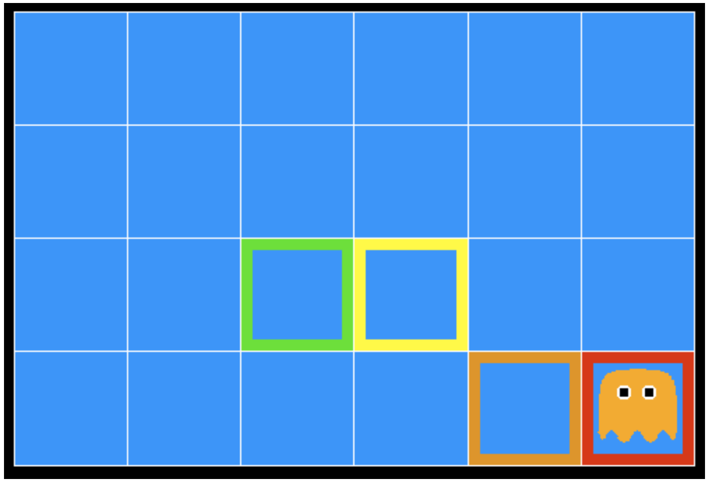
$$\boldsymbol{g} = \sum_{X_i} \prod_{j=1}^{k} \boldsymbol{f}_j$$

- Multiply all remaining factors
- Normalize $P(\boldsymbol{Q}, \boldsymbol{x_e})$ to obtain $P(\boldsymbol{Q}|\boldsymbol{x_e})$

After this summation, X_i is eliminated

Inference in Ghostbusters

- A ghost is in the grid somewhere
- Sensor readings tell how close a square is to the ghost
  - On the ghost: red
  - 1 or 2 away: orange
  - 3 or 4 away: yellow
  - 5+ away: green



- Sensors are noisy, but we know P(Color | Distance)

P(red | 3)	P(orange | 3)	P(yellow | 3)	P(green | 3)
0.05	0.15	0.5	0.3

Video of Demo Ghostbusters

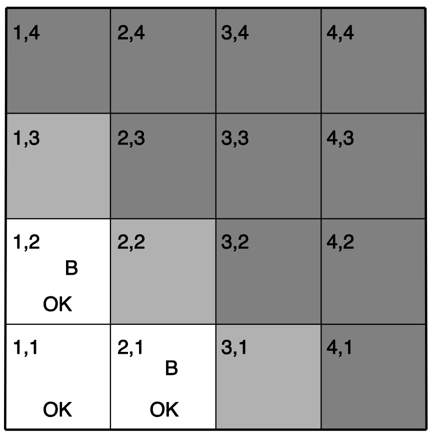
Wampus Example

Environment:

Each square other than [1,1] can be a pit with probability 0.2

It the squares adjacent to a pit, agent perceives a Breeze

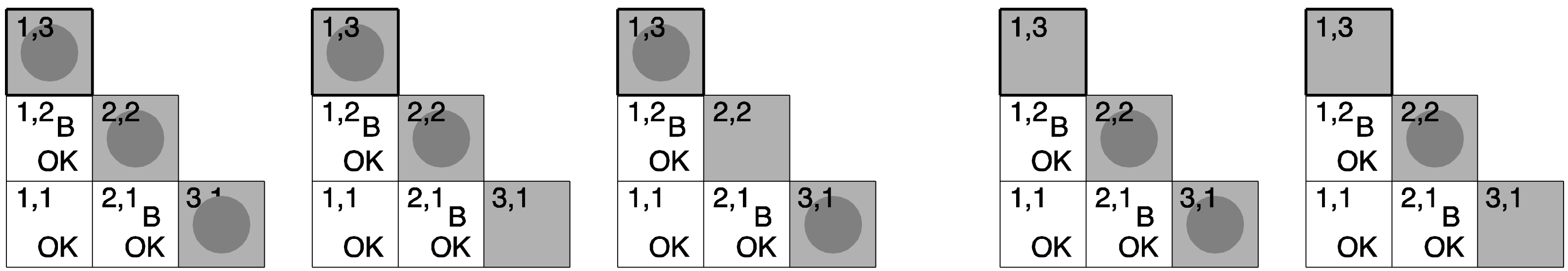
Game ends when the agent enters a pit



$$evidence = \neg b_{1,1} \wedge b_{1,2} \wedge b_{2,1} \wedge \neg p_{1,1} \wedge \neg p_{1,2} \wedge \neg p_{2,1}$$

$$P\left(P_{1,3} \middle| evidence\right) = ?$$

Wumpus Example



0.2 x 0.2 = 0.04 0.2 x 0.8 = 0.16 0.8 x 0.2 = 0.16 0.2 x 0.2 = 0.04 0.2 x 0.8 = 0.16

(a) (b)

Possible worlds with $P_{1,3} = true$ Possible worlds with $P_{1,3} = false$

$$P(P_{1,3} = True \mid evidence) \propto 0.2 \times [0.2\times0.2 + 0.2\times0.8 + 0.8\times0.2]$$
$$P(P_{1,3} = False \mid evidence) \propto 0.8 \times [0.2\times0.2 + 0.2\times0.8]$$
$$\Rightarrow P(P_{1,3} = True \mid evidence) = 0.31$$

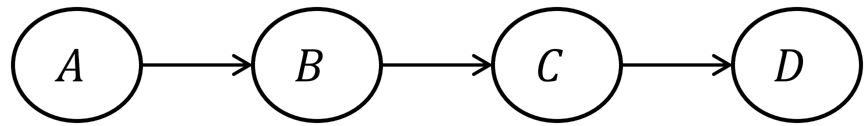
Variable Elimination Complexity

- Eliminates by summation non-observed non-query variables one by one by distributing the sum over the product

- Complexity determined by the size of the largest factor

- Variable elimination can lead to significant costs saving but its efficiency depends on the network structure .
  - there are still cases in which this algorithm we lead to exponential time.

