Graphical Models

Lecture 9:

Variable Elimination

Andrew McCallum
mccallum@cs.umass.edu

Thanks to Noah Smith and Carlos Guestrin for some slide materials.

Probabilistic Inference

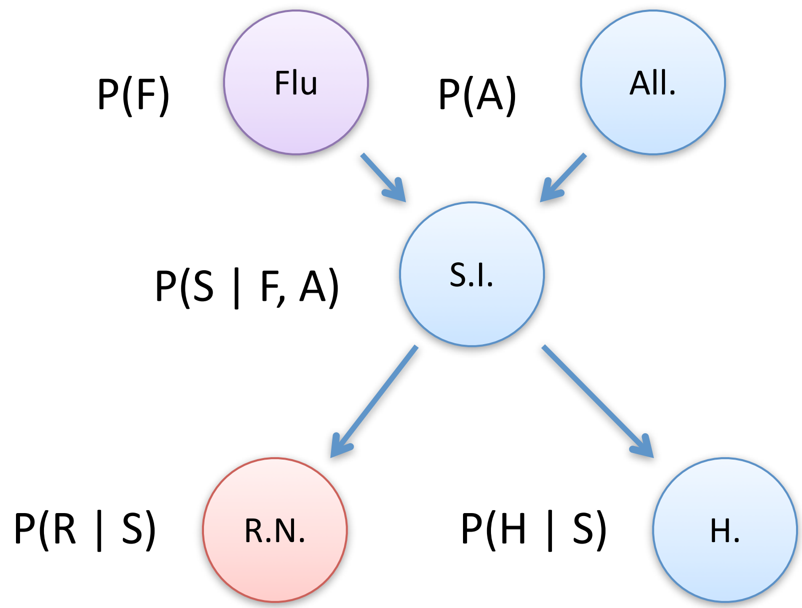
- Assume we are given a graphical model.
- Want:

$$\begin{aligned} P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e}) &= \frac{P(\boldsymbol{X}, \boldsymbol{E} = \boldsymbol{e})}{P(\boldsymbol{E} = \boldsymbol{e})} \\ &\propto P(\boldsymbol{X}, \boldsymbol{E} = \boldsymbol{e}) \\ &= \sum_{\boldsymbol{y} \in \mathrm{Val}(\boldsymbol{Y})} P(\boldsymbol{X}, \boldsymbol{E} = \boldsymbol{e}, \boldsymbol{Y} = \boldsymbol{y}) \end{aligned}$$

Example

- Query:
  P(Flu | runny nose)

- Requires accounting for all possibilities for A, S, and H.



$$
\begin{aligned}
P(F \mid R) &= \frac{P(F, R)}{P(R)} \\
&= \frac{\sum_{A,S,H} P(F, A, S, R, H)}{\sum_{F,A,S,H} P(F, A, S, R, H)}
\end{aligned}
$$

Probabilistic Inference

- In general it is intractable. ☹
- In practice we can often do it anyway.
- Today: Exact inference via variable elimination.

- Later:
  - *Approximate* inference
  - Maximum *a posteriori* inference (find the best explanation, not necessarily the whole distribution)

Decision Version of Inference

- Given a Bayesian network over **X**, and a value $x \in Val(X_i)$ decide whether $P(X_i = x) > 0$.

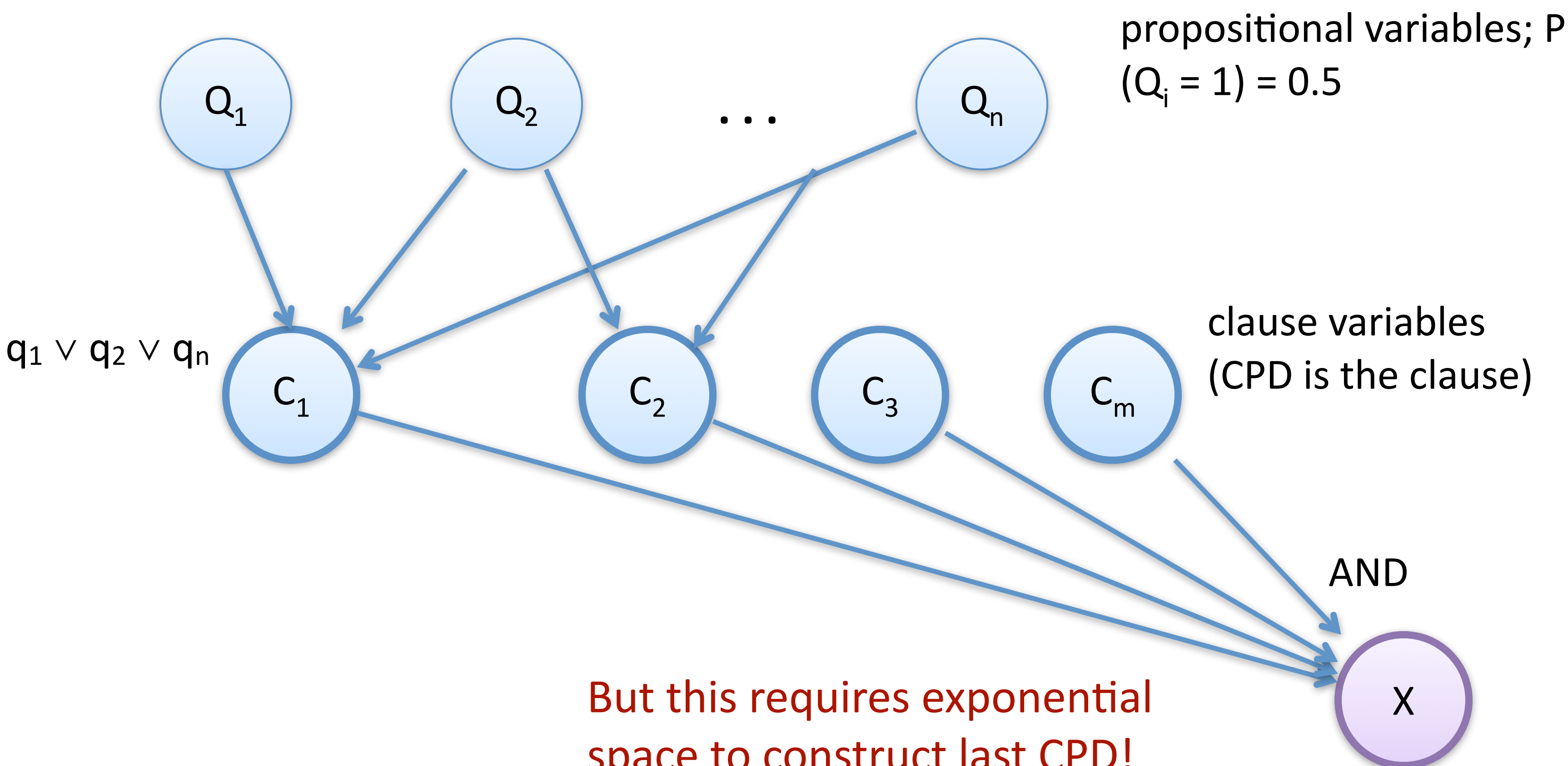
- Witness is full set of random variables **x** such that $x_i = x$.
- Can verify that P(**X** = **x**) > 0 in polynomial time.
- Therefore this problem is in NP.

