Probability and Structure in Natural Language Processing

Noah Smith

Heidelberg University, November 2014

Introduction

Motivation

- Statistical methods in NLP arrived ~20 years ago and now dominate.
- Mercer was right: "There's no data like more data."
  - And there's more and more data.

- Lots of new applications and new statistical techniques – it's formidable to learn and keep up with all of them.

Thesis

- Most of the main ideas are related and similar to each other.
  - Different approaches to decoding.
  - Different learning criteria.
  - Supervised and unsupervised learning.
- Umbrella: probabilistic reasoning about discrete linguistic structures.

- This is good news!

Plan

1.	Graphical models and inference	Monday
2.	Decoding and structures	Tuesday
3.	Supervised learning	Wednesday
4.	Hidden variables	Thursday

Exhortations

- The content is formal, but the style doesn't need to be.
- Ask questions!
  - Help me find the right pace.
  - Lecture 4 can be dropped/reduced if needed.

Lecture 1: Graphical Models and Inference

Random Variables

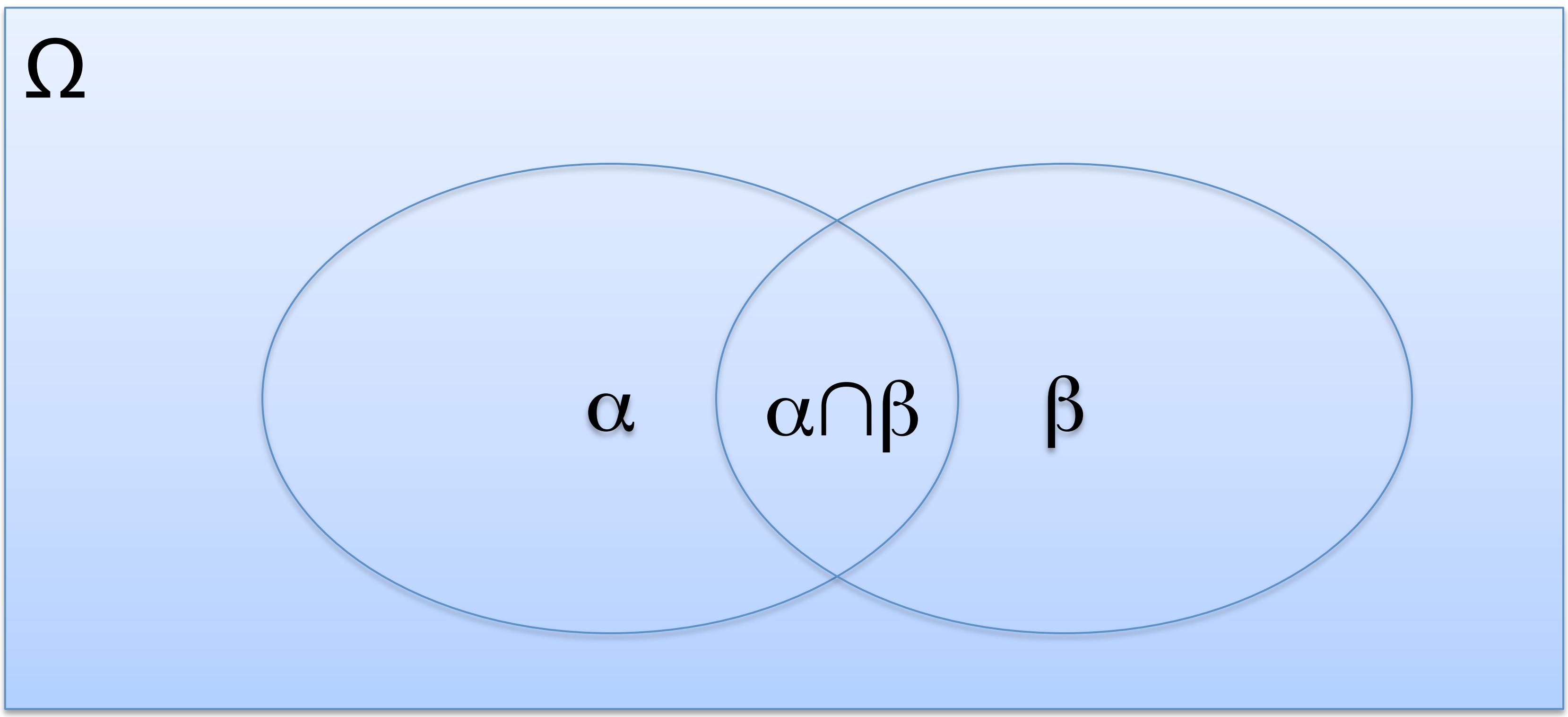
- Probability distributions usually defined by **events**
- Events are complicated!
  - We tend to *group* events by **attributes**
  - Person → Age, Grade, HairColor
- **Random variables** formalize attributes:
  - "Grade = A" is shorthand for event

$$\{\omega \in \Omega : f_{\text{Grade}}(\omega) = A\}$$

- Properties of random variable X:
  - Val(X) = possible values of X
  - For discrete (categorical): $\sum_{x \in \text{Val}(X)} P(X = x) = 1$
  - For continuous: $\int P(X = x)dx = 1$
  - Nonnegativity: $\forall x \in \text{Val}(X), P(X = x) \geq 0$

Conditional Probabilities

- After learning that α is true, how do we feel about β? $P(\beta \mid \alpha)$



Chain Rule

$$P(\alpha \cap \beta) = P(\alpha)P(\beta \mid \alpha)$$

$$P(\alpha_1 \cap \cdots \cap \alpha_k) = P(\alpha_1)P(\alpha_2 \mid \alpha_1) \cdots P(\alpha_k \mid \alpha_1 \cap \ldots \cap \alpha_{k-1})$$

Bayes Rule

likelihood

prior

$$P(\alpha \mid \beta) = \frac{P(\beta \mid \alpha)P(\alpha)}{P(\beta)}$$

posterior

normalization constant

$$P(\alpha \mid \beta \cap \gamma) = \frac{P(\beta \mid \alpha \cap \gamma)P(\alpha \mid \gamma)}{P(\beta \mid \gamma)}$$

γ is an “external event”

Independence

- α and β are **independent** if $P(\beta|\alpha) = P(\beta)$

$$P \rightarrow (\alpha \perp \beta)$$

- **Proposition:** α and β are **independent** if and only if $P(\alpha \cap \beta) = P(\alpha)\, P(\beta)$

Conditional Independence

- Independence is rarely true.

- α and β are **conditionally independent** given γ if
  $P(\beta \mid \alpha \cap \gamma) = P(\beta \mid \gamma)$

$$P \rightarrow (\alpha \perp \beta \mid \gamma)$$

Proposition: $P \rightarrow (\alpha \perp \beta \mid \gamma)$ if and only if

$P(\alpha \cap \beta \mid \gamma) = P(\alpha \mid \gamma)\, P(\beta \mid \gamma)$

Joint Distribution and Marginalization

$$P(\text{Grade}, \text{Intelligence}) =$$

	Intelligence = very high	Intelligence = high
Grade = A	0.70	0.10
Grade = B	0.15	0.05

- Compute the marginal over each individual random variable?

Marginalization: General Case

$$p(X_1 = x) = \sum_{x_2 \in \mathrm{Val}(X_2)} \cdots \sum_{x_n \in \mathrm{Val}(X_n)} P(X_1 = x, X_2 = x_2, \ldots, X_n = x_n)$$

How many terms?

Basic Concepts So Far

- Atomic outcomes: assignment of $x_1,\ldots,x_n$ to $X_1,\ldots, X_n$
- Conditional probability: $P(X, Y) = P(X)\, P(Y|X)$
- Bayes rule: $P(X|Y) = P(Y|X)\, P(X) / P(Y)$
- Chain rule: $P(X_1,\ldots,X_n) = P(X_1)\, P(X_2|X_1)$
  $\ldots\, P(X_k|X_1,\ldots,X_{k-1})$
- Marginals: deriving $P(X = x)$ from $P(X, Y)$

Sets of Variables

- **Sets** of variables **X**, **Y**, **Z**
- **X** is independent of **Y** given **Z** if

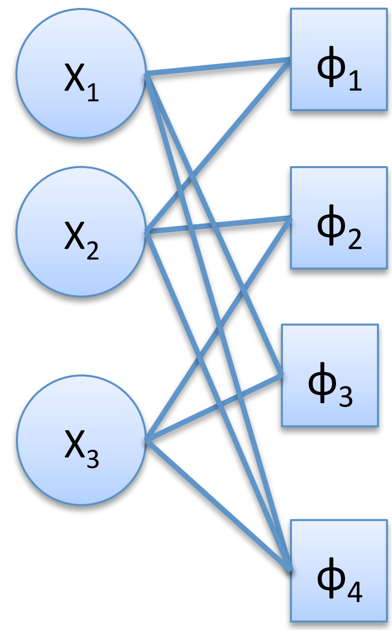
  $P \models (\mathbf{X}=\mathbf{x} \perp \mathbf{Y}=\mathbf{y} | \mathbf{Z}=\mathbf{z})$,
  $\forall\ \mathbf{x} \in Val(\mathbf{X}), \mathbf{y} \in Val(\mathbf{Y}), \mathbf{z} \in Val(\mathbf{Z})$
- Shorthand:
  - Conditional independence: $P \models (\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$
  - For $P \models (\mathbf{X} \perp \mathbf{Y} \mid \emptyset)$, write $P \models (\mathbf{X} \perp \mathbf{Y})$

- **Proposition:** P satisfies $(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$ if and only if $P(\mathbf{X},\mathbf{Y}|\mathbf{Z}) = P(\mathbf{X}|\mathbf{Z})\ P(\mathbf{Y}|\mathbf{Z})$

Factor Graphs

Factor Graphs

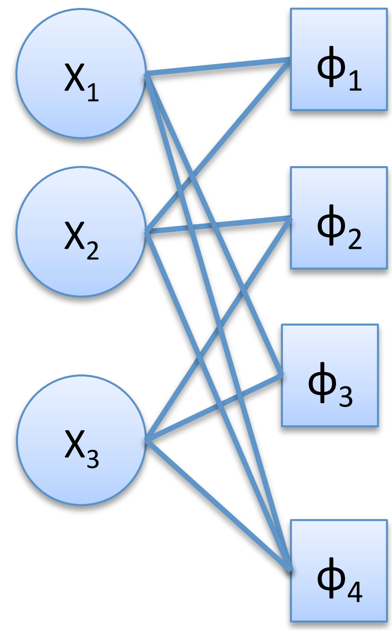
- Random variable nodes (circles)
- Factor nodes (squares)
- Edge between variable and factor if the factor depends on that variable.
  - The graph is bipartite.
- A factor is a function from tuples of r.v. values to nonnegative numbers.



$$P(\boldsymbol{X} = \boldsymbol{x}) \propto \prod_j \phi_j(\boldsymbol{x}_j)$$

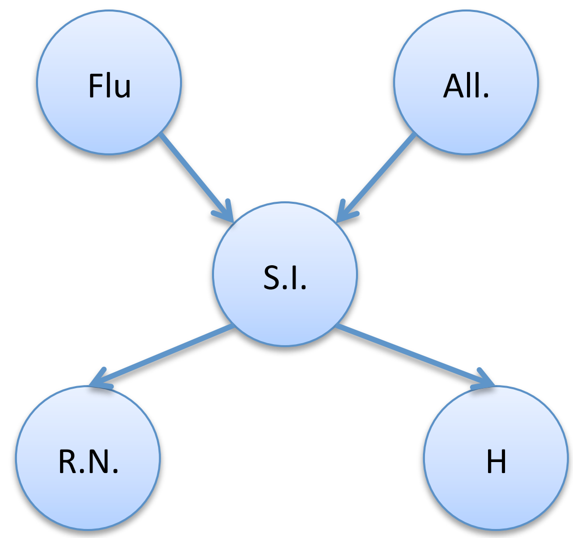
Two Kinds of Factors

- Conditional probability tables
  - E.g., $P(X_2 \mid X_1, X_3)$
  - Leads to Bayesian networks, causal explanations
- Potential functions
  - Arbitrary positive scores
  - Leads to Markov networks

