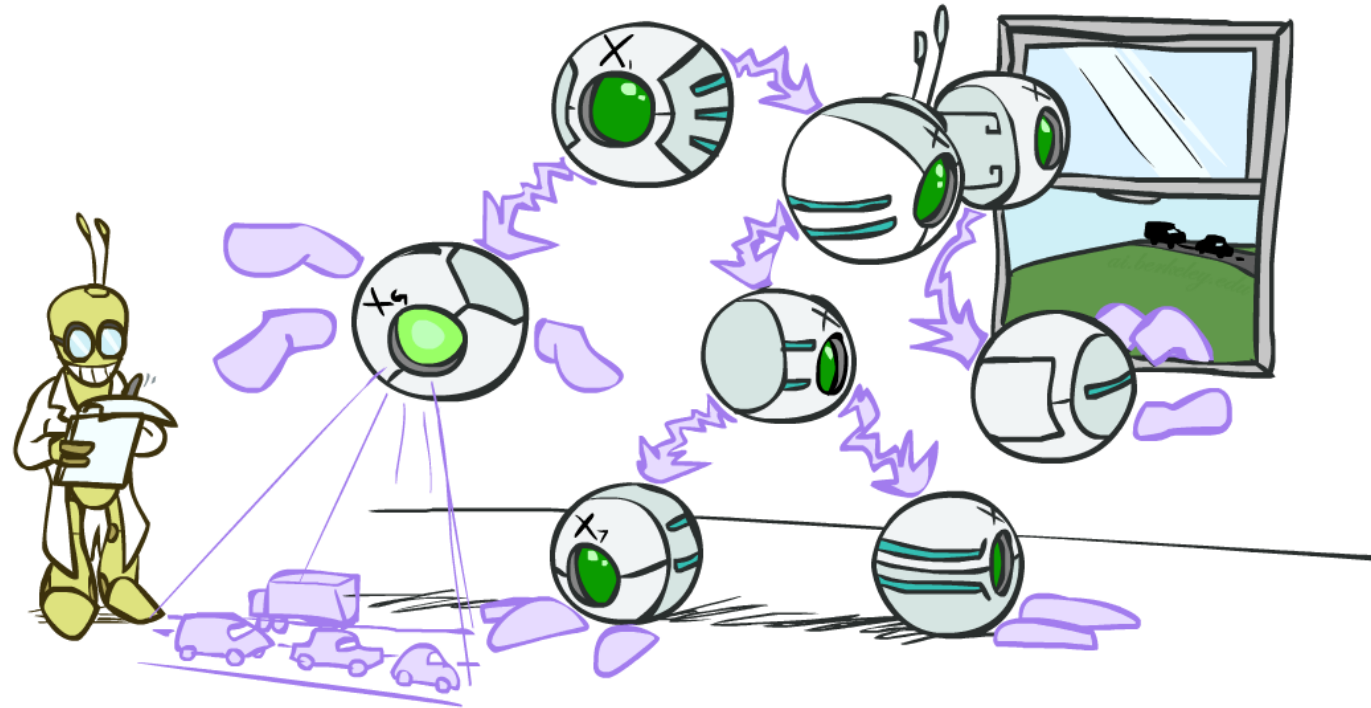


# Bayes Nets: Exact Inference



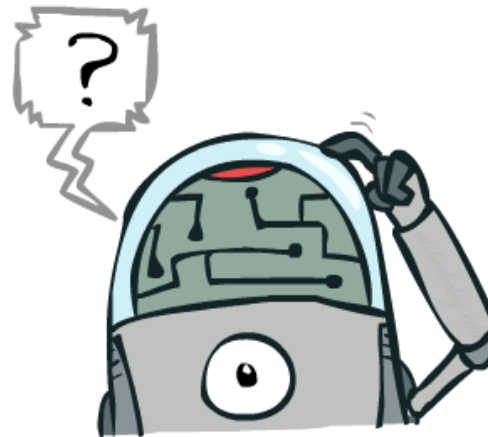
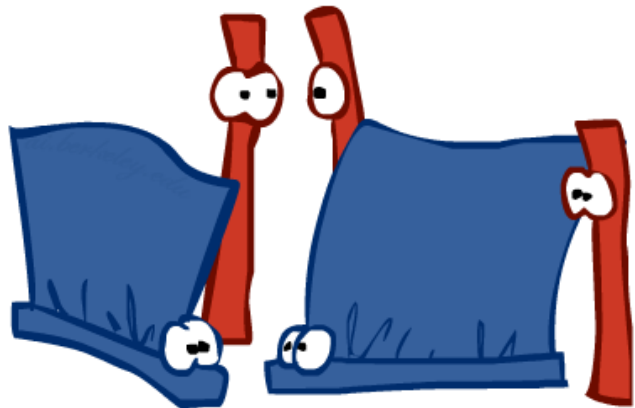
AIMA Chapter 14.4, PRML Chapter 8.4

# Inference

- Inference: calculating some useful quantity from a probabilistic model (joint probability distribution)

- Examples:

- Posterior marginal probability
  - $P(Q|e_1, \dots, e_k)$
  - E.g., what disease might I have?
- Most likely explanation:
  - $\operatorname{argmax}_q P(Q=q|e_1, \dots, e_k)$
  - E.g., what did he say?



# Inference by Enumeration

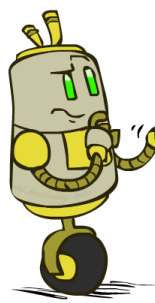
## General case:

- Evidence variables:  $E_1 \dots E_k = e_1 \dots e_k$
  - Query variable:  $Q$
  - Hidden variables:  $H_1 \dots H_r$
- $\left. \begin{array}{l} X_1, X_2, \dots, X_n \\ \text{All variables} \end{array} \right\}$

## We want:

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

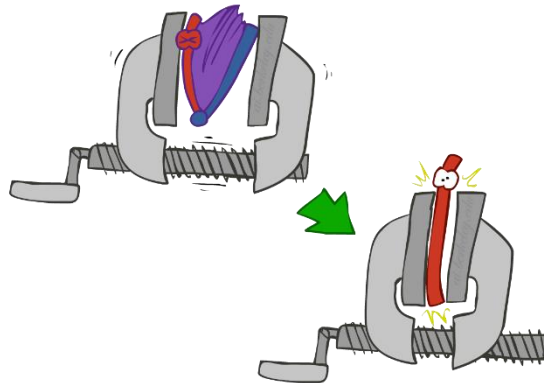
- Step 1: Select the entries consistent with the evidence



x	P(x)
-3	0.05
-1	0.25
0	0.07
1	0.2
5	0.01

2      0.15

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



- Step 3: Normalize

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

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

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

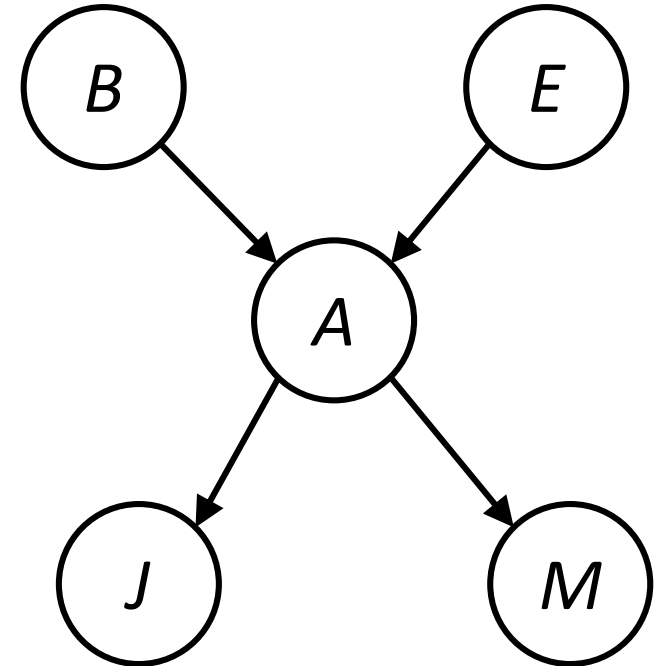
$$P(Q, e_1 \dots e_k) = \sum_{h_1 \dots h_r} \underbrace{P(Q, h_1 \dots h_r, e_1 \dots e_k)}_{X_1, X_2, \dots, X_n}$$

# Inference by Enumeration in Bayes Net

- The joint distribution can be computed from a BN by multiplying the conditional distributions
- Then we can do inference by enumeration

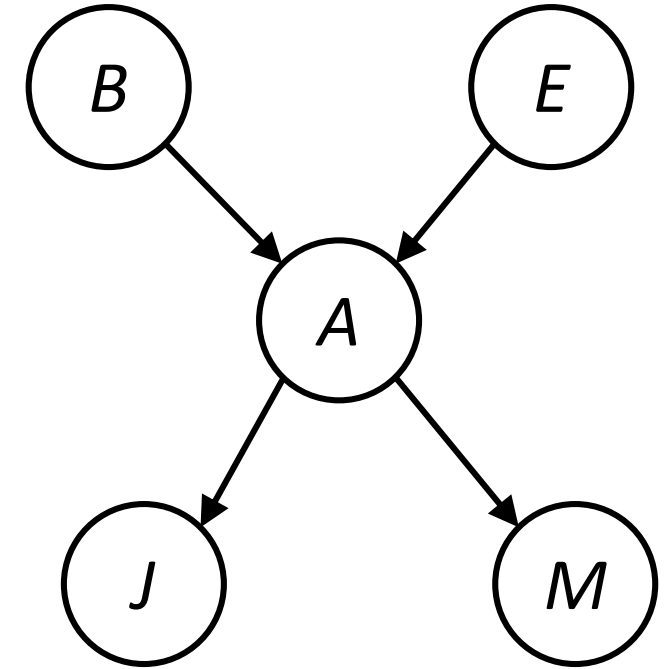
$$\begin{aligned} P(B \mid +j, +m) &\propto_B P(B, +j, +m) \\ &= \sum_{e,a} P(B, e, a, +j, +m) \\ &= \sum_{e,a} P(B)P(e)P(a|B, e)P(+j|a)P(+m|a) \end{aligned}$$

- Problem: sums of **exponentially many** products!