Reduction from 3-SAT

- Given a 3-SAT formula, construct a Bayesian network and variable X such that the decision inference solution yields a 3-SAT solution.

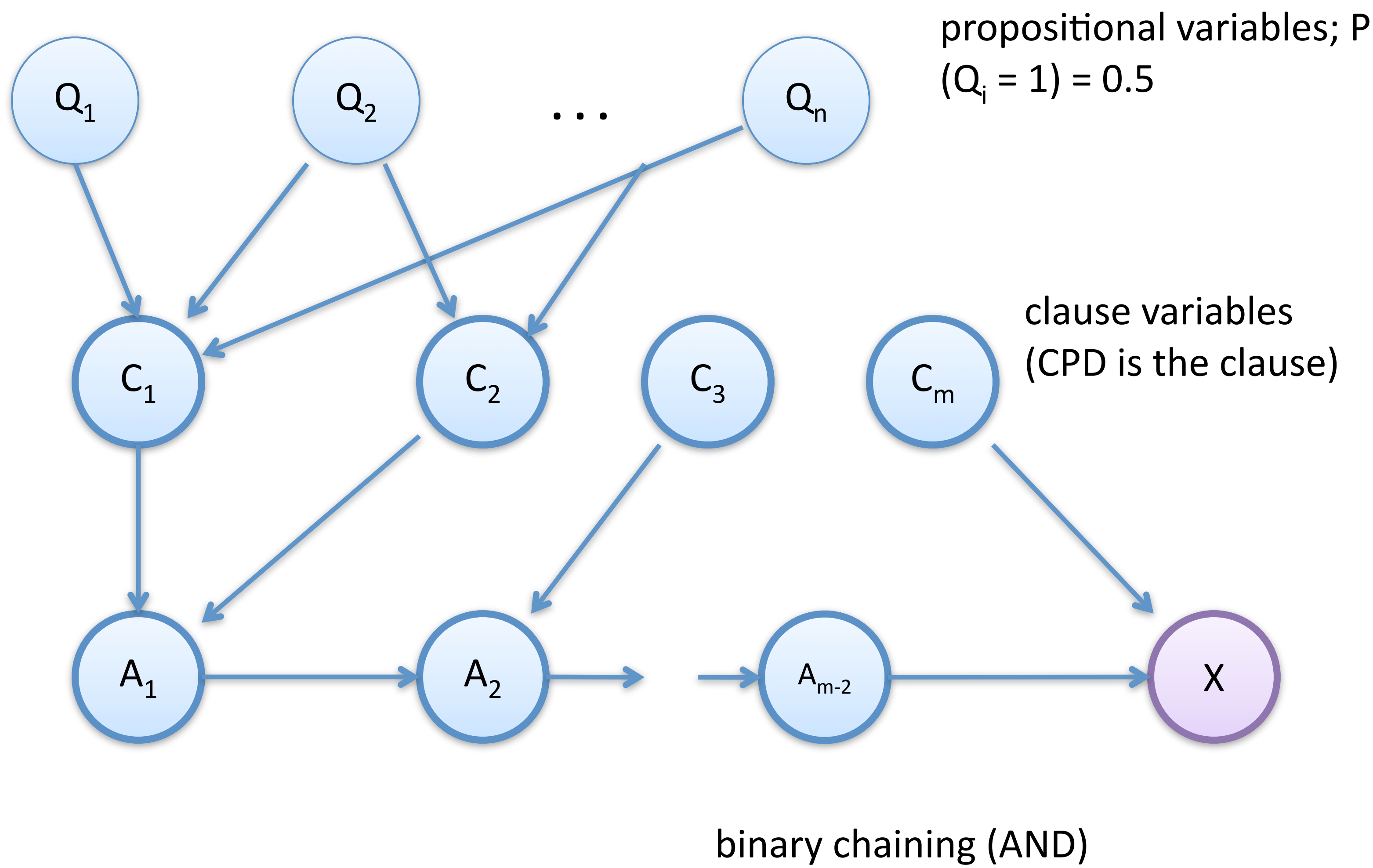
Remember what 3-SAT structure is?

3-SAT Bayesian Network



But this requires exponential space to construct last CPD!

3-SAT Bayesian Network



3-SAT Bayesian Network

- $P(X = 1 \mid \mathbf{Q} = \mathbf{q})$ if and only if **q** is a satisfying assignment for the 3-SAT problem.

- $P(X = 1)$ is the number of satisfying assignments divided by 2^n.
  - If positive, the formula is satisfiable.

The Real Inference Problem

- Given a Bayesian network over **X**, and a value x ∈ Val(X_i), compute P(X_i = x).

- Similar to problems of the form "how many solutions satisfy the requirements?"
- This problem is **#P-complete**.
  – *#P-complete* problems: if a poly-time algorithm exists, then P = PH and therefore P = NP.

PH contains P, NP and Co-NP.

Exact inference is hopeless in general.

Approximations: Absolute Error

- Let ρ denote an estimate:

$$|P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{E} = \boldsymbol{e}) - \rho| \quad \leq \quad \epsilon$$

- Inappropriate for rare events!
- Approximating P(**X** = **x**) up to some fixed absolute error ε has a randomized polynomial time algorithm.
- But this goes away with evidence for $\varepsilon < 0.5$.

Approximations: Relative Error

$$\frac{\rho}{1+\epsilon} \quad \leq \quad P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{E} = \boldsymbol{e}) \quad \leq \quad \rho(1+\epsilon)$$

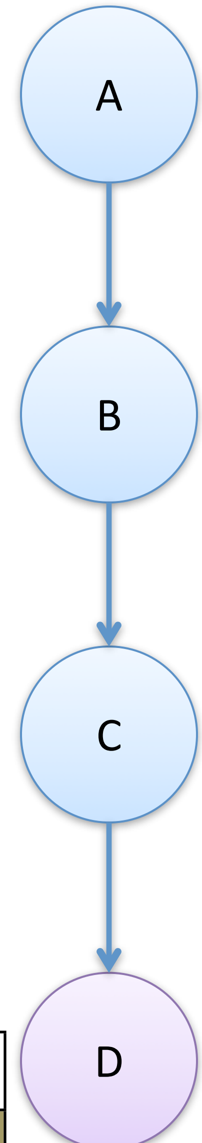
- Given relative error ε, the problem of finding an estimate ρ with that relative error (i.e., satisfying the bounds above) is NP-hard.

Let’s just try it anyway.

Markov Chain

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

0	
1	



P(B |A)	0	1
0		
1		

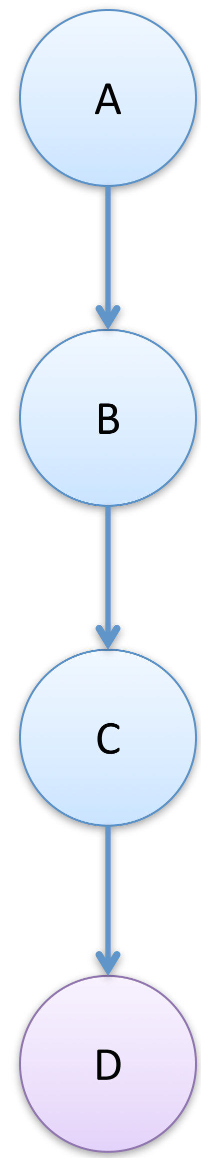
P(C |B)	0	1
0		
1		

P(D |C)	0	1
0		
1		

Markov Chain

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

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

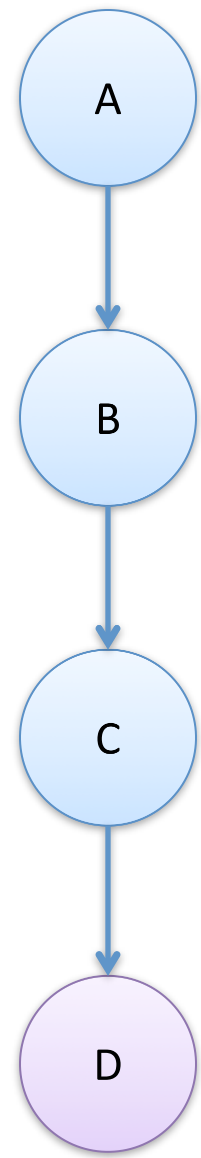


Markov Chain

- 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.



