Graphical Models

Lecture 10:

Variable Elimination, continued

Andrew McCallum
mccallum@cs.umass.edu

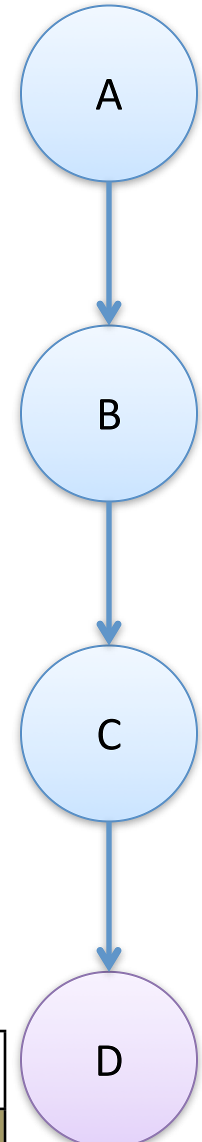
Thanks to Noah Smith and Carlos Guestrin for some slide materials.

Last Time

- Probabilistic inference is the goal: $P(\mathbf{X} \mid \mathbf{E} = \mathbf{e})$.
  - #P-complete in general
- Do it anyway! Variable elimination …

Markov Chain Example

0	
1	



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

P(B |A)	0	1
0		
1		

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

P(C |B)	0	1
0		
1		

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

P(D |C)	0	1
0		
1		

Last Time

- Probabilistic inference is the goal: P(**X** | **E** = **e**).
  - #P-complete in general
- Do it anyway! Variable elimination …
  - Work on factors (algebra of factors)
  - Generally: "sum-product" inference $\sum_{\mathbf{Z}} \prod_{\phi \in \Phi} \phi$

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

·

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:

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

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

Last Time

- Probabilistic inference is the goal: P(**X** | **E** = **e**).
  - #P-complete in general
- Do it anyway! Variable elimination …
  - Work on factors (algebra of factors)
  - How to eliminate one variable (marginalize a product of factors)

Eliminating One Variable

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

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

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

Last Time

- Probabilistic inference is the goal: P(**X** | **E** = **e**).
  - #P-complete in general
- Do it anyway! Variable elimination …
  - Work on factors (algebra of factors)
  - Generally: "sum-product" inference $$\sum_{\mathbf{Z}} \prod_{\phi \in \Phi} \phi$$
  - How to eliminate one variable (marginalize a product of factors)
  - How to eliminate a bunch of variables

Variable Elimination

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

Output: new factor ψ

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

Today

- Variable elimination for inference (with evidence)
- Complexity analysis of VE
- Elimination orderings

Probabilistic Inference

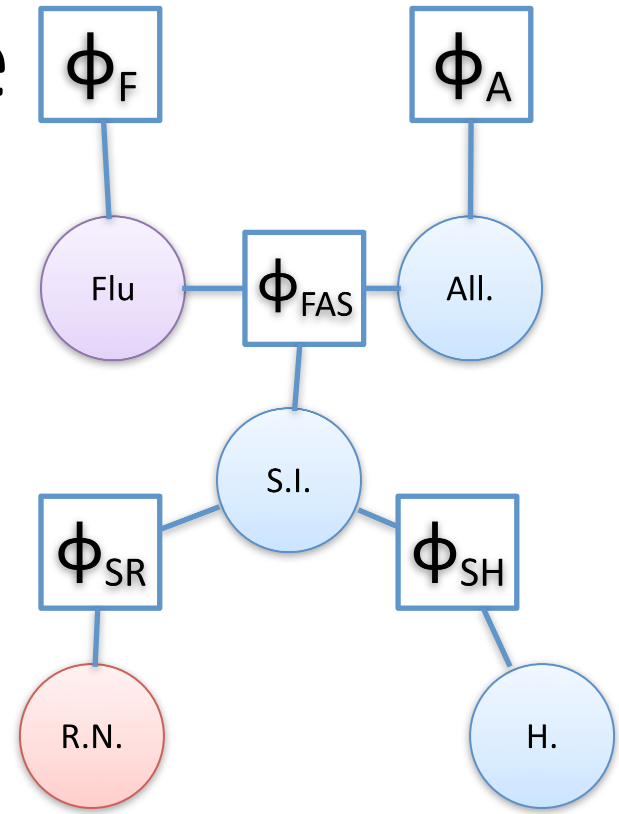
- Assume we are given a graphical model.
- Want:

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

Adding Evidence

- Conditional distributions are Gibbs; can be represented as factor graphs!
- Everything is essentially the same, but we *reduce* the factors to match the evidence.
  - Previously normalized factors may not be normalized any longer, but this is not a problem.
- Prune anything not on an active trail to query variables.

Example



- Query:
  P(Flu | runny nose)

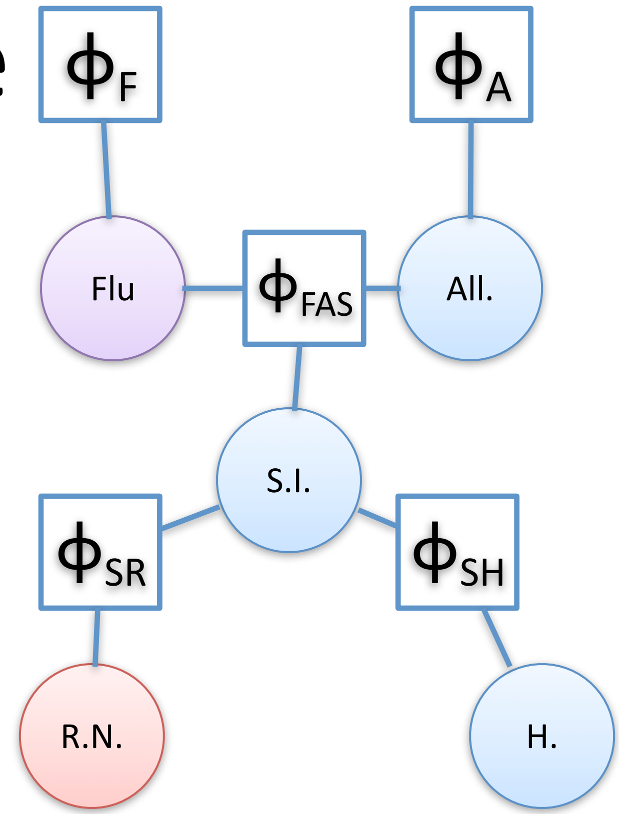
- Let's reduce to R = true (runny nose).

P(R | S)	0	1
0		
1		

Example

- Query:
  P(Flu | runny nose)

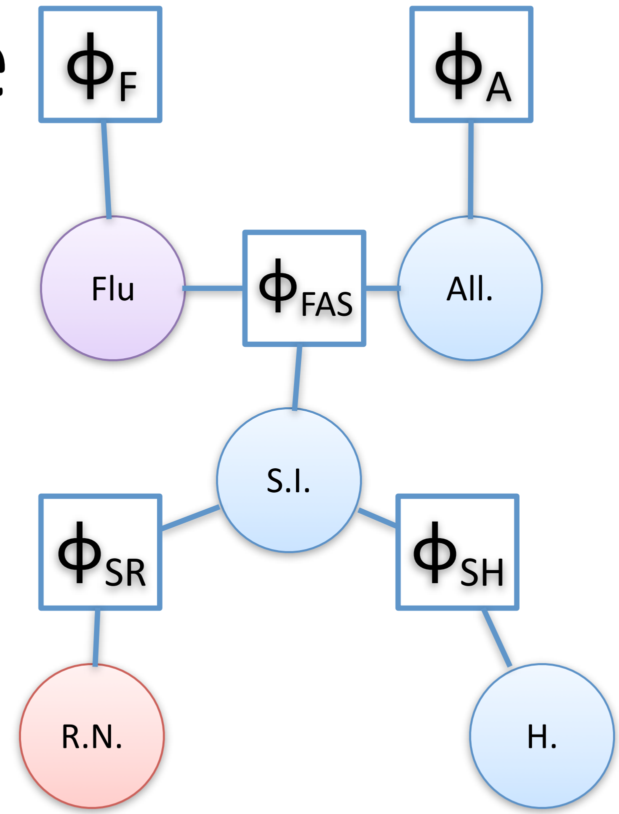
- Let’s reduce to R = true (runny nose).



