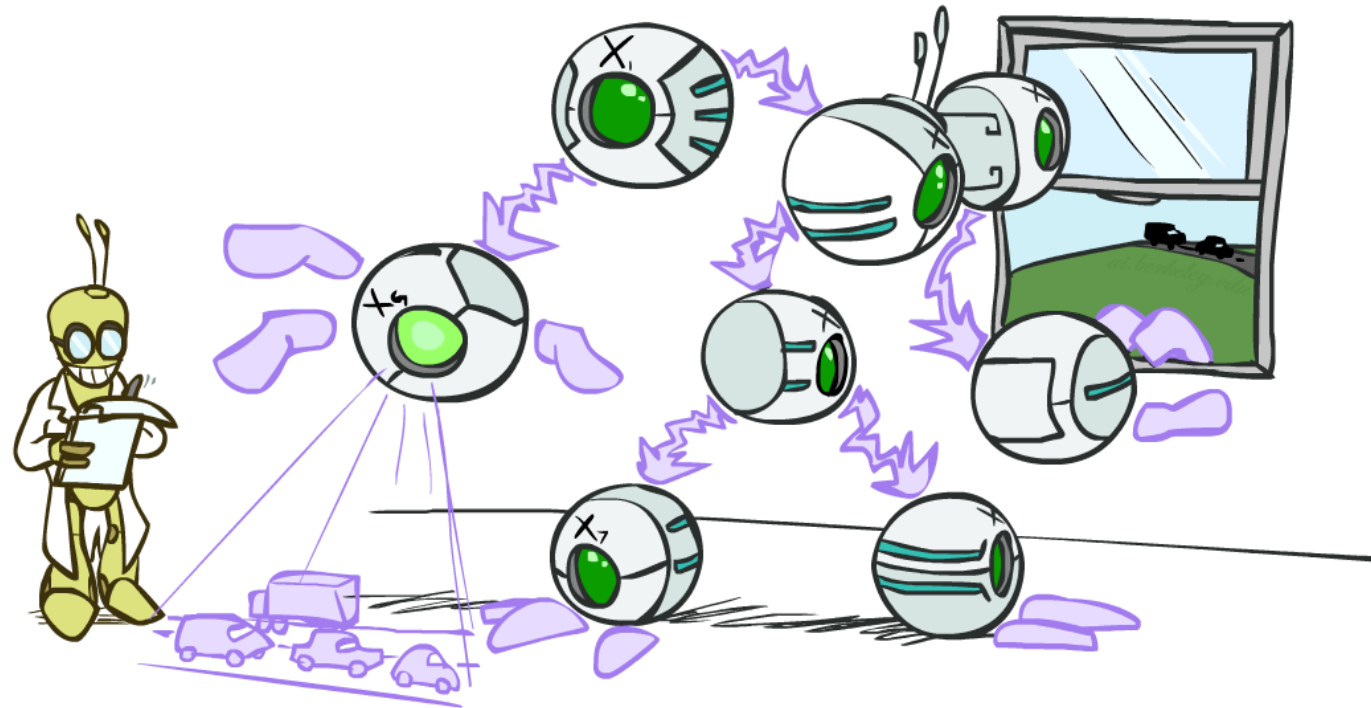


# CS 188: Artificial Intelligence

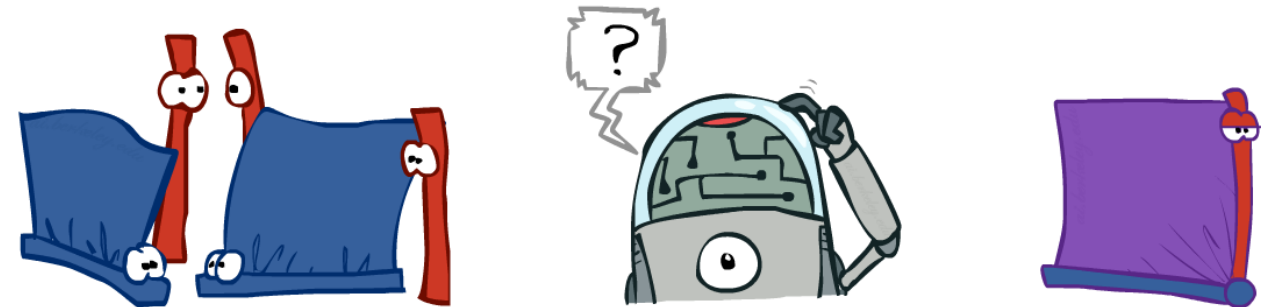
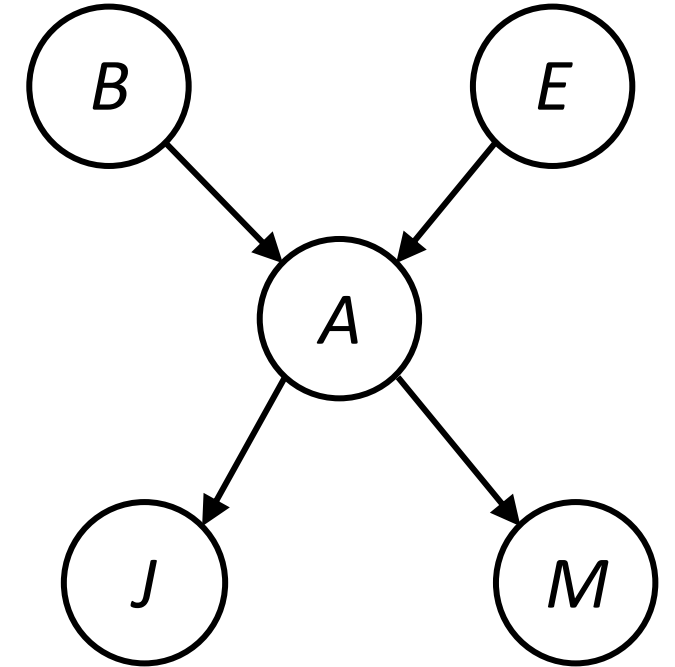
## Bayes Nets: Exact Inference



Instructor: Dan Klein and Stuart Russell

# Inference by Enumeration in Bayes Net

- Reminder of inference by enumeration:
  - Any probability of interest can be computed by summing entries from the joint distribution:  $P(Q | e) = \alpha \sum_h P(Q, h, e)$
  - Entries from the joint distribution can be obtained from a BN by multiplying the corresponding conditional probabilities
- $P(B | j, m) = \alpha \sum_{e,a} P(B, e, a, j, m)$
- $= \alpha \sum_{e,a} P(B) P(e) P(a | B, e) P(j | a) P(m | a)$
- So inference in Bayes nets means computing sums of products of numbers: sounds easy!!
- Problem: sums of **exponentially many** products!