# Inference by Enumeration in Bayes Net

$$\begin{aligned}P(B \mid +j, +m) &\propto_B P(B, +j, +m) \\&= \sum_{e,a} P(B, e, a, +j, +m) \\&= \sum_{e,a} P(B)P(e)P(a|B, e)P(+j|a)P(+m|a)\end{aligned}$$



$$\begin{aligned}&= P(B)P(+e)P(+a|B, +e)P(+j|+a)P(+m|+a) + P(B)P(+e)P(-a|B, +e)P(+j|-a)P(+m|-a) \\&+ P(B)P(-e)P(+a|B, -e)P(+j|+a)P(+m|+a) + P(B)P(-e)P(-a|B, -e)P(+j|-a)P(+m|-a)\end{aligned}$$

Lots of repeated subexpressions!

# Can we do better?

---

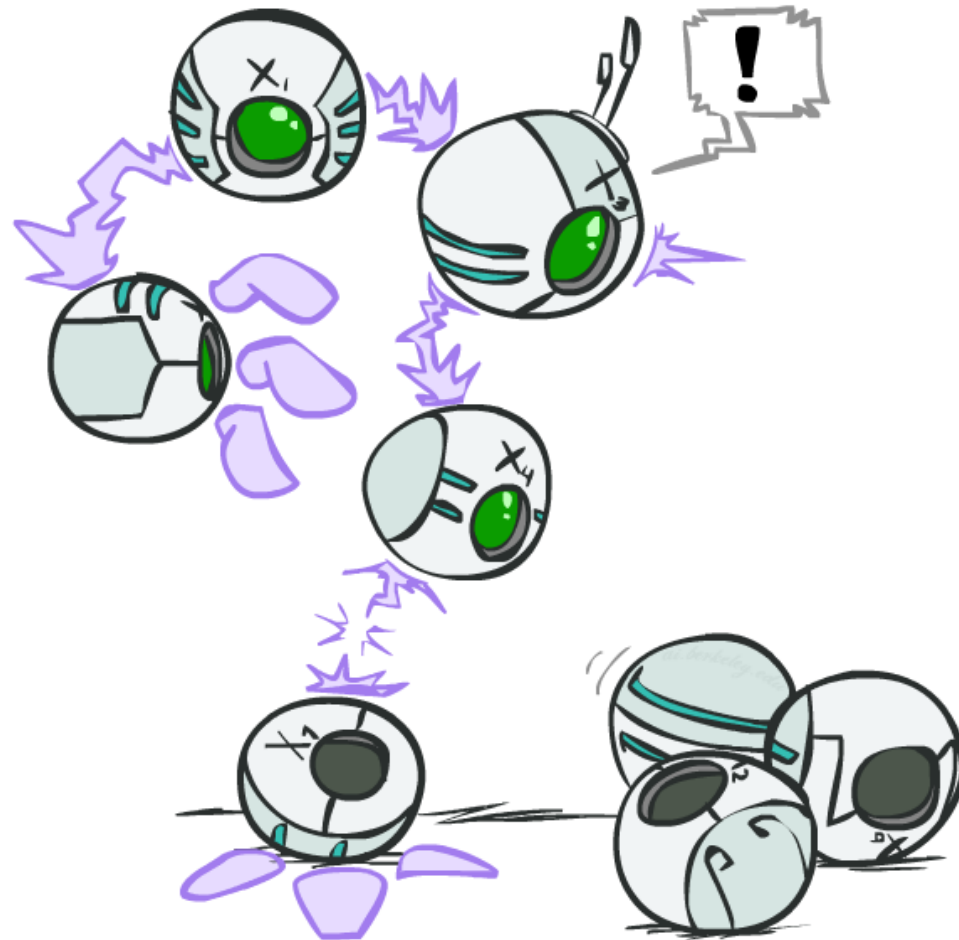
- Consider  $uw y + uw z + ux y + ux z + vw y + vw z + vx y + vx z$ 
  - 16 multiplies, 7 adds
  - Lots of repeated subexpressions!
- Rewrite as  $(u+v)(w+x)(y+z)$ 
  - 2 multiplies, 3 adds
- Idea: Move summations inwards as far as possible

# Variable elimination: The basic ideas

---

- Move summations inwards as far as possible
  - $$\begin{aligned} P(B \mid j, m) &= \alpha \sum_{e,a} P(B) P(e) P(a \mid B, e) P(j \mid a) P(m \mid a) \\ &= \alpha P(B) \sum_e P(e) \sum_a P(a \mid B, e) P(j \mid a) P(m \mid a) \end{aligned}$$
  - Note:  $P(a \mid B, e)$  isn't a single number, it's a bunch of different numbers depending on the values of  $a$ ,  $B$  and  $e$
  - It's clearer to view the computation with operations on **factors** (arrays of numbers)

# Operations on Factors





# Factors

- A **factor** is a multi-dimensional array to represent  $P(Y_1 \dots Y_N \mid X_1 \dots X_M)$ 
  - If a variable is assigned (represented with lower-case), its dimension is missing from the array

- Joint distribution:  $P(X,Y)$

- Entries  $P(x,y)$  for all  $x, y$
- Sums to 1

$$P(T, W)$$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

- Selected joint:  $P(x,Y)$

- A slice of the joint distribution
- Entries  $P(x,y)$  for fixed  $x$ , all  $y$
- Sums to  $P(x)$

$$P(\text{cold}, W)$$

T	W	P
cold	sun	0.2
cold	rain	0.3

# Factors

- A **factor** is a multi-dimensional array to represent  $P(Y_1 \dots Y_N \mid X_1 \dots X_M)$ 
  - If a variable is assigned (represented with lower-case), its dimension is missing from the array

- Single conditional:  $P(Y \mid x)$

- Entries  $P(y \mid x)$  for fixed  $x$ , all  $y$
- Sums to 1

$$P(W \mid cold)$$

T	W	P
cold	sun	0.4
cold	rain	0.6

- Family of conditionals:

$$P(X \mid Y)$$

- Multiple conditionals
- Entries  $P(x \mid y)$  for all  $x, y$
- Sums to  $|Y|$

$$P(W \mid T)$$

T	W	P
hot	sun	0.8
hot	rain	0.2
cold	sun	0.4
cold	rain	0.6

$$P(W \mid hot)$$

$$P(W \mid cold)$$

# Factors

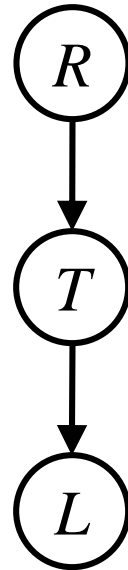
- A **factor** is a multi-dimensional array to represent  $P(Y_1 \dots Y_N \mid X_1 \dots X_M)$ 
  - If a variable is assigned (represented with lower-case), its dimension is missing from the array
- Specified family:  $P(y \mid X)$ 
  - Entries  $P(y \mid x)$  for fixed  $y$ , but for all  $x$
  - Sums to ... who knows!