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)

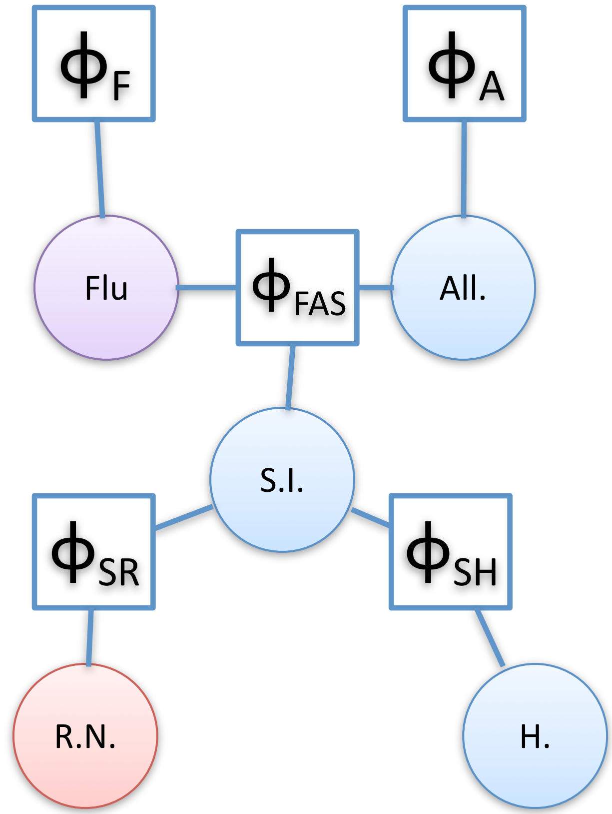
- Let’s reduce to R = true (runny nose).

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		

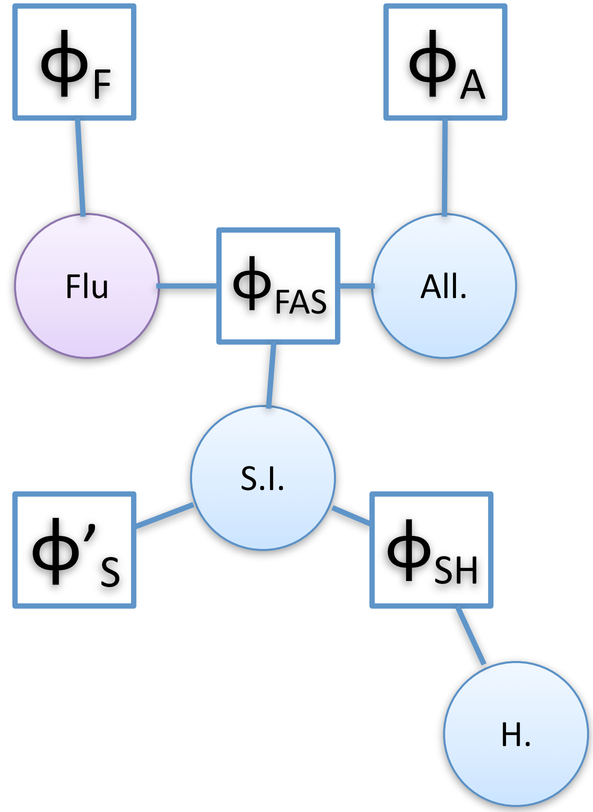
S	R	ϕ_{SR} (S, R)
0	0	
0	1	
1	0	
1	1	

S	R	ϕ'_S (S)
0	1	
1	1	

Example

- Query:
  P(Flu | runny nose)