# Can we do better?

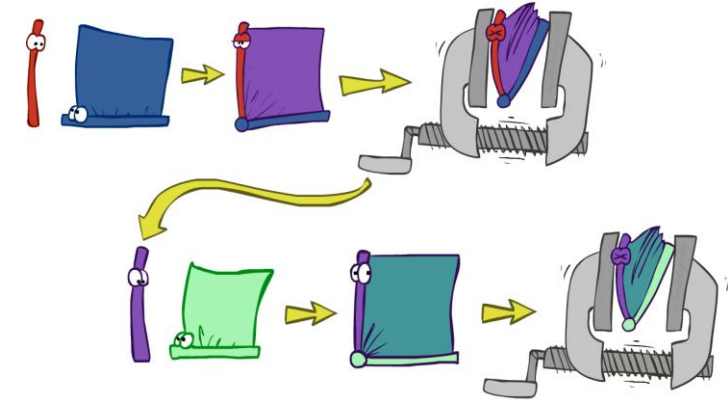
- Consider  $uwy + uwz + uxy + uxz + vwy + vwz + vxy + vxz$ 
  - 16 multiplies, 7 adds
  - Lots of repeated subexpressions!
- Rewrite as  $(u+v)(w+x)(y+z)$ 
  - 2 multiplies, 3 adds
- $\sum_{e,a} P(B) P(e) P(a | B, e) P(j | a) P(m | a)$
- $= P(B)P(e)P(a | B, e)P(j | a)P(m | a) + P(B)P(\neg e)P(a | B, \neg e)P(j | a)P(m | a)$   
 $+ P(B)P(e)P(\neg a | B, e)P(j | \neg a)P(m | \neg a) + P(B)P(\neg e)P(\neg a | B, \neg e)P(j | \neg a)P(m | \neg a)$ 

Lots of repeated subexpressions!

# Variable elimination: The basic ideas

- Move summations inwards as far as possible

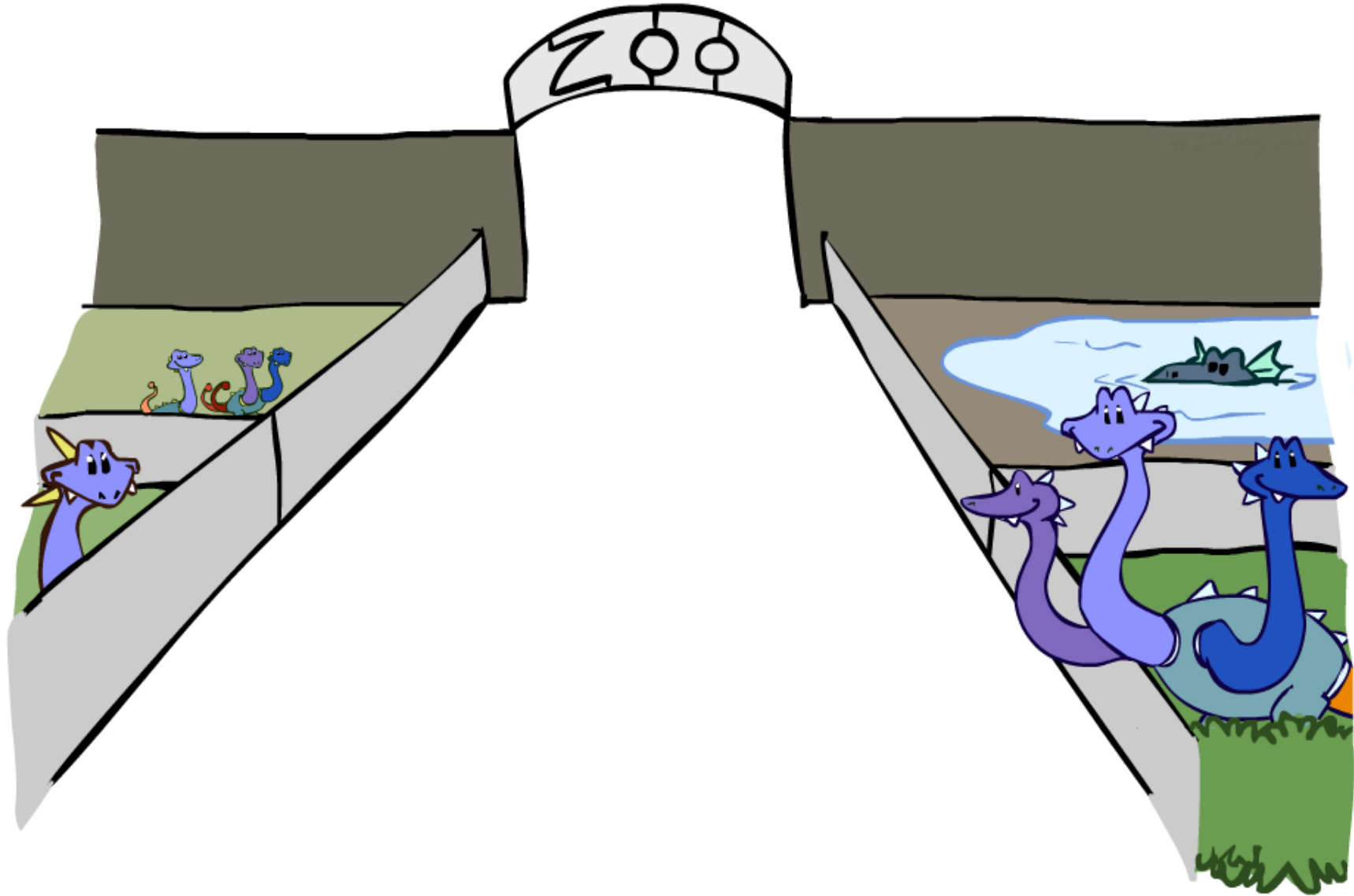
- $P(B | j, m) = \alpha \sum_{e,a} P(B) P(e) P(a|B,e) P(j|a) P(m|a)$
- $= \alpha P(B) \sum_e P(e) \sum_a P(a|B,e) P(j|a) P(m|a)$