$P(rain T)$			
T	W	P	
hot	rain	0.2	} $P(rain hot)$ $P(rain cold)$
cold	rain	0.6	

# Running Example: Traffic Domain

## ■ Random Variables

- R: Raining
- T: Traffic
- L: Late



$$P(R)$$

+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

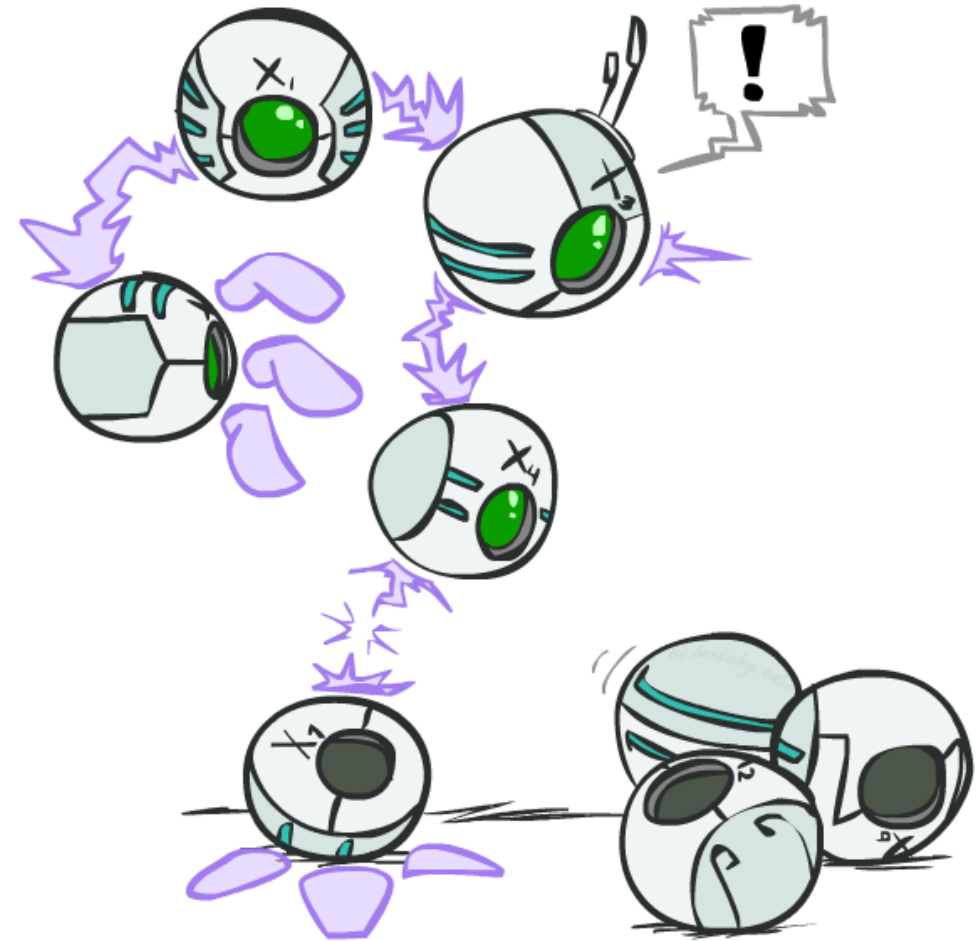
# Running Example: Traffic Domain

- Initial factors are local CPTs (one per node)

$P(R)$		$P(T R)$			$P(L T)$		
+r	0.1	+r	+t	0.8	+t	+l	0.3
-r	0.9	+r	-t	0.2	+t	-l	0.7
		-r	+t	0.1	-t	+l	0.1
		-r	-t	0.9	-t	-l	0.9

- Any known values are selected
  - E.g. if we know  $L = +\ell$ , the initial factors are

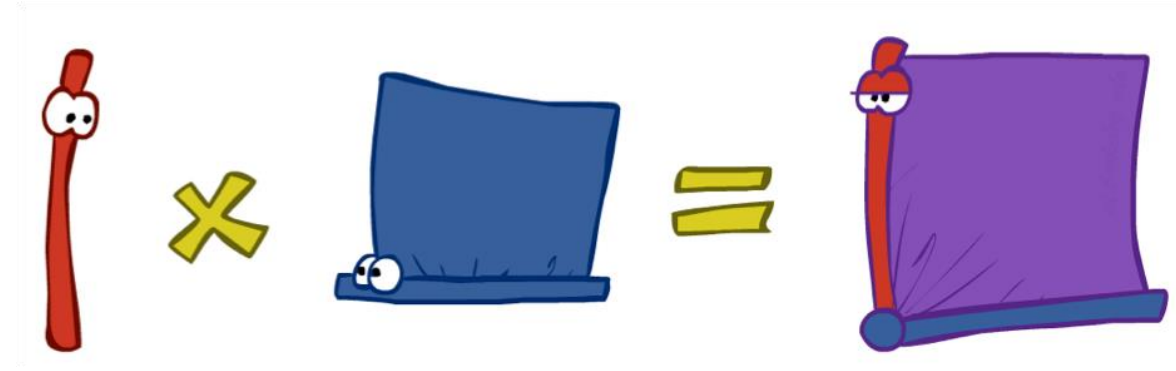
$P(R)$		$P(T R)$			$P(+\ell T)$		
+r	0.1	+r	+t	0.8	+t	+l	0.3
-r	0.9	+r	-t	0.2	-t	+l	0.1
		-r	+t	0.1			
		-r	-t	0.9			



# Operation 1: Join Factors

- First basic operation: **joining factors**

- Just like a database join
- Given multiple factors, build a new factor over the union of the variables involved
- Each entry is computed by **pointwise products**



- Example:

Diagram illustrating the join operation for two factors:

Factor 1 (Left): A node  $R$  pointing to a node  $T$ .

Factor 2 (Middle): A table representing  $P(R)$ :

+r	0.1
-r	0.9

Factor 3 (Right): A table representing  $P(T|R)$ :

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

Result (Far Right): A table representing  $P(R, T)$ :

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

The result is a factor over the union of variables  $R, T$ .

Equation:  $\forall r, t : P(r, t) = P(r) \cdot P(t|r)$


# Operation 2: Eliminate

- Second basic operation: **eliminating a variable**
  - Take a factor and **sum out** (marginalize) a variable
- Example:

