Probability and Structure in Natural Language Processing

Noah Smith, Carnegie Mellon University

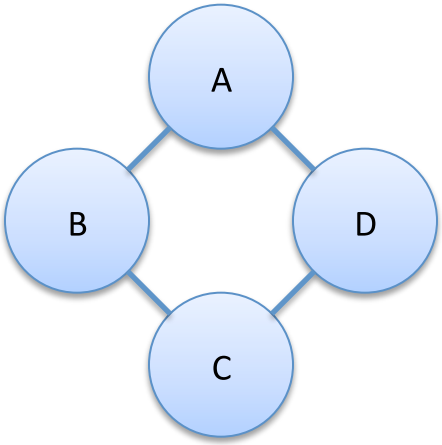
2012 International Summer School in Language and Speech Technologies

Slides Online!

- http://tinyurl.com/psnlp2012
- (I'll post the slides after each lecture.)

Markov Networks

- 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.

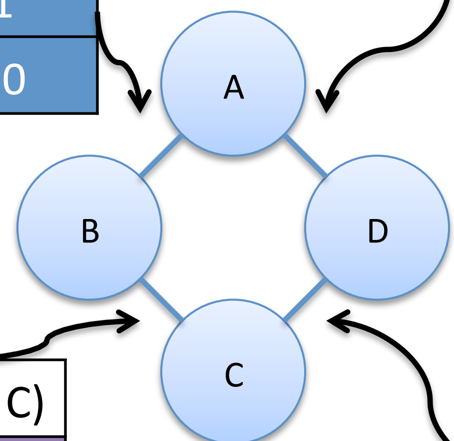


Markov Networks

- 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

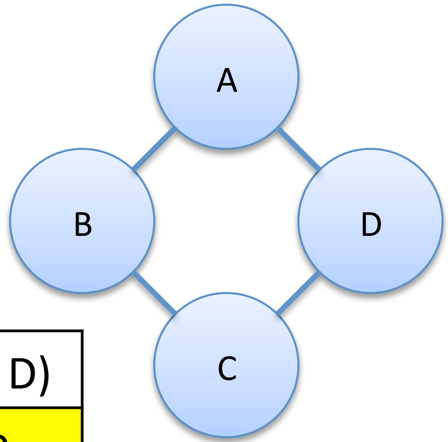
- Probability distribution:

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

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

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

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

Markov Networks

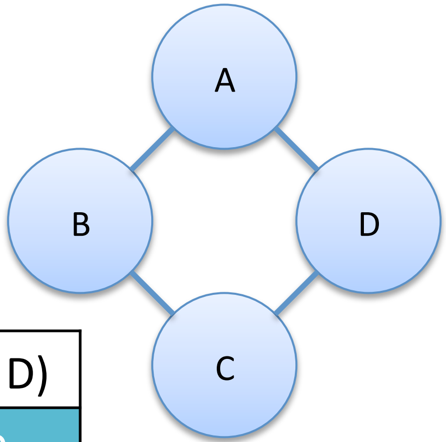
- Probability distribution:

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

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

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

= 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

Markov Networks (General Form)

- Let $\mathbf{D}_i$ denote the set of variables (subset of **X**) in the ith clique.
- Probability distribution is a **Gibbs** distribution:

$$
\begin{aligned}
P(\boldsymbol{X}) &= \frac{U(\boldsymbol{X})}{Z} \\
U(\boldsymbol{X}) &= \prod_{i=1}^{m} \phi_i(\boldsymbol{D}_i) \\
Z &= \sum_{\boldsymbol{x} \in \mathrm{Val}(\boldsymbol{X})} U(\boldsymbol{x})
\end{aligned}
$$

Notes

- Z might be hard to calculate.
  - "Normalization constant"
  - "Partition function"
- Can get efficient calculation in some cases.
  - This is an **inference** problem; it's equivalent to marginalizing over everything.
- *Ratios* of probabilities are easy.

$$\frac{P(\boldsymbol{x})}{P(\boldsymbol{x}')} = \frac{U(\boldsymbol{x})/Z}{U(\boldsymbol{x}')/Z} = \frac{U(\boldsymbol{x})}{U(\boldsymbol{x}')}$$

Independence in Markov Networks

- Given a set of observed nodes **Z**, a path X_1-X_2-X_3-…-X_k is **active** if no nodes on the path are observed.

Independence in Markov Networks

- Given a set of observed nodes **Z**, a path X_1-X_2-X_3-…-X_k is **active** if no nodes on the path are observed.
- Two sets of nodes **X** and **Y** in $\mathcal{H}$ are **separated** given **Z** if there is no active path between any $X_i \in \mathbf{X}$ and any $Y_i \in \mathbf{Y}$.
  - Denoted: $\text{sep}_{\mathcal{H}}(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z})$

Independence in Markov Networks

- Given a set of observed nodes **Z**, a path X_1-X_2-X_3-…-X_k is **active** if no nodes on the path are observed.
- Two sets of nodes **X** and **Y** in $\mathcal{H}$ are **separated** given **Z** if there is no active path between any $X_i \in$ **X** and any $Y_i \in$ **Y**.
  - Denoted: $\text{sep}_{\mathcal{H}}(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z})$
- Global Markov assumption: $\text{sep}_{\mathcal{H}}(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z}) \Rightarrow \mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}$

Representation Theorems

- Bayesian networks …

The Bayesian network graph's independencies are a subset of those in P.

$\Leftrightarrow$

$$P(\boldsymbol{X}) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Parents}(X_i))$$

  - Independencies give you the Bayesian network.
  - Bayesian network reveals independencies.

Representation Theorems

- Bayesian networks …

The Bayesian network graph's independencies are a subset of those in P. ⇔ $P(\boldsymbol{X}) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Parents}(X_i))$

- Markov networks …

Representation Theorems

- Bayesian networks …

The Bayesian network graph's independencies are a subset of those in P.

$$P(\boldsymbol{X}) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Parents}(X_i))$$

- Markov networks …

The Markov network graph's independencies are a subset of those in P.

***

$$P(\boldsymbol{X}) = \frac{1}{Z} \prod_{i=1}^{m} \phi_i(\boldsymbol{D}_i)$$

Hammersley-Clifford Theorem

- Other direction succeeds if P(**x**) > 0 for all **x**.
- Hammersley-Clifford Theorem

The Markov network graph's independencies are a subset of those in P *and P is nonnegative*.

⇒

$$P(\boldsymbol{X}) = \frac{1}{Z}\prod_{i=1}^{m}\phi_i(\boldsymbol{D}_i)$$

Completeness of Separation

- For almost all P that factorize, I(H) = I(P).
  - "Almost all" is the same hedge as in the Bayesian network case. A measure-zero set of parameterizations might make stronger independence assumptions than P does.

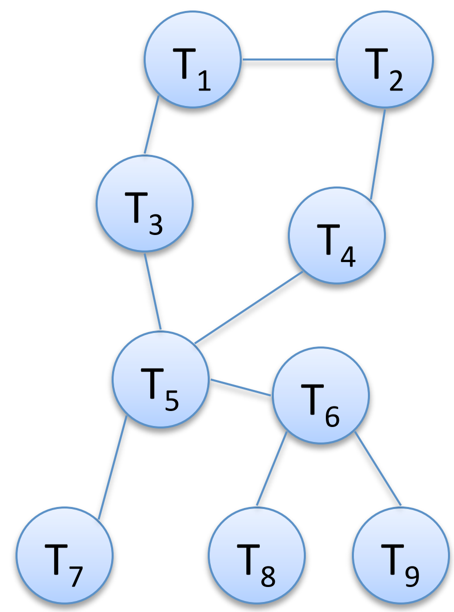
Graphs and Independencies

	Bayesian Networks	**Markov Networks**
local independencies	local Markov assumption	?
global independencies	d-separation	separation

- With Bayesian networks, we had the local Markov assumptions
- Is there anything similar in Markov networks?

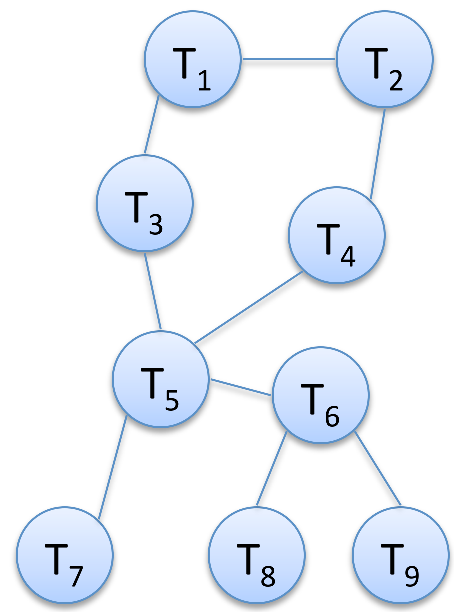
Local Independence Assumptions in Markov Networks

- **Separation** defines global independencies.



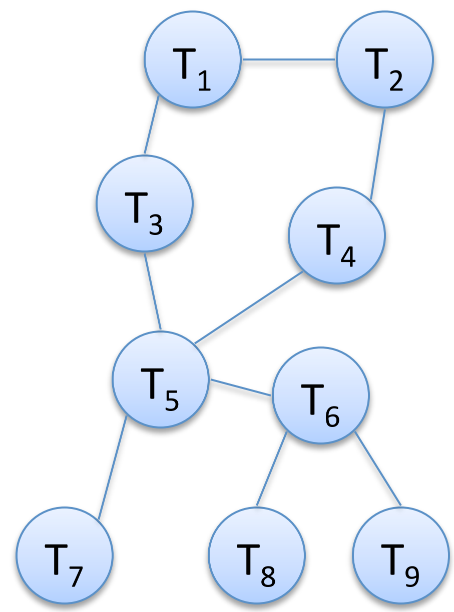
Local Independence Assumptions in Markov Networks

- **Pairwise** Markov independence: pairs of non-adjacent variables are independent given everything else.



Local Independence Assumptions in Markov Networks

- **Markov blanket**: each variable is independent of the rest given its *neighbors*.