- 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  $x, y$
- $|X| \times |Y|$  matrix
- Sums to 1

$P(A,J)$

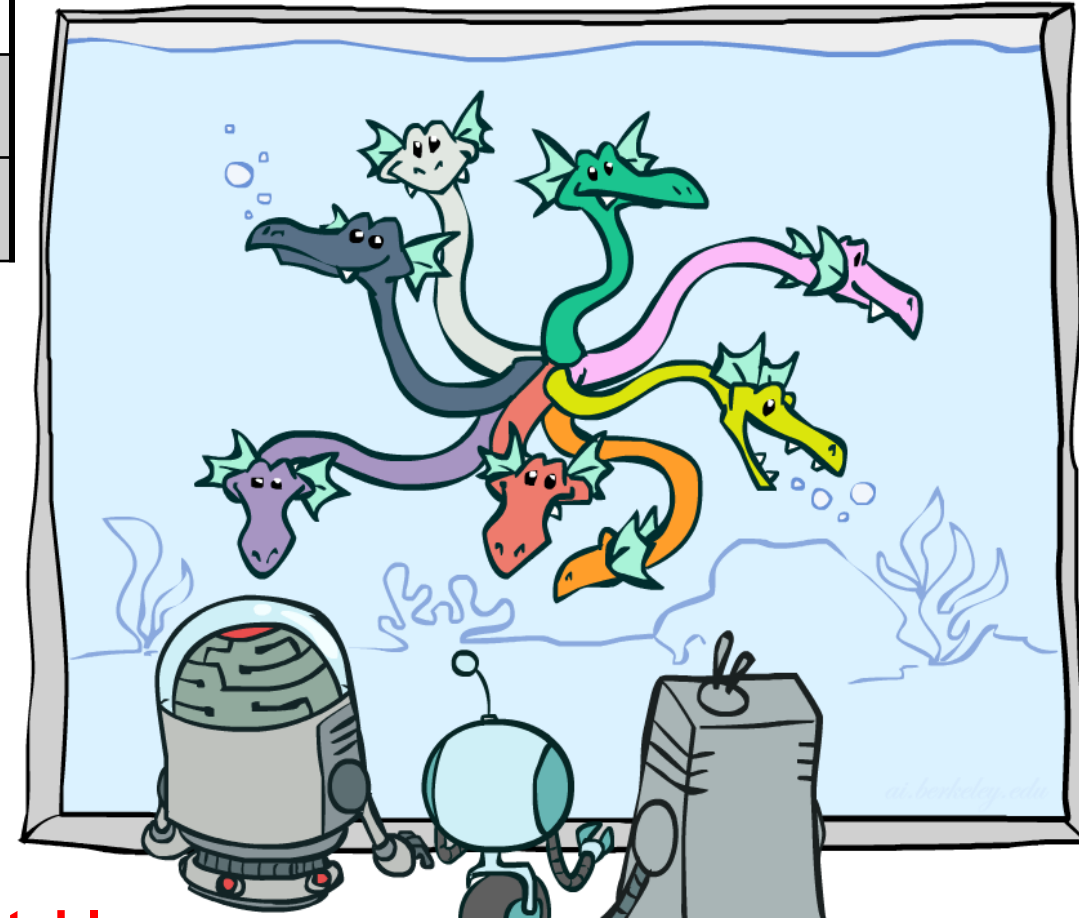
$A \setminus J$	true	false
true	0.09	0.01
false	0.045	0.855

- 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  $P(x)$

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

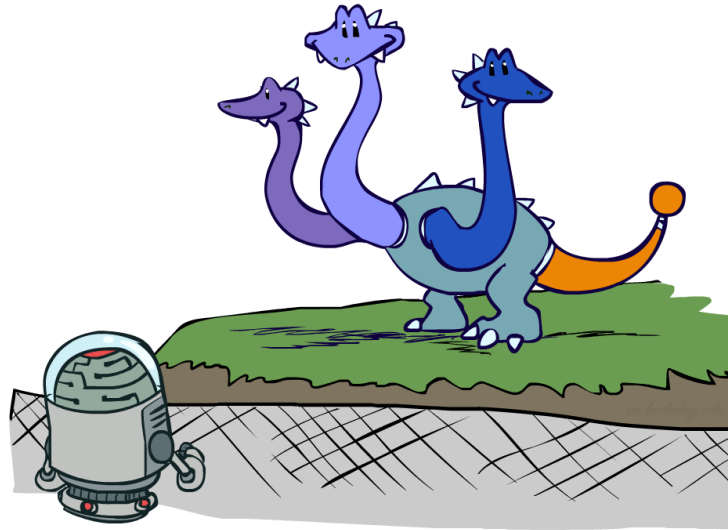
$A \setminus J$	true	false
true	0.09	0.01