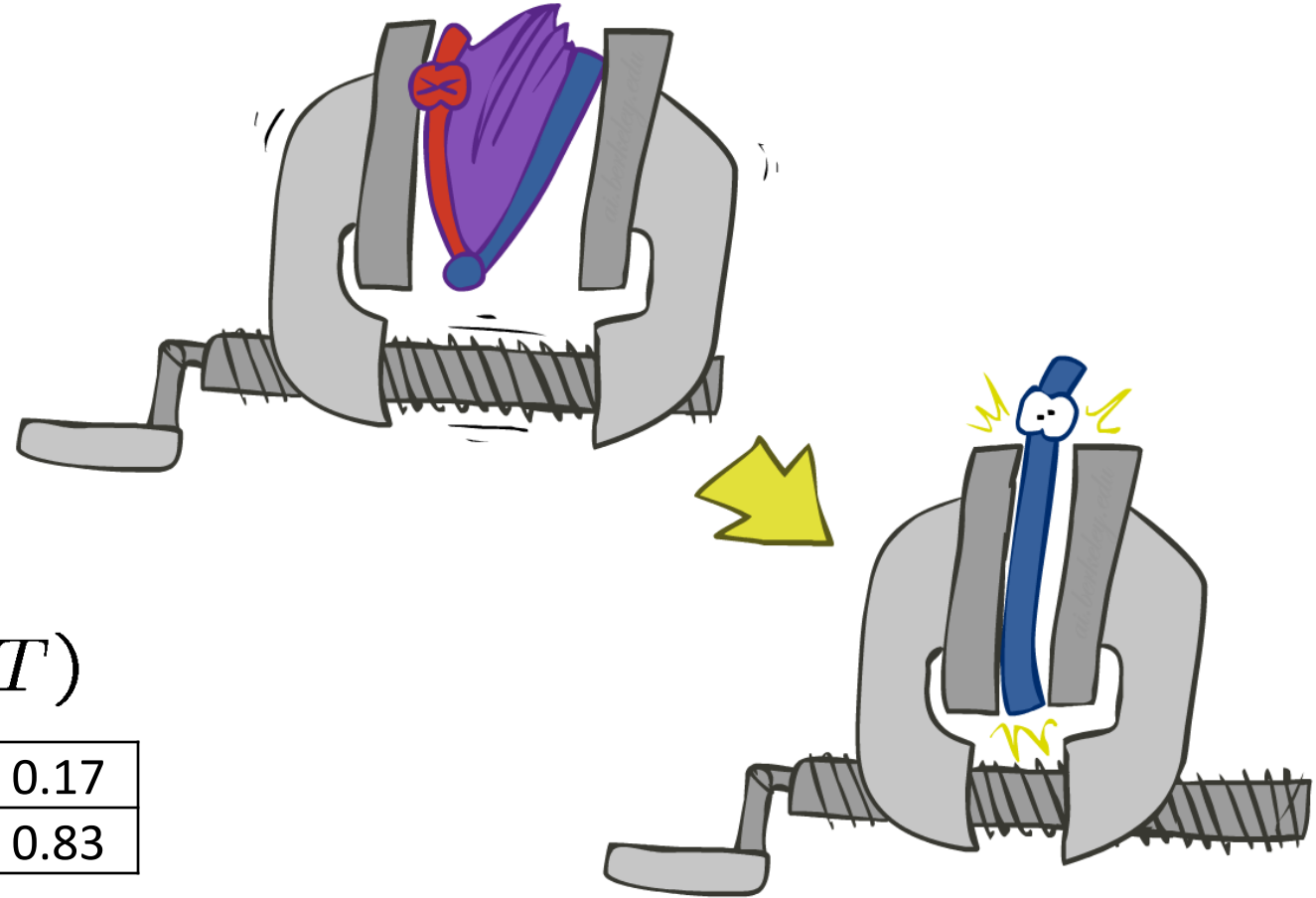
$$P(R, T)$$

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

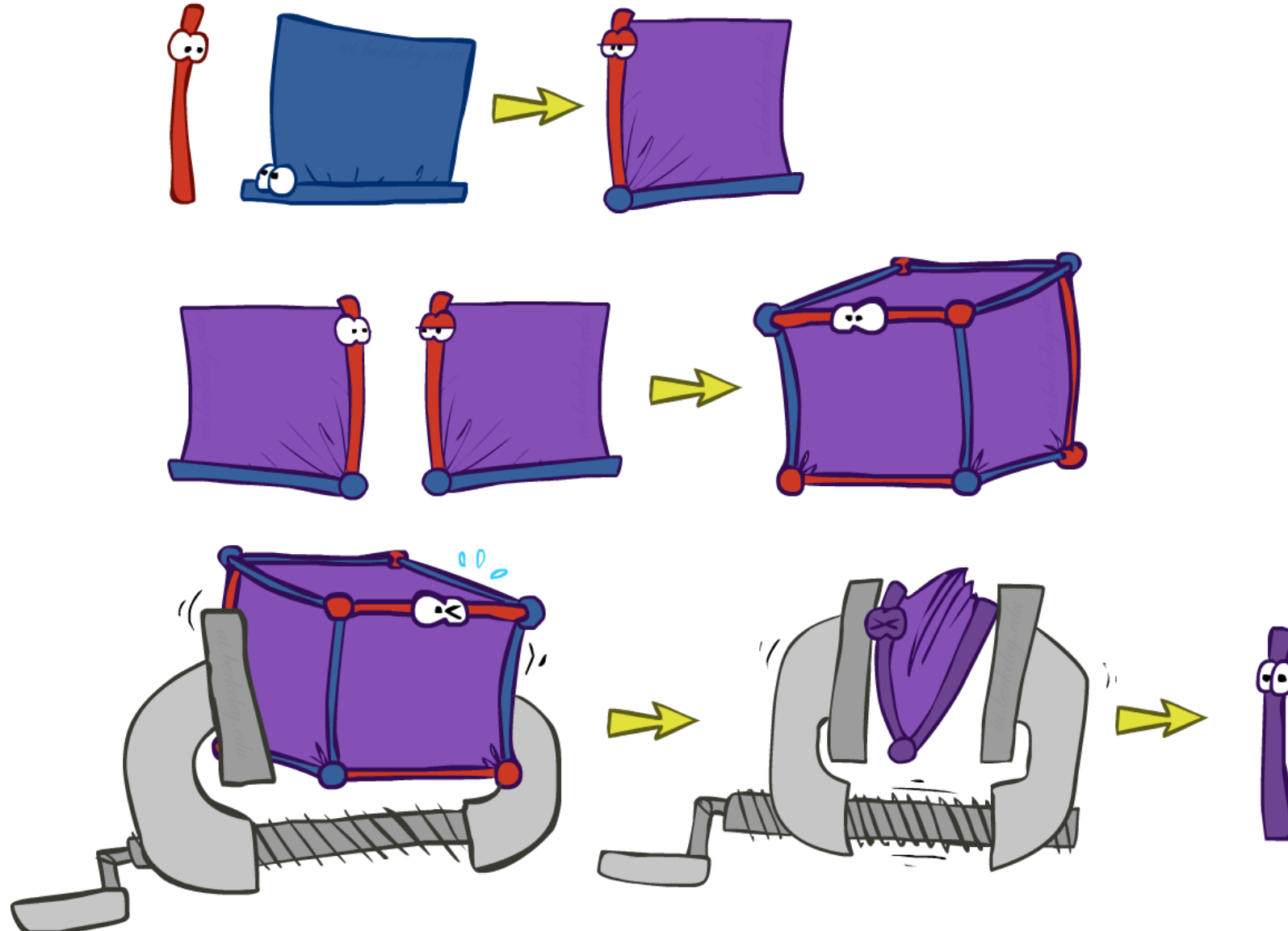
sum  $R$


$$P(T)$$

+t	0.17
-t	0.83

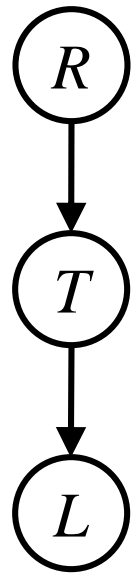


# Inference by Enumeration in BN = Multiple Join + Multiple Eliminate





# Computing $P(L)$ : Multiple Joins



$P(R)$

+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

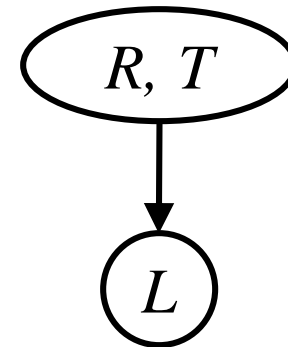


$P(R, T)$

+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



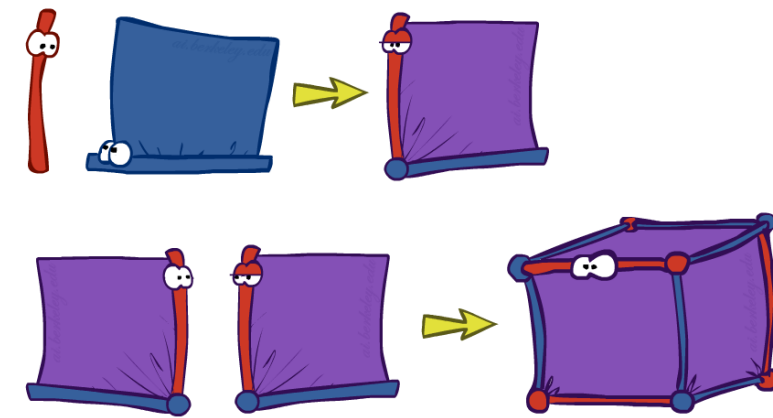
Join



$R, T, L$

$P(R, T, L)$

+r	+t	+l	0.024
+r	+t	-l	0.056
+r	-t	+l	0.002
+r	-t	-l	0.018
-r	+t	+l	0.027
-r	+t	-l	0.063
-r	-t	+l	0.081
-r	-t	-l	0.729



# Computing $P(L)$ : Multiple Elimination

$P(R, T, L)$

$R, T, L$			
+r	+t	+l	0.024
+r	+t	-l	0.056
+r	-t	+l	0.002
+r	-t	-l	0.018
-r	+t	+l	0.027
-r	+t	-l	0.063
-r	-t	+l	0.081
-r	-t	-l	0.729

Sum out R

$P(T, L)$

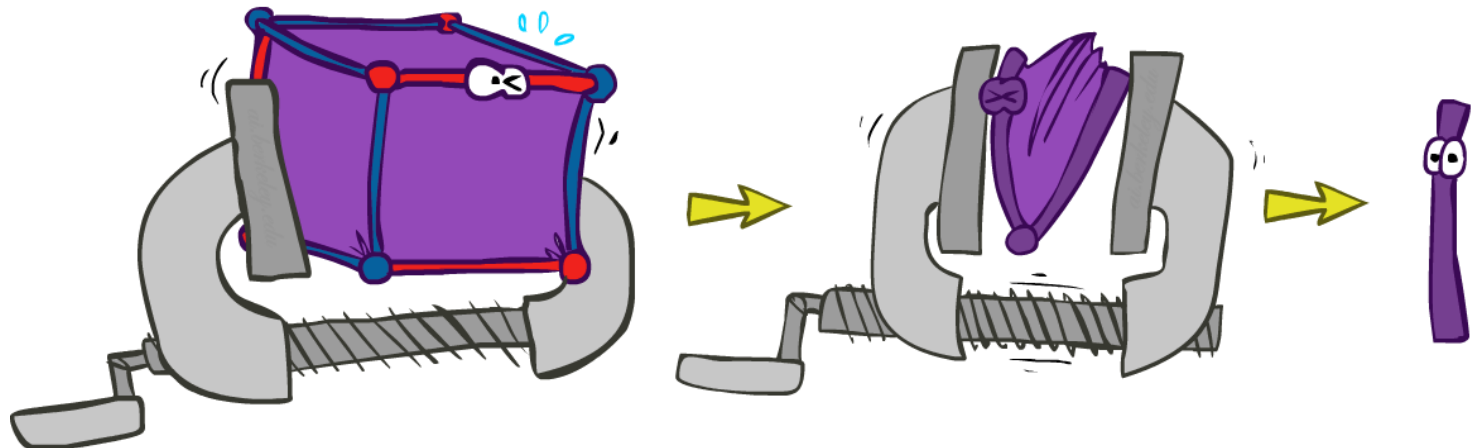
$T, L$		
+t	+l	0.051
+t	-l	0.119
-t	+l	0.083
-t	-l	0.747