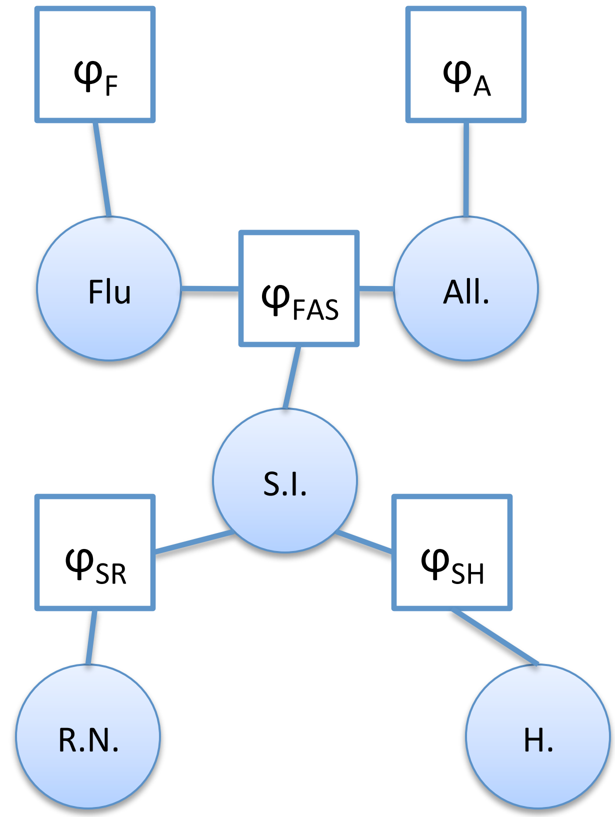


Example: Bayesian Network

- The flu causes sinus inflammation
- Allergies *also* cause sinus inflammation
- Sinus inflammation causes a runny nose
- Sinus inflammation causes headaches



- The flu causes sinus inflammation
- Allergies *also* cause sinus inflammation
- Sinus inflammation causes a runny nose
- Sinus inflammation causes headaches



"Some local configurations are more likely than others."

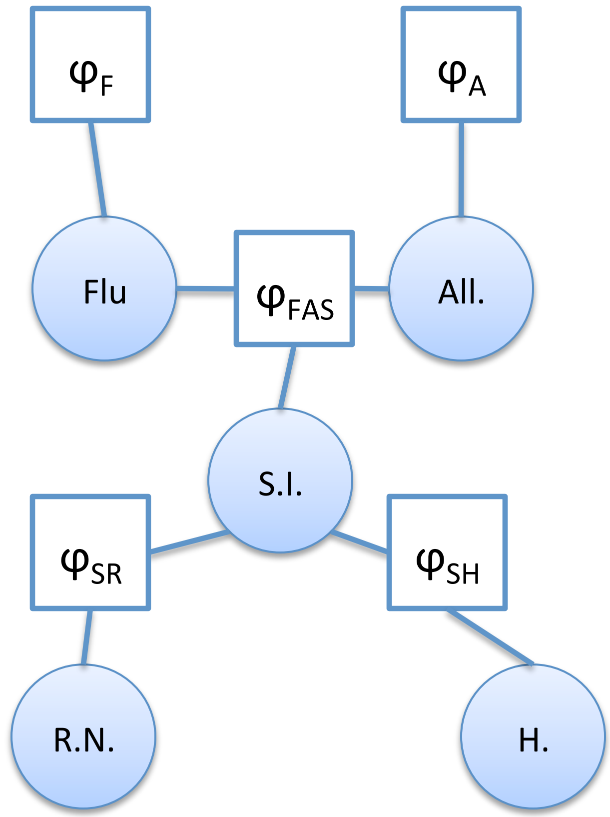
F	$\varphi_F(F)$
0	
1	

A	$\varphi_A(A)$
0	
1	

F	A	S	$\varphi_{FAS}(F,A,S)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

S	R	$\varphi_{SR}(S, R)$
0	0	
0	1	
1	0	
1	1	

S	H	$\varphi_{SH}(S, H)$
0	0	
0	1	
1	0	
1	1	



"Some local configurations are more likely than others."

Example: Markov Network

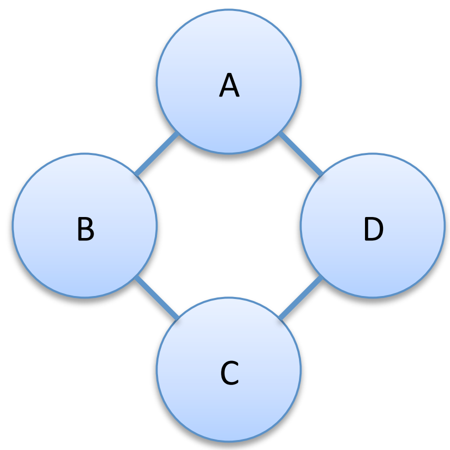
- Swinging couples or confused students

$A \perp C \mid B, D$

$B \perp D \mid A, C$

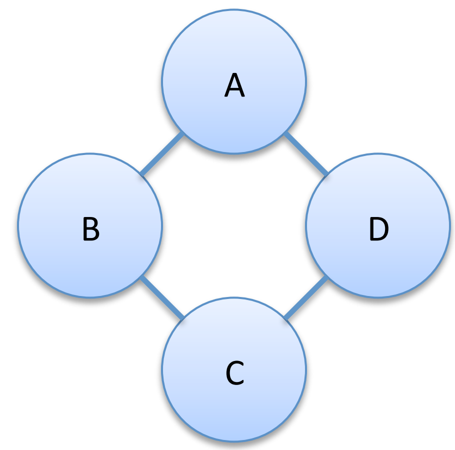
$\neg\, B \perp D$

$\neg\, A \perp C$



Example: Markov Network

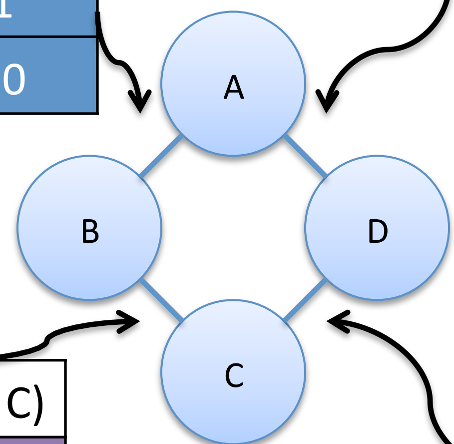
- Each random variable is a vertex.
- Undirected edges.
- **Factors** are associated with subsets of nodes that form cliques.
  - A factor maps assignments of its nodes to nonnegative values.



- In this example, associate a factor with each edge.
  - Could also have factors for single nodes!

A	B	$\varphi_1(A, B)$
0	0	30
0	1	5
1	0	1
1	1	10

A	D	$\varphi_4(A, D)$
0	0	100
0	1	1
1	0	1
1	1	100



B	C	$\varphi_2(B, C)$
0	0	100
0	1	1
1	0	1
1	1	100

C	D	$\varphi_3(C, D)$
0	0	1
0	1	100
1	0	100
1	1	1

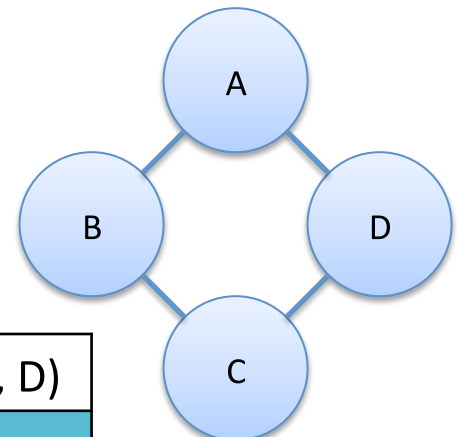
Markov Networks

- Probability distribution:

$$P(a,b,c,d) \propto \phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(a,d)$$

$$P(a,b,c,d) = \frac{\phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(a,d)}{\sum_{a',b',c',d'} \phi_1(a',b')\phi_2(b',c')\phi_3(c',d')\phi_4(a',d')}$$

$$Z = \sum_{a',b',c',d'} \phi_1(a',b')\phi_2(b',c')\phi_3(c',d')\phi_4(a',d')$$



A	B	φ_1(A, B)
0	0	30
0	1	5
1	0	1
1	1	10

B	C	φ_2(B, C)
0	0	100
0	1	1
1	0	1
1	1	100

C	D	φ_3(C, D)
0	0	1
0	1	100
1	0	100
1	1	1

A	D	φ_4(A, D)
0	0	100
0	1	1
1	0	1
1	1	100

Example: Markov Network

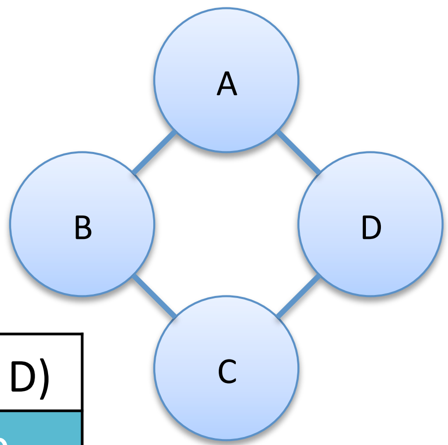
- Probability distribution:

$$P(a,b,c,d) \propto \phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(a,d)$$

$$P(a,b,c,d) = \frac{\phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(a,d)}{\sum_{a',b',c',d'} \phi_1(a',b')\phi_2(b',c')\phi_3(c',d')\phi_4(a',d')}$$

$$Z = \sum_{a',b',c',d'} \phi_1(a',b')\phi_2(b',c')\phi_3(c',d')\phi_4(a',d')$$

= 7,201,840



A	B	$\varphi_1(A, B)$
0	0	30
0	1	5
1	0	1
1	1	10

B	C	$\varphi_2(B, C)$
0	0	100
0	1	1
1	0	1
1	1	100

C	D	$\varphi_3(C, D)$
0	0	1
0	1	100
1	0	100
1	1	1

A	D	$\varphi_4(A, D)$
0	0	100
0	1	1
1	0	1
1	1	100

Example: Markov Network

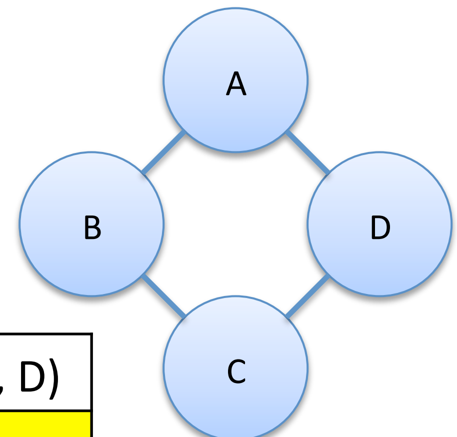
- Probability distribution:

$$P(a,b,c,d) \propto \phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(a,d)$$

$$P(a,b,c,d) = \frac{\phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(a,d)}{\sum_{a',b',c',d'} \phi_1(a',b')\phi_2(b',c')\phi_3(c',d')\phi_4(a',d')}$$

$$Z = \sum_{a',b',c',d'} \phi_1(a',b')\phi_2(b',c')\phi_3(c',d')\phi_4(a',d')$$

= 7,201,840



A	B	$\varphi_1(A, B)$
0	0	30
0	1	5
1	0	1
1	1	10

B	C	$\varphi_2(B, C)$
0	0	100
0	1	1
1	0	1
1	1	100

C	D	$\varphi_3(C, D)$
0	0	1
0	1	100
1	0	100
1	1	1

A	D	$\varphi_4(A, D)$
0	0	100
0	1	1
1	0	1
1	1	100

P(0, 1, 1, 0)
= 5,000,000 / Z
= 0.69

Example: Markov Network

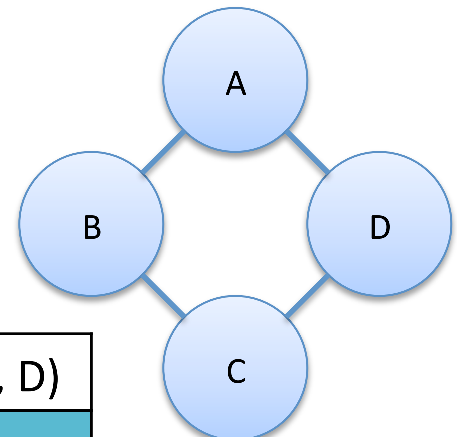
- Probability distribution:

$$P(a,b,c,d) \propto \phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(a,d)$$

$$P(a,b,c,d) = \frac{\phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(a,d)}{\sum_{a',b',c',d'} \phi_1(a',b')\phi_2(b',c')\phi_3(c',d')\phi_4(a',d')}$$

$$Z = \sum_{a',b',c',d'} \phi_1(a',b')\phi_2(b',c')\phi_3(c',d')\phi_4(a',d')$$

= 7,201,840



A	B	φ_1(A, B)
0	0	30
0	1	5
1	0	1
1	1	10

B	C	φ_2(B, C)
0	0	100
0	1	1
1	0	1
1	1	100

C	D	φ_3(C, D)
0	0	1
0	1	100
1	0	100
1	1	1

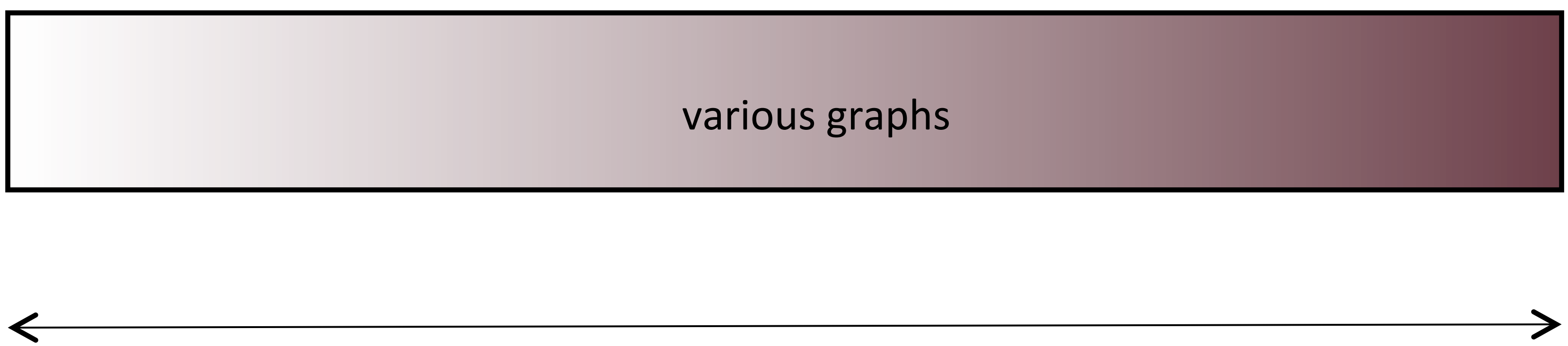
A	D	φ_4(A, D)
0	0	100
0	1	1
1	0	1
1	1	100

P(1, 1, 0, 0)
= 10 / Z
= 0.0000014

Independence and Structure

- There's a *lot* of theory about how BNs and MNs encode conditional independence assumptions.
  - **BNs:** A variable X is independent of its non-descendants given its parents.
  - **MNs**: Conditional independence derived from "Markov blanket" and *separation* properties.
  - Local configurations can be used to check *all* conditional independence questions; almost no need to look at the values in the factors!

Independence Spectrum



$$\prod_i \phi_i(x_i)$$

full independence assumptions

$$\phi(\boldsymbol{x})$$

everything is dependent

Products of Factors

- Given two factors with different scopes, we can calculate a new factor equal to their products.

$$\phi_{product}(\boldsymbol{x} \cup \boldsymbol{y}) \quad = \quad \phi_1(\boldsymbol{x}) \cdot \phi_2(\boldsymbol{y})$$

Products of Factors

- Given two factors with different scopes, we can calculate a new factor equal to their products.

A	B	$\varphi_1(A, B)$
0	0	30
0	1	5
1	0	1
1	1	10

·

B	C	$\varphi_2(B, C)$
0	0	100
0	1	1
1	0	1
1	1	100

=

A	B	C	$\varphi_3(A, B, C)$
0	0	0	3000
0	0	1	30
0	1	0	5
0	1	1	500
1	0	0	100
1	0	1	1
1	1	0	10
1	1	1	1000

Factor Maximization

- Given **X** and Y ($Y \notin \mathbf{X}$), we can turn a factor φ(**X**, Y) into a factor ψ(**X**) via maximization:

$$\psi(\boldsymbol{X}) = \max_{Y} \phi(\boldsymbol{X}, Y)$$

- We can refer to this new factor by $\max_Y \varphi$.

Factor Maximization

- Given **X** and Y (Y ∉ **X**), we can turn a factor φ(**X**, Y) into a factor ψ(**X**) via maximization:

$$\psi(\boldsymbol{X}) = \max_Y \phi(\boldsymbol{X}, Y)$$

A	B	C	φ (A, B, C)
0	0	0	0.9
0	0	1	0.3
0	1	0	1.1
0	1	1	1.7
1	0	0	0.4
1	0	1	0.7
1	1	0	1.1
1	1	1	0.2

"maximizing out" B

A	C	ψ(A, C)	
0	0	1.1	B=1
0	1	1.7	B=1
1	0	1.1	B=1
1	1	0.7	B=0

Factor Marginalization

- Given **X** and Y (Y ∉ **X**), we can turn a factor φ(**X**, Y) into a factor ψ(**X**) via marginalization:

$$\psi(\boldsymbol{X}) \quad = \quad \sum_{y \in \mathrm{Val}(Y)} \phi(\boldsymbol{X}, y)$$

Factor Marginalization

- Given **X** and Y (Y ∉ **X**), we can turn a factor φ(**X**, Y) into a factor ψ(**X**) via marginalization:

$$\psi(\boldsymbol{X}) = \sum_{y \in \mathrm{Val}(Y)} \phi(\boldsymbol{X}, y)$$

A	B	C	φ (A, B, C)
0	0	0	0.9
0	0	1	0.3
0	1	0	1.1
0	1	1	1.7
1	0	0	0.4
1	0	1	0.7
1	1	0	1.1
1	1	1	0.2

"summing out" B

A	C	ψ(A, C)
0	0	2.0
0	1	2.0
1	0	1.5
1	1	0.9

Factor Marginalization

- Given **X** and Y (Y ∉ **X**), we can turn a factor φ(**X**, Y) into a factor ψ(**X**) via marginalization:

$$\psi(\boldsymbol{X}) = \sum_{y \in \mathrm{Val}(Y)} \phi(\boldsymbol{X}, y)$$

A	B	C	φ (A, B, C)
0	0	0	0.9
0	0	1	0.3
0	1	0	1.1
0	1	1	1.7
1	0	0	0.4
1	0	1	0.7
1	1	0	1.1
1	1	1	0.2

"summing out" C

A	B	ψ(A, B)
0	0	1.2
0	1	2.8
1	0	1.1
1	1	1.3

Factor Marginalization

- Given **X** and Y ($Y \notin \mathbf{X}$), we can turn a factor φ(**X**, Y) into a factor ψ(**X**) via marginalization:

$$\psi(\boldsymbol{X}) = \sum_{y \in \mathrm{Val}(Y)} \phi(\boldsymbol{X}, y)$$

- We can refer to this new factor by $\sum_{Y} \varphi$.

Marginalizing Everything?

- Take a factor graph's "everything factor" by multiplying *all* of its factors.
- Sum out all the variables (one by one).

- What do you get?

Factors Are Like Numbers

- Products are commutative: $\varphi_1 \cdot \varphi_2 = \varphi_2 \cdot \varphi_1$
- Products are associative: $(\varphi_1 \cdot \varphi_2) \cdot \varphi_3 = \varphi_1 \cdot (\varphi_2 \cdot \varphi_3)$
- Sums are commutative: $\sum_X \sum_Y \varphi = \sum_Y \sum_X \varphi$ (max, too).
- Distributivity of multiplication over marginalization and maximization:

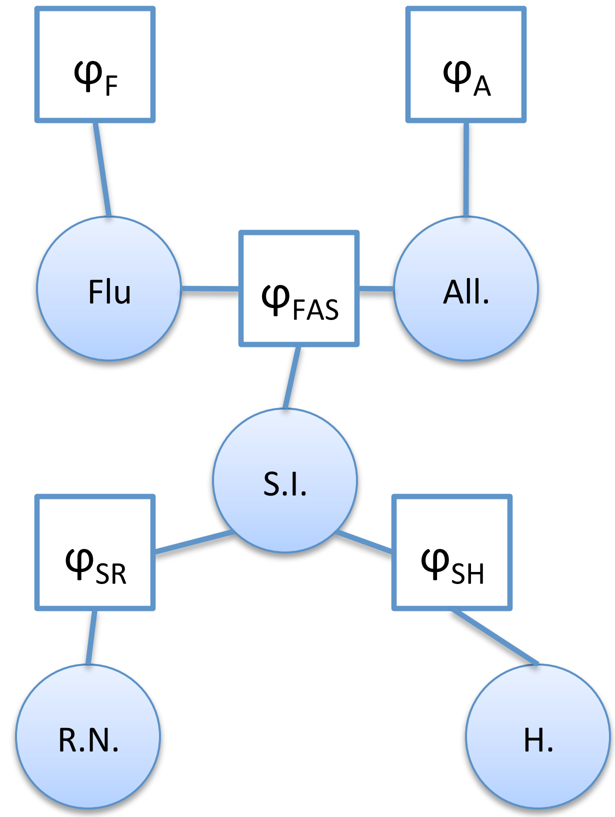
$$X \notin \text{Scope}(\phi_1) \quad \Rightarrow \quad \sum_X (\phi_1 \cdot \phi_2) = \phi_1 \cdot \sum_X \phi_2$$

$$\max_X (\phi_1 \cdot \phi_2) = \phi_1 \cdot \max_X \phi_2$$

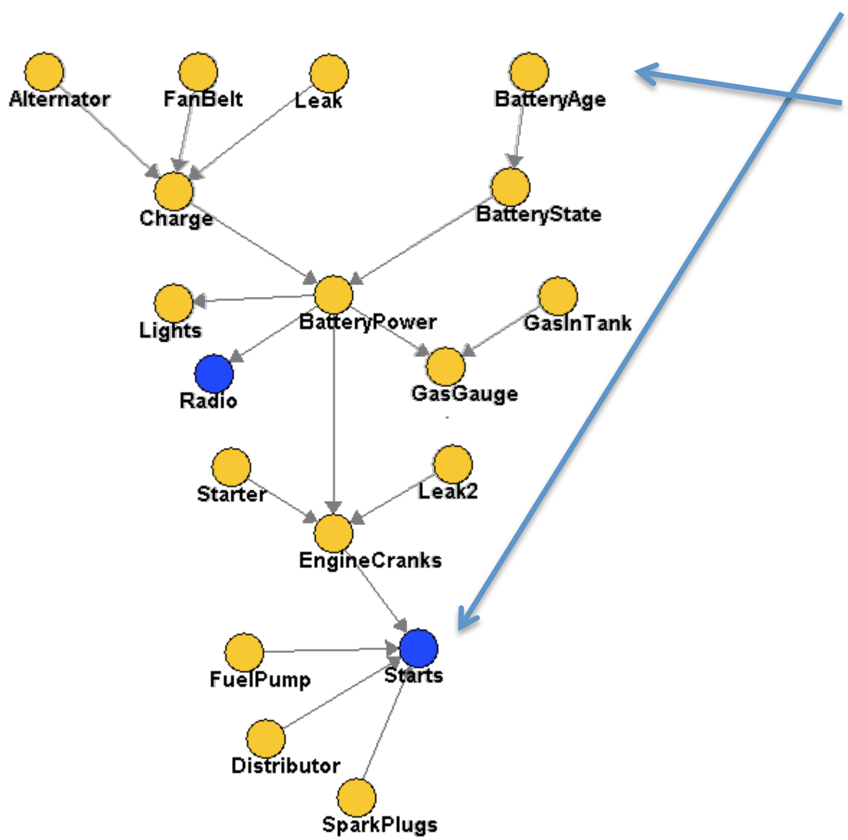
Inference

Querying the Model

- Inference (e.g., do you have allergies or the flu?)
- What's the best explanation for your symptoms?
- Active data collection (what is the next best r.v. to observe?)