Example: Inference on a Chain

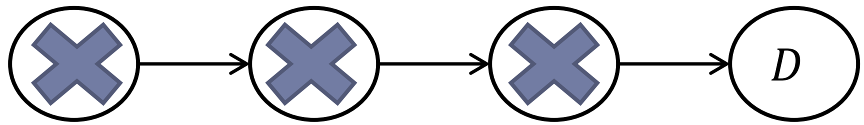


$$P(D) = \sum_A \sum_B \sum_C P(A, B, C, D)$$

$$P(D) = \sum_A \sum_B \sum_C P(A)P(B|A)P(C|B)P(D|C)$$

- A naïve summation needs to enumerate over an exponential number of terms

Inference on a Chain:



$$P(D) = \sum_A \sum_B \sum_C P(A)P(B|A)P(C|B)P(D|C)$$

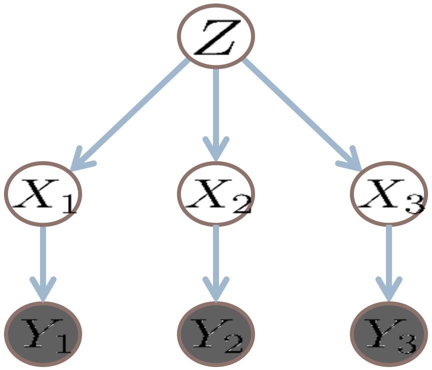
$$= \sum_C \sum_B \sum_A P(A)P(B|A)P(C|B)P(D|C)$$

$$= \sum_C P(D|C) \underbrace{\sum_B P(C|B) \underbrace{\sum_A P(A)\, P(B|A)}_{f(B)}}_{f(C)}$$

- In a chain of n nodes each having d values, $O(nd^2)$ instead of $O(d^n)$

Another Variable Elimination Example

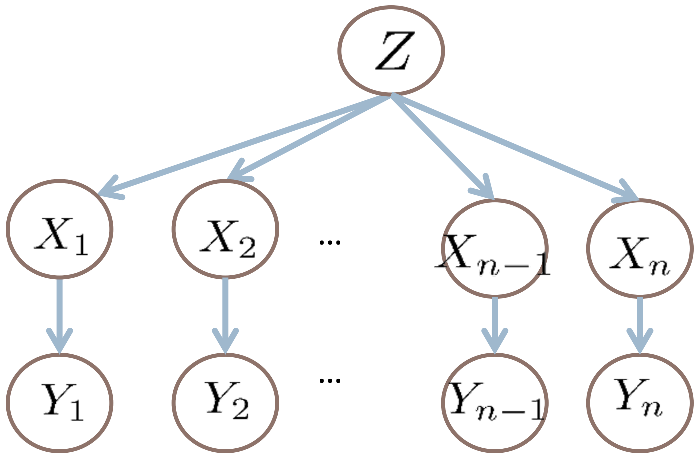
Query: $P(X_3|Y_1 = y_1, Y_2 = y_2, Y_3 = y_3)$



Computational complexity critically depends on the largest factor being generated in this process. Size of factor = number of entries in table. In example above (assuming binary) all factors generated are of size 2 --- as they all only have one variable (Z, Z, and X_3 respectively).

Variable Elimination Ordering

- For the query $P(X_n|y_1,\ldots,y_n)$ work through the following two different orderings as done in previous slide: $Z, X_1, \ldots, X_{n-1}$ and $X_1, \ldots, X_{n-1}, Z$. What is the size of the maximum factor generated for each of the orderings?



- Answer: 2^{n+1} versus 2^2 (assuming binary)

- In general: the ordering can greatly affect efficiency.

VE: Computational and Space Complexity

- The computational and space complexity of variable elimination is determined by the largest factor

- 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!

Worst Case Complexity?

- CSP:

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4 \vee \neg x_5) \wedge (x_2 \vee x_5 \vee x_7) \wedge (x_4 \vee x_5 \vee x_6) \wedge (\neg x_5 \vee x_6 \vee \neg x_7) \wedge (\neg x_5 \vee \neg x_6 \vee x_7)$$

$$P(X_i = 0) = P(X_i = 1) = 0.5$$

$$Y_1 = X_1 \vee X_2 \vee \neg X_3$$

$$\ldots$$

$$Y_8 = \neg X_5 \vee X_6 \vee X_7$$

$$Y_{1,2} = Y_1 \wedge Y_2$$

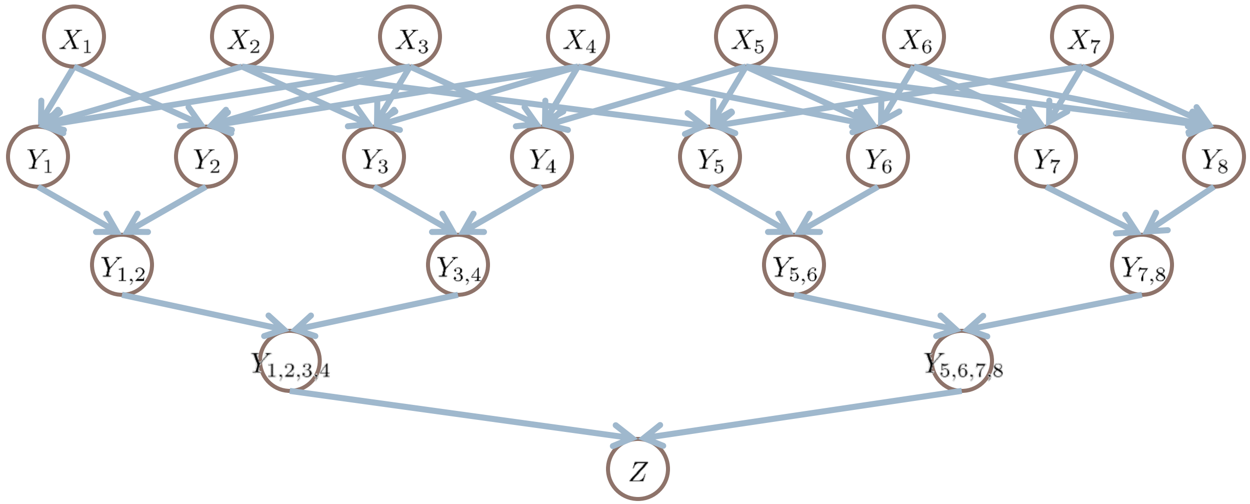
$$\ldots$$

$$Y_{7,8} = Y_7 \wedge Y_8$$

$$Y_{1,2,3,4} = Y_{1,2} \wedge Y_{3,4}$$

$$Y_{5,6,7,8} = Y_{5,6} \wedge Y_{7,8}$$

$$Z = Y_{1,2,3,4} \wedge Y_{5,6,7,8}$$

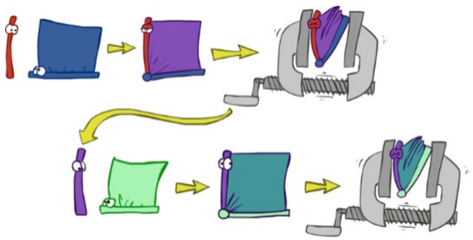


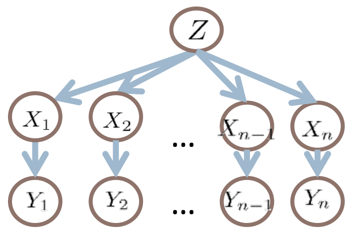
- If we can answer P(z) equal to zero or not, we answered whether the 3-SAT problem has a solution.