**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

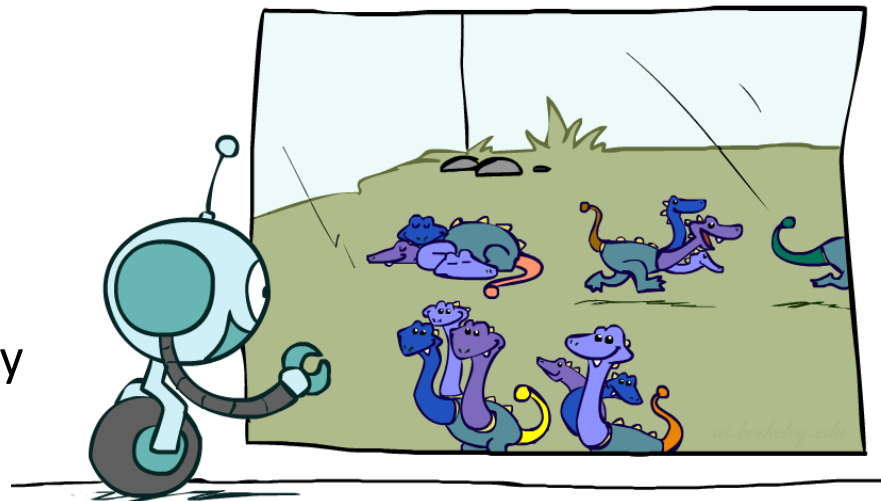


$$P(J|a)$$

$A \setminus J$	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  $|Y|$



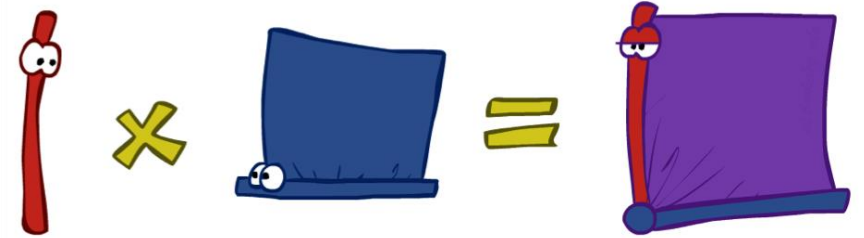
$$P(J|A)$$

$A \setminus J$	true	false
true	0.9	0.1
false	0.05	0.95

} -  $P(J|a)$   
} -  $P(J|\neg a)$

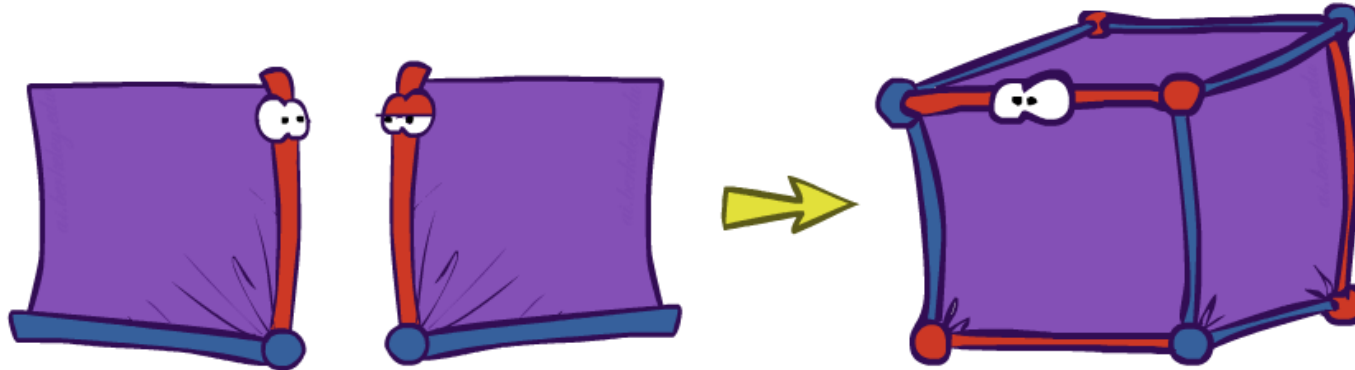
# Operation 1: Pointwise product

- 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:  $P(J|A) \times P(A) = P(A,J)$



$P(A)$			$P(J A)$			$P(A,J)$		
true	0.1	$\times$	A \ J	true	false	A \ J	true	false
false	0.9		true	0.9	0.1	true	0.09	0.01
			false	0.05	0.95	false	0.045	0.855

# Example: Making larger factors



- Example:  $P(A,J) \times P(A,M) = P(A,J,M)$

$P(A,J)$                        $P(A,M)$                        $P(A,J,M)$

A \ J	true	false
true	0.09	0.01
false	0.045	0.855

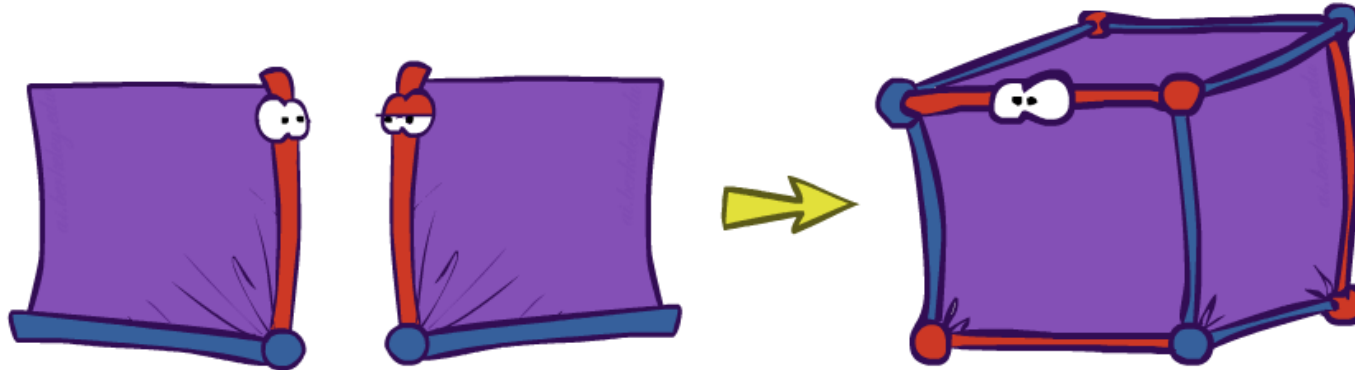
×

A \ M	true	false
true	0.07	0.03
false	0.009	0.891

=

		J \ M	true	false	
J \ M	true	false			
	true				18
	false			.0003	
					A=false
					A=true