Local Independence Assumptions in Markov Networks

- Separation:
  $\text{sep}_{\mathcal{H}}(\mathbf{W}, \mathbf{Y} \mid \mathbf{Z}) \Rightarrow \mathbf{W} \perp \mathbf{Y} \mid \mathbf{Z}$

- Pairwise Markov:
  $A \perp B \mid \mathbf{X} \setminus \{A, B\}$

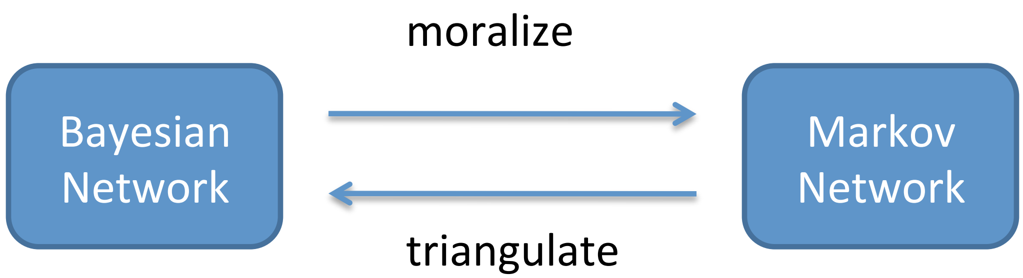
- Markov blanket:
  $A \perp \mathbf{X} \setminus \text{Neighbors}(A) \mid \text{Neighbors}(A)$

Soundness

- For a positive distribution P, the three statements are equivalent:
  - P entails the global independencies of $\mathcal{H}$ (strongest)
  - P entails the Markov blanket independencies of $\mathcal{H}$
  - P entails the pairwise independencies of $\mathcal{H}$ (weakest)
- For nonpositive distributions, we can find cases that satisfy each property, but not the stronger one!
  - Examples in K&F 4.3.

Bayesian Networks and Markov Networks

	Bayesian Networks	**Markov Networks**
local independencies	local Markov assumption	pairwise; Markov blanket
global independencies	d-separation	separation
relative advantages	• v-structures handled elegantly • CPDs are conditional probabilities • probability of full instantiation is easy (no partition function)	• cycles allowed • perfect maps for swinging couples


moralize
Bayesian
Network
Markov
Network
triangulate

From Bayesian Networks to Markov Networks

- Each CPT can be thought of as a factor
- Requires us to connect all parents of each node together
  - Also called "moralization"

From Markov Networks to Bayesian Networks

- Conversion from MN to BN requires triangulation.
  - May lose some independence information.
  - May involve a lot of additional edges.

Summary

- BNs and MNs offer a way to encode a set of independence assumptions
- There is a way to transform from one to another, but it can be at the cost of losing independence assumptions
- This afternoon: inference

Lecture 2: Inference

Inference: An Ubiquitous Obstacle

- Decoding is inference.
- Subroutines for learning are inference.
- Learning is inference.

- 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 $\mathbf{V} \setminus \mathbf{X}$?

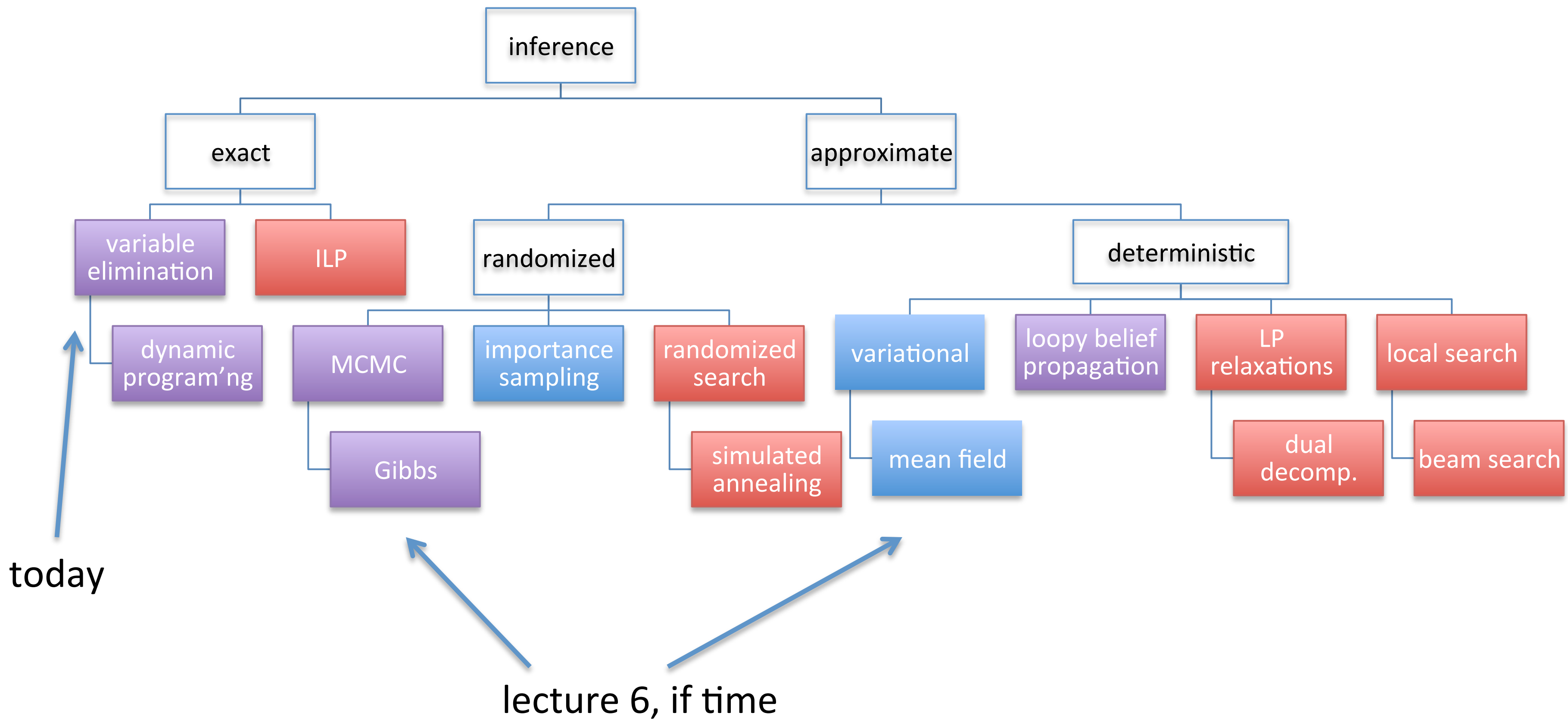
(More generally …)

- 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

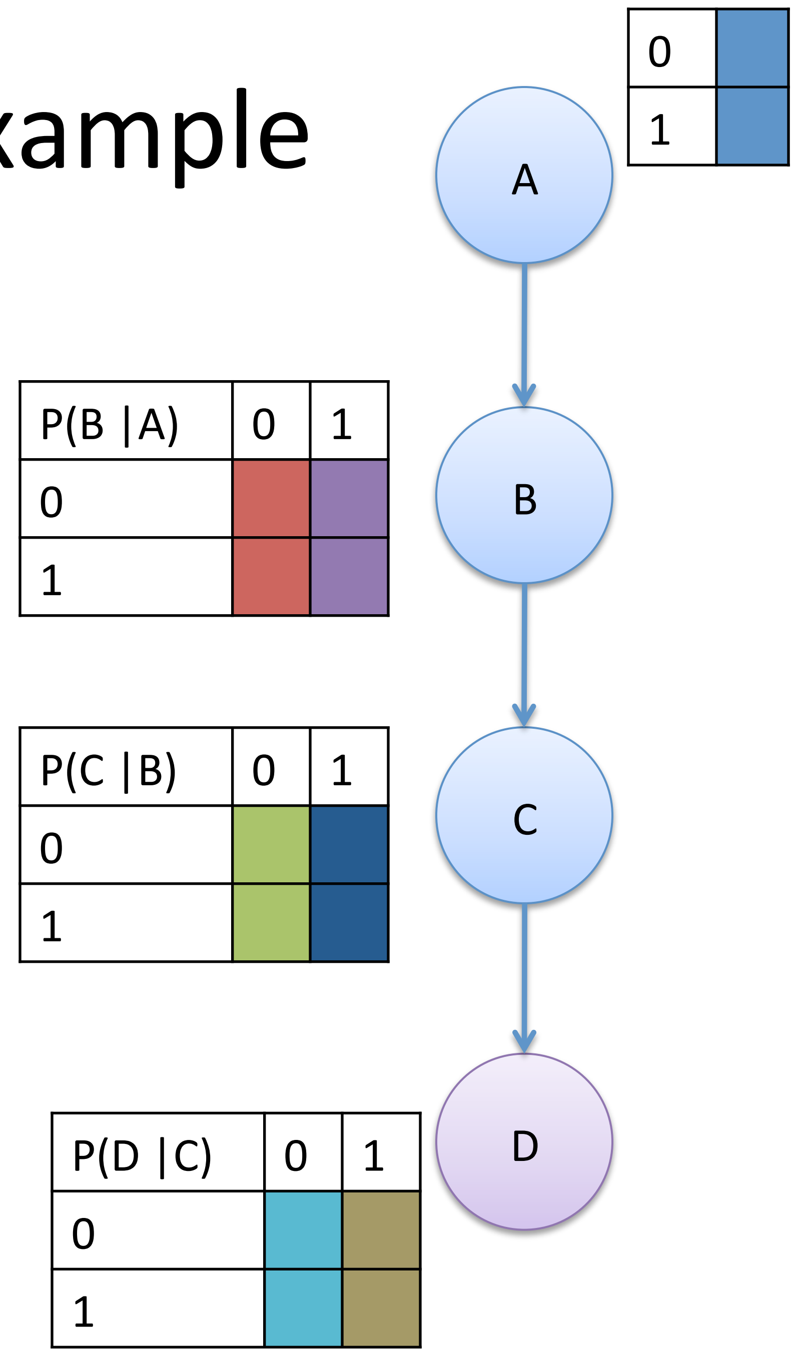


Exact Marginal for Y

- This will be a generalization of algorithms you already know: 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.