Markov Chain

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

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

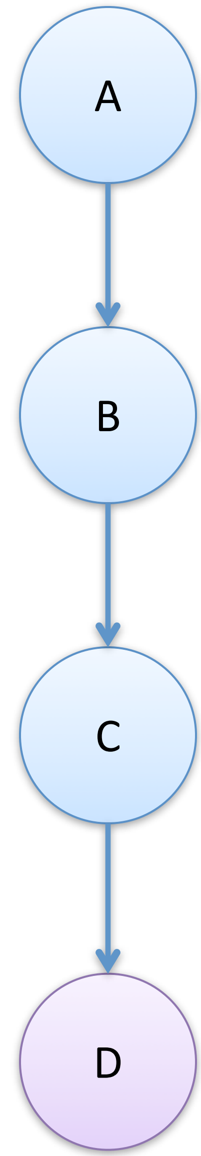
0	
1	

T

P(B | A)	0	1
0		
1		

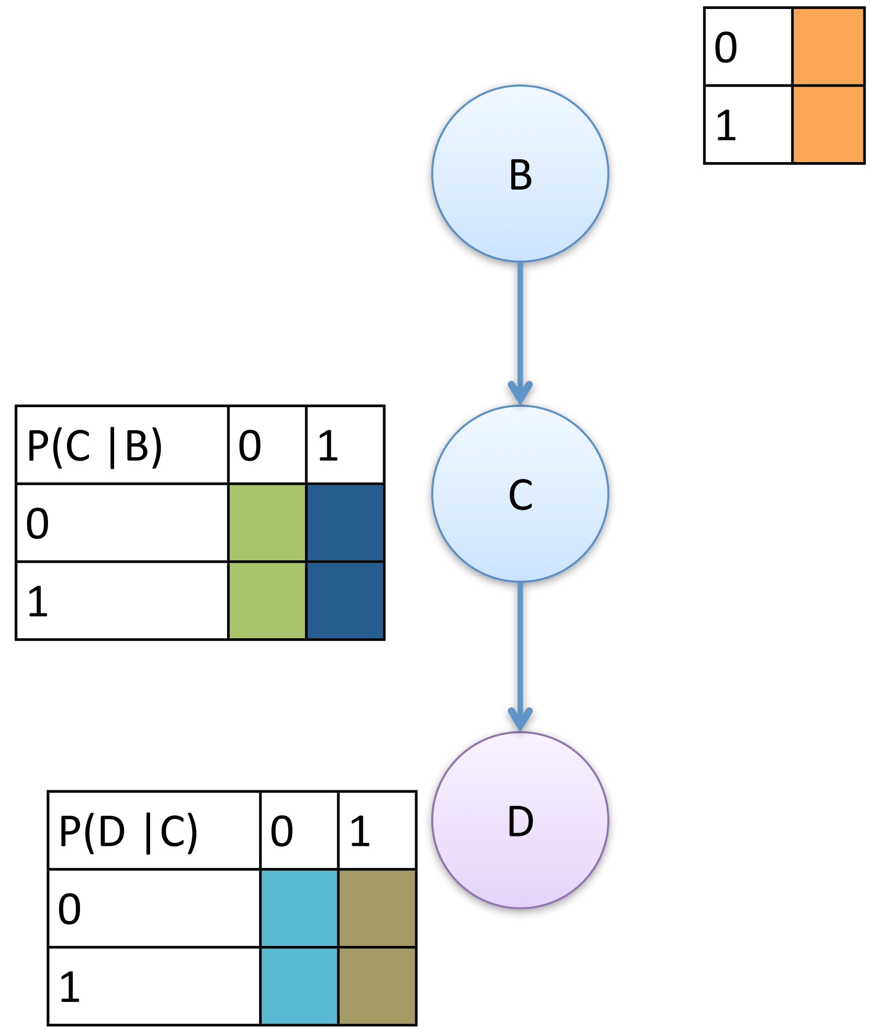
=

0	
1	



Markov Chain

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

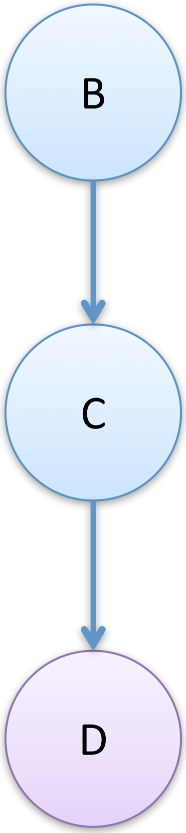


Markov Chain

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

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

- We already have P(B)!



Markov Chain

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

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

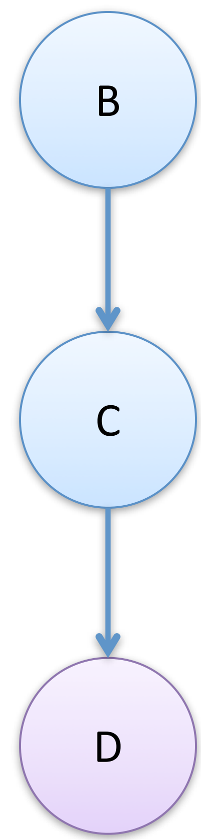
0	
1	

T

P(C | B)	0	1
0		
1		

=

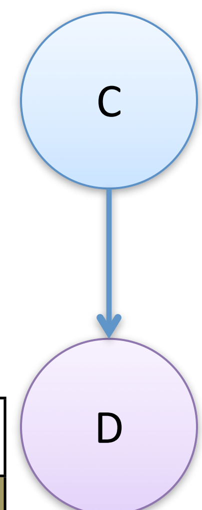
0	
1	



Markov Chain

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

0	
1	



P(D |C)	0	1
0		
1		

Markov Chain

- Last step to get P(D):

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

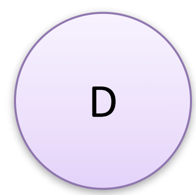
0	
1	

T

P(D | C)	0	1
0		
1		

=

0	
1	



Markov Chain

- Notice that the same step happened for each random variable:
  - We created a new CPD over a 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!
- We exploited the Bayesian network structure (each subexpression only depends on a small number of variables).
- Exponential blowup avoided!
- But: is there a general technique for any graphical model?