Sum out T

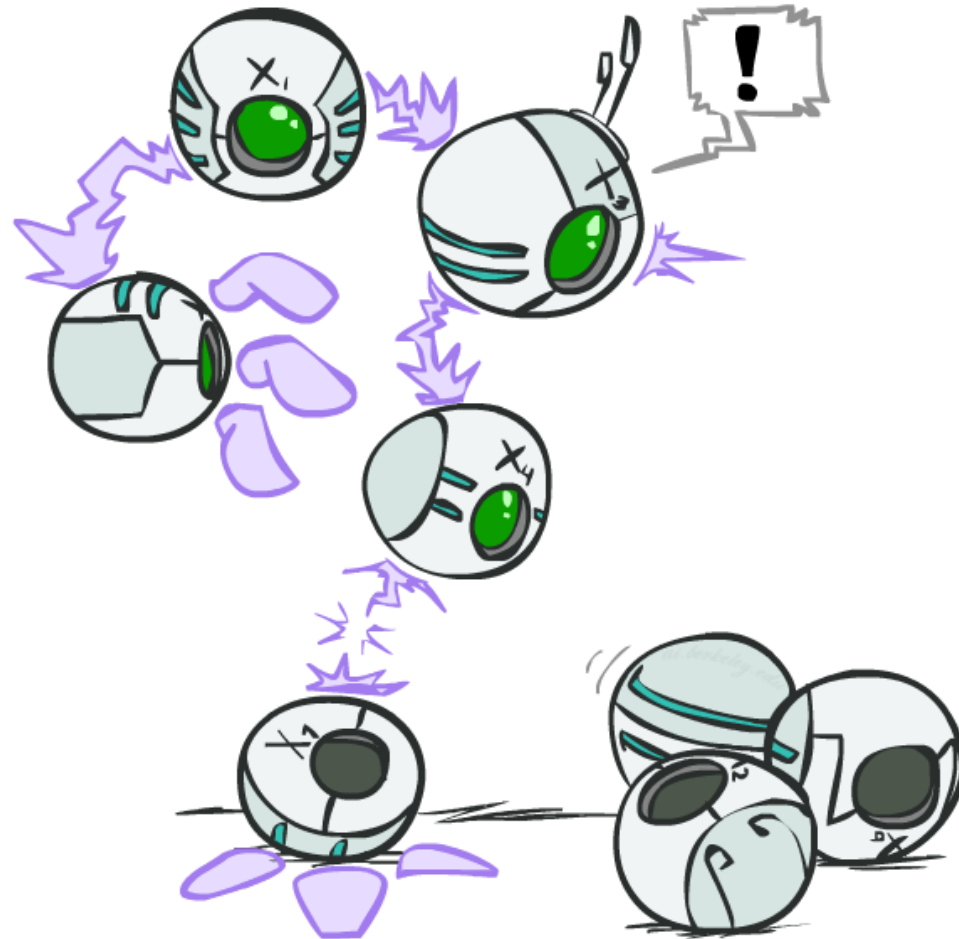
$P(L)$

$L$	
+l	0.134
-l	0.866

A factor of  
exponential  
size!



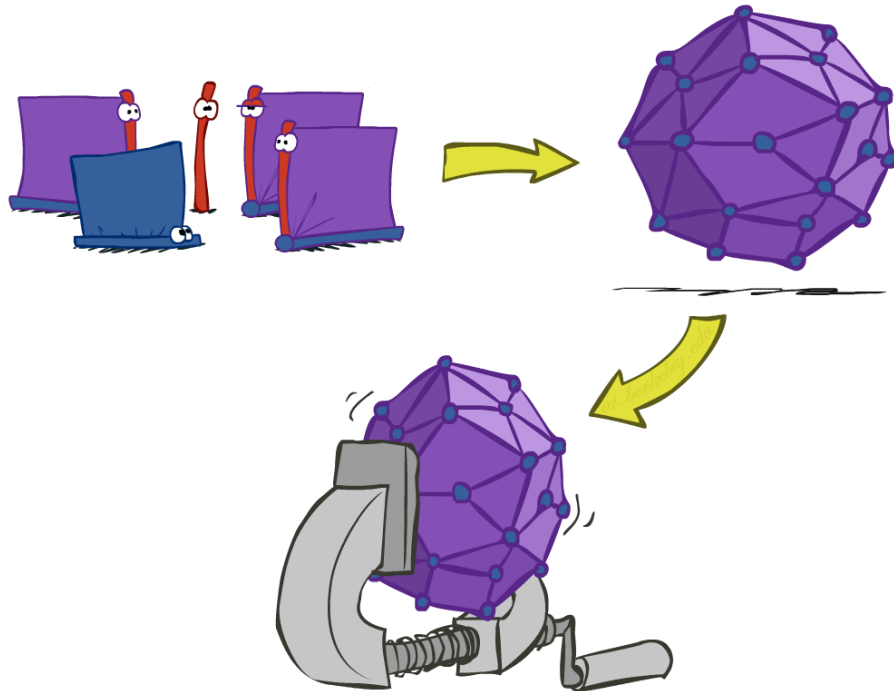
# Variable Elimination



# Inference by Enumeration vs. Variable Elimination

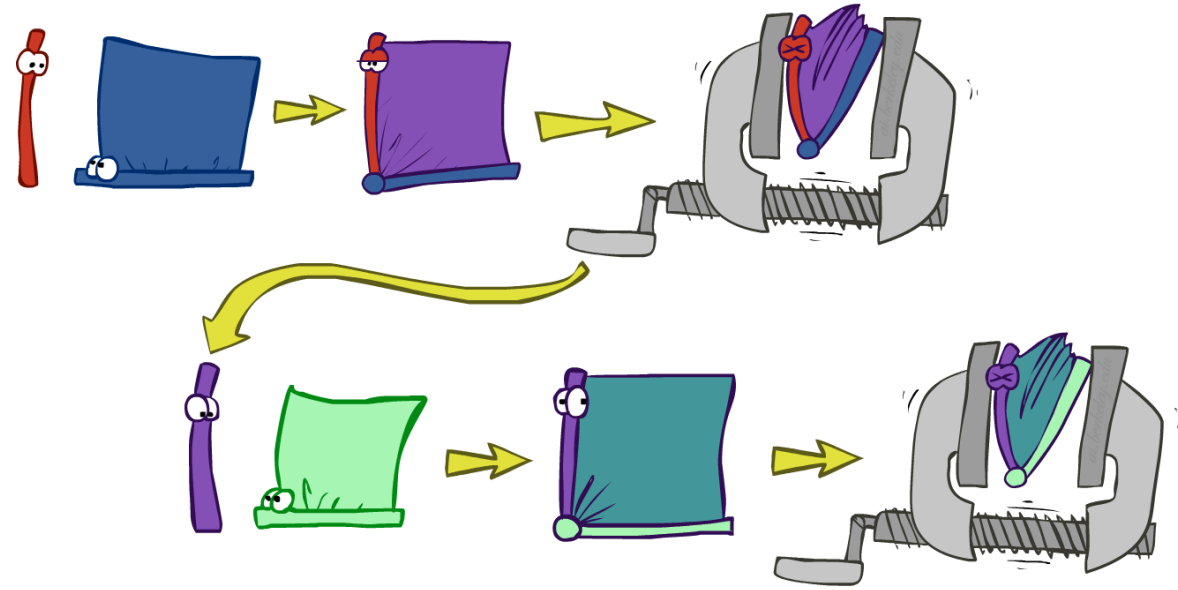
- Why is inference by enumeration so slow?

- You join up the whole joint distribution before you sum out the hidden variables



- Idea: interleave joining and elimination!

- Called “Variable Elimination”
- Still NP-hard, but usually much faster than inference by enumeration

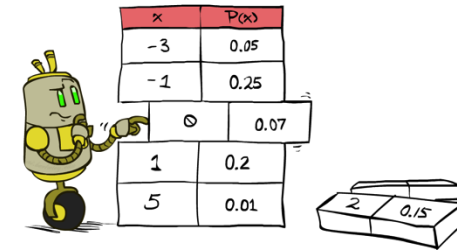


# Variable Elimination

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

- Start with initial factors:

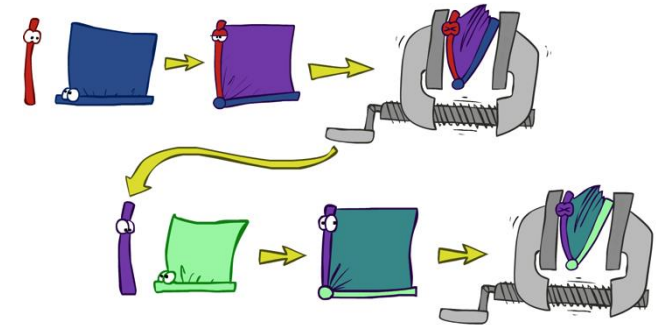
- Local CPTs (but instantiated by evidence)



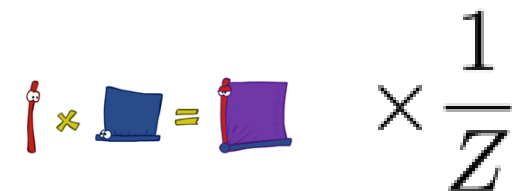
x	P(x)
-3	0.05
-1	0.25
0	0.07
1	0.2
5	0.01

- While there are still hidden variables (not Q or evidence):

- Pick a hidden variable H
  - Join all factors mentioning H
  - Eliminate (sum out) H