A Bigger Example: Your Car



- The car doesn't start.
- What do we conclude about the battery age?

- 18 random variables
- 2^{18} possible scenarios

Inference: An Ubiquitous Obstacle

- Decoding is inference (lecture 2).
- Learning is inference (lectures 3 and 4).

- Exact inference is #P-complete.
  - Even approximations within a given absolute or relative error are hard.

Probabilistic Inference Problems

Given values for some random variables ($\mathbf{X} \subset \mathbf{V}$) …

- Most Probable Explanation: what are the *most probable* values of the *rest* of the r.v.s $\mathbf{V} \setminus \mathbf{X}$?

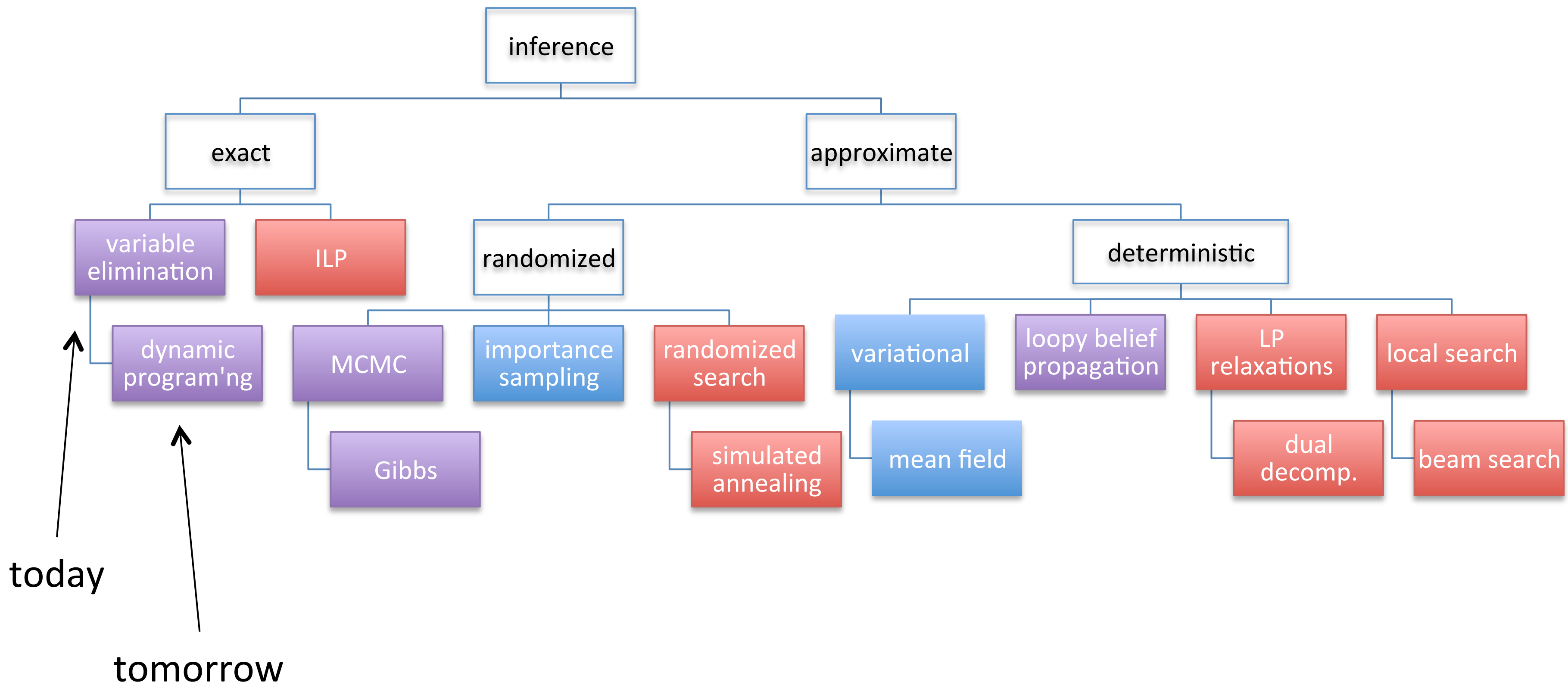
(More generally …)

- Maximum *A Posteriori* (MAP): what are the most probable values of *some* other r.v.s, $\mathbf{Y} \subset (\mathbf{V} \setminus \mathbf{X})$?

- Random sampling from the posterior over values of **Y**
- Full posterior over values of **Y**
- Marginal probabilities from the posterior over **Y**

- Minimum Bayes risk: What is the **Y** with the lowest expected cost?
- Cost-augmented decoding: What is the most *dangerous* **Y**?

Approaches to Inference



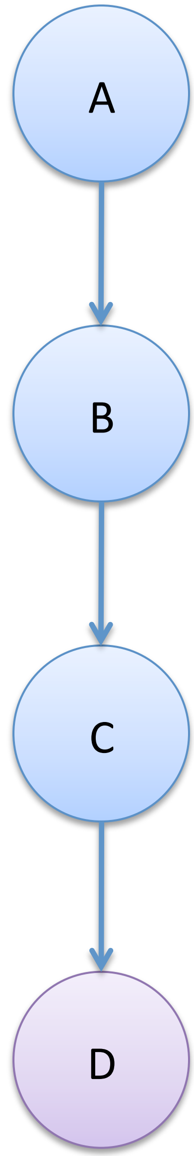
hard inference methods; soft inference methods; methods for both

Exact Marginal for Y

- This will be a generalization of algorithms you may already have seen: the *forward* and *backward* algorithms.
- The general name is variable elimination.
- After we see it for the marginal, we'll see how to use it for the MAP.

Inference Example

- Goal: P(D)



A	$P(A) = \varphi_A(A)$
0	
1	

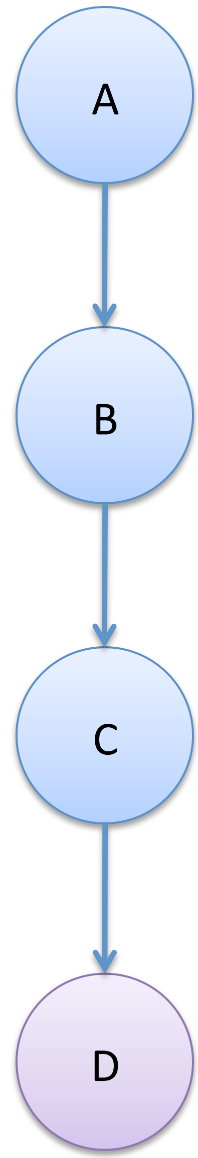
A	B	$P(B \mid A) = \varphi_{AB}(A, B)$
0	0	
0	1	
1	0	
1	1	

B	C	$P(C \mid B) = \varphi_{BC}(B, C)$
0	0	
0	1	
1	0	
1	1	

C	D	$P(D \mid C) = \varphi_{CD}(C, D)$
0	0	
0	1	
1	0	
1	1	

Inference Example

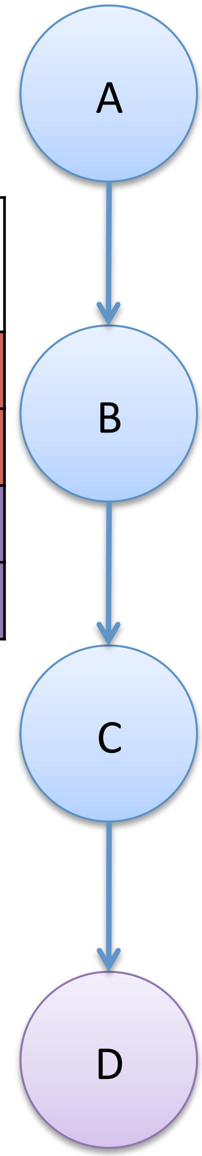
- Let's calculate P(B) first.



Inference Example

- Let's calculate P(B) first.

A	B	P(B | A) = $\varphi_{AB}(A, B)$
0	0	
0	1	
1	0	
1	1	



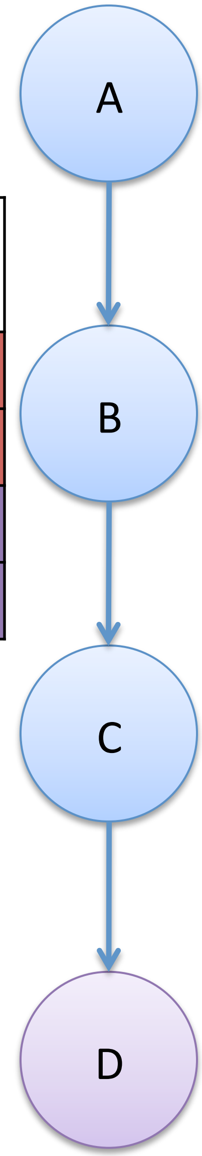
A	P(A) = $\varphi_A(A)$
0	
1	

$$P(B) = \sum_{a \in \mathrm{Val}(A)} P(A = a) P(B \mid A = a)$$

Inference Example

- Let's calculate P(B) first.

A	B	P(B | A) = $\varphi_{AB}(A, B)$
0	0	
0	1	
1	0	
1	1	



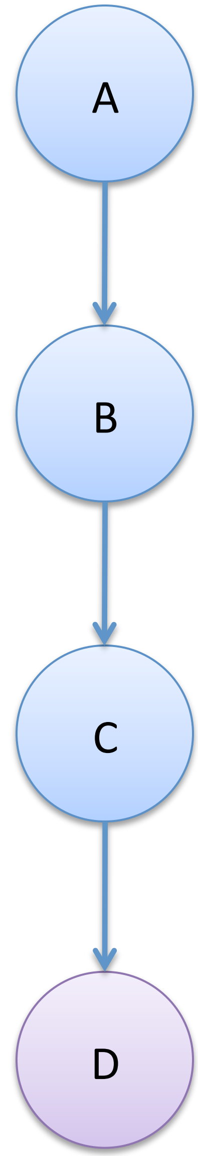
A	P(A) = $\varphi_A(A)$
0	
1	

$$P(B) = \sum_{a \in \mathrm{Val}(A)} P(A = a) P(B \mid A = a)$$

- Note: C and D don't matter.