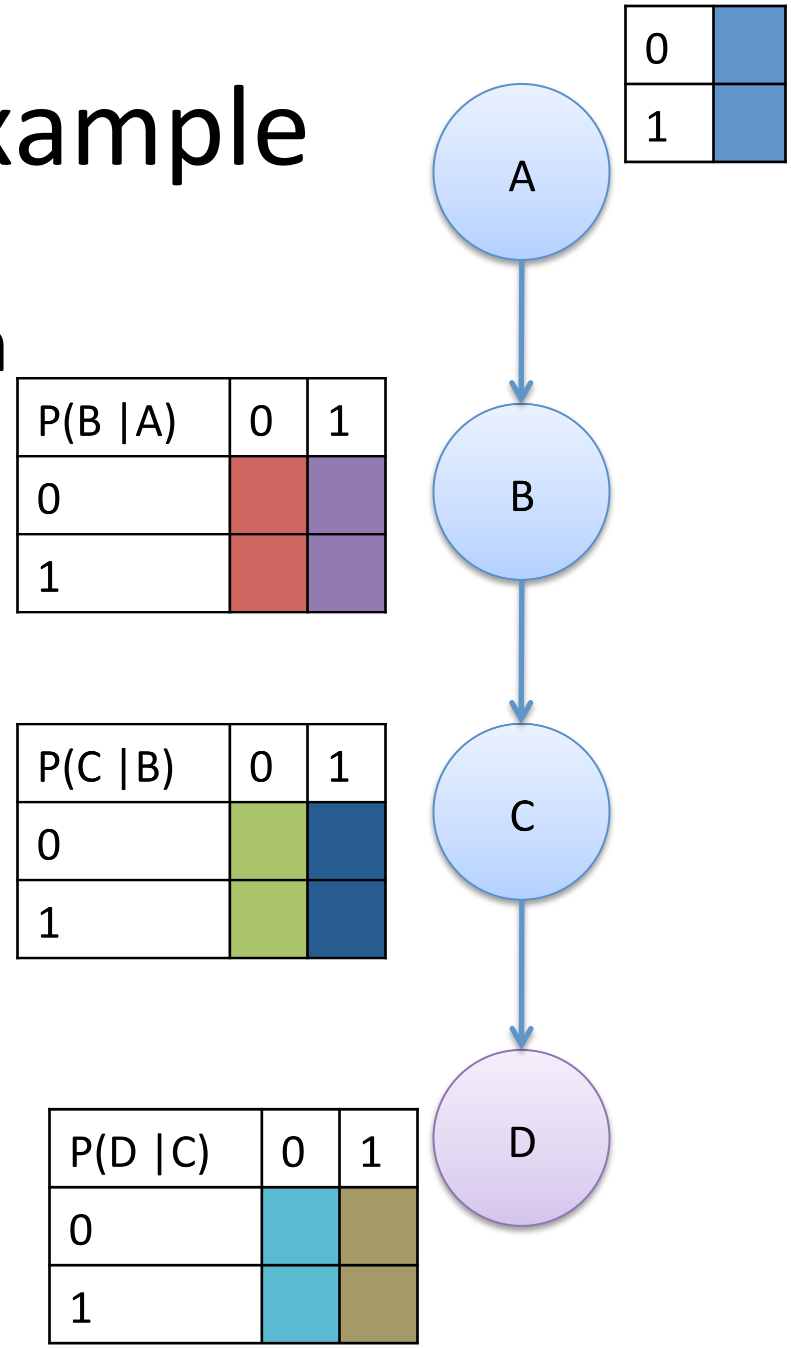
Simple Inference Example

- Goal: P(D)



Simple Inference Example

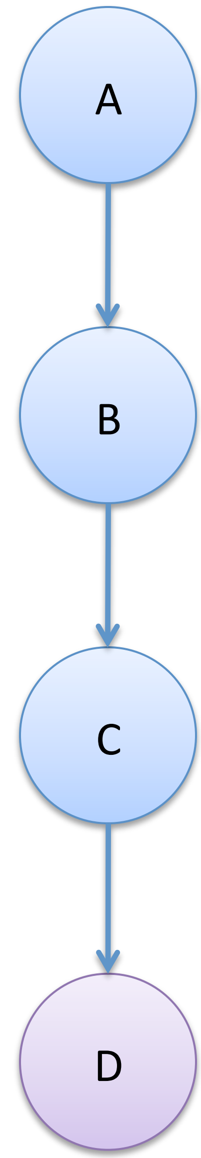
- Let’s calculate P(B) from things we have.



Simple Inference Example

- Let' s calculate P(B) from things we have.

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

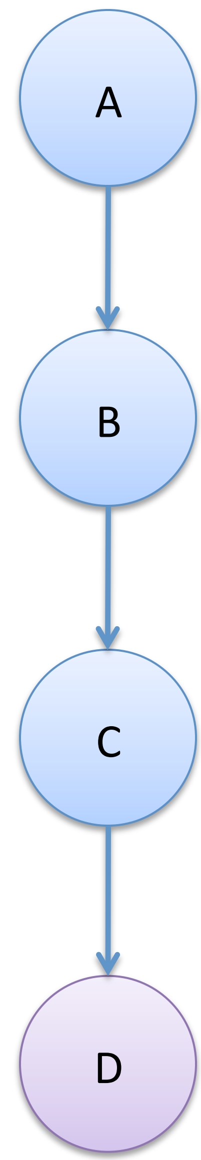


Simple Inference Example

- Let’s calculate P(B) from things we have.

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

- Note that C and D do not matter.



Simple Inference Example

- Let's calculate P(B) from things we have.

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

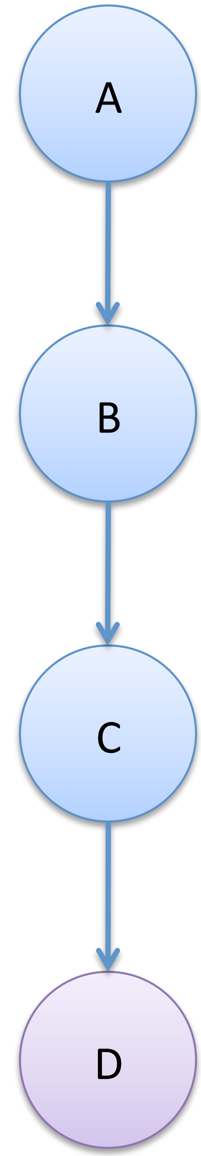
0	
1	

T

P(B | A)	0	1
0		
1		

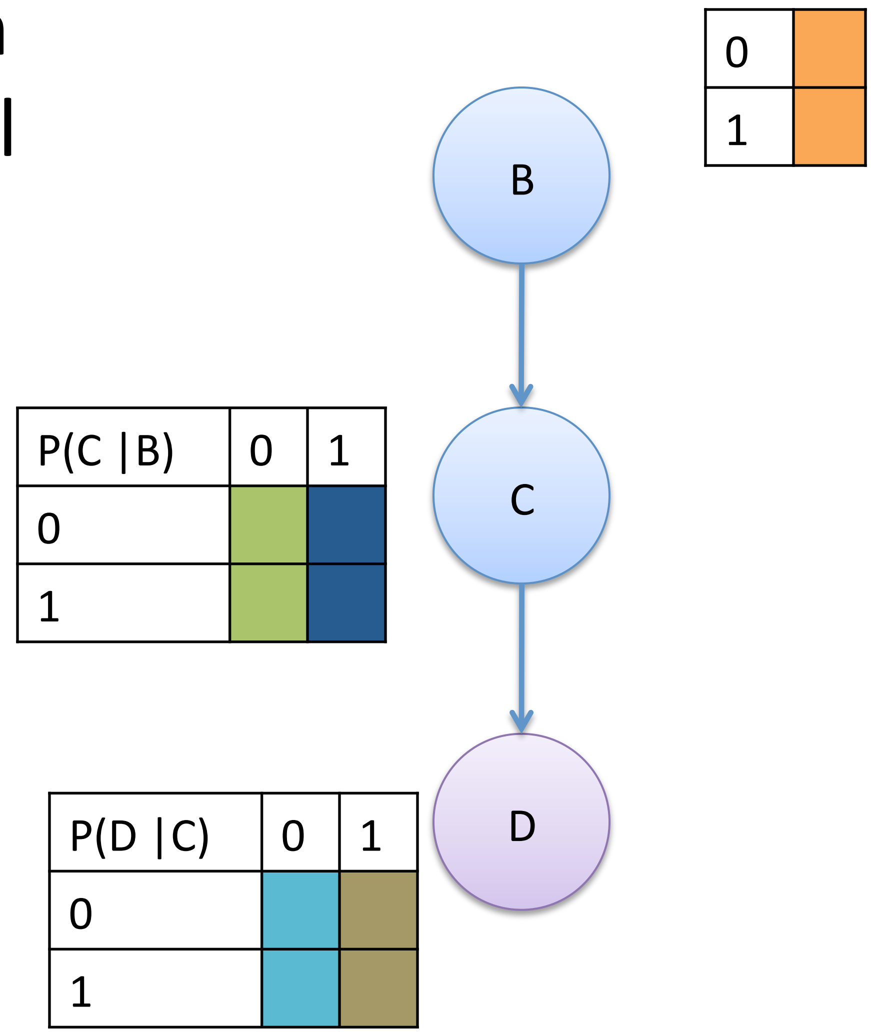
=

0	
1	



Simple Inference Example

- We now have a Bayesian network for the marginal distribution P(B, C, D).

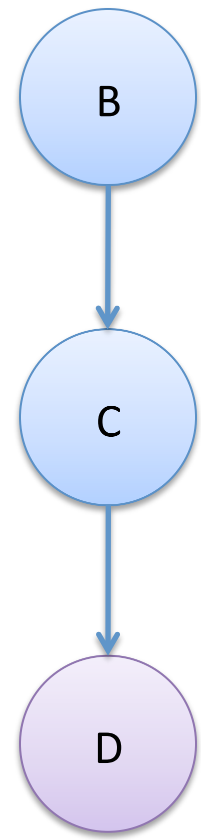


Simple Inference Example

- We can repeat the same process to calculate P(C).

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

- We already have P(B)!



Simple Inference Example

- We can repeat the same process to calculate P(C).