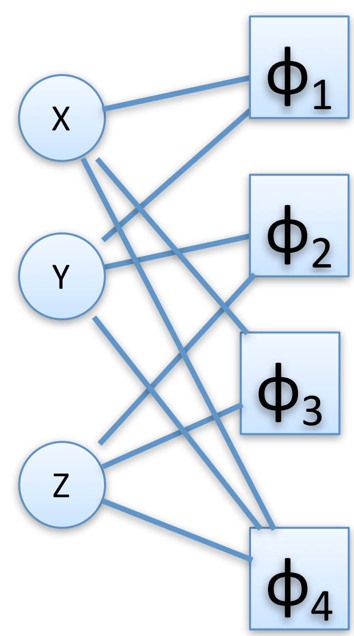
What's Next?

- Some machinery
- Variable elimination algorithm
- Analysis

Factor Graphs

Lecture 5!

- Bipartite graph
  - Variable nodes (circles)
  - Factor nodes (squares)
  - Edge between variable and factor if the factor depends on that variable.
- Makes the factors more obvious.
- CPDs can be seen as factors.



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	$\phi_1(A, B)$
0	0	30
0	1	5
1	0	1
1	1	10

$\cdot$

B	C	$\phi_2(B, C)$
0	0	100
0	1	1
1	0	1
1	1	100

=

A	B	C	$\phi_3(A, B, C)$
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 $\notin$ **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

A	B	C	ψ
0	0	0	0.5
0	0	1	0.5
0	1	0	0.4
0	1	1	0.6
1	0	0	0.2
1	0	1	0.8
1	1	0	0.1
1	1	1	0.9

"summing out" B

A	C	ψ(A, C)
0	0	0.9
0	1	1.1
1	0	0.3
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

A	B	C	ψ
0	0	0	0.5
0	0	1	0.5
0	1	0	0.4
0	1	1	0.6
1	0	0	0.2
1	0	1	0.8
1	1	0	0.1
1	1	1	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 $\notin$ **X**), we can turn a factor ϕ(**X**, Y) into a factor ψ(**X**) via marginalization:

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

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

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: $\phi_1 \cdot \phi_2 = \phi_2 \cdot \phi_1$
- Products are associative: $(\phi_1 \cdot \phi_2) \cdot \phi_3 = \phi_1 \cdot (\phi_2 \cdot \phi_3)$
- Sums are commutative: $\sum_X \sum_Y \phi = \sum_Y \sum_X \phi$
- Distributivity of multiplication over summation:

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

General Recipe

Eliminating One Variable

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

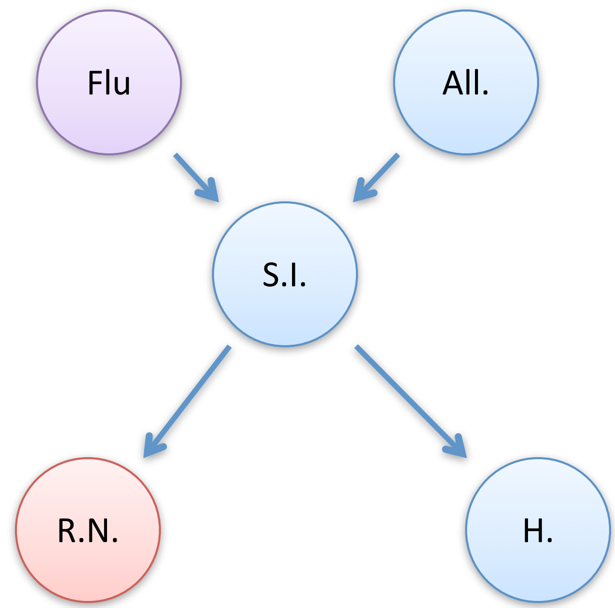
Output: new set of factors $\mathbf{\Psi}$

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

Example

- Query:
  P(Flu | runny nose)

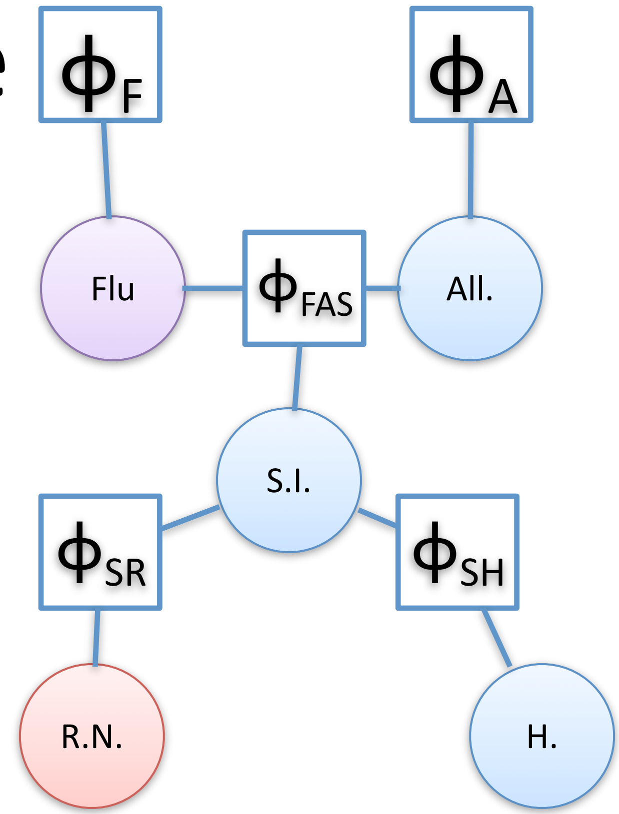
- Let's eliminate H.



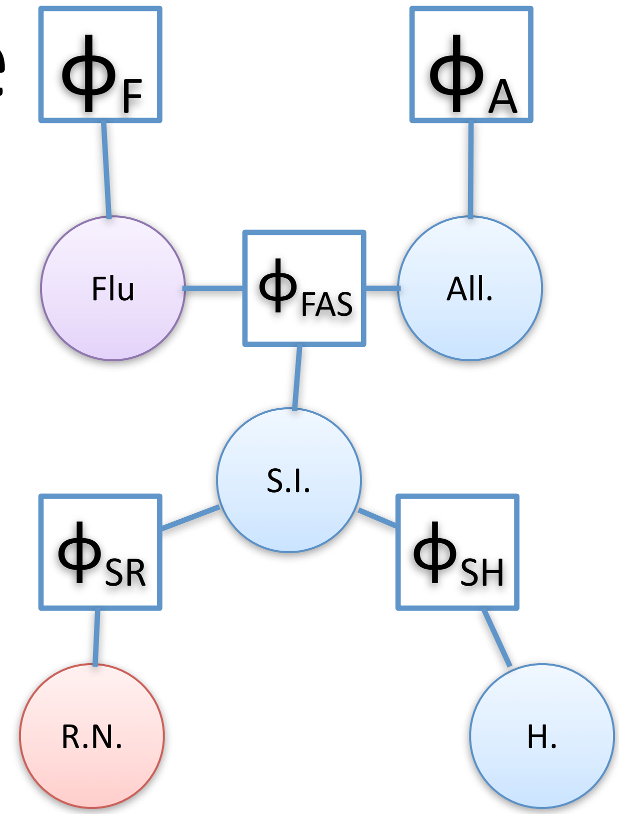
Example

- Query:
  P(Flu | runny nose)

- Let's eliminate H.