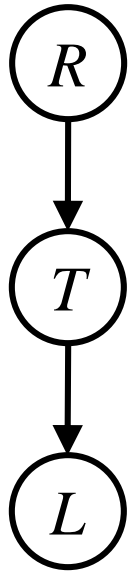


- Join all remaining factors and normalize



$$\text{red stick figure} \times \text{blue square} = \text{purple square} \times \frac{1}{Z}$$

# Traffic Domain



$$P(L) = ?$$

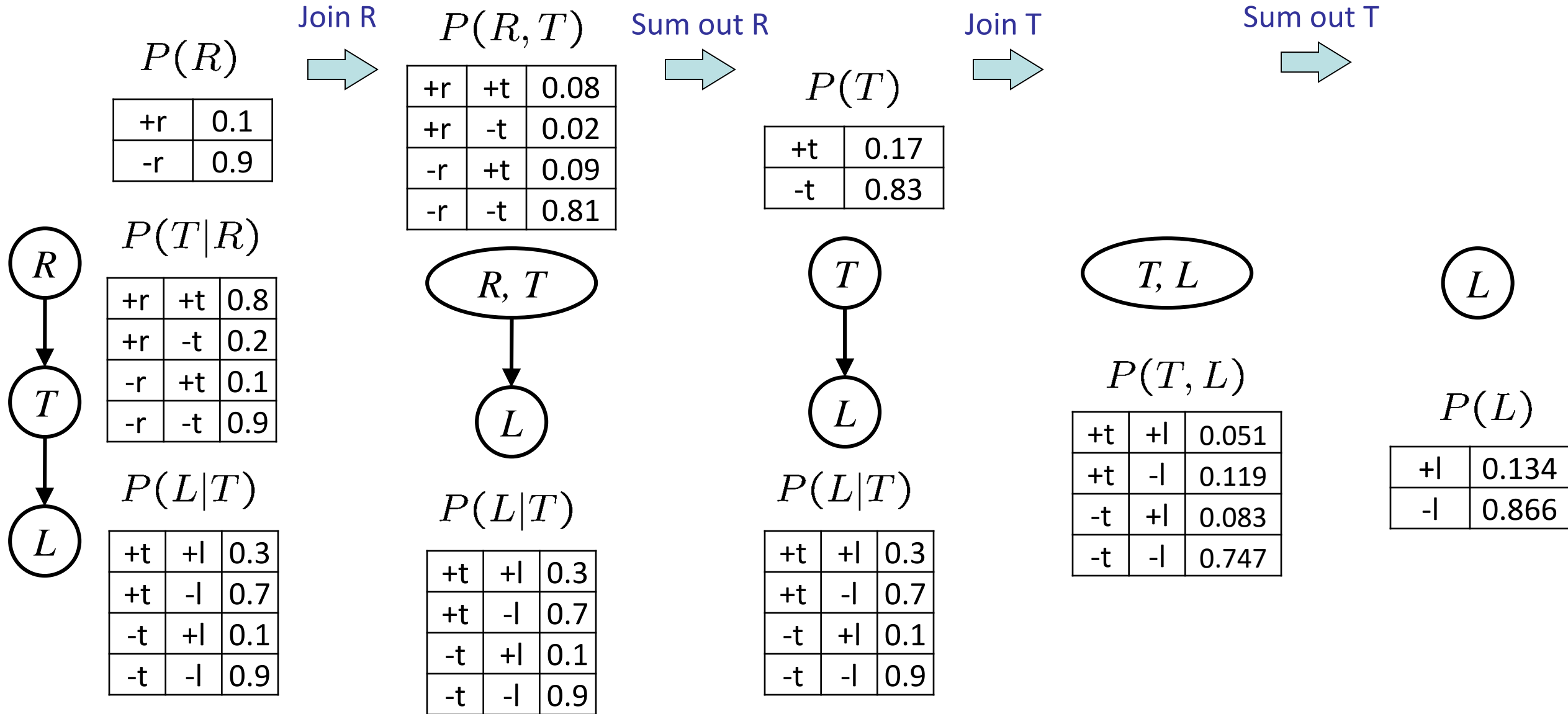
## ■ Inference by Enumeration

$$= \sum_t \sum_r \underbrace{P(L|t)P(r)P(t|r)}_{\text{Join on } r}$$
$$\underbrace{\hspace{10em}}_{\text{Join on } t}$$
$$\underbrace{\hspace{10em}}_{\text{Eliminate } r}$$
$$\underbrace{\hspace{10em}}_{\text{Eliminate } t}$$

## ■ Variable Elimination

$$= \sum_t P(L|t) \underbrace{\sum_r P(r)P(t|r)}_{\text{Join on } r}$$
$$\underbrace{\hspace{10em}}_{\text{Eliminate } r}$$
$$\underbrace{\hspace{10em}}_{\text{Join on } t}$$
$$\underbrace{\hspace{10em}}_{\text{Eliminate } t}$$

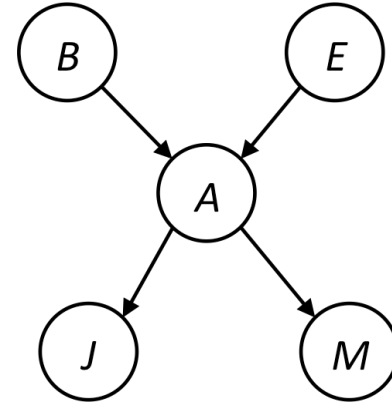
# Variable Elimination



# Example

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

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

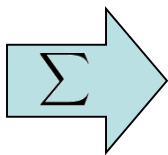
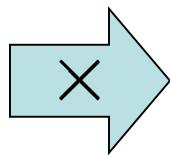


Choose A

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

$$P(j|A)$$

$$P(m|A)$$



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

$P(B)$	$P(E)$	$P(j, m B, E)$
--------	--------	----------------



# Example

$P(B)$	$P(E)$	$P(j, m B, E)$
--------	--------	----------------

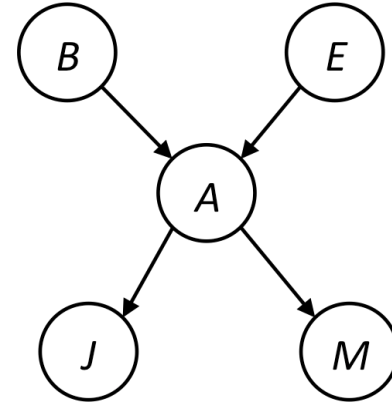
Choose E

$$\begin{array}{l} P(E) \\ P(j, m|B, E) \end{array} \xrightarrow{\times} \xrightarrow{\Sigma} P(j, m|B)$$

$P(B)$	$P(j, m B)$
--------	-------------

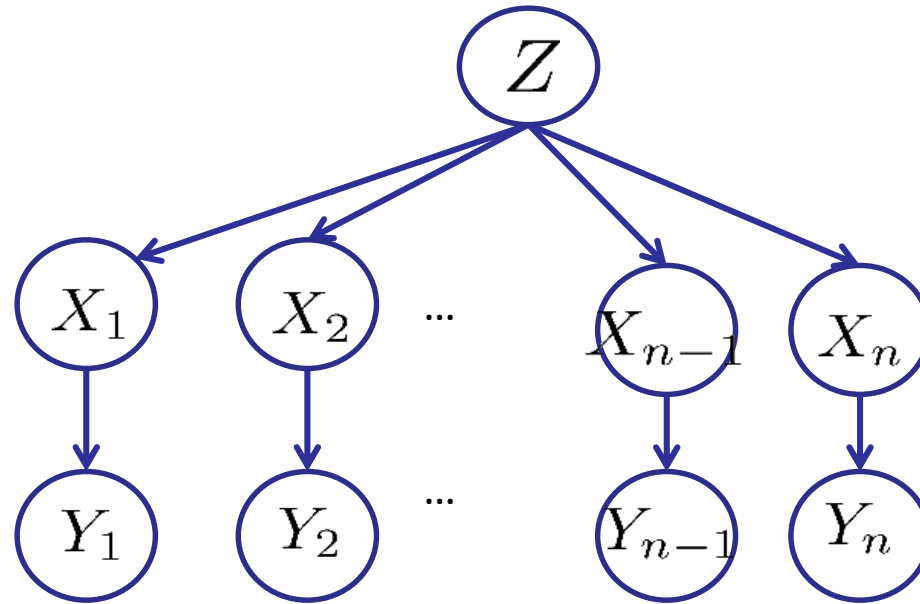
Finish with B

$$\begin{array}{l} P(B) \\ P(j, m|B) \end{array} \xrightarrow{\times} P(j, m, B) \xrightarrow{\text{Normalize}} P(B|j, m)$$