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

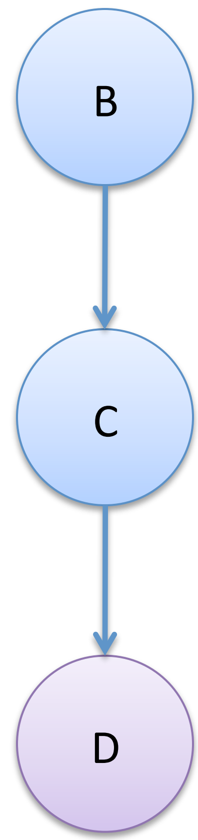
0	
1	

T

P(C | B)	0	1
0		
1		

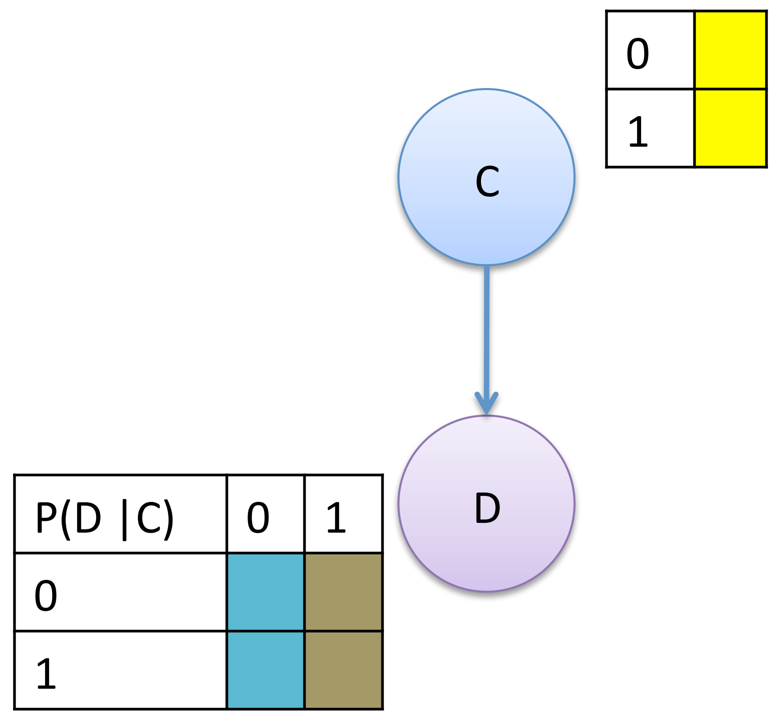
=

0	
1	



Simple Inference Example

- We now have P(C, D).
- Marginalizing out A and B happened in two steps, and we are exploiting the Bayesian network structure.



Simple Inference Example

- Last step to get P(D):

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

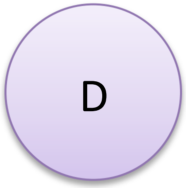
0	
1	

T

P(D | C)	0	1
0		
1		

=

0	
1	



Simple Inference Example

- Notice that the same step happened for each random variable:
  - We created a new CPD 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 Bayesian network structure (each subexpression only depends on a small number of variables).
- Exponential blowup avoided!

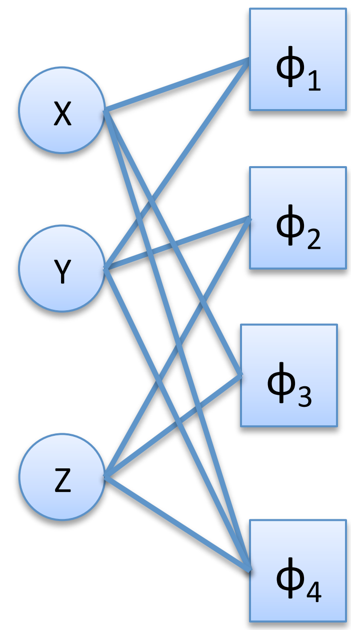
What Remains

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

Factor Graphs

- Variable nodes (circles)
- Factor nodes (squares)
  - Can be MN factors or BN conditional probability distributions!
- Edge between variable and factor if the factor depends on that variable.

- The graph is bipartite.



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 Marginalization

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

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

Factor Marginalization

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

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

P(C | A, B)	0, 0	0, 1	1, 0	1,1
0	0.5	0.4	0.2	0.1
1	0.5	0.6	0.8	0.9

"summing out" B

A	C	ψ(A, C)
0	0	0.9
0	1	0.3
1	0	1.1
1	1	1.7

Factor Marginalization

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

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

P(C | A, B)	0, 0	0, 1	1, 0	1,1
0	0.5	0.4	0.2	0.1
1	0.5	0.6	0.8	0.9

"summing out" C

A	B	ψ(A, B)
0	0	1
0	1	1
1	0	1
1	1	1

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:

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

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

Marginalizing Everything?

- Take a Markov network's "product 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: $\sum_X \sum_Y \varphi = \sum_Y \sum_X \varphi$
- Distributivity of multliplication over summation:

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

Eliminating One Variable

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

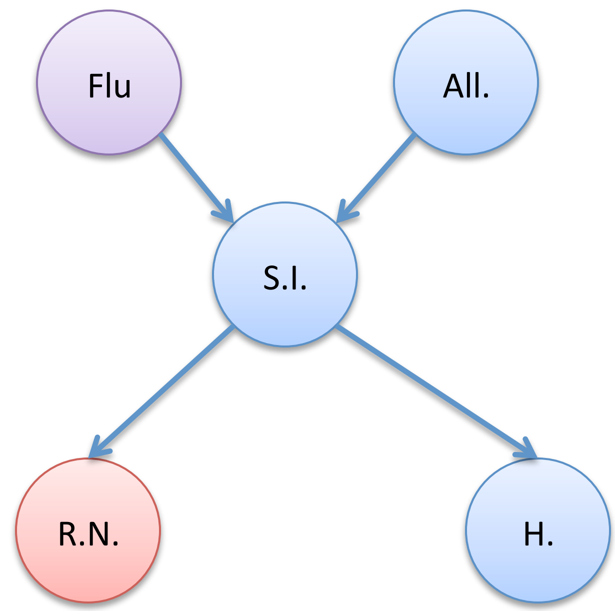
Output: new set of factors $\mathbf{\Psi}$

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)

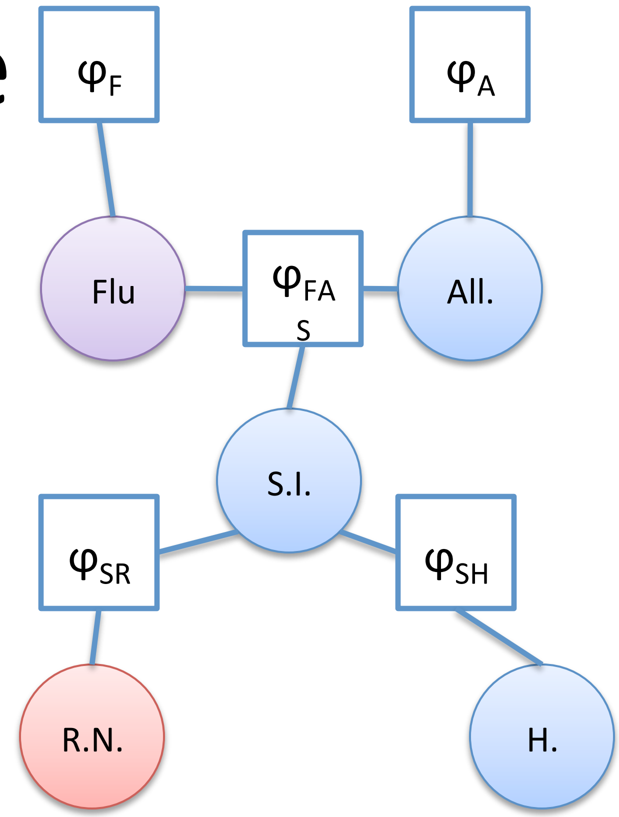
- Let's eliminate H.



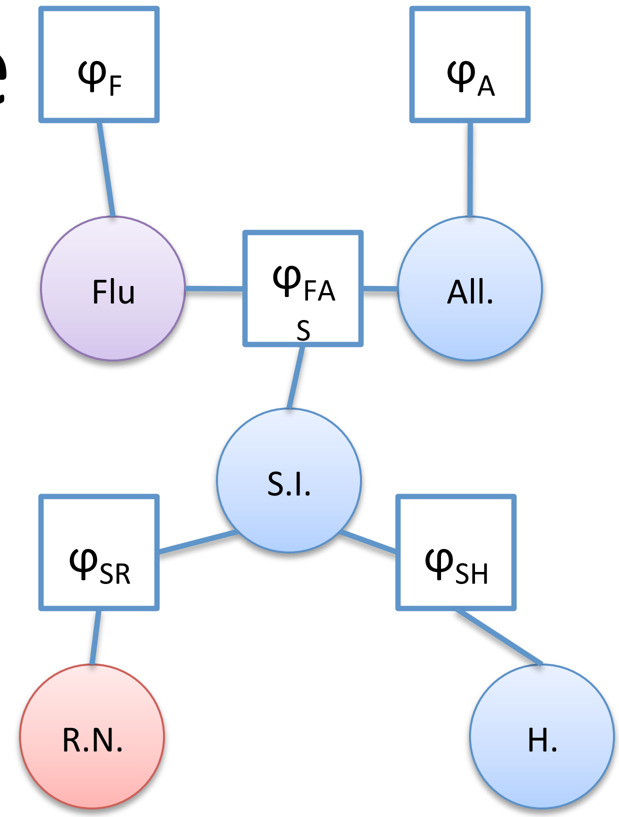
Example

- Query:
  P(Flu | runny nose)

- Let's eliminate H.



Example



- Query: P(Flu | runny nose)

- Let's eliminate H.