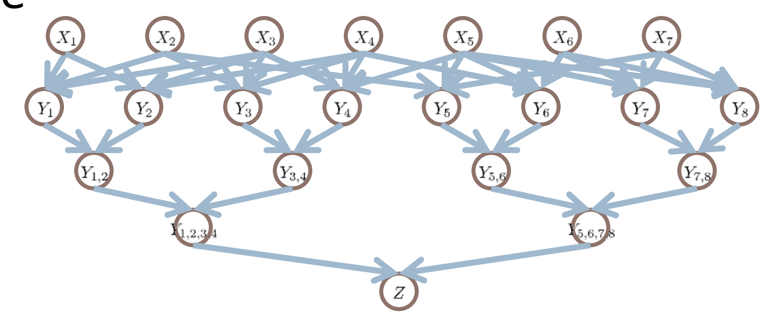
- Hence inference in Bayes' nets is NP-hard. No known efficient probabilistic inference in general.

Variable Elimination: Summary

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



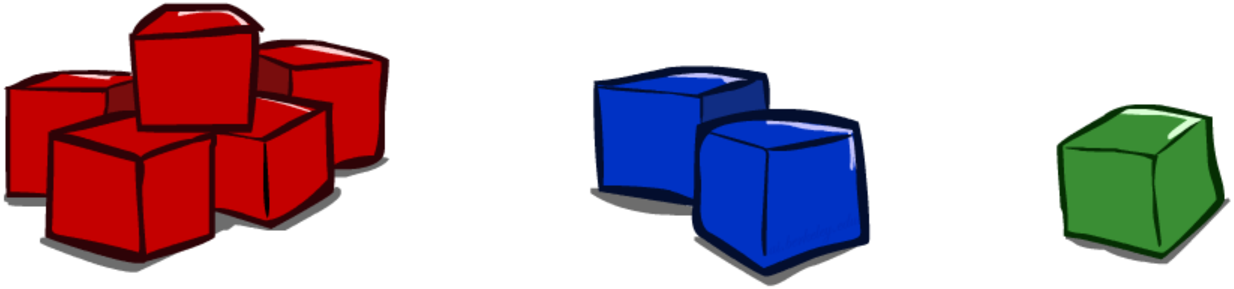




Bayes' Nets

- ✓ Representation
- ✓ Conditional Independences
- Probabilistic Inference
  - ✓ Enumeration (exact, exponential complexity)
  - ✓ Variable elimination (exact, worst-case exponential complexity, often better)
  - ✓ Inference is NP-complete
  - Sampling (approximate)
- Learning Bayes' Nets from Data

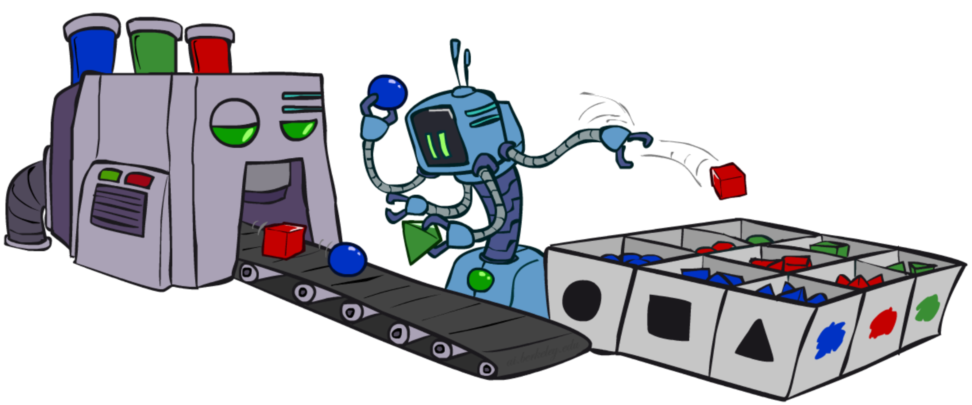
Approximate Inference: Sampling



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?
  - Often very fast yo get a descent 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
    - 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

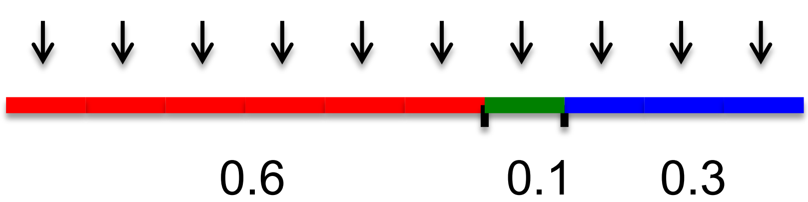
Sampling

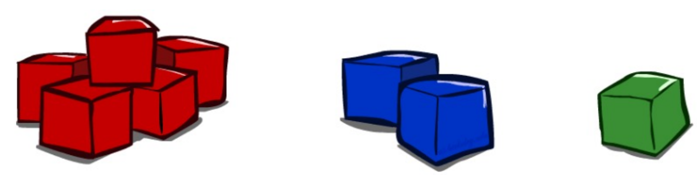
- Sampling from a given distribution
  - Step 1: Get sample *u* from uniform distribution over [0, 1)
    - E.g. random() in python
  - Step 2: Convert this sample *u* into an outcome for the given distribution by having each outcome associated with a sub-interval of [0,1) with sub-interval size equal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

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:

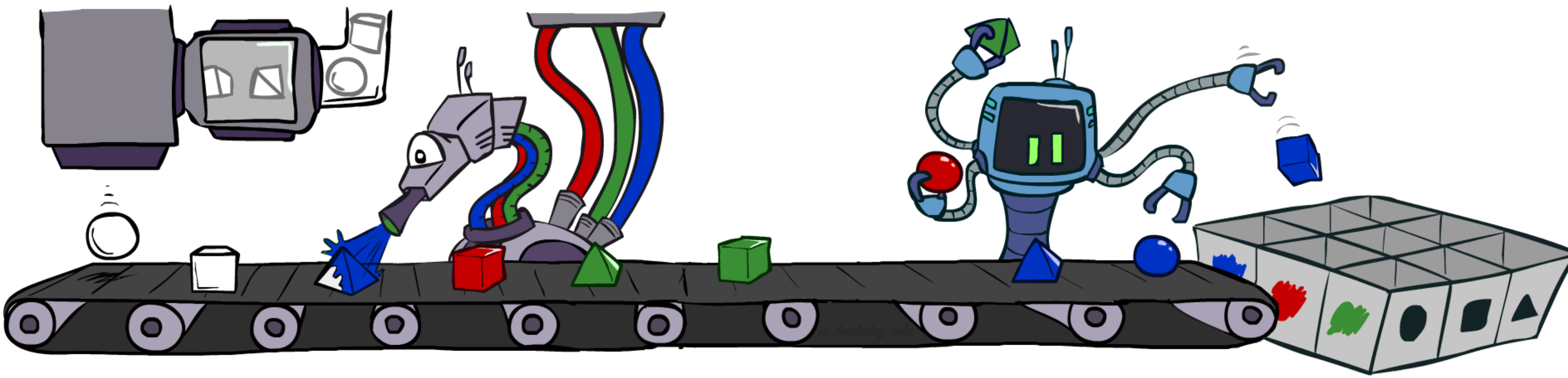


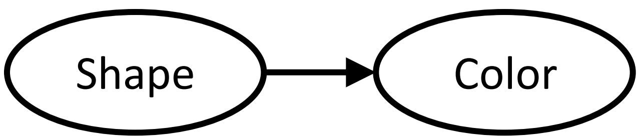


Sampling in Bayes' Nets

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

Prior Sampling





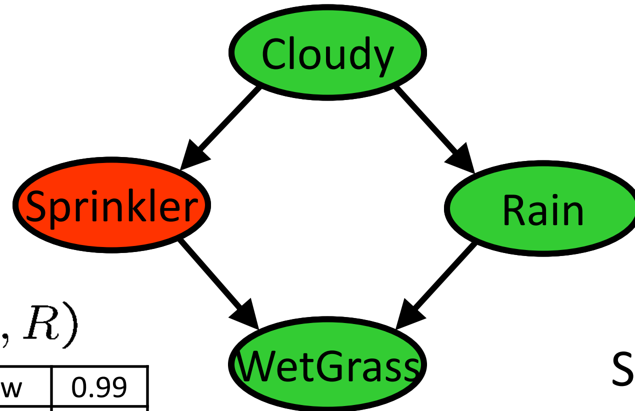
Prior Sampling

$P(C)$

+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

$P(W|S,R)$

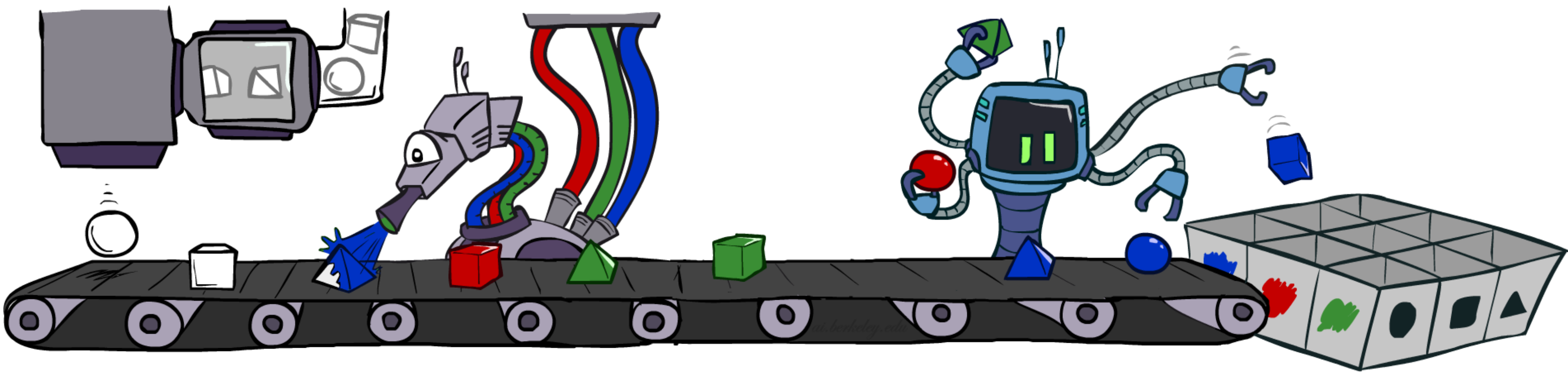
+s	+r	+w	0.99
		-w	0.01
	-r	+w	0.90
		-w	0.10
-s	+r	+w	0.90
		-w	0.10
	-r	+w	0.01
		-w	0.99