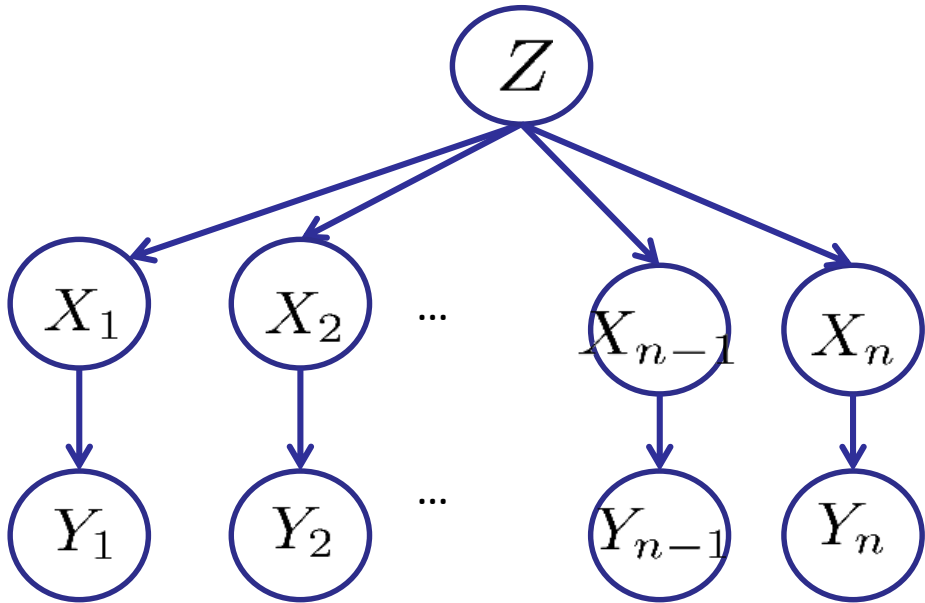
# Variable Elimination Ordering

- Query:  $P(X_n | y_1, \dots, y_n)$
- Two different orderings:  $Z, X_1, \dots, X_{n-1}$  and  $X_1, \dots, X_{n-1}, Z$ .
- What is the size of the maximum factor generated for each of the orderings?



# Variable Elimination Ordering

- $Z, X_1, \dots, X_{n-1}$



$$P(Z)P(X_1|Z)P(X_2|Z), \dots, P(X_n|Z)$$



$$f_1(X_1, X_2, \dots, X_n)$$



$$f_2(y_1, X_2, \dots, X_n)$$



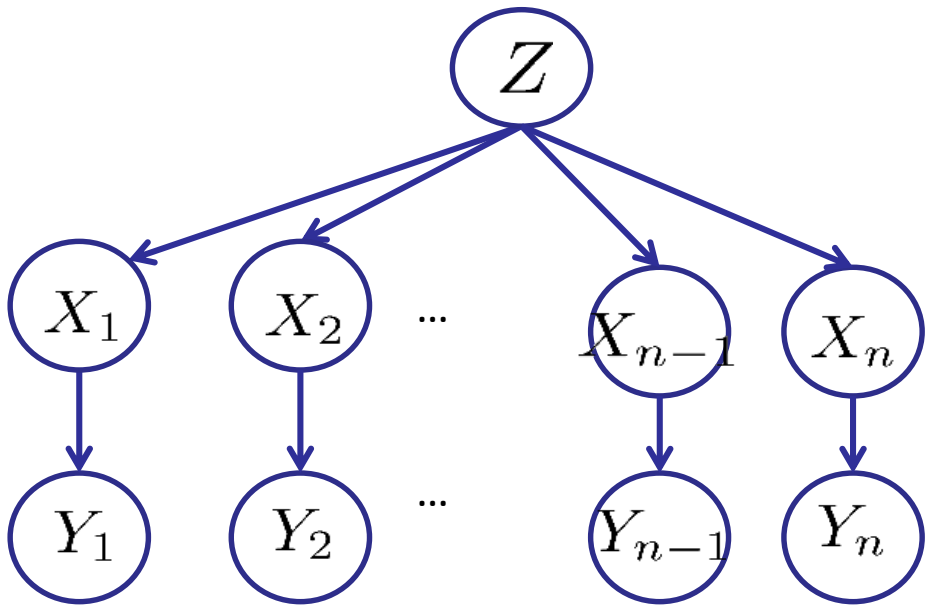
$$f_3(y_1, y_2, \dots, X_n)$$



$$2^{n+1}$$

# Variable Elimination Ordering

- $X_1, \dots, X_{n-1}, Z$



$$P(X_1|Z)P(y_1|X_1)$$



$$f_1(Z, y_1)$$



$$f_1(Z, y_1), f_2(Z, y_2), \dots, f_{n-1}(Z, y_{n-1}), P(Z), P(X_n|Z)$$



$$f_n(X_n, y_1, \dots, y_{n-1})$$



$$2^2$$

# VE: Computational Complexity

---

- The size of the largest factor determines the time and space complexity of VE
- The elimination ordering can greatly affect the size of the largest factor.
  - E.g., previous slide's example  $2^{n+1}$  vs.  $2^2$
- Does there always exist an ordering that only results in small factors?
  - No!

# Reduction from 3SAT

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

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

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

...

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

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

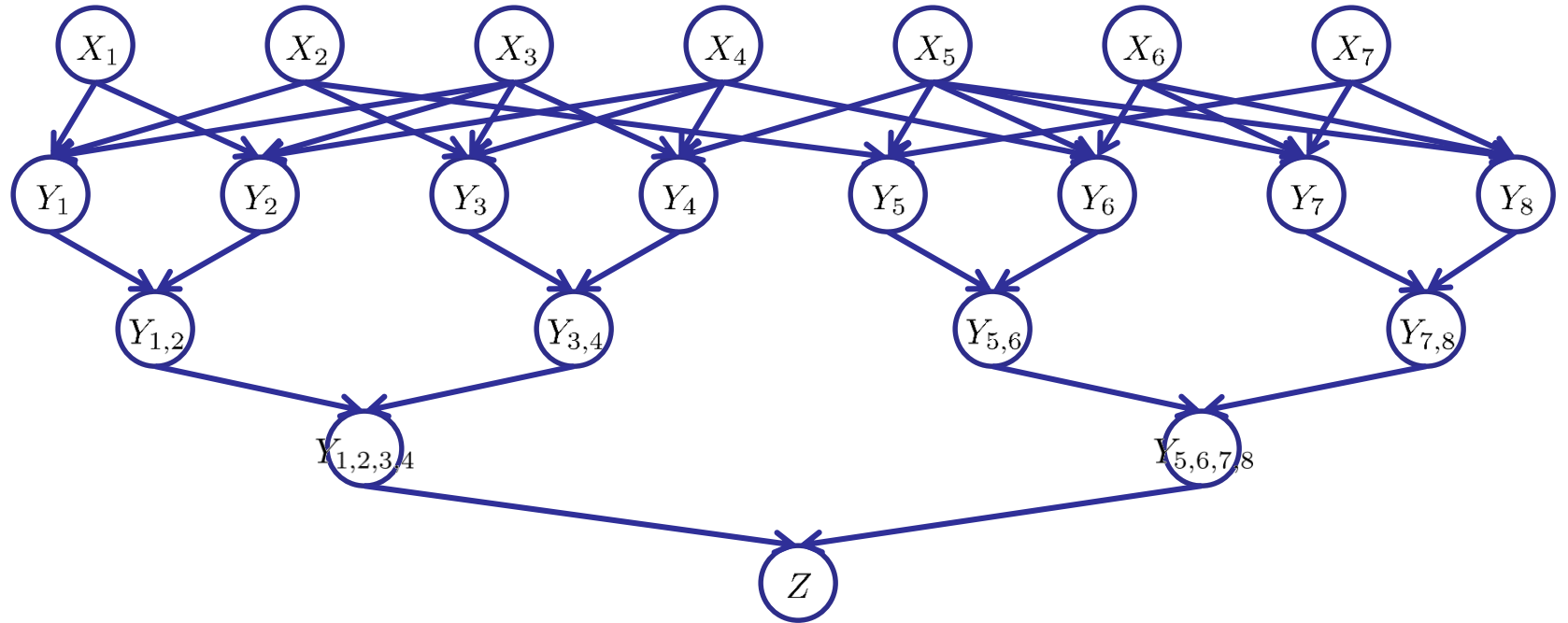
...