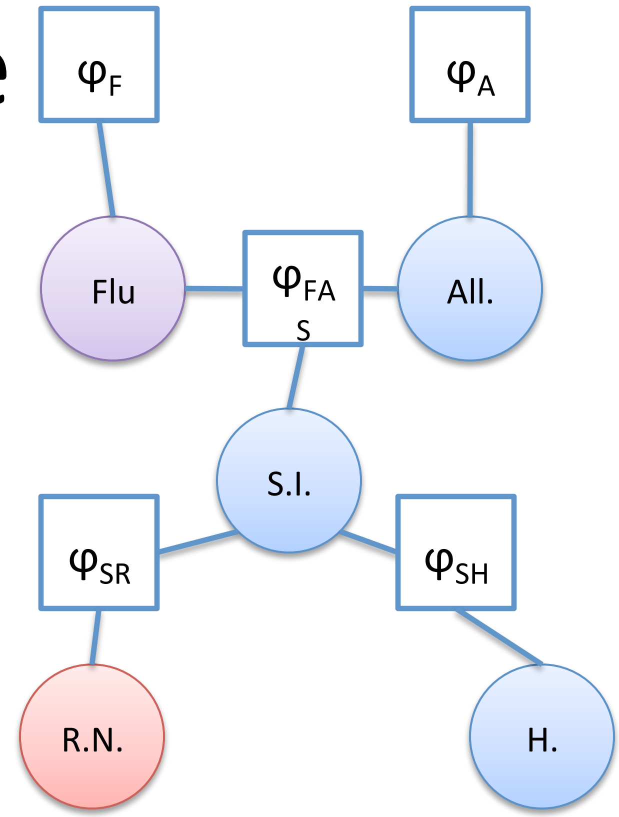
1. $\mathbf{\Phi}' = \{\varphi_{SH}\}$
2. $\mathbf{\Psi} = \{\varphi_F, \varphi_A, \varphi_{FAS}, \varphi_{SR}\}$
3. $\psi = \sum_H \prod_{\varphi \in \mathbf{\Phi}'} \varphi$
4. Return $\mathbf{\Psi} \cup \{\psi\}$

Example

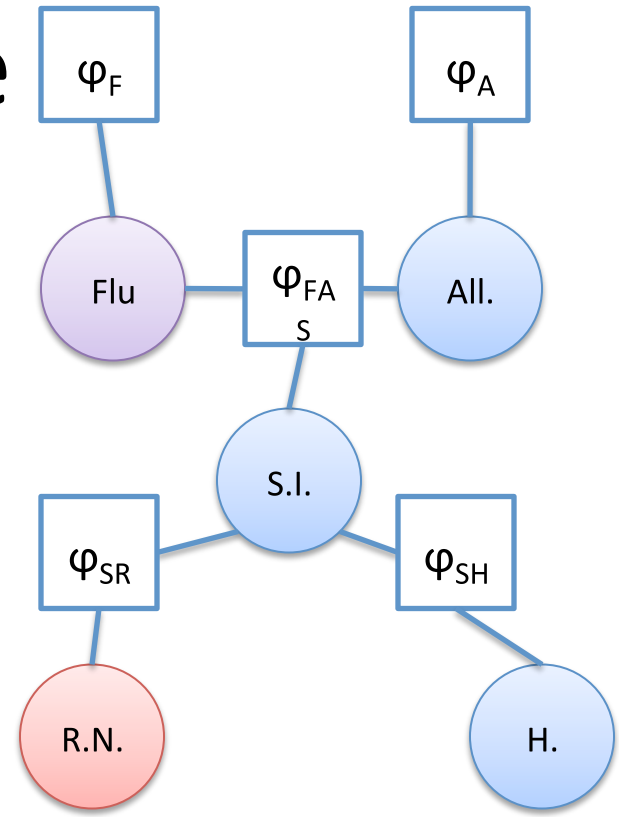
- Query:
  P(Flu | runny nose)

- 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)

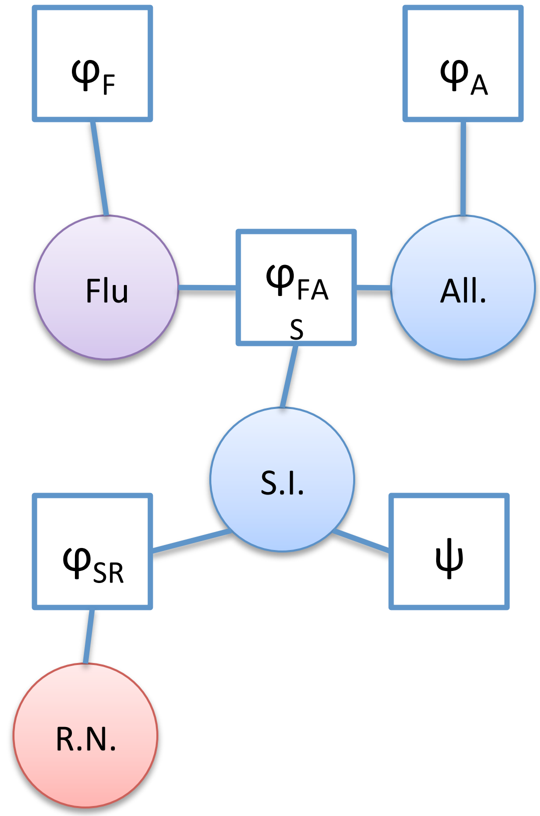
- Let's eliminate H.

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

P(H | S)	0	1
0	0.8	0.1
1	0.2	0.9

S	ψ(S)
0	1.0
1	1.0

Example



- Query:
  P(Flu | runny nose)

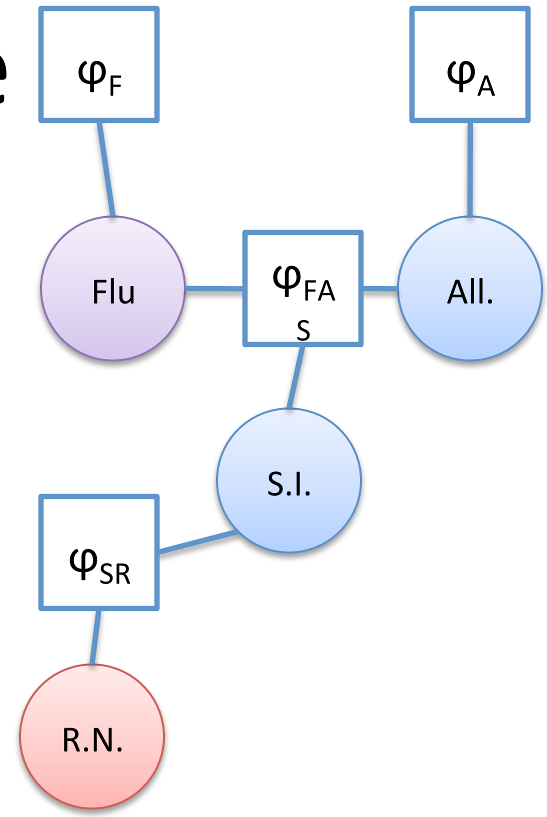
- Let's eliminate H.

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

P(H | S)	0	1
0	0.8	0.1
1	0.2	0.9

S	ψ(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

Variable Elimination

Input: Set of factors **Φ**, ordered list of variables **Z** to eliminate

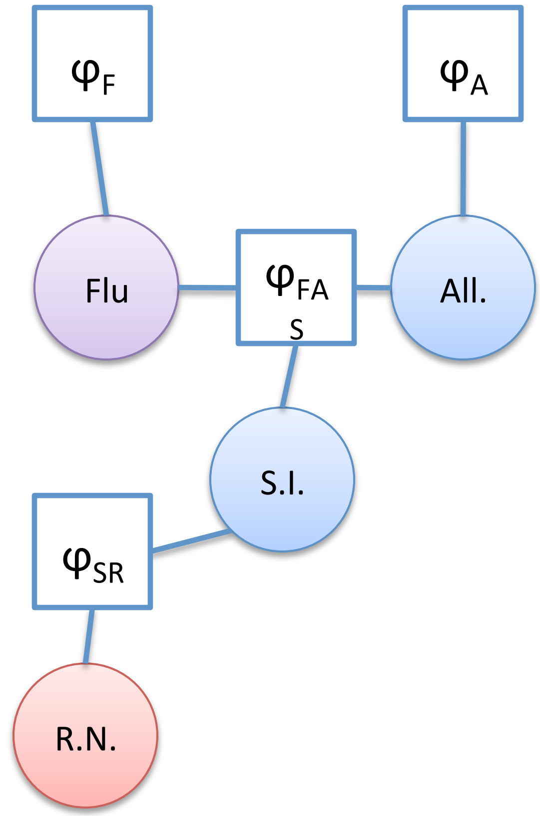
Output: new factor ψ

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

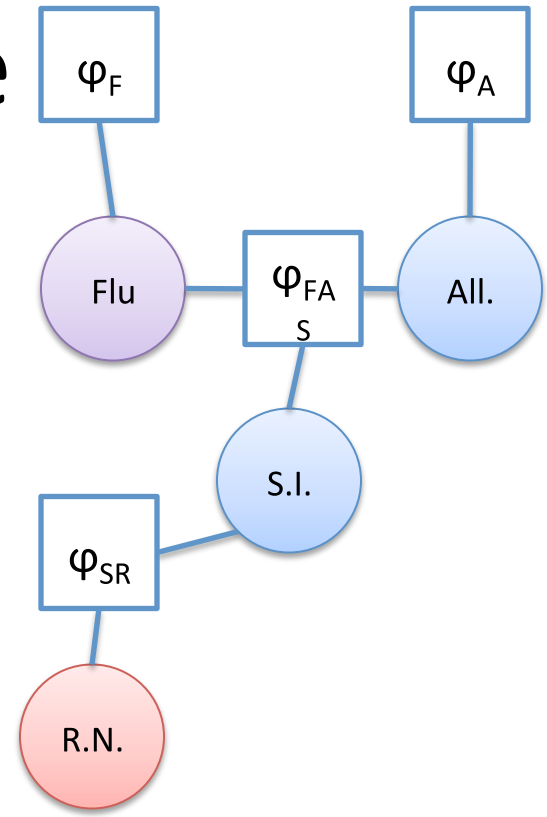
Example

- Query:
  P(Flu | runny nose)

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



Example

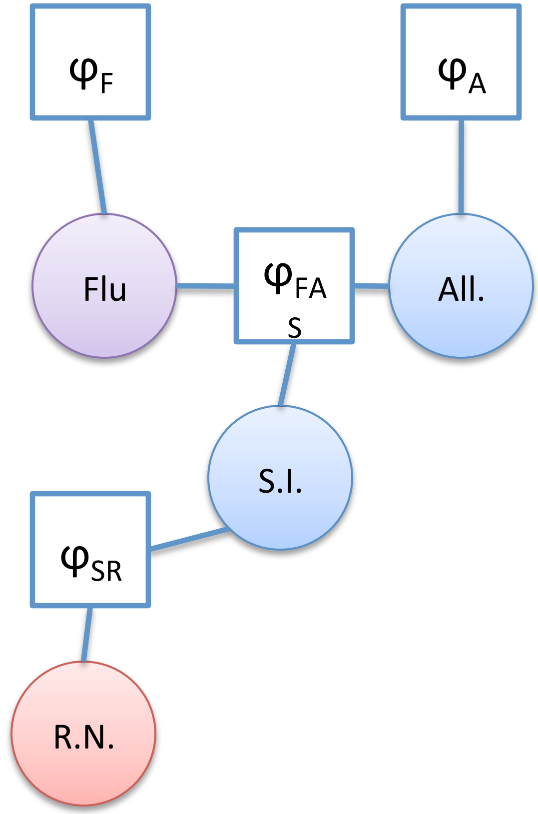


- Query:
  P(Flu | runny nose)

- Eliminating S.