# Example: Making larger factors



- Example:  $P(U,V) \times P(V,W) \times P(W,X) = P(U,V,W,X)$
- Sizes:  $[10,10] \times [10,10] \times [10,10] = [10,10,10,10]$
- 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:  $\sum_j P(A,J) = P(A,j) + P(A,\neg j) = P(A)$

$P(A,J)$

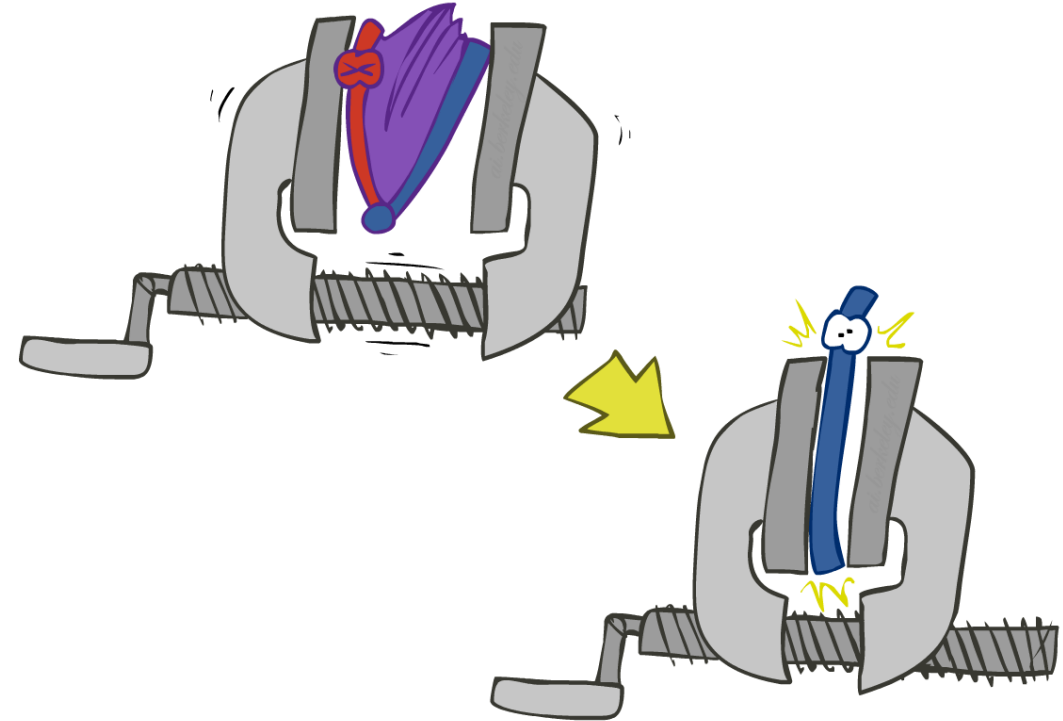
A \ J	true	false
true	0.09	0.01
false	0.045	0.855