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

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

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

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



$$P(+z) = \sum_{\mathbf{x}, \mathbf{y}} P(\mathbf{x}, \mathbf{y}, +z) = \sum_{\mathbf{x} \text{ s.t. } z=T} P(\mathbf{x})$$

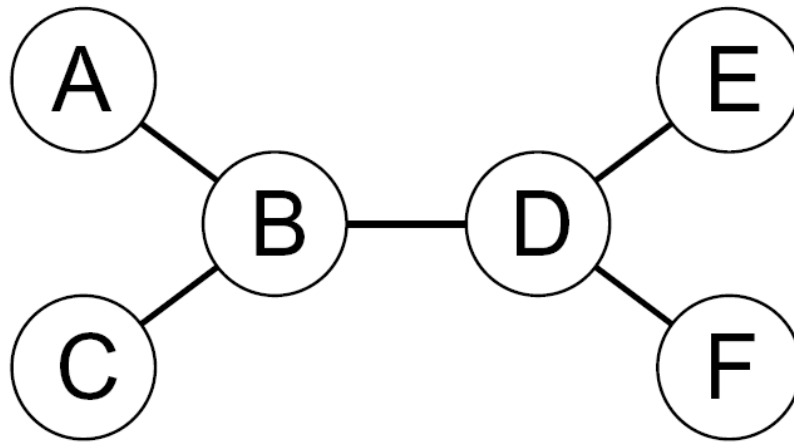
- $P(+z) > 0$  iff the sentence is satisfiable  $\rightarrow$  NP-hard
- $P(+z) = S \times 0.5^7$  where  $S$  is the number of satisfying assignments  $\rightarrow$  #P-hard

# When do we have tractable inference?

---

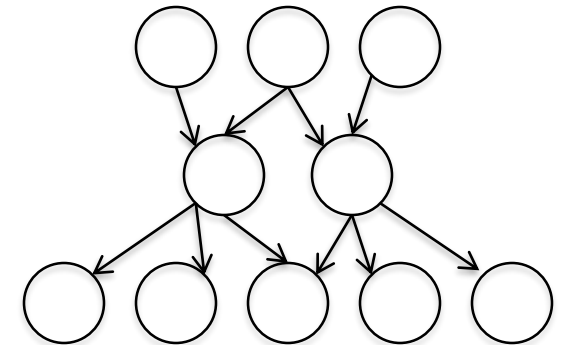
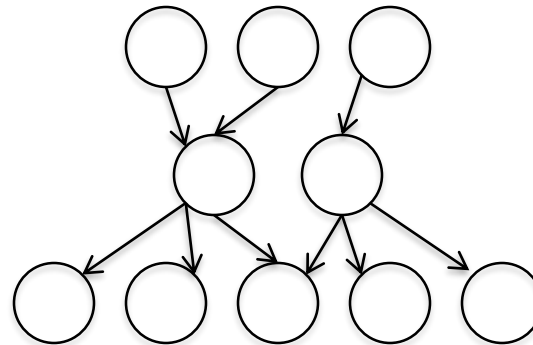
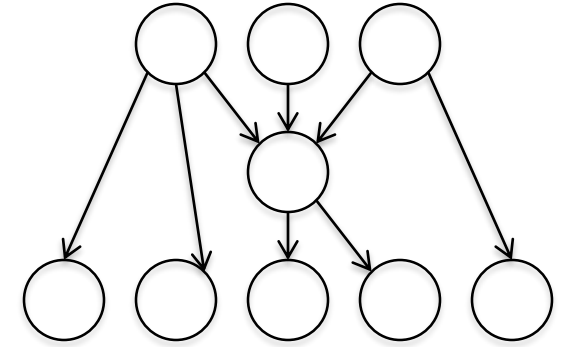
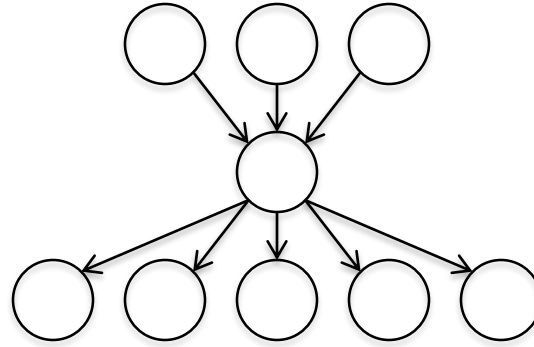
- Recall: Tree-Structured CSPs

- CSP is NP-hard in general
- If the constraint graph has no loops (i.e., tree), the CSP can be solved in linear time!



# Polytrees

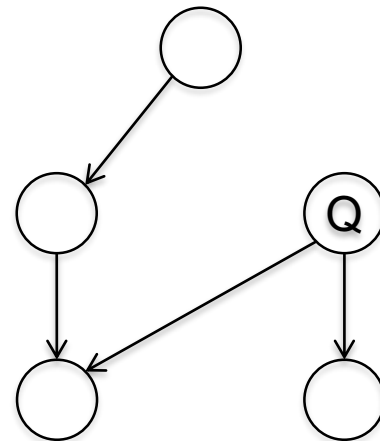
- A polytree is a directed graph with no undirected cycles



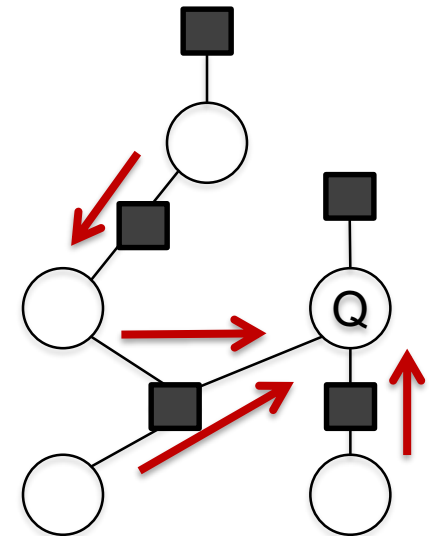


# Variable Elimination on Polytrees

- For poly-tree BNs, the complexity of VE is *linear in the BN size* (number of CPT entries) with the following elimination ordering:
  - Convert to a factor graph
  - Take Q as the root
  - Eliminate from the leaves towards the root



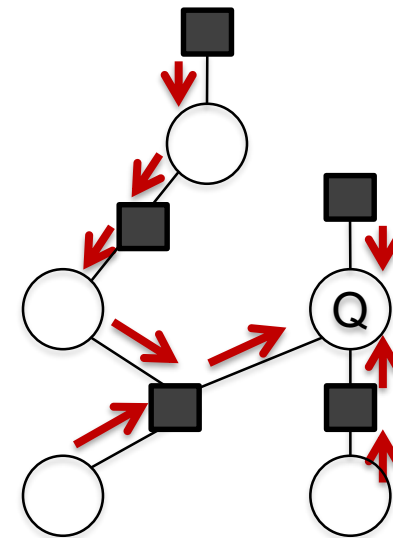
Bayesian Network



Factor Graph

# Variable Elimination on Polytrees

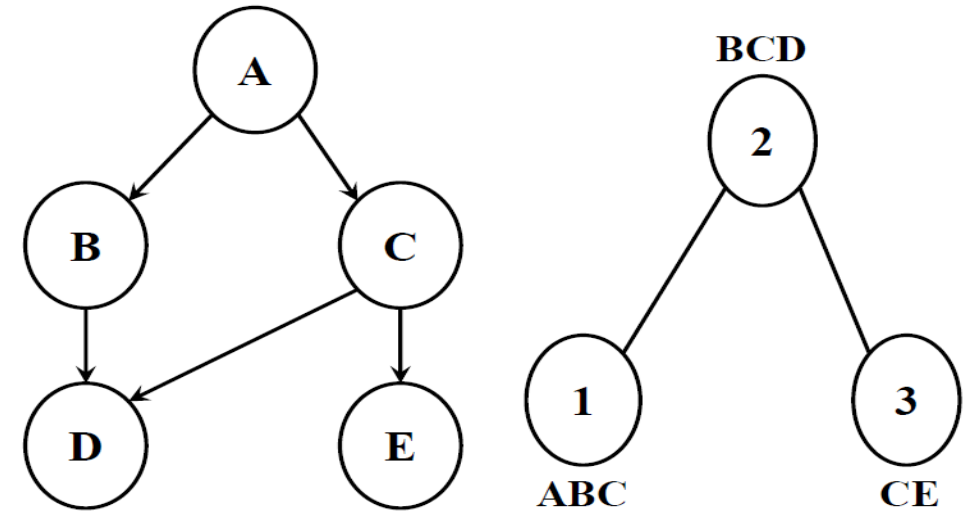
- VE for poly-tree BNs is equivalent to
  - Sum-product message passing algorithm or belief propagation algorithm (i.e., passing messages/beliefs from leaf nodes to the root node)
  - “Messages” are just 1d factors resulted from joining/elimination



# Message Passing on General Graphs

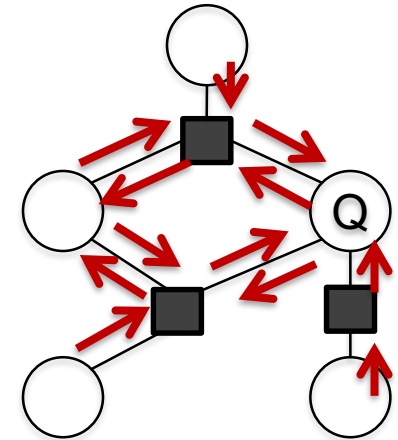
- Exact inference: Junction Tree Algorithm

- Group individual nodes to form cluster nodes in such a way that the resulting network is a polytree (called a **junction tree** or **join tree**)
- Run a sum-product-like algorithm on the junction tree.
- Intractable* on graphs with large cluster nodes (i.e., large **tree-width**).



# Message Passing on General Graphs

- Approximate inference: Loopy Belief Propagation
  - Simply pass the messages on the general graph
    - Will not terminate with loops
    - Run until convergence (not guaranteed!)
  - *Approximate* but *tractable* for large graphs.
  - Sometime works well, sometimes not at all.



# Summary

---

- Exact inference of Bayesian networks
  - Enumeration
    - exponential complexity
  - Variable Eliminating
    - worst-case exponential complexity, often better
  - VE on polytrees
    - linear complexity
    - = message passing
  - Message passing on general graphs
    - Junction Tree Algorithm
    - Loopy Belief Propagation: no longer exact