Inference Example

- Let's calculate P(B) first.



$$P(B) = \sum_{a \in \mathrm{Val}(A)} P(A = a) P(B \mid A = a)$$

B	P(B) = $\varphi_B(B)$
0	
1	

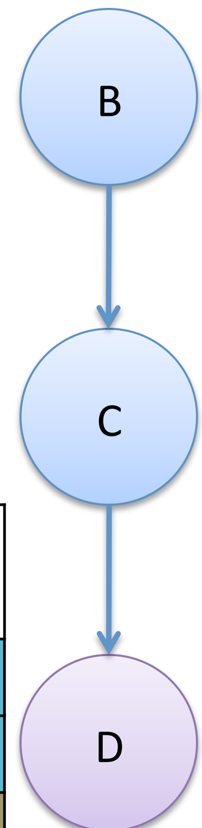
A	P(A) = $\varphi_A(A)$
0	
1	

A	B	P(B | A) = $\varphi_{AB}(A, B)$
0	0	
0	1	
1	0	
1	1	

Inference Example

- New model in which A is eliminated; defines P(B, C, D)

B	P(B) = $\varphi_B(B)$
0	
1	

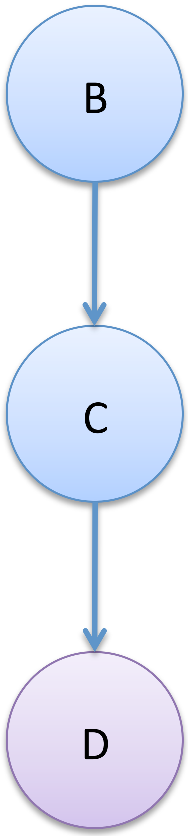


B	C	P(C | B) = $\varphi_{BC}(B, C)$
0	0	
0	1	
1	0	
1	1	

C	D	P(D | C) = $\varphi_{CD}(C, D)$
0	0	
0	1	
1	0	
1	1	

Inference Example

- Same thing to eliminate B.



$$P(C) = \sum_{b \in \mathrm{Val}(B)} P(B=b)P(C \mid B=b)$$

C	P(C) = $\varphi_C(C)$
0	
1	

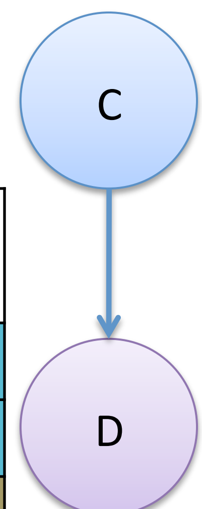
B	P(B) = $\varphi_B(B)$
0	
1	

B	C	P(C | B) = $\varphi_{BC}(B, C)$
0	0	
0	1	
1	0	
1	1	

Inference Example

- New model in which B is eliminated; defines P(C, D)

C	P(C) = $\varphi_C(C)$
0	
1	



C	D	P(D | C) = $\varphi_{CD}(C, D)$
0	0	
0	1	
1	0	
1	1	

Simple Inference Example

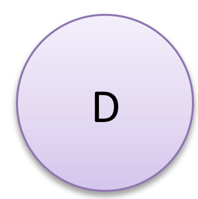
- Last step to get P(D):

$$P(D) = \sum_{c \in \mathrm{Val}(C)} P(C = c) P(D \mid C = c)$$

D	P(D) = $\varphi_D(D)$
0	
1	

C	P(C) = $\varphi_C(C)$
0	
1	

C	D	P(D | C) = $\varphi_{CD}(C, D)$
0	0	
0	1	
1	0	
1	1	



Simple Inference Example

- Notice that the same step happened for each random variable:
  - We created a new factor over the variable and its "successor"
  - We summed out (marginalized) the variable.

$$
\begin{aligned}
P(D) &= \sum_{a \in \mathrm{Val}(A)} \sum_{b \in \mathrm{Val}(B)} \sum_{c \in \mathrm{Val}(C)} P(A = a) P(B = b \mid A = a) P(C = c \mid B = b) P(D \mid C = c) \\
&= \sum_{c \in \mathrm{Val}(C)} P(D \mid C = c) \sum_{b \in \mathrm{Val}(B)} P(C = c \mid B = b) \sum_{a \in \mathrm{Val}(A)} P(A = a) P(B = b \mid A = a)
\end{aligned}
$$

That Was Variable Elimination

- We reused computation from previous steps and avoided doing the same work more than once.
  - Dynamic programming à la forward algorithm!
- We exploited the graph structure (each subexpression only depends on a small number of variables).
- Exponential blowup avoided!

What Remains

- Variable elimination in general
- The maximization version (for MAP inference)
- A bit about approximate inference

Eliminating One Variable

Input: Set of factors $\mathbf{\Phi}$, variable Z to eliminate

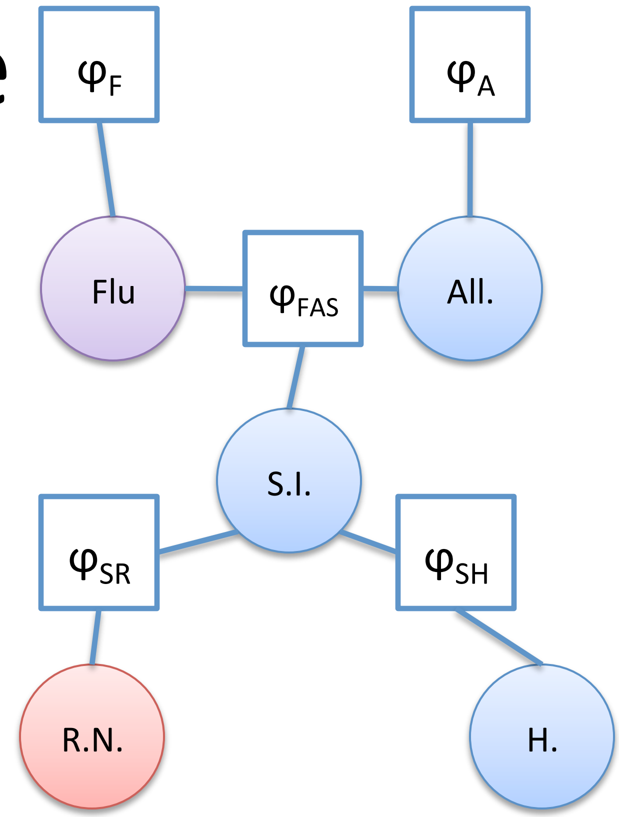
Output: new set of factors $\mathbf{\Psi}$

1. Let $\mathbf{\Phi'} = \{\varphi \in \mathbf{\Phi} \mid Z \in \text{Scope}(\varphi)\}$
2. Let $\mathbf{\Psi} = \{\varphi \in \mathbf{\Phi} \mid Z \notin \text{Scope}(\varphi)\}$
3. Let ψ be $\sum_Z \prod_{\varphi \in \mathbf{\Phi'}} \varphi$
4. Return $\mathbf{\Psi} \cup \{\psi\}$

Example

- Query:
  P(Flu | runny nose)

- Let's eliminate H.

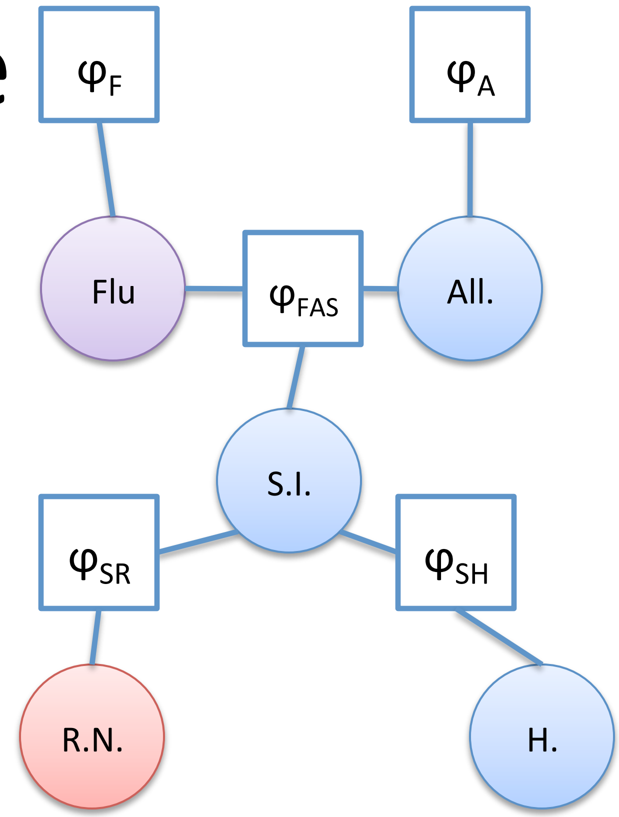


Example

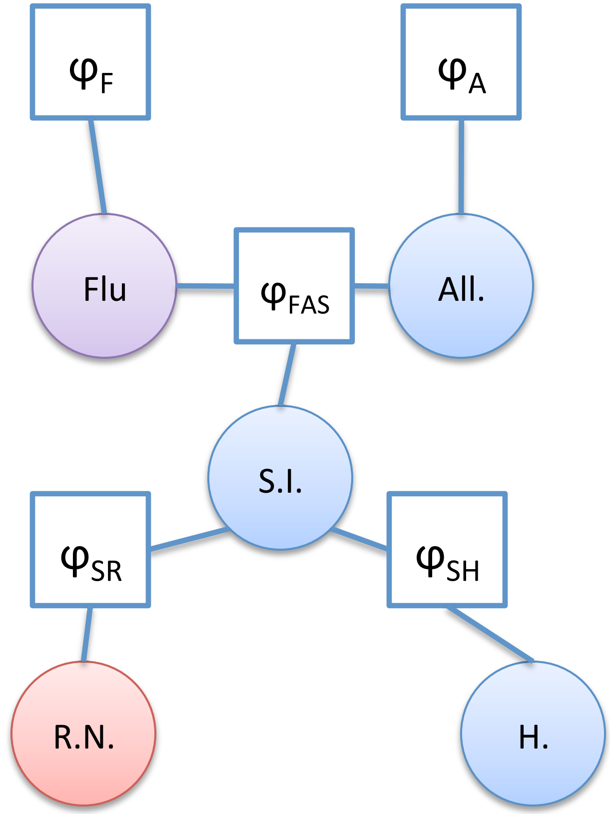
- Query:
  P(Flu | runny nose)

- Let's eliminate H.

1. $\boldsymbol{\Phi}' = \{\varphi_{SH}\}$
2. $\boldsymbol{\Psi} = \{\varphi_F, \varphi_A, \varphi_{FAS}, \varphi_{SR}\}$
3. $\psi = \sum_H \prod_{\varphi \in \boldsymbol{\Phi}'} \varphi$
4. Return $\boldsymbol{\Psi} \cup \{\psi\}$



Example

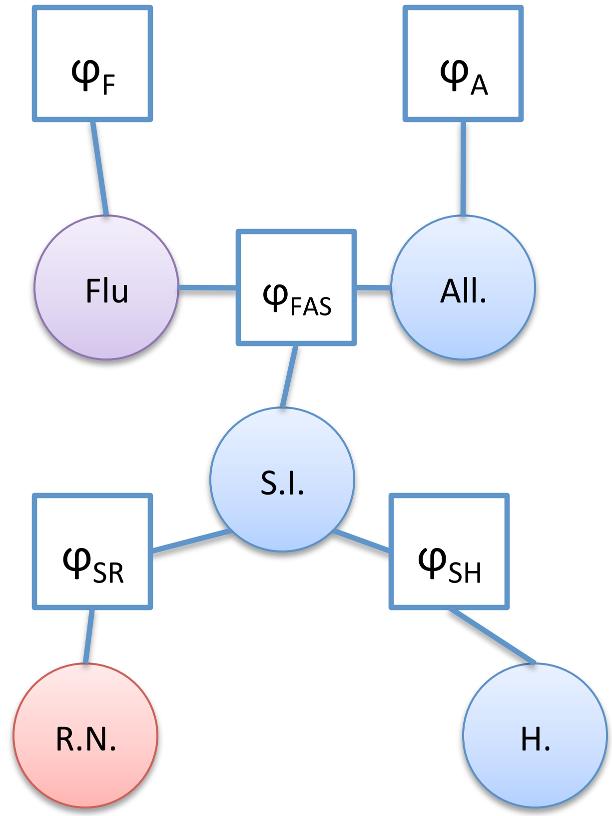


- Query:
  P(Flu | runny nose)

- Let's eliminate H.

1. $\boldsymbol{\Phi}' = \{\varphi_{SH}\}$
2. $\boldsymbol{\Psi} = \{\varphi_F, \varphi_A, \varphi_{FAS}, \varphi_{SR}\}$
3. $\psi = \sum_H \varphi_{SH}$
4. Return $\boldsymbol{\Psi} \cup \{\psi\}$

Example



- Query:
  P(Flu | runny nose)

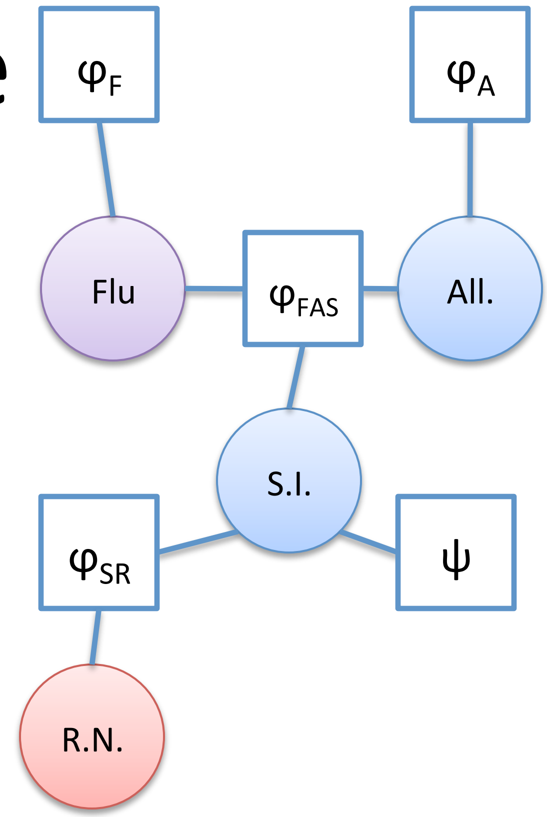
- Let's eliminate H.