Example

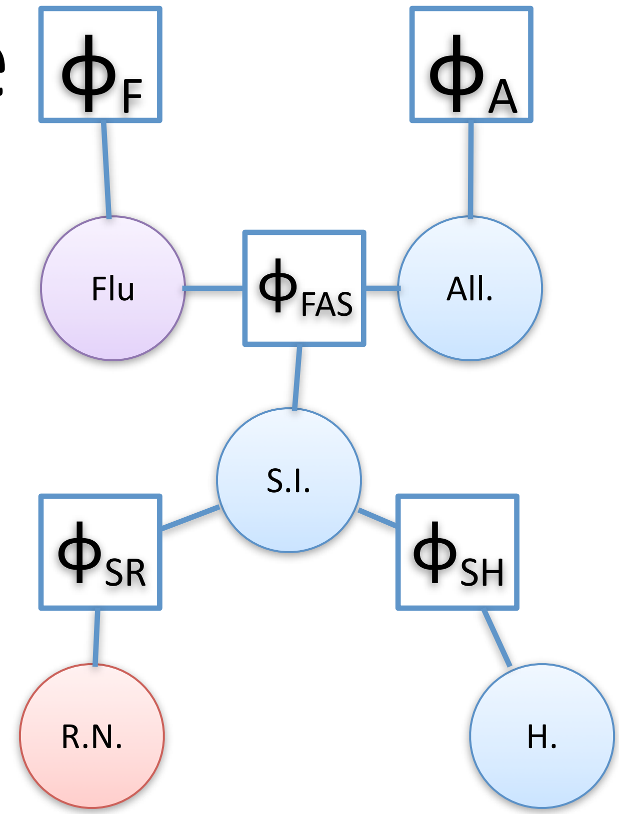


- Query:
  P(Flu | runny nose)

- Let's eliminate H.

1. $\boldsymbol{\Phi}' = \{\phi_{SH}\}$
2. $\boldsymbol{\Psi} = \{\phi_F, \phi_A, \phi_{FAS}, \phi_{SR}\}$
3. $\psi = \sum_H \prod_{\phi \in \boldsymbol{\Phi}'} \phi$
4. Return $\boldsymbol{\Psi} \cup \{\psi\}$

Example

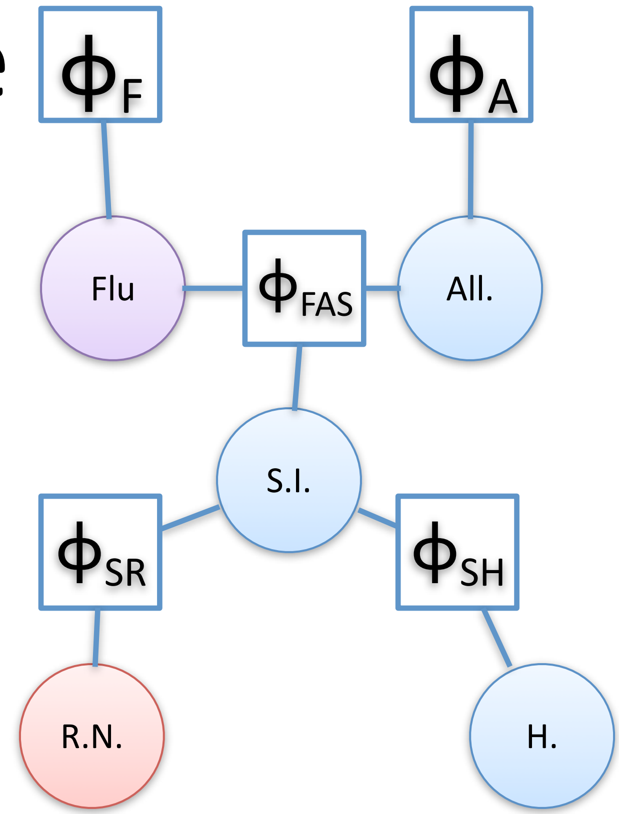


- Query: P(Flu | runny nose)

- Let's eliminate H.

1. $\mathbf{\Phi'} = \{\phi_{SH}\}$
2. $\mathbf{\Psi} = \{\phi_F, \phi_A, \phi_{FAS}, \phi_{SR}\}$
3. $\psi = \sum_H \phi_{SH}$
4. Return $\mathbf{\Psi} \cup \{\psi\}$

Example



- Query:
  P(Flu | runny nose)

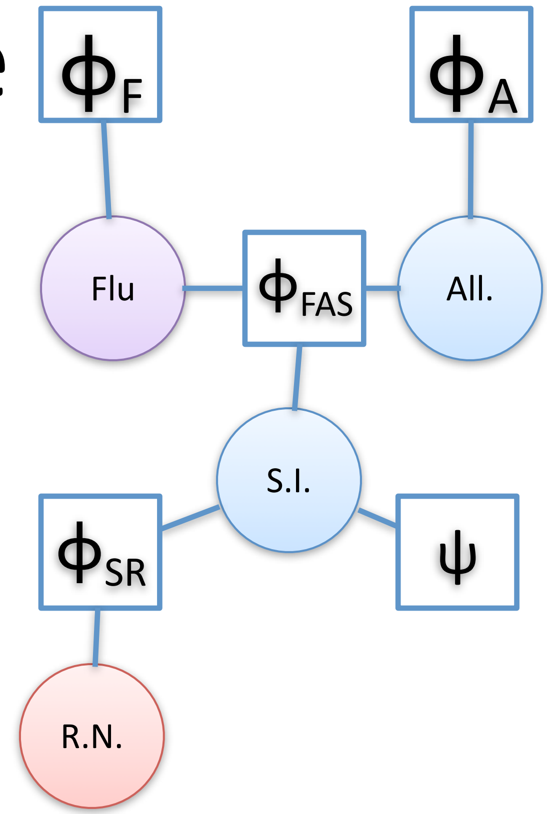
- Let's eliminate H.

1. $\mathbf{\Phi'} = \{\phi_{SH}\}$
2. $\mathbf{\Psi} = \{\phi_F, \phi_A, \phi_{FAS}, \phi_{SR}\}$
3. $\psi = \sum_H \phi_{SH}$
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.

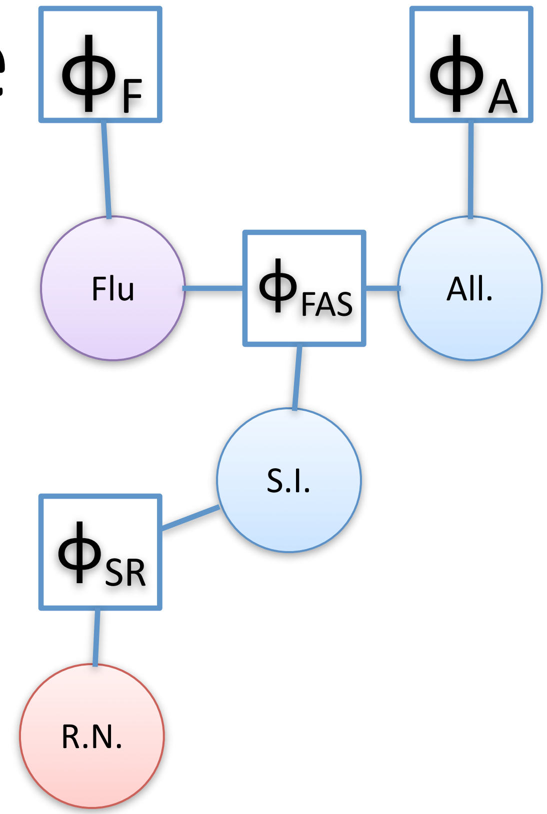
1. $\mathbf{\Phi}' = \{\phi_{SH}\}$
2. $\mathbf{\Psi} = \{\phi_F, \phi_A, \phi_{FAS}, \phi_{SR}\}$
3. $\psi = \sum_H \phi_{SH}$
4. Return $\mathbf{\Psi} \cup \{\psi\}$