Sum out  $J$



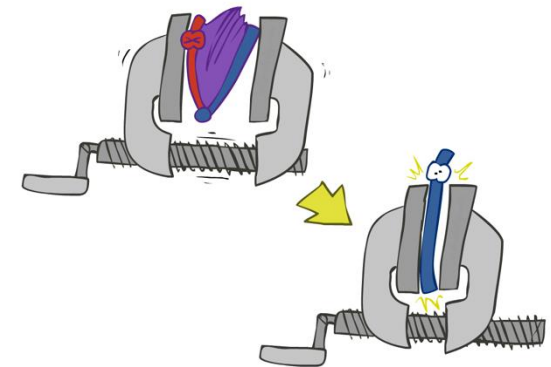
$P(A)$

true	0.1
false	0.9

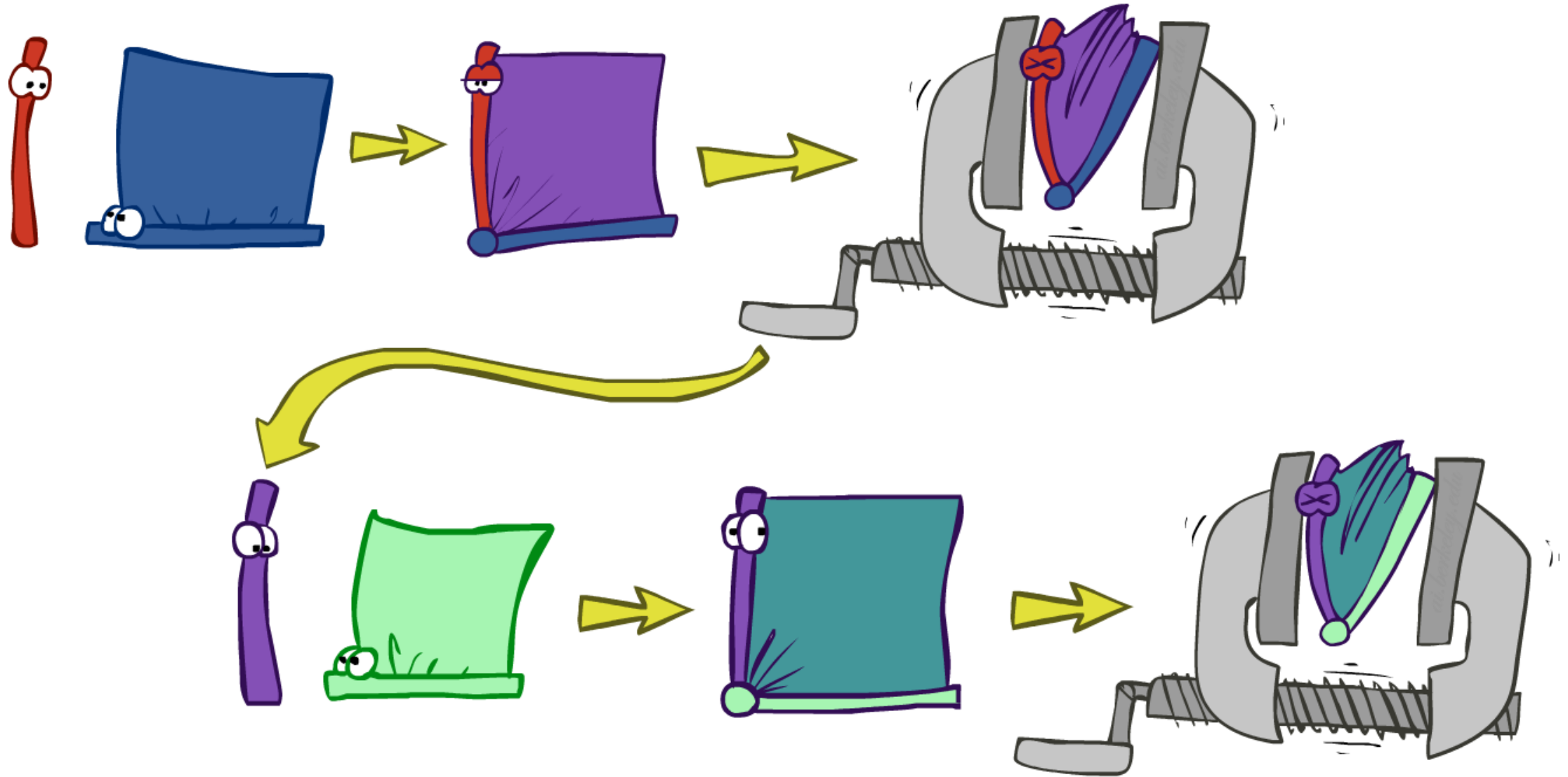


# Summing out from a product of factors

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

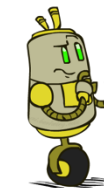


# Variable Elimination

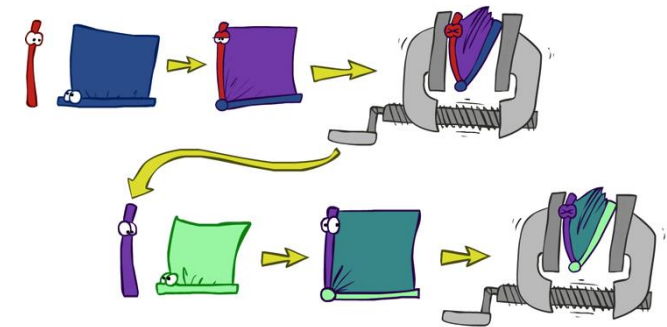



# Variable Elimination

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



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

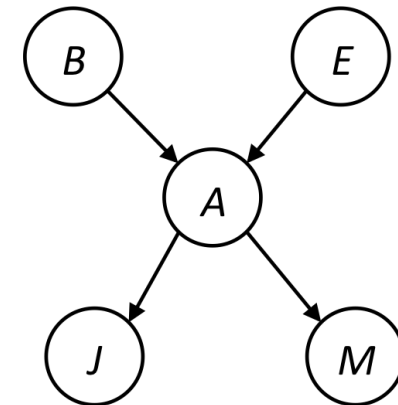



$$\text{stick} \times \text{blue square} = \text{purple square} \times \alpha$$

# Example

Query  $P(B | j,m)$

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



Choose A

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

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

# Example

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

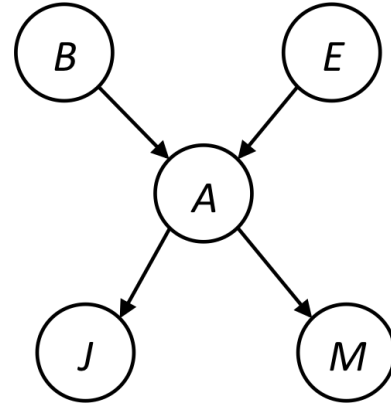
Choose E

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

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

Finish with B

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