1. $\mathbf{\Phi'} = \{\varphi_{SH}\}$
2. $\mathbf{\Psi} = \{\varphi_F, \varphi_A, \varphi_{FAS}, \varphi_{SR}\}$
3. $\psi = \sum_H \varphi_{SH}$
4. Return $\mathbf{\Psi} \cup \{\psi\}$

S	H	$\varphi_{SH}(S, H)$
0	0	0.8
0	1	0.2
1	0	0.1
1	1	0.9

S	$\psi(S)$
0	1.0
1	1.0

Example



- Query:
  P(Flu | runny nose)

- Let's eliminate H.

1. $\mathbf{\Phi'} = \{\varphi_{SH}\}$
2. $\mathbf{\Psi} = \{\varphi_F, \varphi_A, \varphi_{FAS}, \varphi_{SR}\}$
3. $\psi = \sum_H \varphi_{SH}$
4. Return $\mathbf{\Psi} \cup \{\psi\}$

S	H	$\varphi_{SH}(S, H)$
0	0	0.8
0	1	0.2
1	0	0.1
1	1	0.9

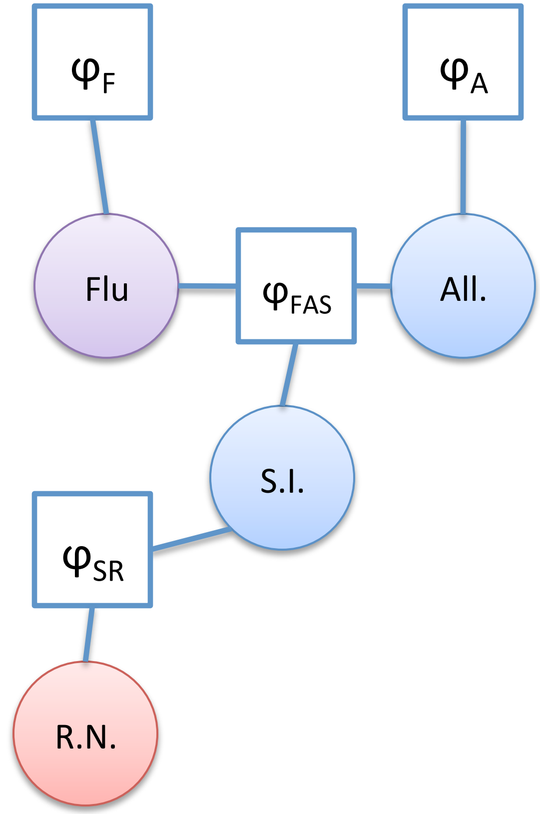
→

S	$\psi(S)$
0	1.0
1	1.0

Example

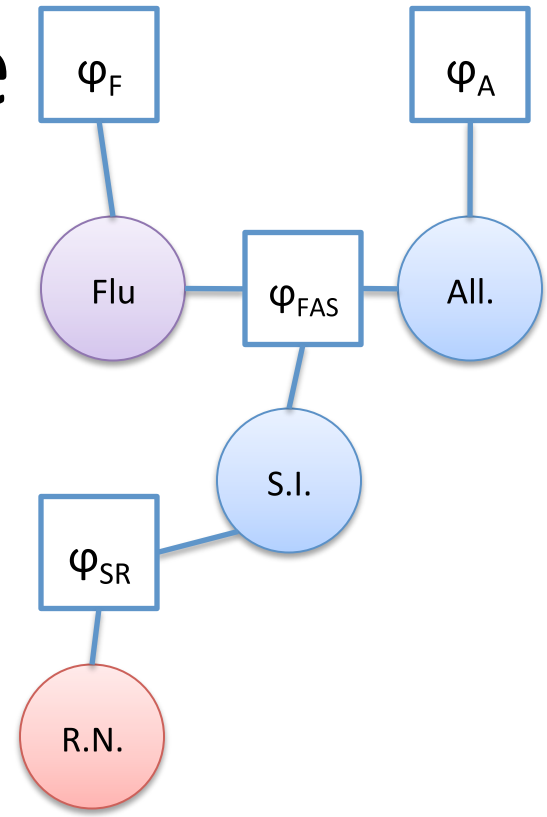
- Query:
  P(Flu | runny nose)

- Let's eliminate H.
- We can actually ignore the new factor, equivalently just deleting H!
  - Why?
  - In some cases eliminating a variable is really easy!



S	ψ(S)
0	1.0
1	1.0

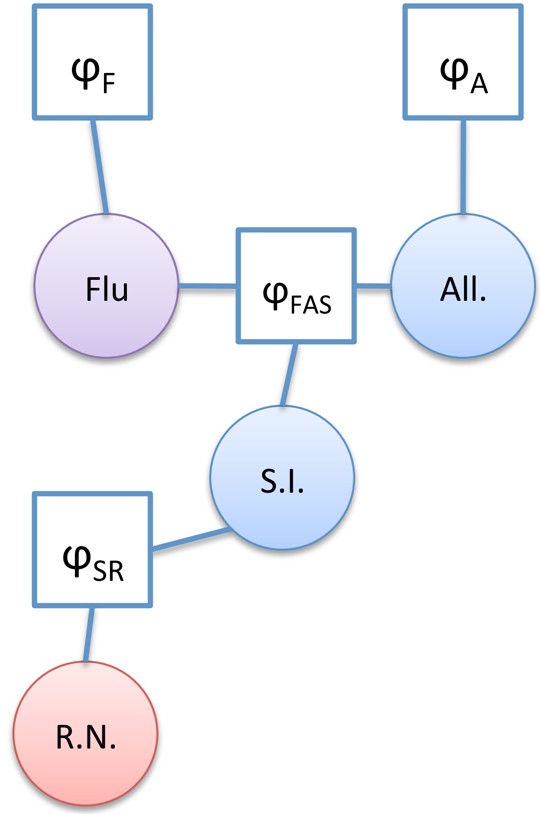
Example



- Query:
  P(Flu | runny nose)

- H is already eliminated.
- Let's now eliminate S.

Example

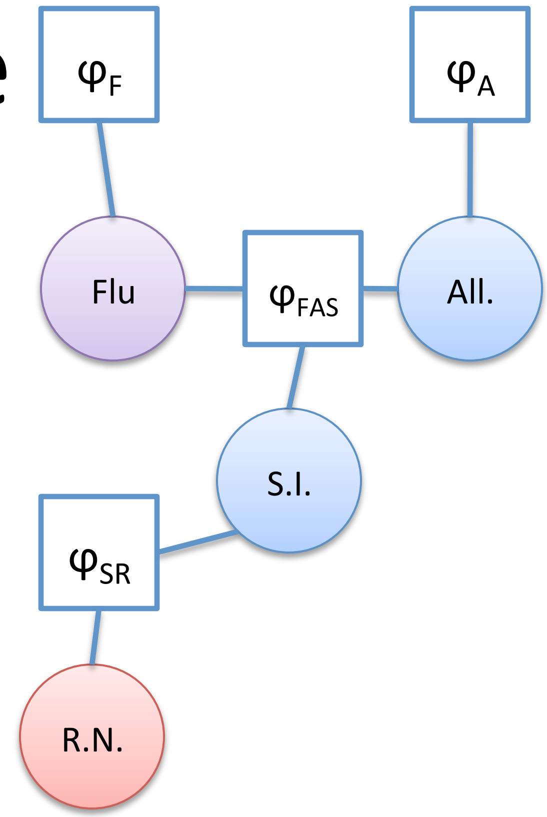


- Query:
  P(Flu | runny nose)

- Eliminating S.

1. $\mathbf{\Phi'} = \{\varphi_{SR}, \varphi_{FAS}\}$
2. $\mathbf{\Psi} = \{\varphi_F, \varphi_A\}$
3. $\psi_{FAR} = \sum_S \prod_{\varphi \in \mathbf{\Phi'}} \varphi$
4. Return $\mathbf{\Psi} \cup \{\psi_{FAR}\}$

Example

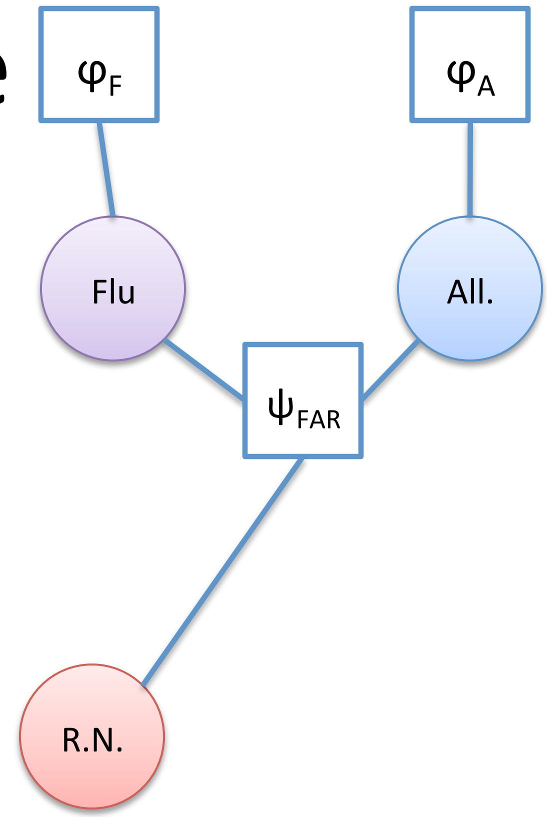


- Query: P(Flu | runny nose)

- Eliminating S.

1. $\mathbf{\Phi'} = \{\varphi_{SR}, \varphi_{FAS}\}$
2. $\mathbf{\Psi} = \{\varphi_F, \varphi_A\}$
3. $\psi_{FAR} = \sum_S \varphi_{SR} \cdot \varphi_{FAS}$
4. Return $\mathbf{\Psi} \cup \{\psi_{FAR}\}$

Example



- Query:
  P(Flu | runny nose)

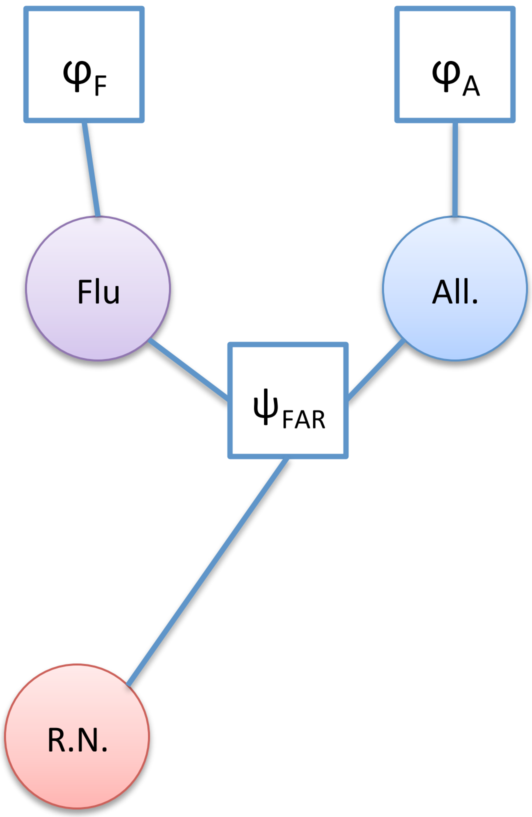
- Eliminating S.