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 $(x_1, x_2, \ldots, x_n)$



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

- I.e., the sampling procedure is consistent

Example

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

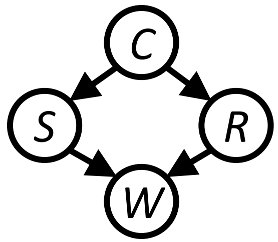
  +c, -s, +r, +w

  +c, +s, +r, +w

  -c, +s, +r, -w

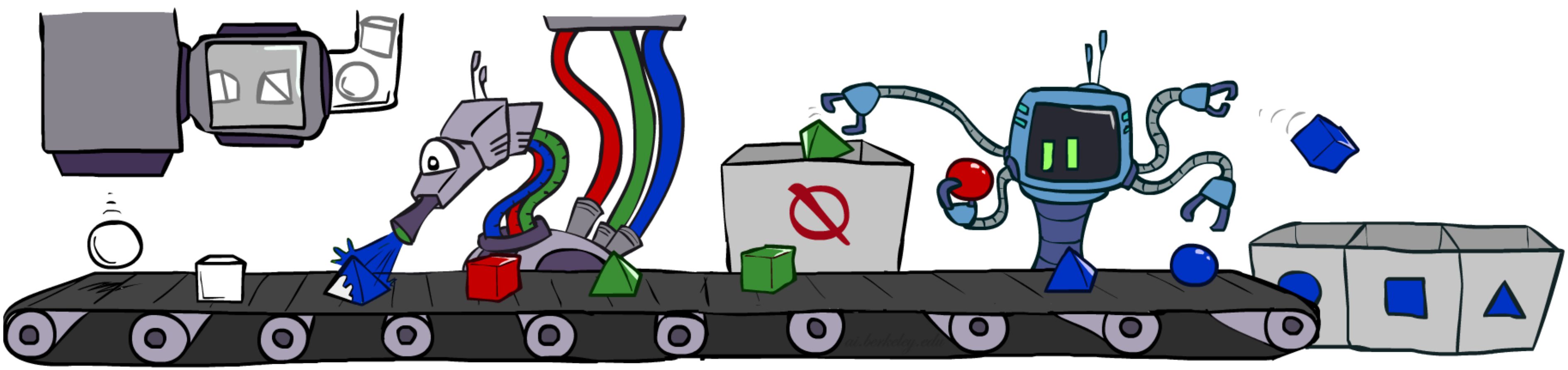
  +c, -s, +r, +w

  -c, -s, -r, +w



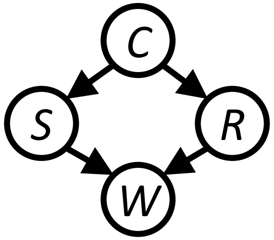
- If we want to know P(W)
  - We have counts <+w:4, -w:1>
  - Normalize to get P(W) = <+w:0.8, -w:0.2>
  - This will get closer to the true distribution with more samples
  - Can estimate anything else, too
  - What about P(C| +w)? P(C| +r, +w)? P(C| -r, -w)?
  - Fast: can use fewer samples if less time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

Rejection Sampling



Rejection Sampling

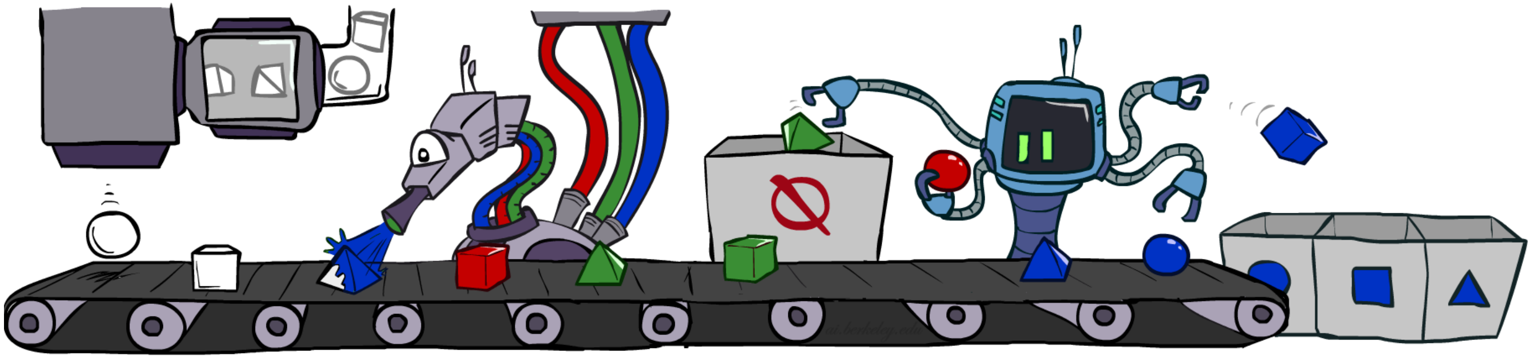
- A simple application of prior sampling for estimating conditional probabilities
  - Let's say we want $P(C \mid r, w) = \alpha P(C, r, w)$
  - For these counts, samples with $\neg r$ or $\neg w$ ***are not relevant***
  - So count the C outcomes for samples with r, w and reject all other samples
-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



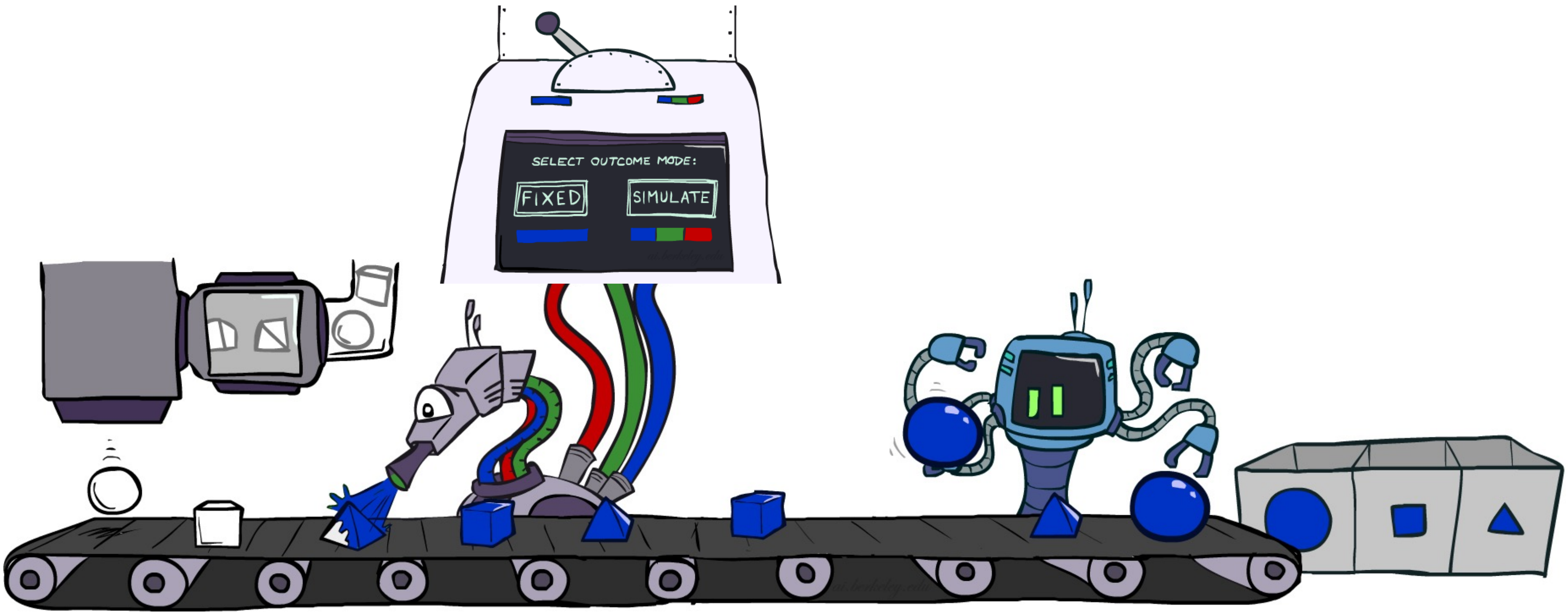
+c, -s, +r, +w
+c, +s, +r, +w
-c, +s, +r, -w
+c, -s, +r, +w
-c, -s, -r, +w