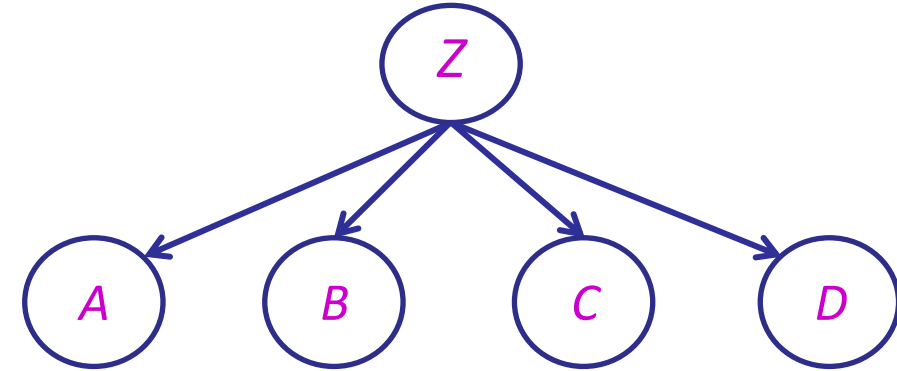
# Order matters

- Order the terms Z, A, B C, D

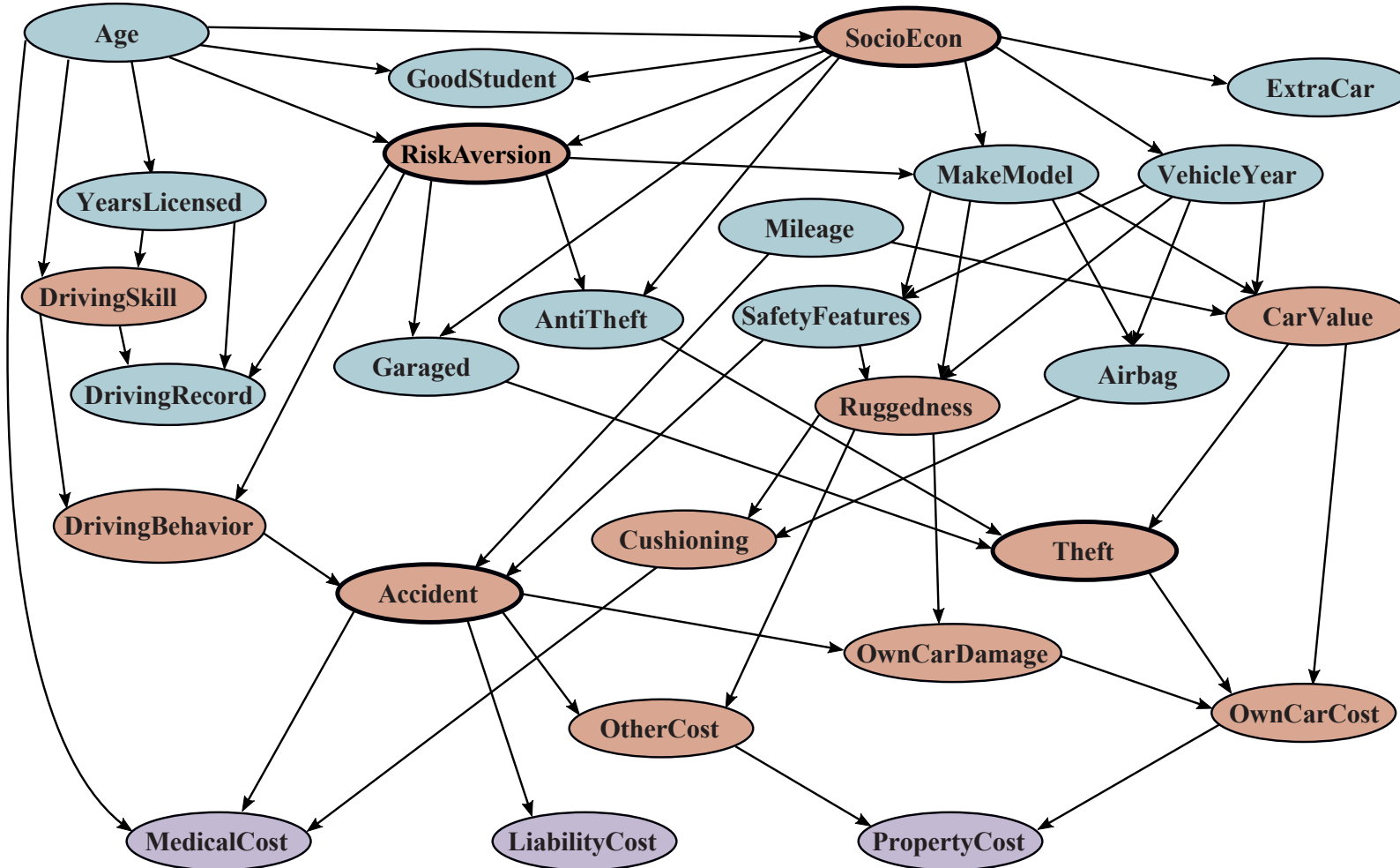
- $P(D) = \alpha \sum_{z,a,b,c} P(z) P(a|z) P(b|z) P(c|z) P(D|z)$
- $= \alpha \sum_z P(z) \sum_a P(a|z) \sum_b P(b|z) \sum_c P(c|z) P(D|z)$
- Largest factor has 2 variables (D,Z)

- Order the terms A, B C, D, Z

- $P(D) = \alpha \sum_{a,b,c,z} P(a|z) P(b|z) P(c|z) P(D|z) P(z)$
- $= \alpha \sum_a \sum_b \sum_c \sum_z P(a|z) P(b|z) P(c|z) P(D|z) P(z)$
- Largest factor has 4 variables (A,B,C,D)
- In general, with  $n$  leaves, factor of size  $2^n$



# Example Bayes' Net: Car Insurance



Enumeration: **227M** operations

Elimination: **221K** operations

# Computational and Space Complexity

---

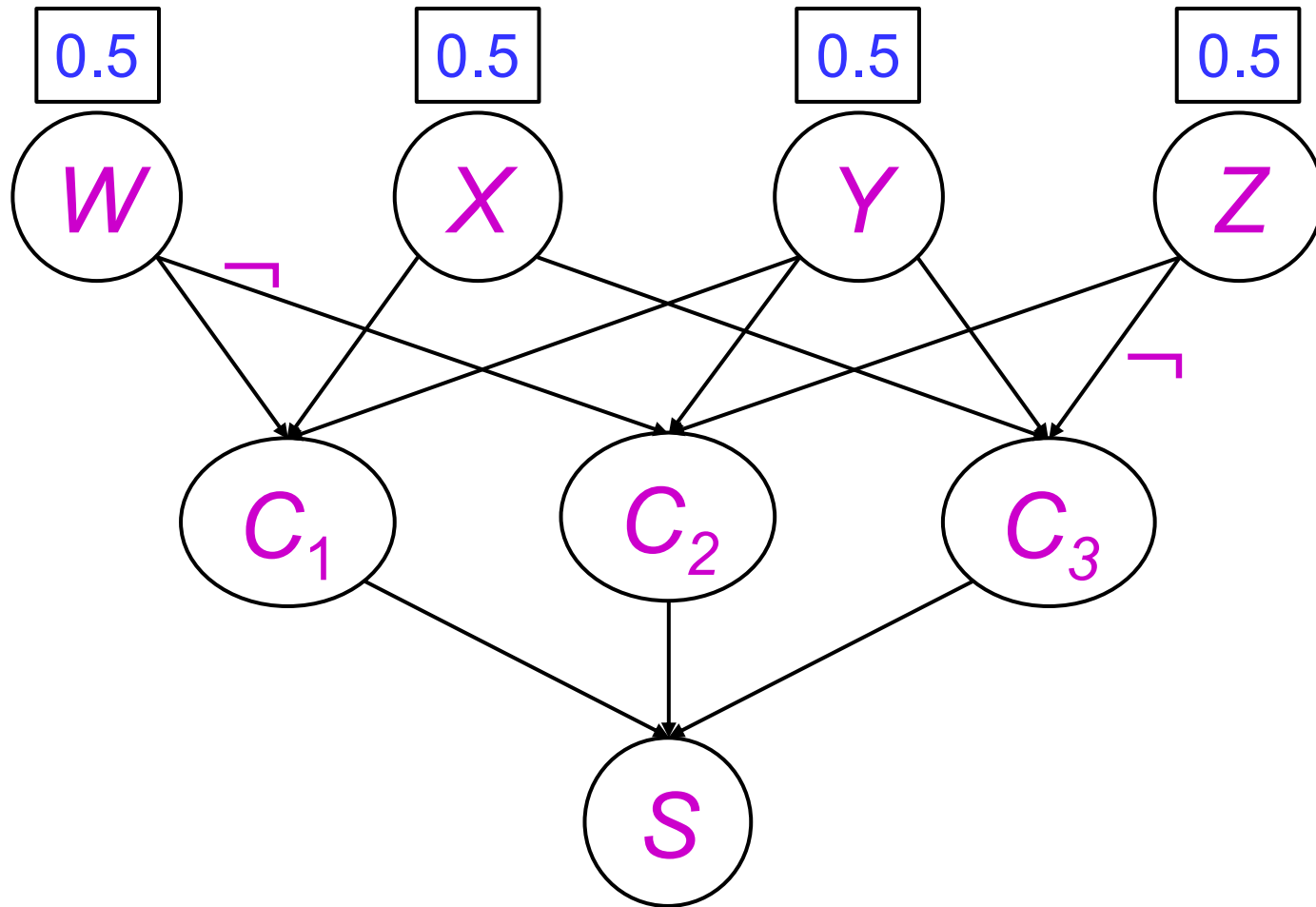
- The computational and space complexity of variable elimination is determined by the largest factor (and it's space that kills you)
- The elimination ordering can greatly affect the size of the largest factor.
  - E.g., ZABCD example  $2^n$  vs. 2
- Does there always exist an ordering that only results in small factors?
  - **No!**