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

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

General Recipe

Variable Elimination

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

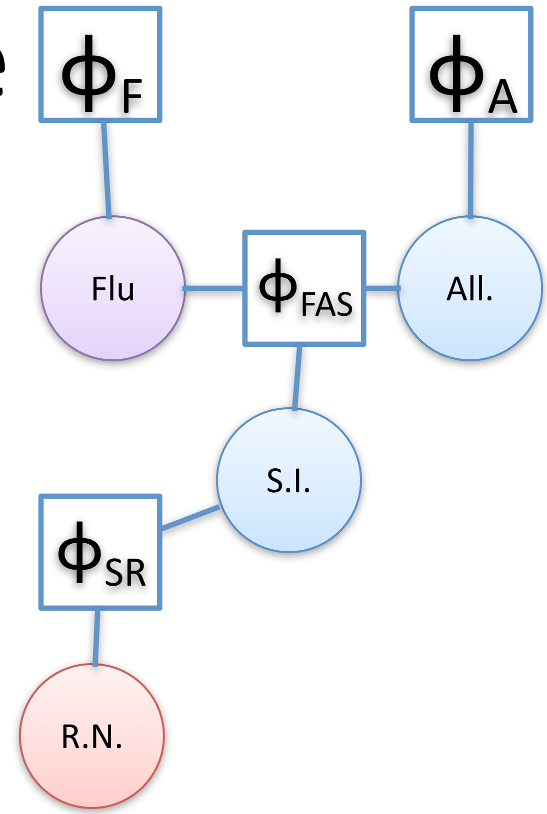
Output: new factor ψ

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

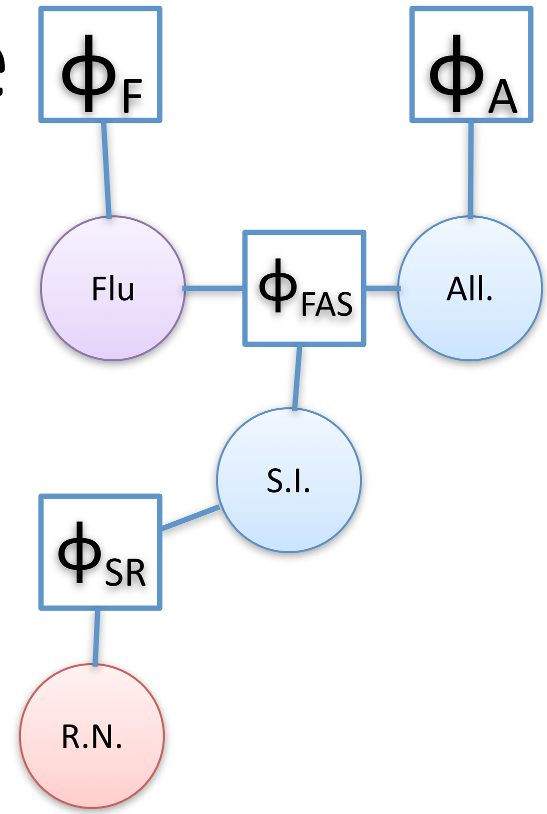
Example

- Query:
  P(Flu | runny nose)

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



Example

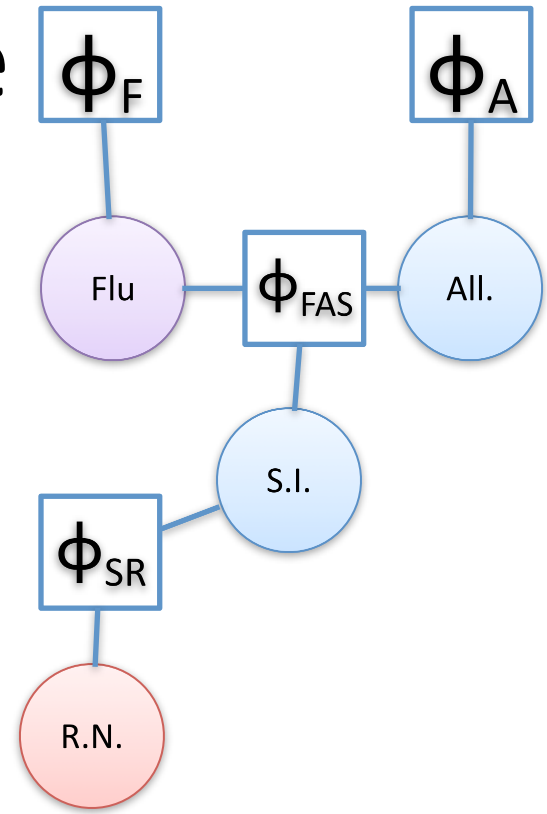


- Query:
  P(Flu | runny nose)

- Eliminating S.

1. $\mathbf{\Phi}' = \{\phi_{SR}, \phi_{FAS}\}$
2. $\mathbf{\Psi} = \{\phi_F, \phi_A\}$
3. $\psi_{FAR} = \sum_S \prod_{\phi \in \mathbf{\Phi}'} \phi$
4. Return $\mathbf{\Psi} \cup \{\psi_{FAR}\}$

Example

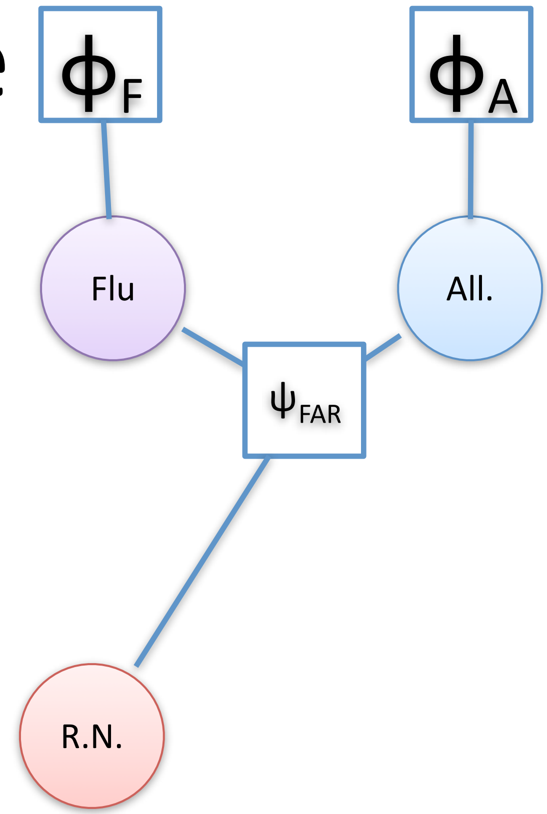


- Query:
  P(Flu | runny nose)

- Eliminating S.