1. $\mathbf{\Phi'} = \{\varphi_{SR}, \varphi_{FAS}\}$
2. $\mathbf{\Psi} = \{\varphi_F, \varphi_A\}$
3. $\psi_{FAR} = \sum_S \varphi_{SR} \cdot \varphi_{FAS}$
4. Return $\mathbf{\Psi} \cup \{\psi_{FAR}\}$

Example

- Query:
  P(Flu | runny nose)

- Finally, eliminate A.

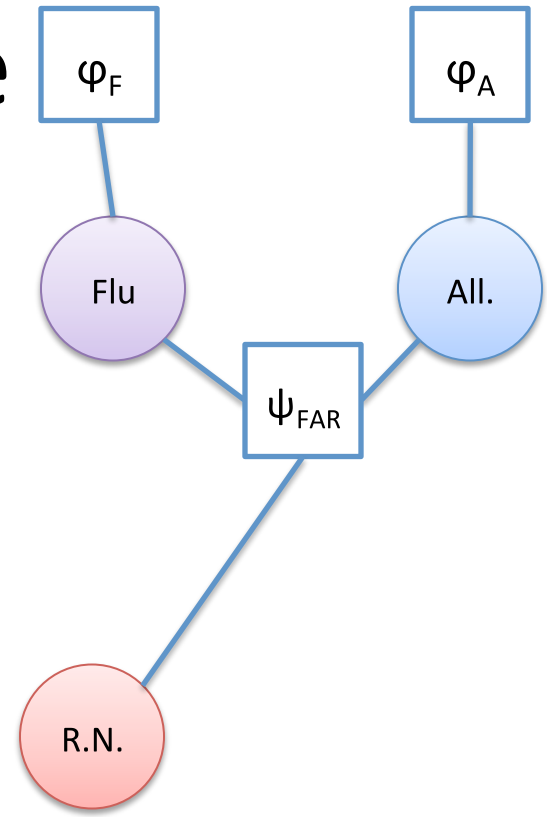


Example

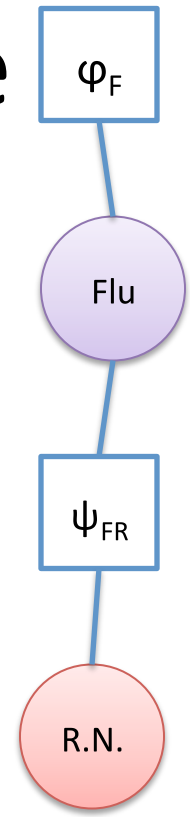
- Query:
  P(Flu | runny nose)

- Eliminating A.

1. $\mathbf{\Phi'} = \{\varphi_A, \varphi_{FAR}\}$
2. $\mathbf{\Psi} = \{\varphi_F\}$
3. $\psi_{FR} = \sum_A \varphi_A \cdot \psi_{FAR}$
4. Return $\mathbf{\Psi} \cup \{\psi_{FR}\}$



Example



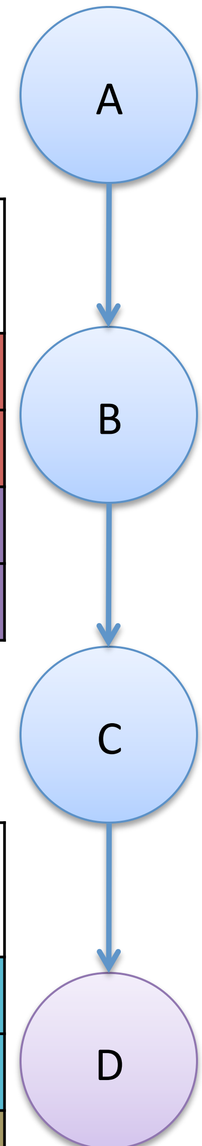
- Query:
  P(Flu | runny nose)

- Eliminating A.

1. $\mathbf{\Phi'} = \{\varphi_A, \varphi_{FAR}\}$
2. $\mathbf{\Psi} = \{\varphi_F\}$
3. $\psi_{FR} = \sum_A \varphi_A \cdot \psi_{FAR}$
4. Return $\mathbf{\Psi} \cup \{\psi_{FR}\}$

Chain, Again

- Goal: P(D)
- Earlier, we eliminated A, then B, then C.



A	$P(A) = \varphi_A(A)$
0	
1	

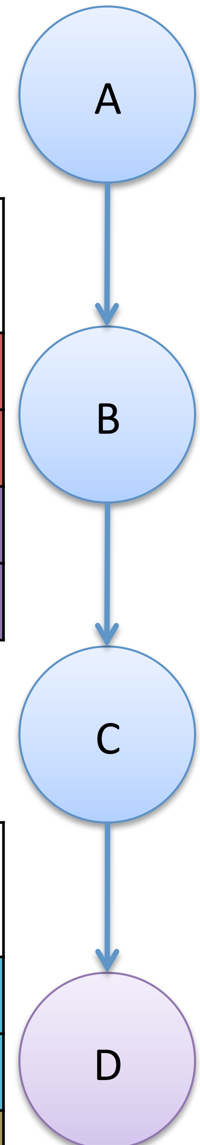
A	B	$P(B \mid A) = \varphi_{AB}(A, B)$
0	0	
0	1	
1	0	
1	1	

B	C	$P(C \mid B) = \varphi_{BC}(B, C)$
0	0	
0	1	
1	0	
1	1	

C	D	$P(D \mid C) = \varphi_{CD}(C, D)$
0	0	
0	1	
1	0	
1	1	

Chain, Again

- Goal: P(D)
- Earlier, we eliminated A, then B, then C.
- Let's start with C.



A	$P(A) = \varphi_A(A)$
0	
1	

A	B	$P(B \mid A) = \varphi_{AB}(A, B)$
0	0	
0	1	
1	0	
1	1	

B	C	$P(C \mid B) = \varphi_{BC}(B, C)$
0	0	
0	1	
1	0	
1	1	

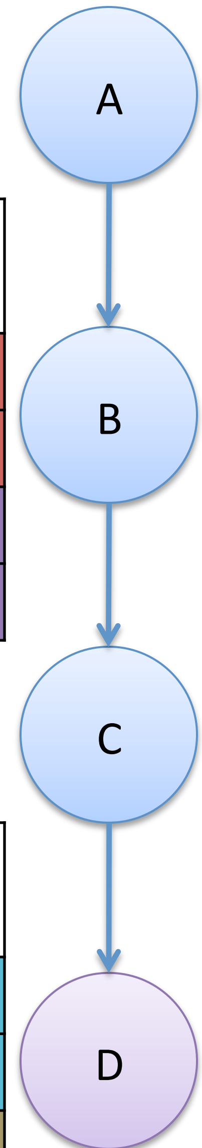
C	D	$P(D \mid C) = \varphi_{CD}(C, D)$
0	0	
0	1	
1	0	
1	1	

Chain, Again

- Goal: P(D)
- Earlier, we eliminated A, then B, then C.
- Let’s start with C.

A	$P(A) = \varphi_A(A)$
0	
1	

A	B	$P(B \mid A) = \varphi_{AB}(A, B)$
0	0	
0	1	
1	0	
1	1	



B	C	$P(C \mid B) = \varphi_{BC}(B, C)$
0	0	
0	1	
1	0	
1	1	

C	D	$P(D \mid C) = \varphi_{CD}(C, D)$
0	0	
0	1	
1	0	
1	1	

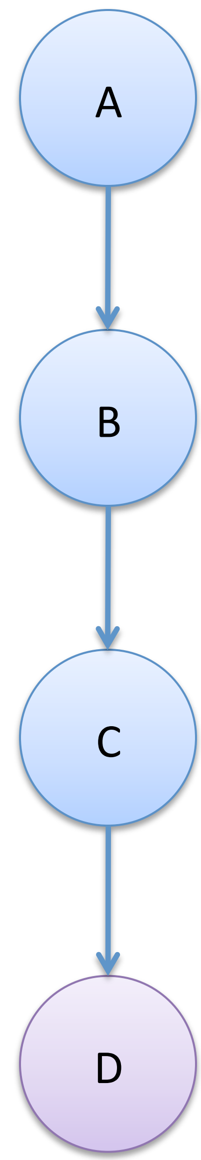
Chain, Again

- Eliminating C.

B	C	$P(C \mid B) = \varphi_{BC}(B, C)$
0	0	
0	1	
1	0	
1	1	

C	D	$P(D \mid C) = \varphi_{CD}(C, D)$
0	0	
0	1	
1	0	
1	1	

B	C	D	$\varphi_{BCD}(B, C, D)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

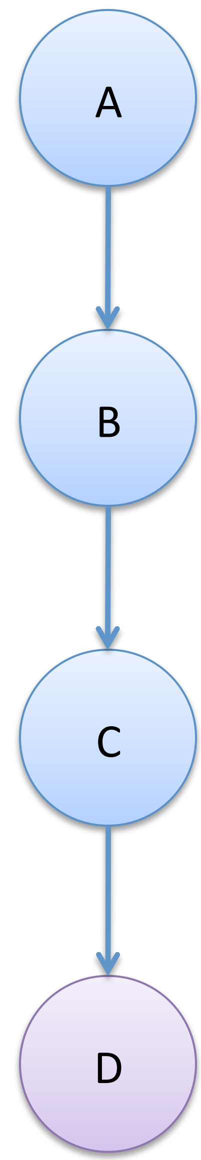


Chain, Again

- Eliminating C.

B	C	D	$\varphi_{BCD}(B, C, D)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

B	D	$\psi(B, D)$
0	0	
0	1	
1	0	
1	1	

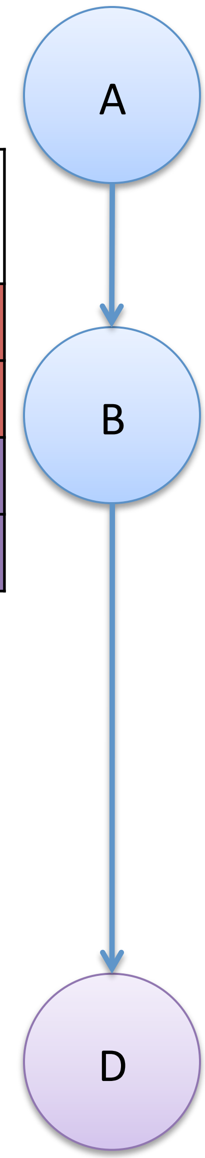


Chain, Again

- Eliminating B will be similarly complex.

A	B	P(B | A) = $\varphi_{AB}(A, B)$
0	0	
0	1	
1	0	
1	1	

B	D	$\psi(B, D)$
0	0	
0	1	
1	0	
1	1	



A	P(A) = $\varphi_A(A)$
0	
1	

Variable Elimination: Comments

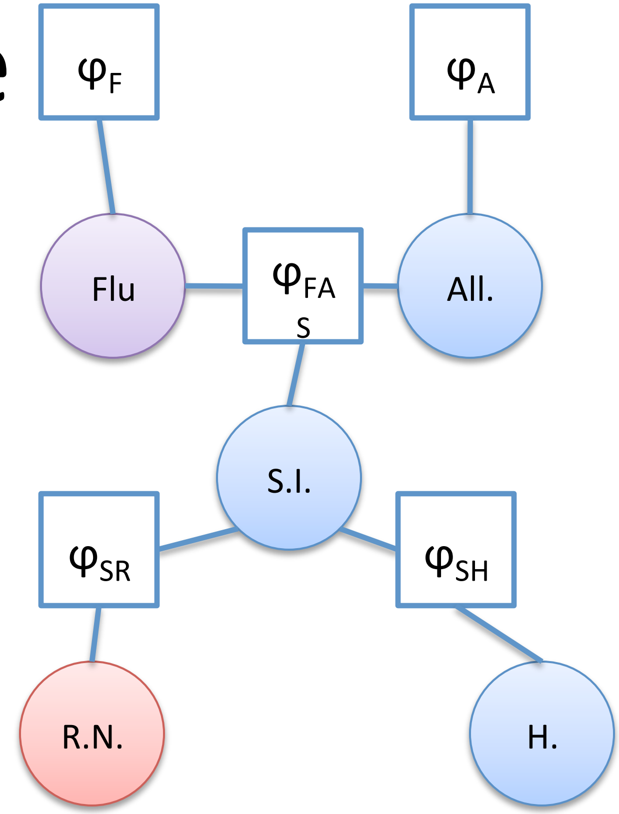
- Can prune away all non-ancestors of the query variables.
- Ordering makes a difference!