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

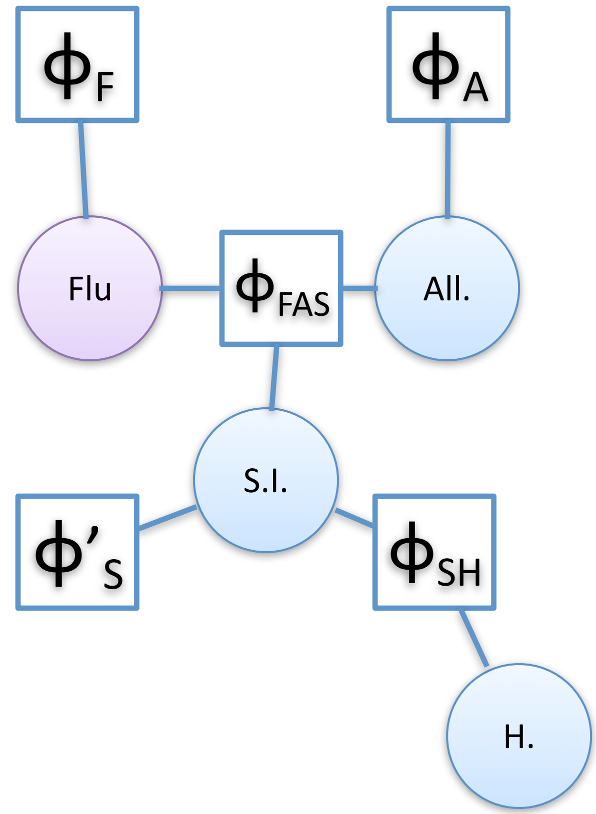


S	R	ϕ'_S (S)
0	1	
1	1	

Example

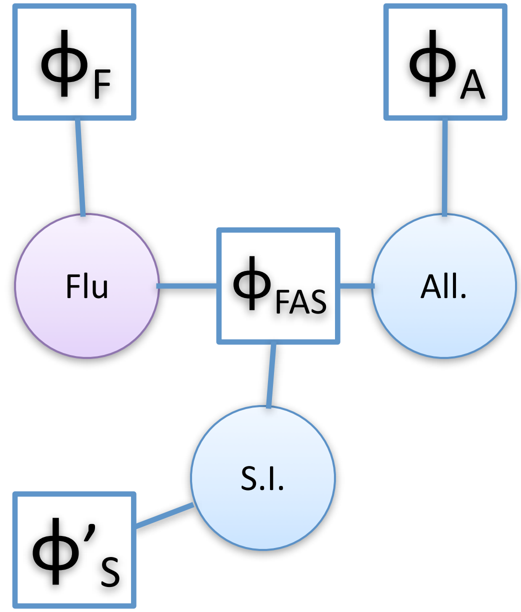
- Query:
  P(Flu | runny nose)

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



H can be pruned
for the same reasons
as before.

Example

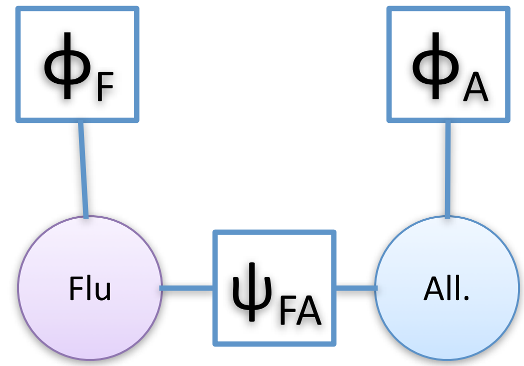


Eliminate S.