1. $\mathbf{\Phi'} = \{\phi_{SR}, \phi_{FAS}\}$
2. $\mathbf{\Psi} = \{\phi_F, \phi_A\}$
3. $\psi_{FAR} = \sum_S \phi_{SR} \cdot \phi_{FAS}$
4. Return $\mathbf{\Psi} \cup \{\psi_{FAR}\}$

Example



- Query:
  P(Flu | runny nose)

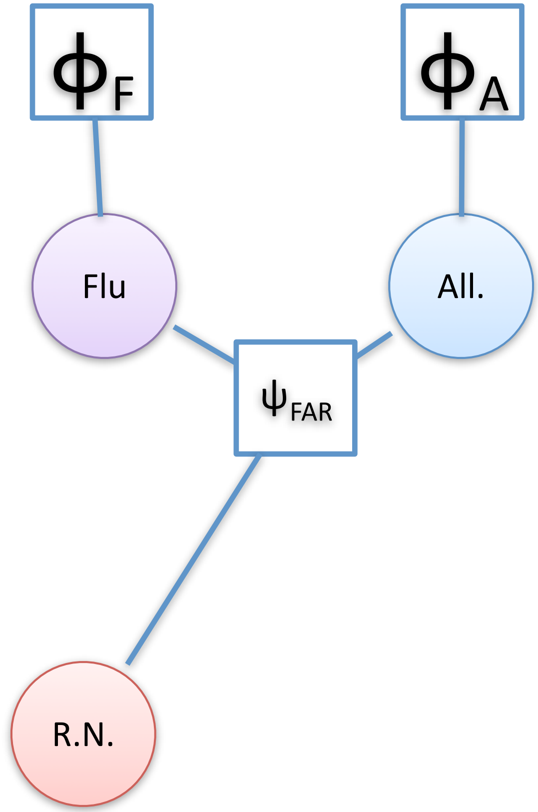
- Eliminating S.

1. $\mathbf{\Phi'} = \{\phi_{SR}, \phi_{FAS}\}$
2. $\mathbf{\Psi} = \{\phi_F, \phi_A\}$
3. $\psi_{FAR} = \sum_S \phi_{SR} \cdot \phi_{FAS}$
4. Return $\mathbf{\Psi} \cup \{\psi_{FAR}\}$

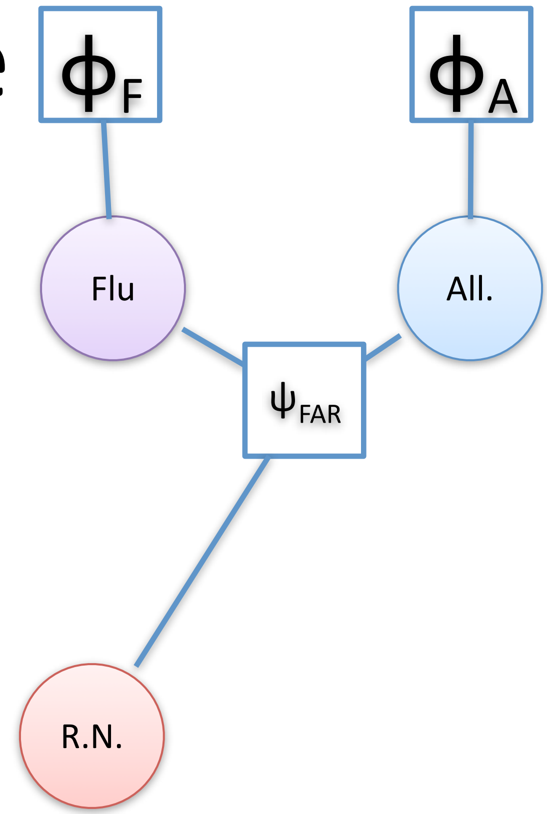
Example

- Query:
  P(Flu | runny nose)

- Finally, eliminate A.



Example

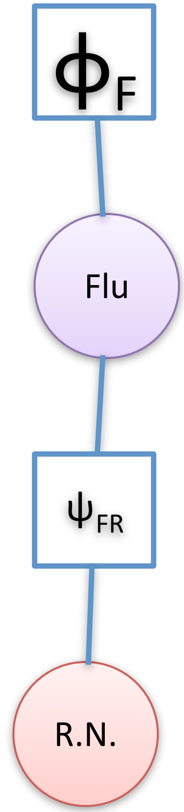


- Query:
  P(Flu | runny nose)

- Eliminating A.

1. $\mathbf{\Phi'} = \{\phi_A, \phi_{FAR}\}$
2. $\mathbf{\Psi} = \{\phi_F\}$
3. $\psi_{FR} = \sum_A \phi_A \cdot \psi_{FAR}$
4. Return $\mathbf{\Psi} \cup \{\psi_{FR}\}$

Example



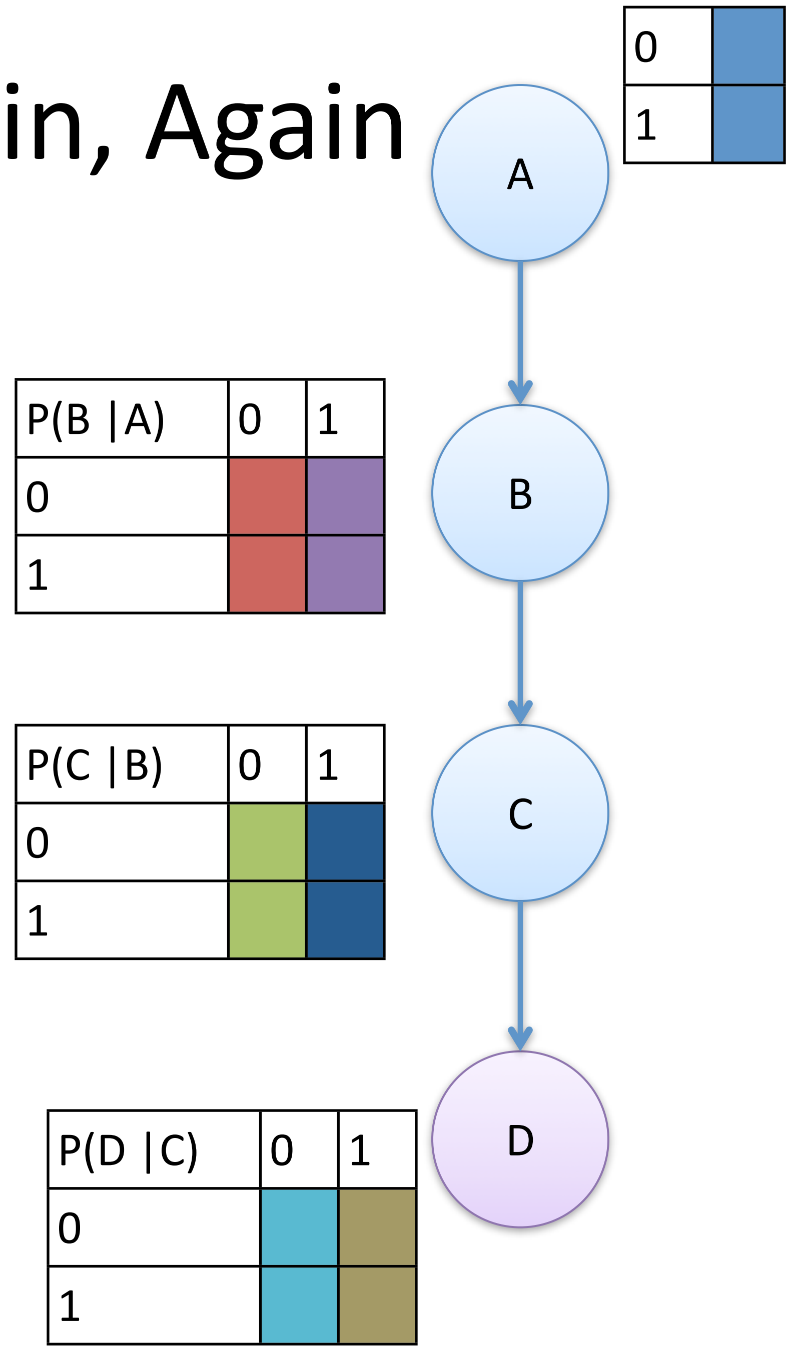
- Query:
  P(Flu | runny nose)

- Eliminating A.