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 $(x_1, x_2, \ldots, x_n)$



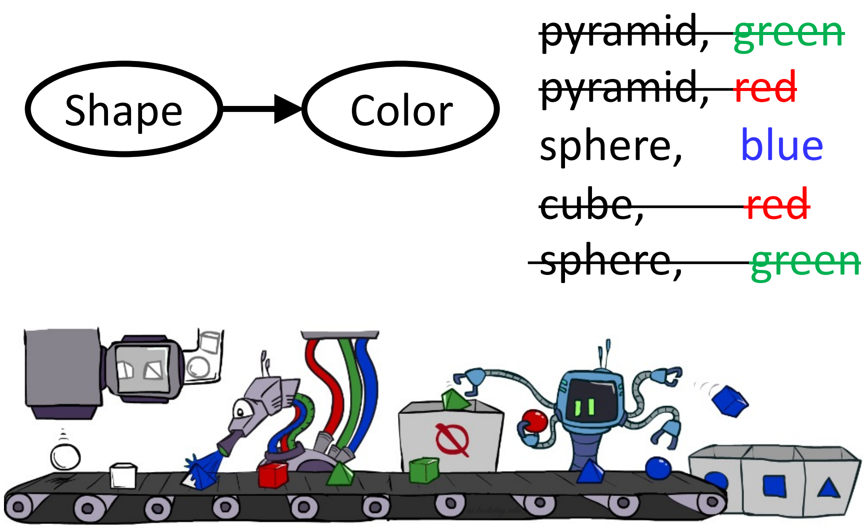
Likelihood Weighting

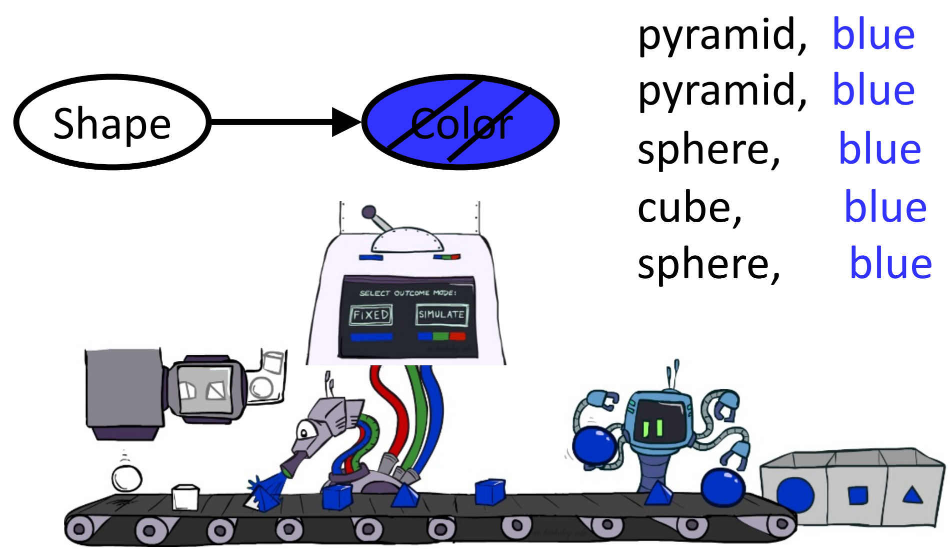


Likelihood Weighting

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

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





Likelihood Weighting

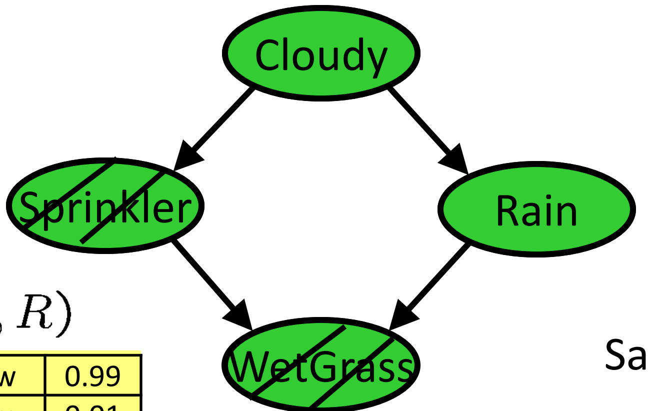
P(C| +s, +w)=?

$P(C)$

+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:

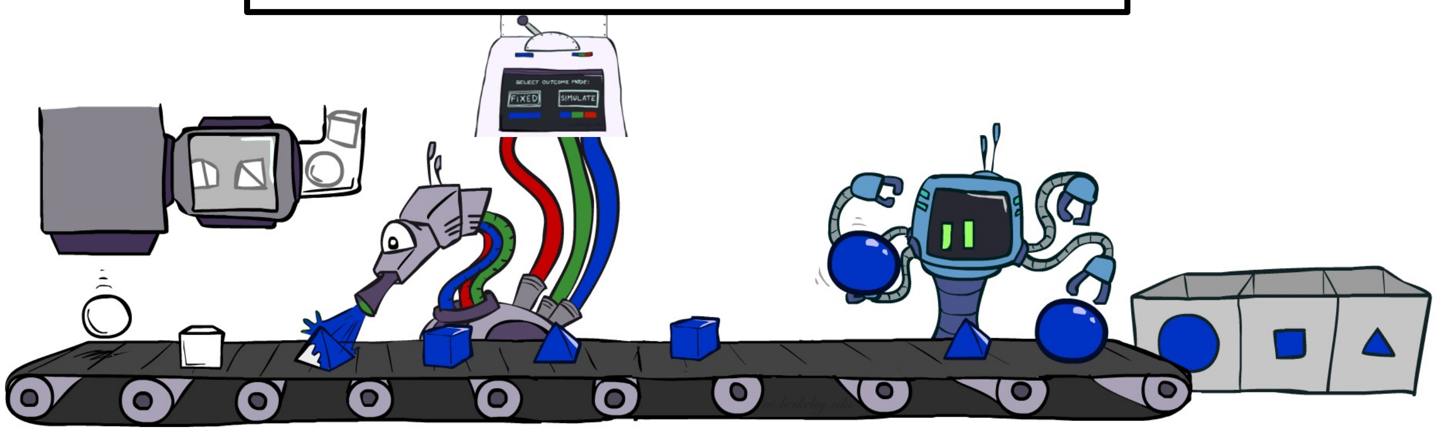
+c, +s, +r, +w

...