- Query:
  P(Flu | runny nose)

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

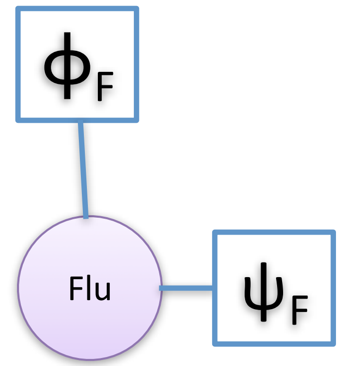
Example



Eliminate A.

- Query:
  P(Flu | runny nose)
- Now run variable elimination all the way down to one factor (for F).

Example



Take final product.

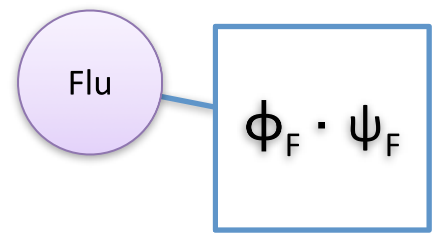
- Query:
  P(Flu | runny nose)

- 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.



General Recipe

Variable Elimination for Conditional Probabilities

Input: Graphical model, set of query variables $\mathbf{Q}$, evidence $\mathbf{E} = \mathbf{e}$

Output: factor ϕ and scalar α

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

Note

- For Bayesian networks, the final factor will be P(**Q**, **E** = **e**) and the sum α = P(**E** = **e**).
- This equates to a Gibbs distribution with partition function = α.

Complexity of Variable Elimination

Complexity of Variable Elimination

- n = number of random variables
- m = number of factors
- In step i, we multiply all factors relating to X_i, resulting in ψ_i, and sum out X_i, giving a new factor τ_i.
  - N_i = number of entries in ψ_i
  - $N_{max} = \max_i N_i$
- If we eliminate everything, m initial factors plus n new ones (the τ_i).

Complexity of Variable Elimination

- m + n factors
- Each is multiplied once, then removed.

Recall: Eliminating One Variable

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

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

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

Complexity of Variable Elimination

- m + n factors
- Each is multiplied once to produce some ψ_i, then removed to produce τ_i.
  - (m + n) N_i multiplications for X_i
  - $O(mN_{max})$
- Marginalization (summing) touches each entry in each ψ_i once:
  - N_i additions for X_i
  - $O(nN_{max})$

Complexity of Variable Elimination

- Overall: $O(mN_{max})$
  - Bayesian network: $m = n$
  - Markov network: $m \geq n$

Complexity of Variable Elimination

- Overall: $O(mN_{max})$
- The size N_{max} of the intermediate factors ψ_i is what makes this blow up.
  - v values per random variable
  - k_i variables for factor ψ_i
  - $N_i = v^{k_i}$

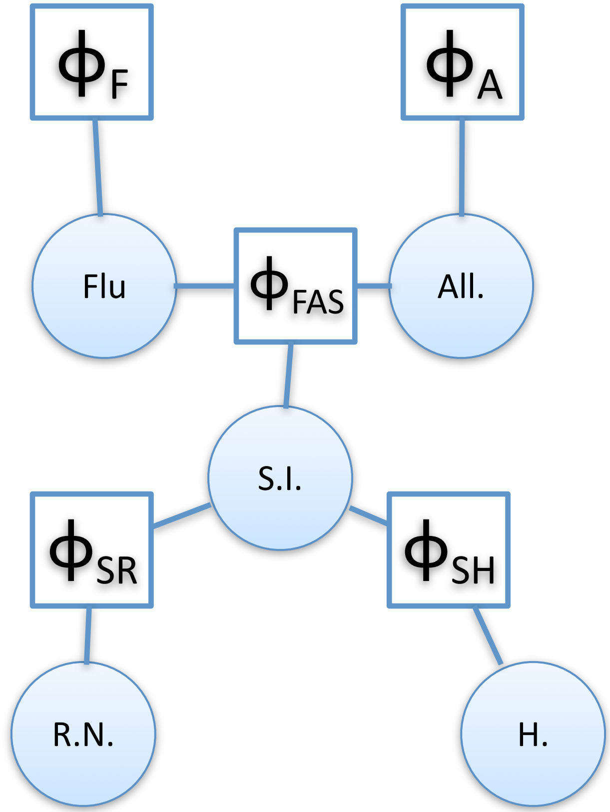
- But really, how bad is it?