1. $\mathbf{\Phi'} = \{\phi_A, \phi_{FAR}\}$
2. $\mathbf{\Psi} = \{\phi_F\}$
3. $\psi_{FR} = \sum_A \phi_A \cdot \psi_{FAR}$
4. Return $\mathbf{\Psi} \cup \{\psi_{FR}\}$

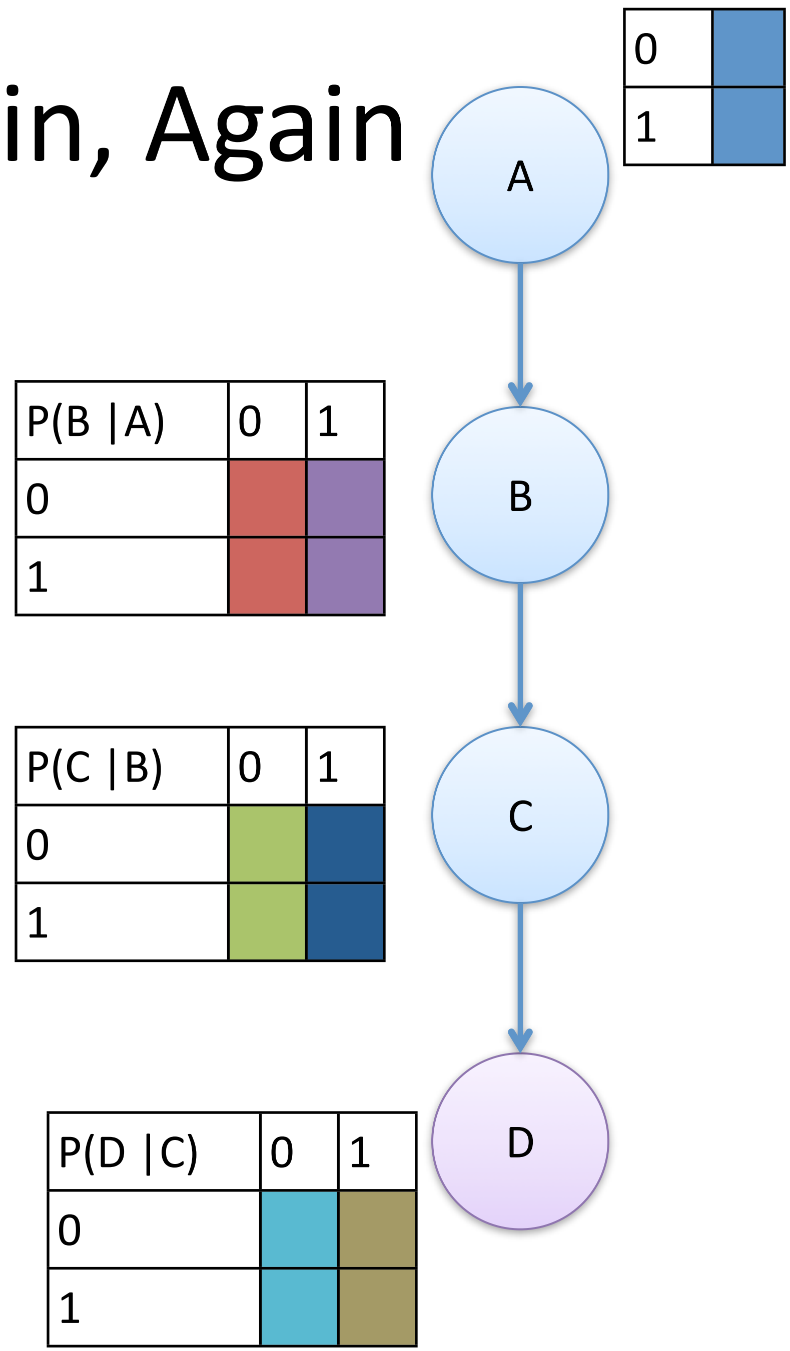
Markov Chain, Again

- 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.

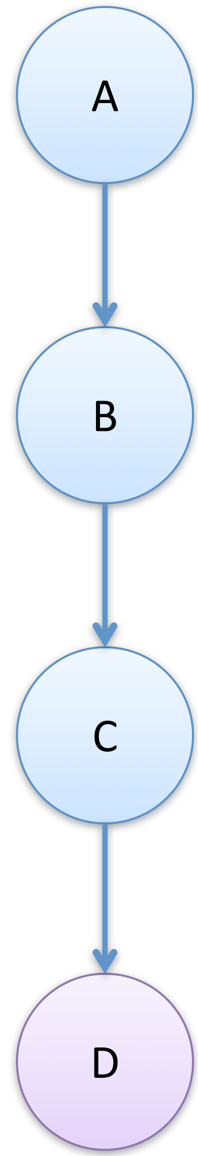
P(C |B)	0	1
0		
1		

·

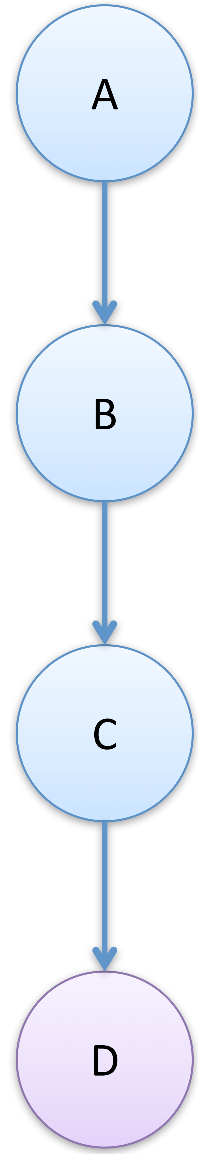
P(D |C)	0	1
0		
1		

=

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.

Σ_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	

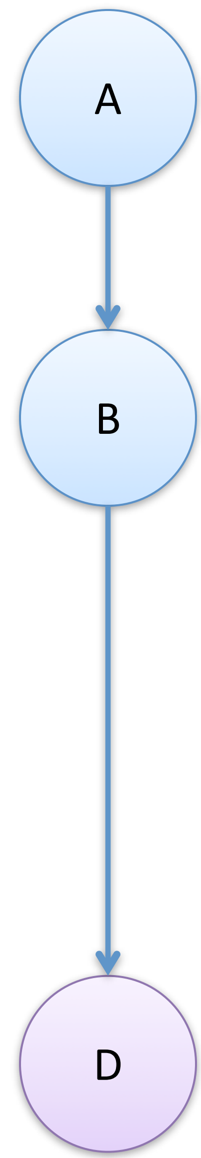
=

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.