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

Likelihood Weighting

- 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 $(x_1, x_2, …, x_n), w$



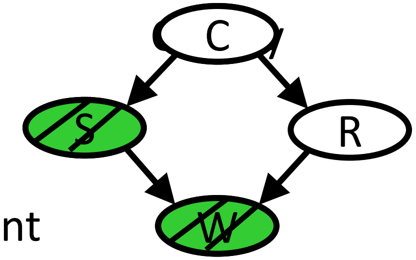
Likelihood Weighting

- 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

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



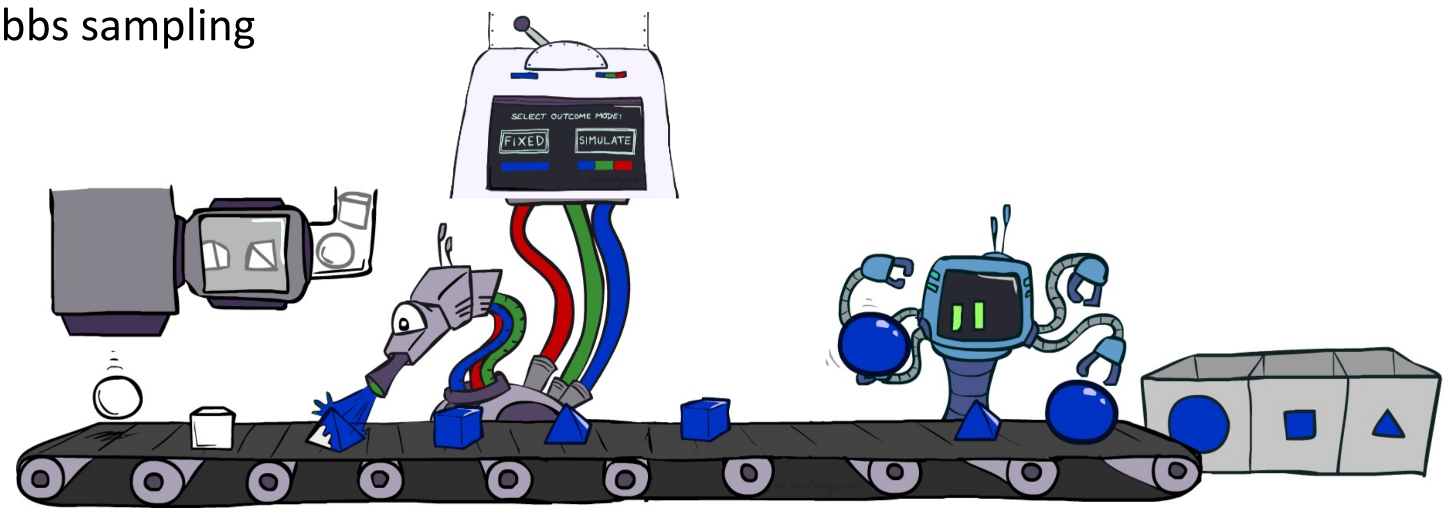
- Together, weighted sampling distribution is consistent