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

Example

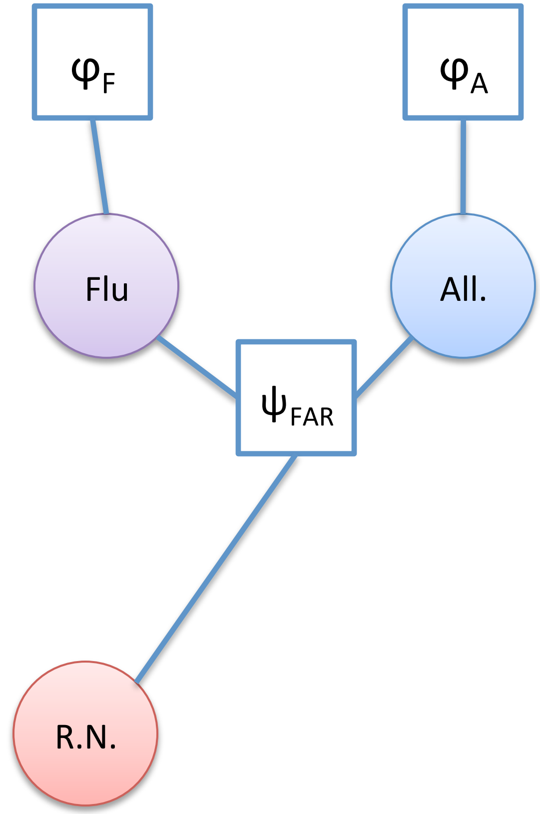


- Query: P(Flu | runny nose)

- 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)

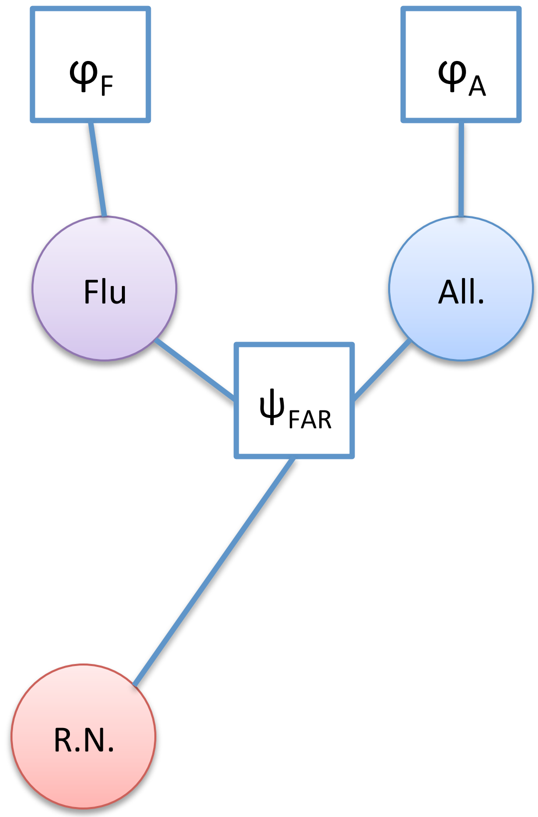
- Eliminating S.

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 $\mathbf{\Psi} \cup \{\psi_{FAR}\}$

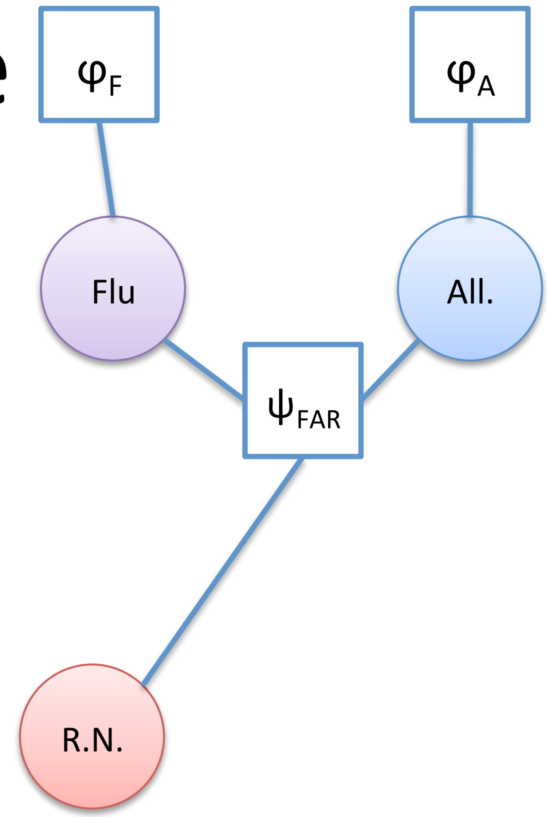
Example

- Query:
  P(Flu | runny nose)

- Finally, eliminate A.



Example

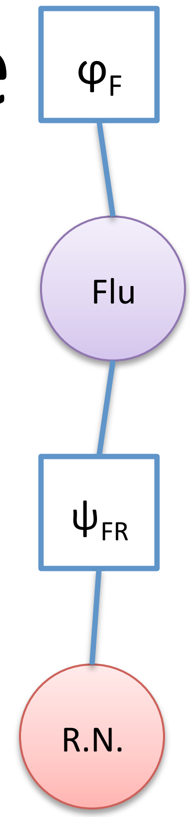


- Query: P(Flu | runny nose)

- 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



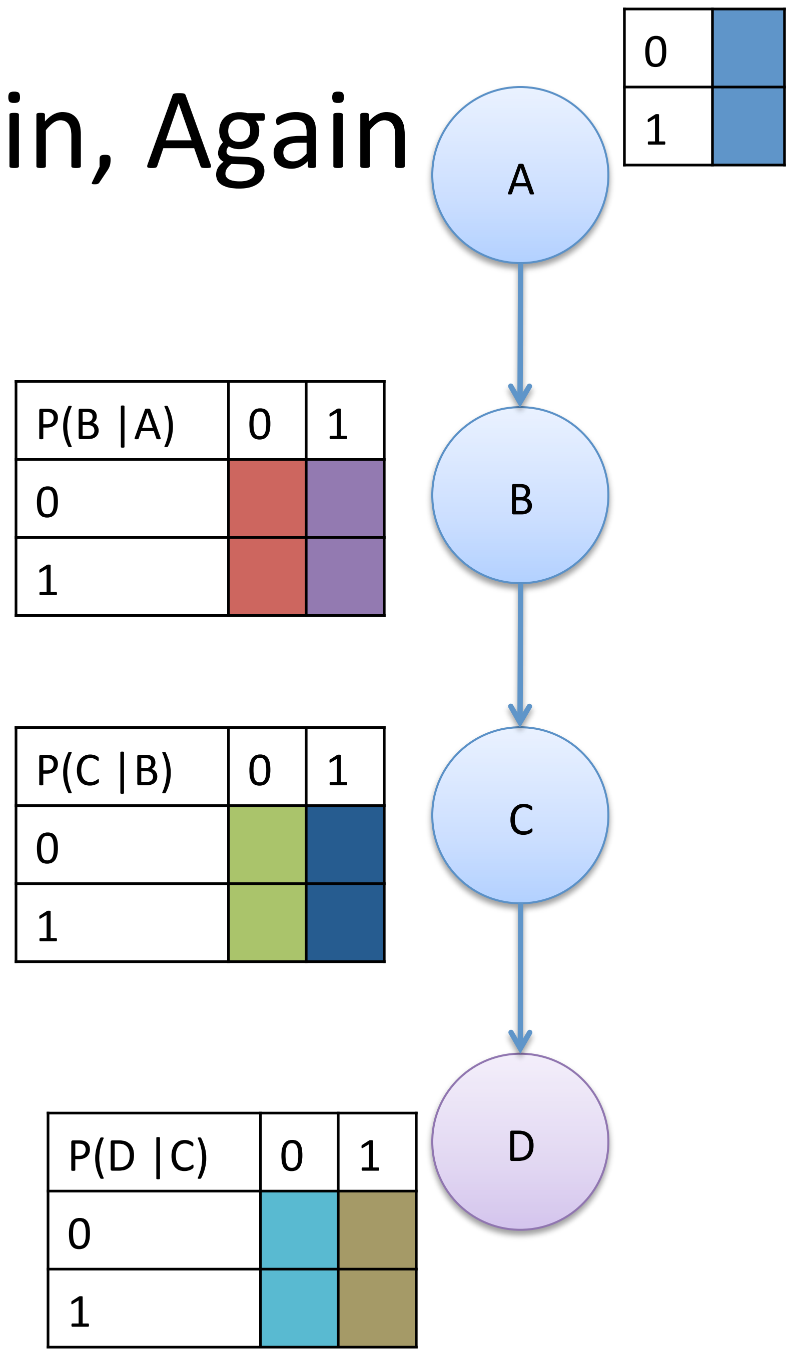
- Query:
  P(Flu | runny nose)

- 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 $\mathbf{\Psi} \cup \{\psi_{FR}\}$