# Worst Case Complexity? Reduction from SAT

---

- SAT (Boolean satisfiability): is there an assignment of true/false to the Boolean variables that makes the given sentence true?
  - E.g., values for  $W, X, Y, Z$  to satisfy  $(W \vee X \vee Y) \wedge (Y \vee Z \vee \neg W) \wedge (X \vee Y \vee \neg Z)$
- Given any Boolean sentence  $S$ , can we construct a Bayes net  $B_S$  such that inference in  $B_S$  gives the answer for satisfiability of  $S$ ?
  - If so, then inference in Bayes nets must be NP-hard

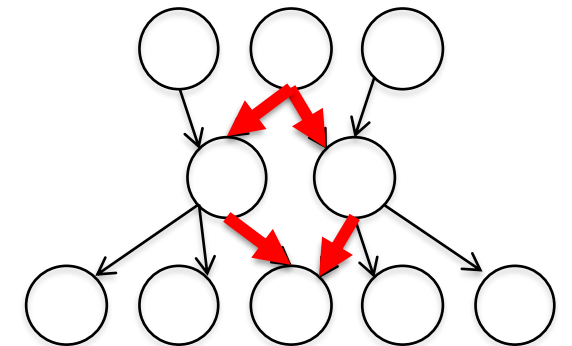
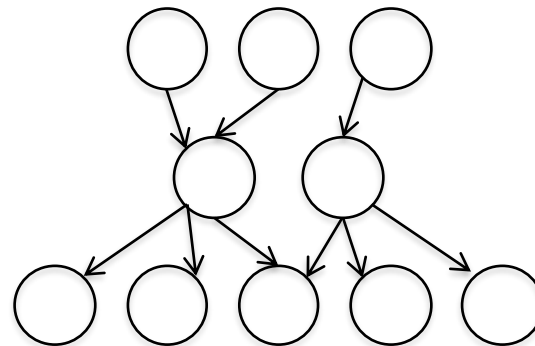
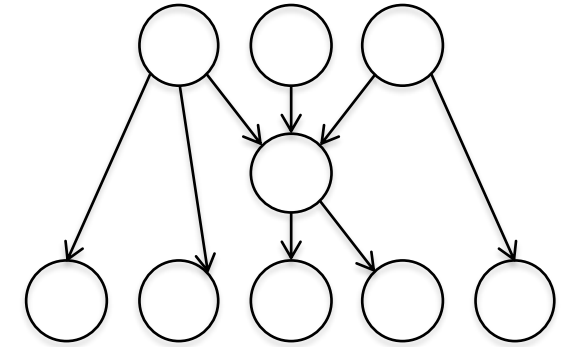
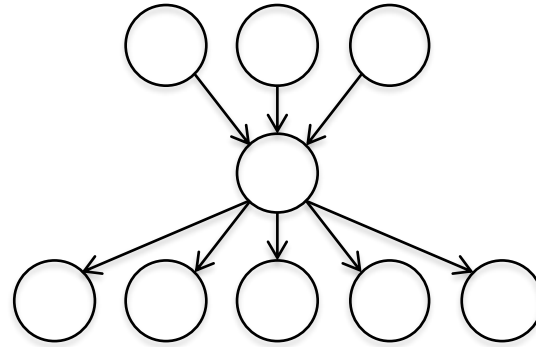
# Worst Case Complexity? Reduction from SAT



- Variables:  $W, X, Y, Z$
- CNF clauses:
  1.  $C_1 = W \vee X \vee Y$
  2.  $C_2 = Y \vee Z \vee \neg W$
  3.  $C_3 = X \vee Y \vee \neg Z$
- Sentence  $S = C_1 \wedge C_2 \wedge C_3$
- $P(S) > 0$  iff  $S$  is satisfiable
  - $\Rightarrow$  **NP-hard**
- $P(S) = K \times 0.5^n$  where  $K$  is the number of satisfying assignments for clauses
  - $\Rightarrow$  **#P-hard**

# Polytrees

- A polytree is a directed graph with no undirected cycles
- For poly-trees the complexity of variable elimination is **linear in the network size** if you eliminate from the leaves towards the roots



# Summary

- Exact inference = sums of products of conditional probabilities from the network
- Enumeration is always exponential
- Variable elimination reduces this by avoiding the recomputation of repeated subexpressions
  - Massive speedups in practice
  - Linear time for polytrees
- Exact inference is #P-hard
- Next: approximate inference