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

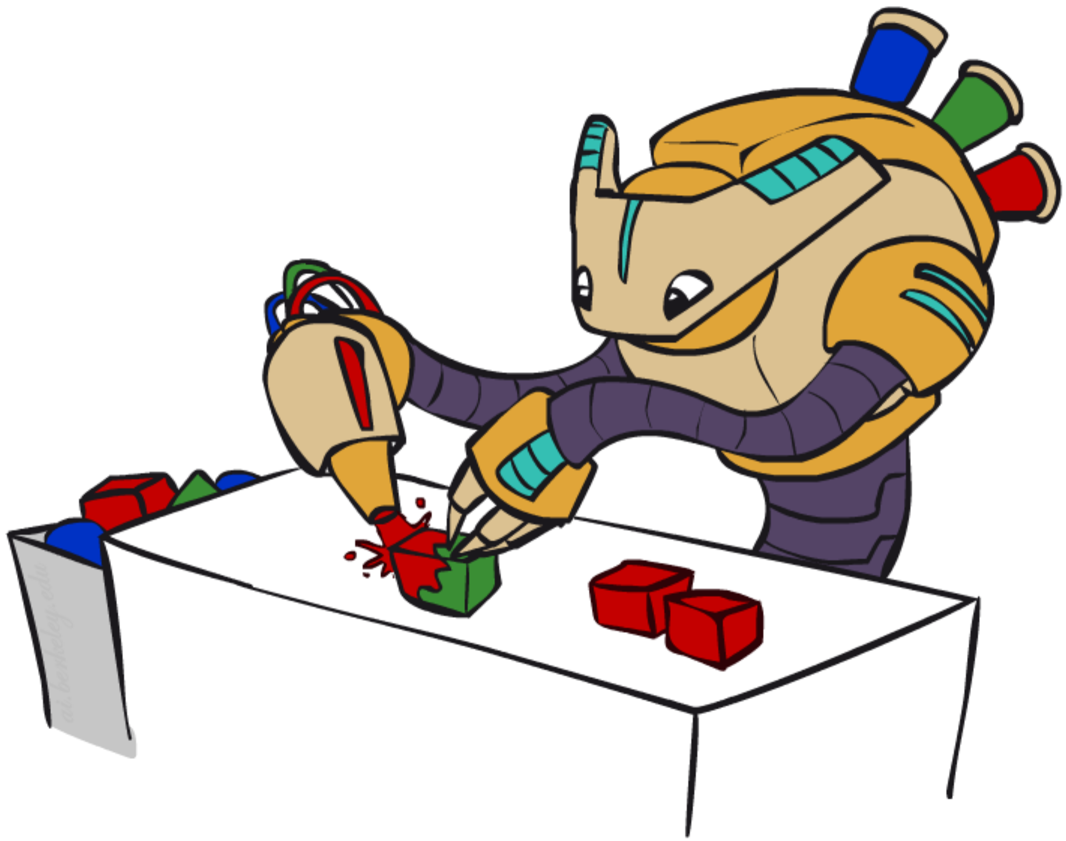
Likelihood Weighting

- 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

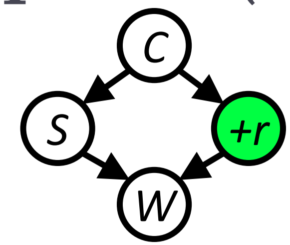


Gibbs Sampling

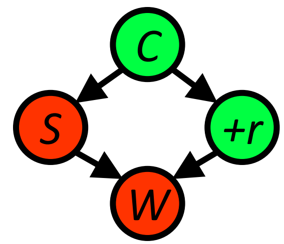
- *Procedure:* keep track of a full instantiation $x_1, x_2, \ldots, x_n$.
  - Start with an arbitrary instantiation consistent with the evidence.
  - Sample one variable at a time, conditioned on all the rest, but keep evidence fixed.
  - Keep repeating this for a long time.
- *Property:* in the limit 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, 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 want high weight.

Gibbs Sampling Example: P(S | +r)

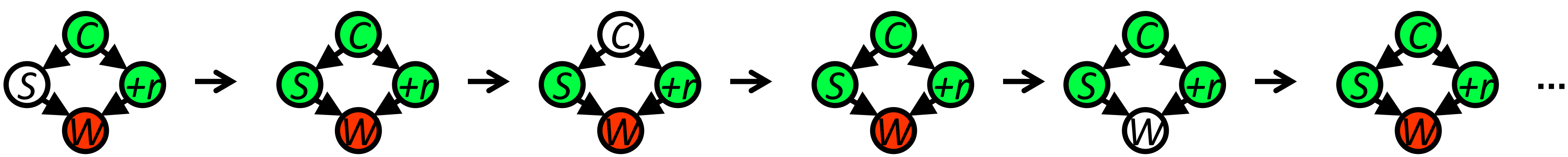
- Step 1: Fix evidence
  - R = +r



- 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 $P(W|+s,+c,+r)$

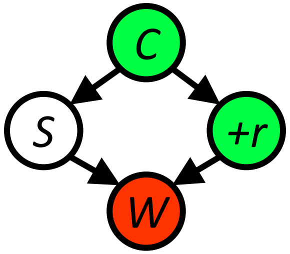
Gibbs Sampling

- 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)

Efficient Resampling of One Variable

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

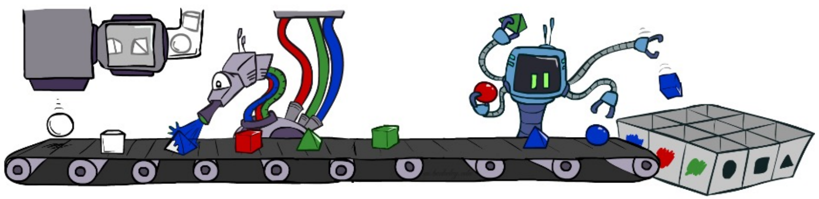
$$
\begin{aligned}
P(S|+c,+r,-w) &= \frac{P(S,+c,+r,-w)}{P(+c,+r,-w)} \\
&= \frac{P(S,+c,+r,-w)}{\sum_s P(s,+c,+r,-w)} \\
&= \frac{P(+c)P(S|+c)P(+r|+c)P(-w|S,+r)}{\sum_s P(+c)P(s|+c)P(+r|+c)P(-w|s,+r)} \\
&= \frac{P(+c)P(S|+c)P(+r|+c)P(-w|S,+r)}{P(+c)P(+r|+c)\sum_s P(s|+c)P(-w|s,+r)} \\
&= \frac{P(S|+c)P(-w|S,+r)}{\sum_s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



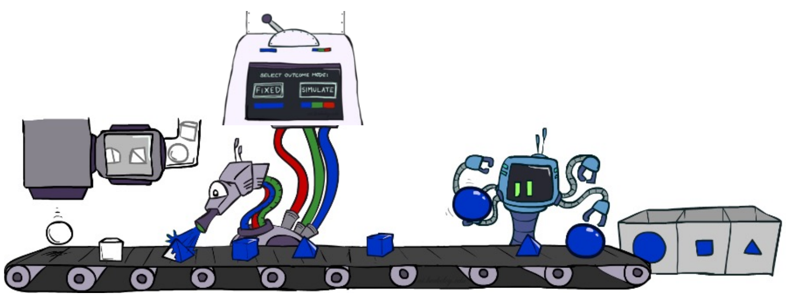
- Many things cancel out – only CPTs with S remain!
- More generally: only CPTs that have resampled variable need to be considered, and joined together

Bayes' Net Sampling Summary

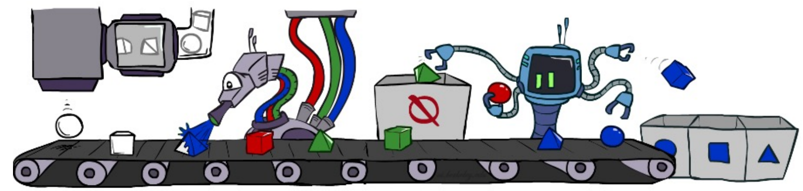
- Prior Sampling P:
  - Generate complete samples from $P(x_1,\ldots,x_n)$



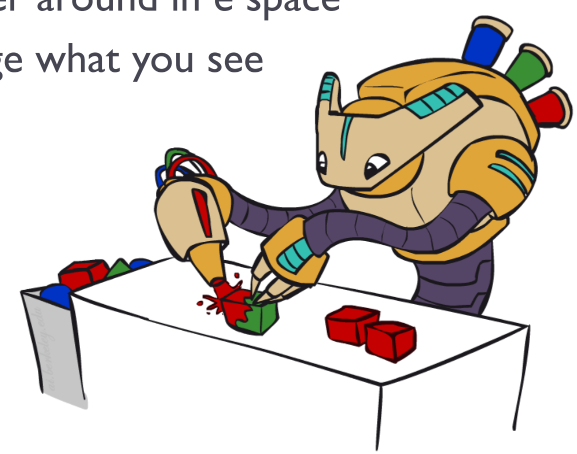
- Likelihood Weighting P(Q | e):
  - Weight samples by how well they predict ***e***



- Rejection Sampling P(Q | e):
  - Reject samples that don't match e



- Gibbs Sampling P(Q | e):
  - Wander around in e space
  - Average what you see



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