What about Evidence?

- So far, we've just considered the posterior/ marginal P(Y).
- Next: conditional distribution P(Y | **X** = **x**).

- It's almost the same: the additional step is to *reduce* factors to respect the evidence.

Example



- Query:
  P(Flu | runny nose)

- Let's reduce to R = true (runny nose).

S	R	φ_{SR} (S, R)
0	0	
0	1	
1	0	
1	1	

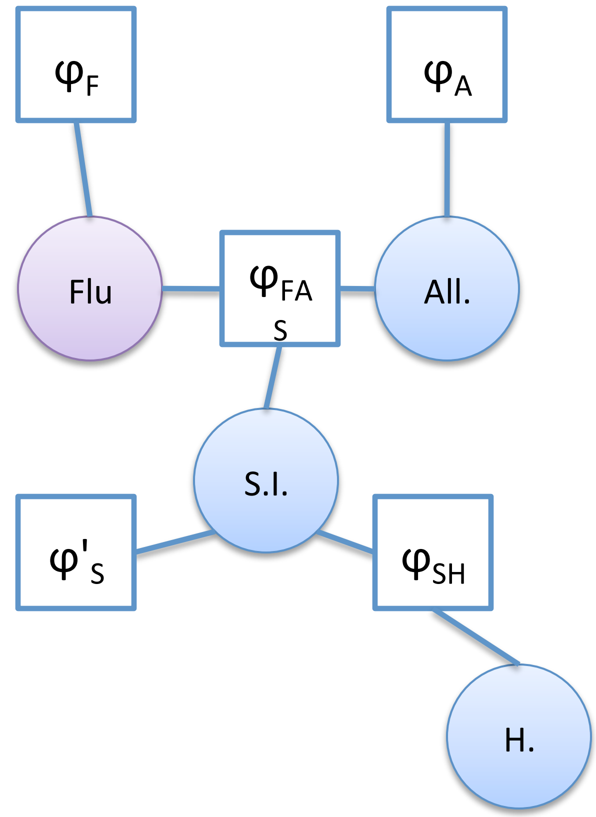
S	R	φ'_S (S)
[redacted]		
0	1	
[redacted]		
1	1	

S	R	φ'_S (S)
0	1	
1	1	

Example

- Query:
  P(Flu | runny nose)

- Let's reduce to R = true (runny nose).

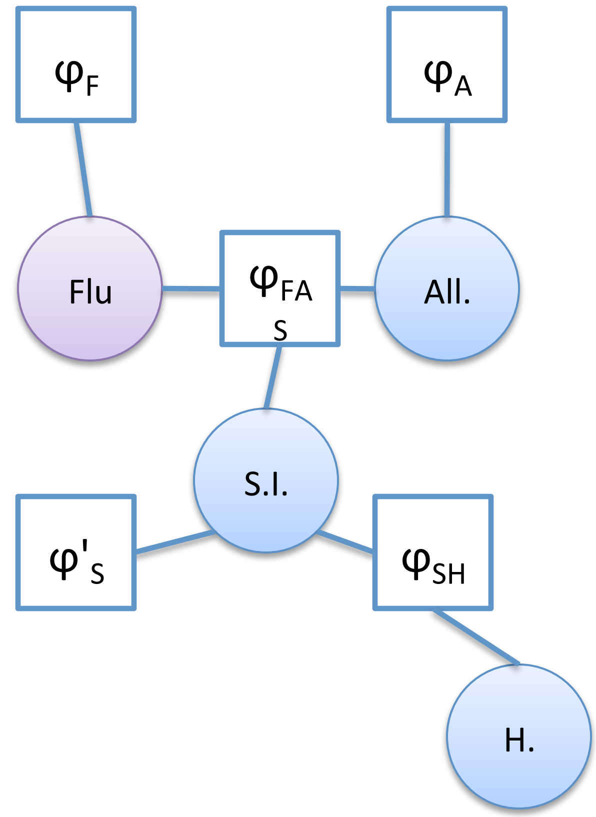


S	R	φ'_S (S)
0	1	
1	1	

Example

- Query:
  P(Flu | runny nose)

- Now run variable elimination all the way down to one factor (for F).

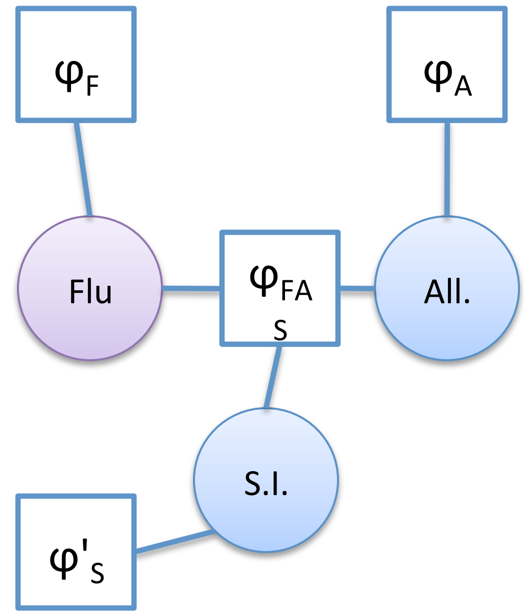


Eliminate H.

Example

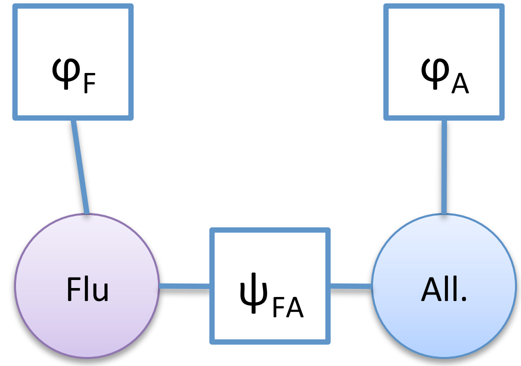
- Query:
  P(Flu | runny nose)

- Now run variable elimination all the way down to one factor (for F).



Eliminate S.

Example

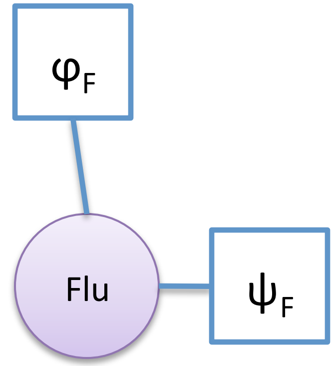


Eliminate A.

- Query:
  P(Flu | runny nose)

- Now run variable elimination all the way down to one factor (for F).

Example



Take final product.

- Query:
  P(Flu | runny nose)

- Now run variable elimination all the way down to one factor (for F).

Example

- Query:
  P(Flu | runny nose)

$\varphi_F \cdot \psi_F$

- Now run variable elimination all the way down to one factor.

Additional Comments

- Runtime depends on the size of the *intermediate* factors.
- Hence, variable elimination ordering matters a lot.
  - But it's NP-hard to find the best one.
  - For MNs, *chordal graphs* permit inference in time linear in the size of the original factors.
  - For BNs, *polytree* structures do the same.
- If you can avoid “big” intermediate factors, you can make inference linear in the size of the original factors.

Variable Elimination for Conditional Probabilities $P(\mathbf{Y} \mid \mathbf{X} = \mathbf{x})$

Input: Graphical model on $\mathbf{V}$, set of query variables $\mathbf{Y}$, evidence $\mathbf{X} = \mathbf{x}$

Output: factor φ and scalar α

1. $\mathbf{\Phi}$ = factors in the model
2. Reduce factors in $\mathbf{\Phi}$ by $\mathbf{X} = \mathbf{x}$
3. Choose variable ordering π on $\mathbf{Z} = \mathbf{V} \setminus \mathbf{Y} \setminus \mathbf{X}$
4. φ = Variable-Elimination($\mathbf{\Phi}$, $\mathbf{Z}$, π)
5. $\alpha = \sum_{\mathbf{z} \in Val(\mathbf{Z})} \varphi(\mathbf{z})$
6. Return φ, α

Getting Back to NLP

- Traditional structured NLP models were sometimes chosen for these properties.
  - HMMs, PCFGs (with a little work)
  - But not: IBM model 3
- To decode, we need MAP inference for decoding!
- When models get complicated, need approximations!

From Marginals to MAP

- Replace factor marginalization steps with *maximization*.
  - Add bookkeeping to keep track of the maximizing values.
- Add a traceback at the end to recover the solution.

- This is analogous to the connection between the forward algorithm and the Viterbi algorithm.
  - Ordering challenge is the same.

Variable Elimination (Max-Product Version with Decoding)

Input: Set of factors $\mathbf{\Phi}$, ordered list of variables $\mathbf{Z}$ to eliminate

Output: new factor

1. For each $Z_i \in \mathbf{Z}$ (in order):
   - Let $(\mathbf{\Phi}, \psi_{Z_i})$ = Eliminate-One$(\mathbf{\Phi}, Z_i)$
2. Return $\prod_{\varphi \in \mathbf{\Phi}} \varphi$, Traceback$(\{\psi_{Z_i}\})$

Eliminating One Variable (Max-Product Version with Bookkeeping)

Input: Set of factors $\mathbf{\Phi}$, variable Z to eliminate

Output: new set of factors $\mathbf{\Psi}$

1. Let $\mathbf{\Phi'} = \{\varphi \in \mathbf{\Phi} \mid Z \in \text{Scope}(\varphi)\}$
2. Let $\mathbf{\Psi} = \{\varphi \in \mathbf{\Phi} \mid Z \notin \text{Scope}(\varphi)\}$
3. Let τ be $\max_Z \prod_{\varphi \in \mathbf{\Phi'}} \varphi$
   – Let ψ be $\prod_{\varphi \in \mathbf{\Phi'}} \varphi$ (bookkeeping)
4. Return $\mathbf{\Psi} \cup \{\tau\}$, ψ

Traceback

Input: Sequence of factors with associated variables: $(\psi_{Z_1}, \ldots, \psi_{Z_k})$

Output: $\mathbf{z}^*$

- Each ψ_Z is a factor with scope including Z and variables eliminated *after* Z.
- Work backwards from i = k to 1:
  - Let $z_i = \arg\max_z \psi_{Z_i}(z, z_{i+1}, z_{i+2}, \ldots, z_k)$
- Return $\mathbf{z}$

About the Traceback

- No extra (asymptotic) expense.
  - Linear traversal over the intermediate factors.
- The factor operations for both sum-product VE and max-product VE can be generalized.
  - Example: get the K most likely assignments

Variable Elimination Tips

- Any ordering will be correct.
- Most orderings will be too expensive.
- There are heuristics for choosing an ordering.
  - If the graph is chain-like, work from one end toward the other.

(Rocket Science: True MAP)

- Evidence: **X** = **x**
- Query: **Y**
- Other variables: **Z** = **V** \ **X** \ **Y**

$$
\begin{aligned}
\boldsymbol{y}^* &= \arg\max_{\boldsymbol{y} \in \mathrm{Val}(\boldsymbol{Y})} P(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{X} = \boldsymbol{x}) \\
&= \arg\max_{\boldsymbol{y} \in \mathrm{Val}(\boldsymbol{Y})} \sum_{\boldsymbol{z} \in \mathrm{Val}(\boldsymbol{Z})} P(\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z} = \boldsymbol{z} \mid \boldsymbol{X} = \boldsymbol{x})
\end{aligned}
$$

- First, marginalize out **Z**, then do MAP inference over **Y** given **X** = **x**

- This is not usually attempted in NLP, with some exceptions.

Parting Shots

- You will probably never implement the general variable elimination algorithm.
- You will rarely use exact inference.
- Understand the *inference problem* would look like in exact form; then approximate.
  - Sometimes you get lucky.
  - You'll appreciate better approximations as they come along.