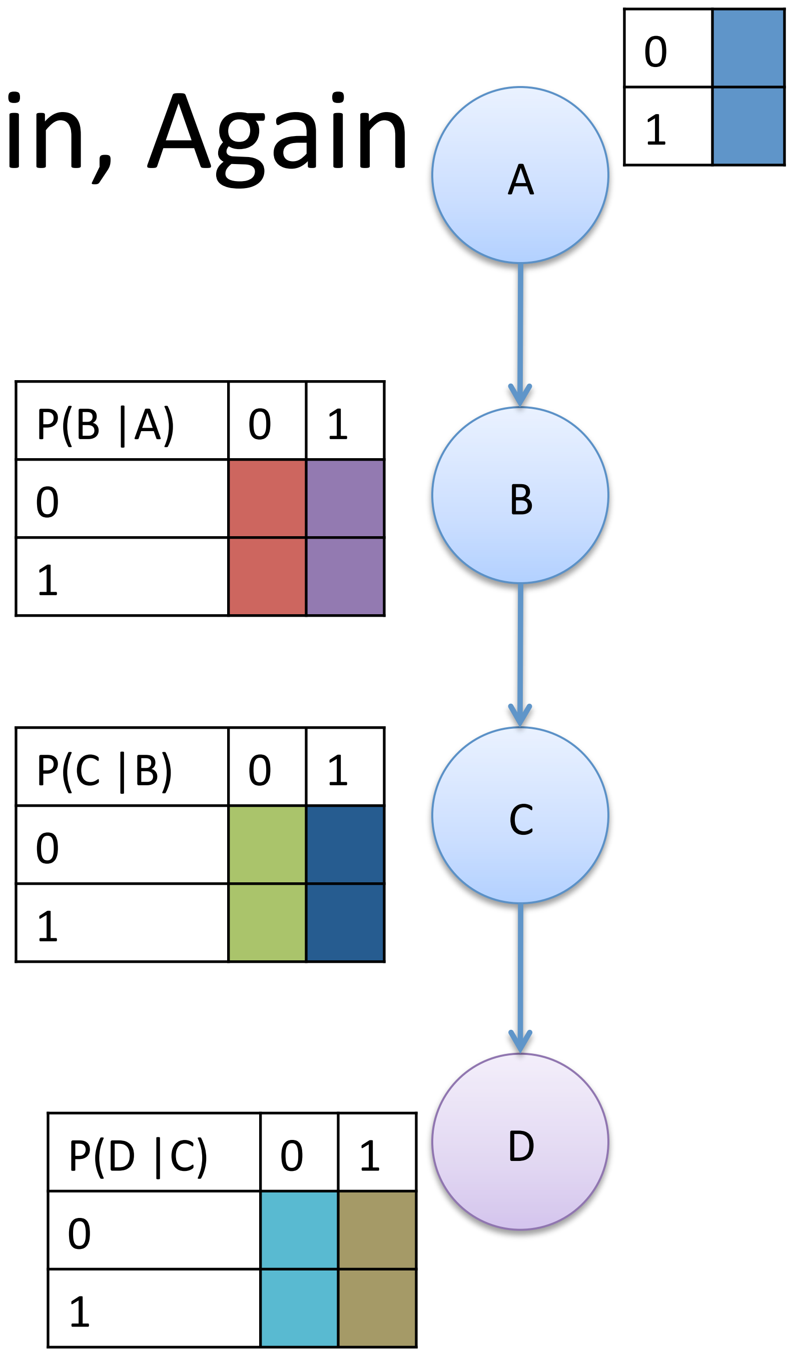
Markov Chain, Again

- Earlier, we eliminated A, then B, then C.



Markov Chain, Again

- Now let's start by eliminating C.



Markov Chain, Again

- Now let's start by eliminating C.

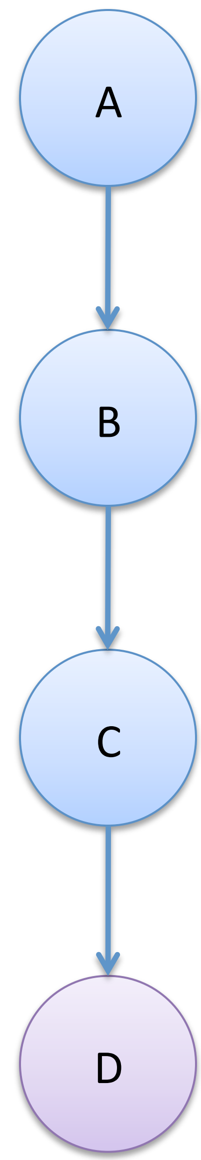
P(C |B)	0	1
0		
1		

·

P(D |C)	0	1
0		
1		

=

B	C	D	φ' (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	



Markov Chain, Again

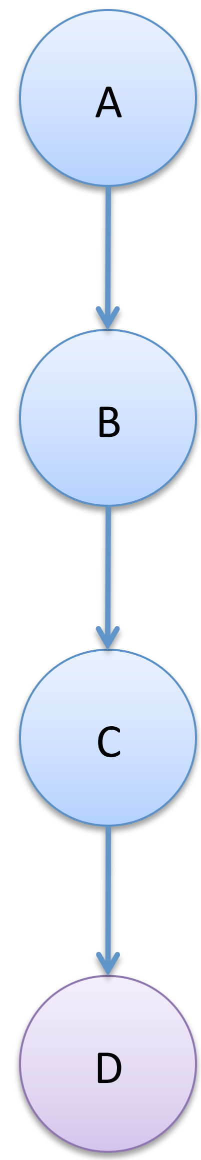
- Now let’s start by eliminating C.

$\sum_C$

B	C	D	φ’(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	

=

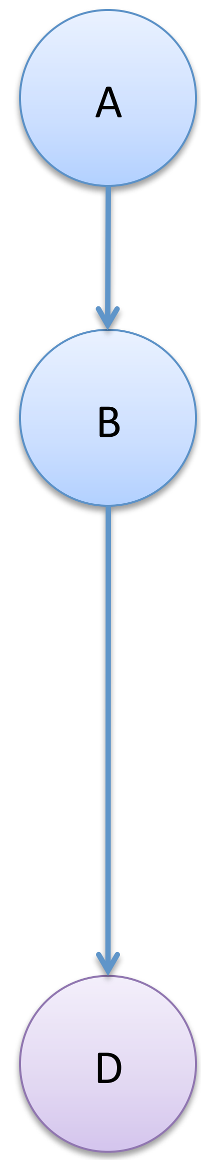
B	D	ψ(B, D)
0	0	
0	1	
1	0	
1	1	



Markov Chain, Again

- Eliminating B will be similarly complex.

B	D	ψ(B, D)
0	0	
0	1	
1	0	
1	1	



Variable Elimination: Comments

- Can prune away all non-ancestors of the query variables.
- Ordering makes a difference!
- Works for Markov networks and Bayesian networks.
  - Factors need not be CPDs and, in general, new factors won't be.

What about Evidence?

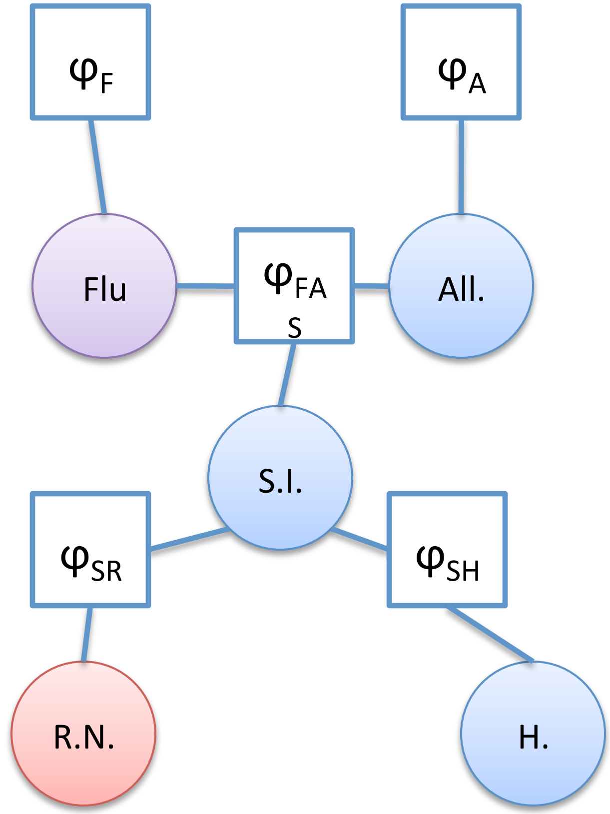
- So far, we've just considered the posterior/ marginal P(Y).
- Next: conditional distribution P(Y | **X** = **x**).

- 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).

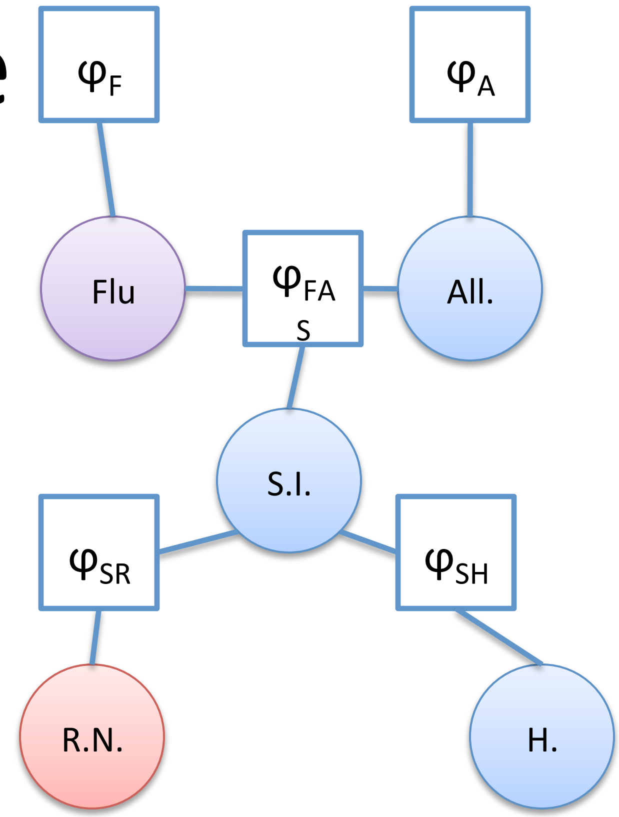


P(R | S)	0	1
0		
1		

Example

- Query:
  P(Flu | runny nose)

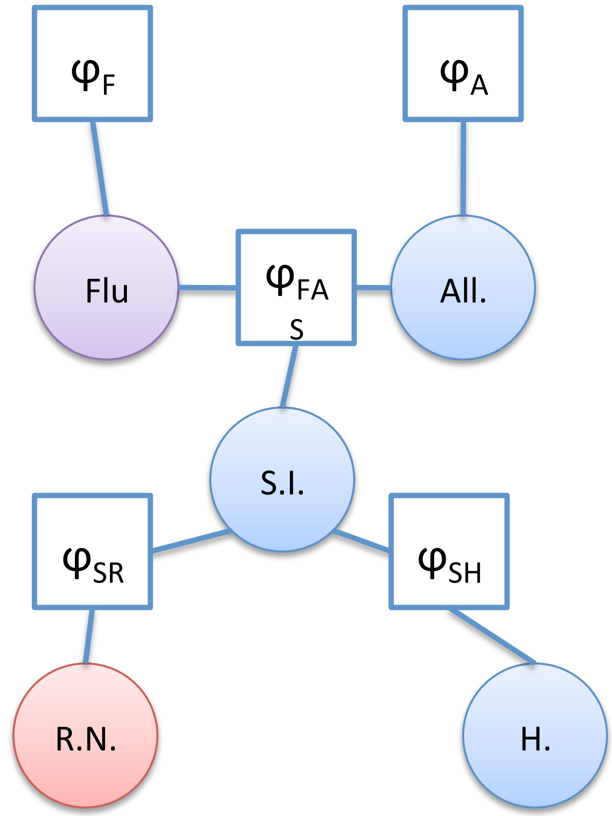
- Let's reduce to R = true (runny nose).



P(R | S)	0	1
0		
1		

S	R	φ_{SR} (S, R)
0	0	
0	1	
1	0	
1	1	

Example



- Query:
  P(Flu | runny nose)

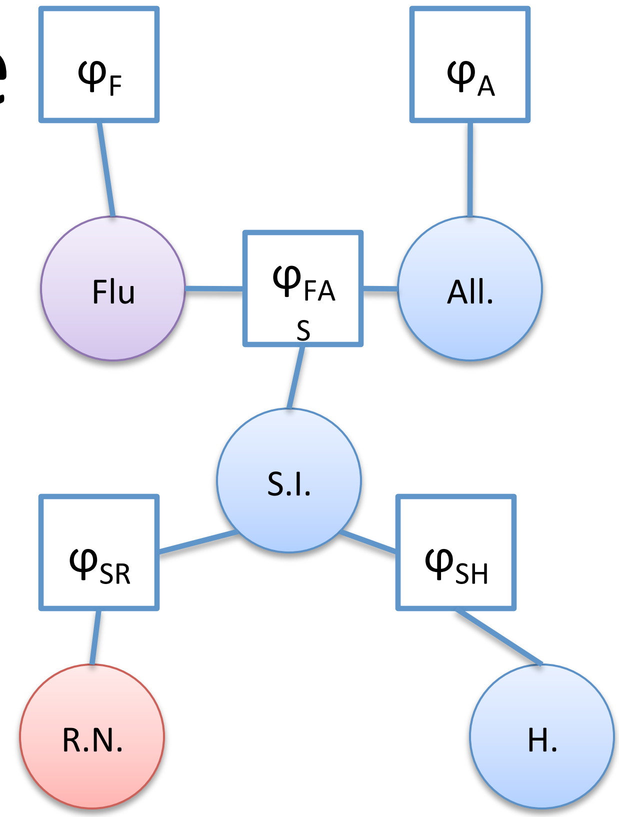
- Let's reduce to R = true (runny nose).

P(R | S)	0	1
0		
1		

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	

Example



- Query:
  P(Flu | runny nose)

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

P(R | S)	0	1
0		
1		

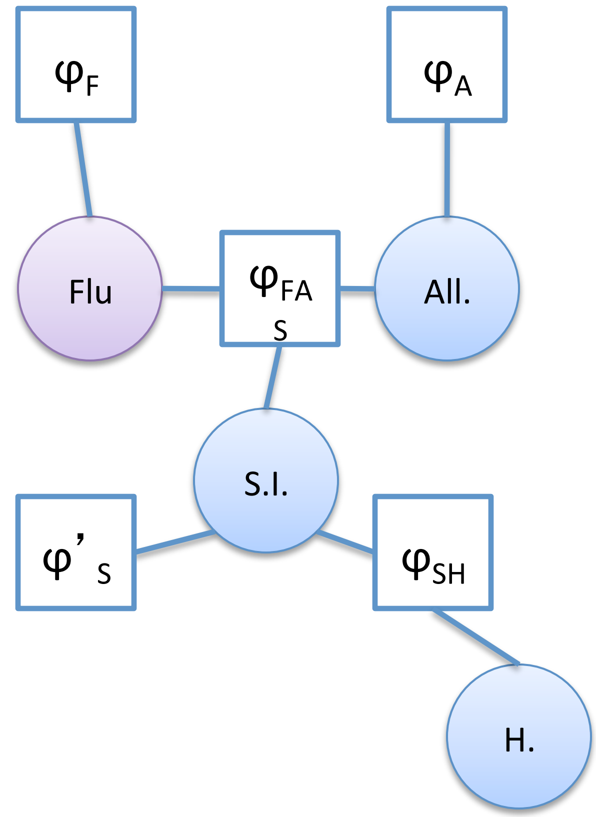
S	R	φ_{SR} (S, R)
0	0	
0	1	
1	0	
1	1	

S	R	φ'_S (S)
0	1	
1	1	

Example

- Query:
  P(Flu | runny nose)

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

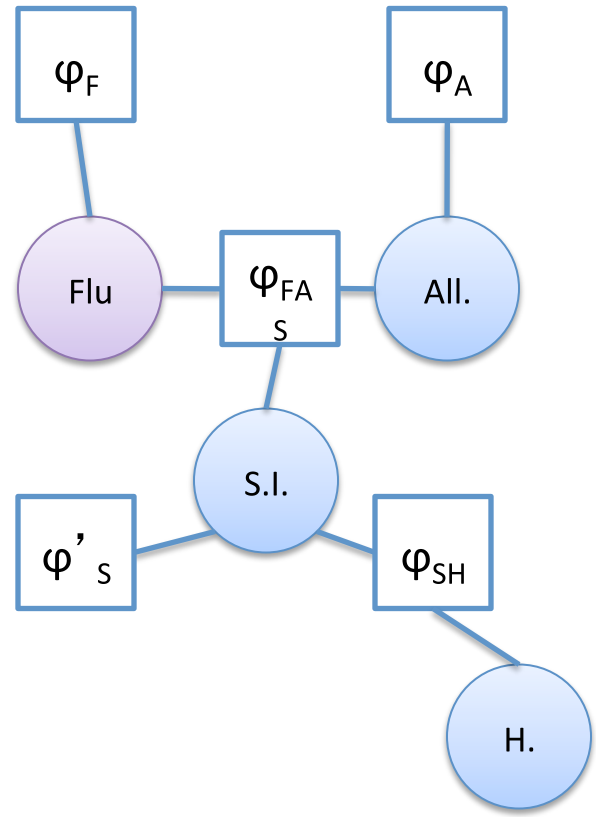


S	R	φ' S (S)
0	1	
1	1	

Example

- Query:
  P(Flu | runny nose)

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

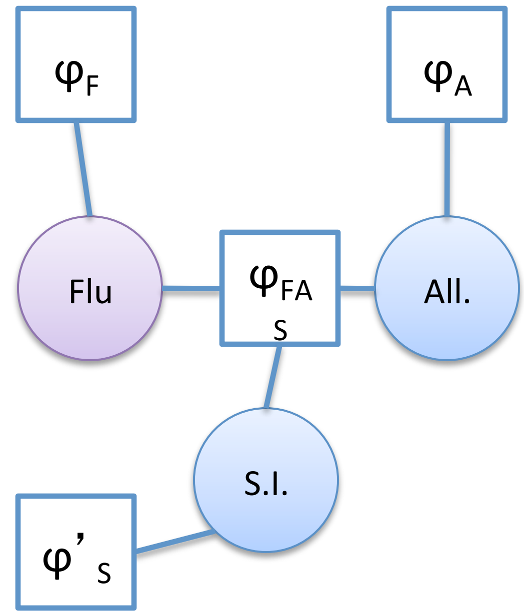


H can be pruned for the same reasons as before.

Example

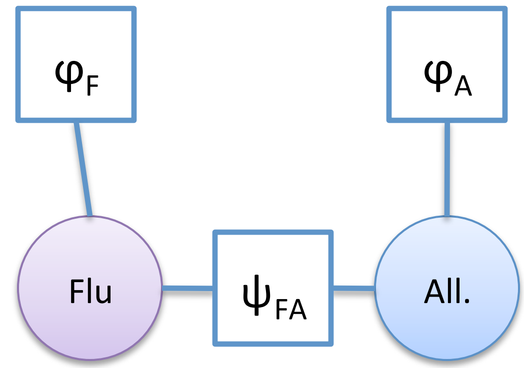
- Query:
  P(Flu | runny nose)

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



Eliminate S.

Example

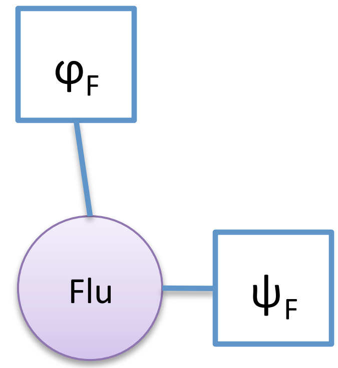


Eliminate A.

- 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)

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

$\varphi_F \cdot \psi_F$

Variable Elimination for Conditional Probabilities

Input: Graphical model on **V**, set of query variables **Y**, evidence **X** = **x**

Output: factor φ and scalar α

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

Note

- For Bayesian networks, the final factor will be $P(\mathbf{Y}, \mathbf{X} = \mathbf{x})$ and the sum $\alpha = P(\mathbf{X} = \mathbf{x})$.
- This equates to a Gibbs distribution with partition function = α.

Variable Elimination

- In general, exponential requirements in induced width corresponding to the ordering you choose.
- It's NP-hard to find the best elimination ordering.
- If you can avoid "big" intermediate factors, you can make inference linear in the size of the original factors.

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.

Getting Back to NLP

- Traditional structured NLP models were sometimes subconsciously chosen for these properties.
  - HMMs, PCFGs (with a little work)
  - But not: IBM model 3
- Need MAP inference for decoding!
- Need approximate inference for complex models!

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.

Factor Maximization

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

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

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

Factor Maximization

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

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

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

Distributive Property

- A useful property we exploited in variable elimination:

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

- Under the same conditions, factor multiplication distributes over max, too:

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