Analyzing VE via the Graph

- Assume a factor graph representation.
- One step of VE, on X_i:
  - Create a single factor ψ that includes X_i and its neighbors (**Y** that share factors with X_i).
  - Marginalize X_i out of ψ, giving new factor τ.
  - If we go back to a Markov network, we have now introduced new edges!

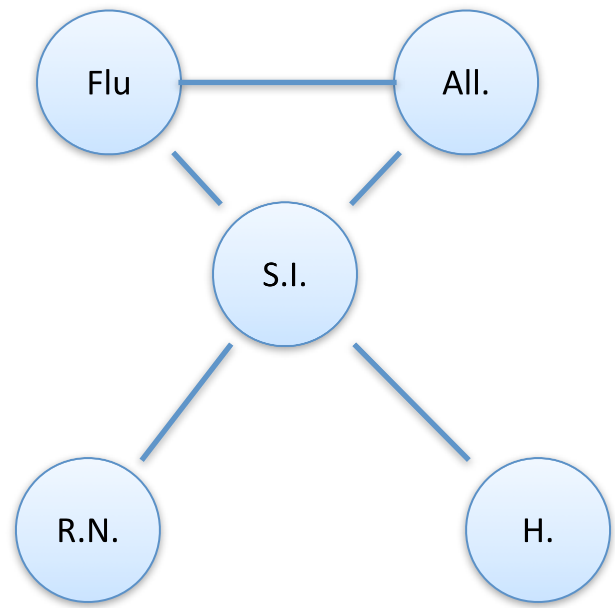
Example

- Factor graph.



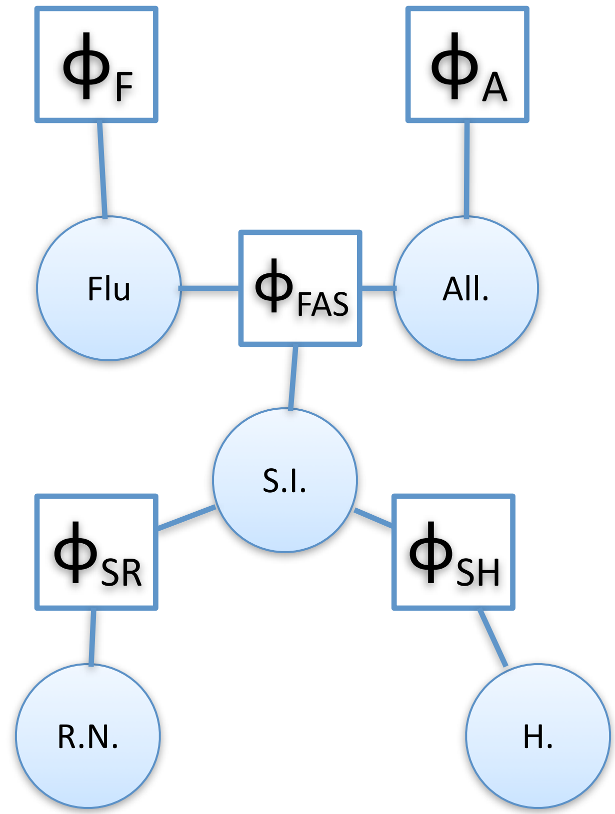
Example

- Markov network.



Example

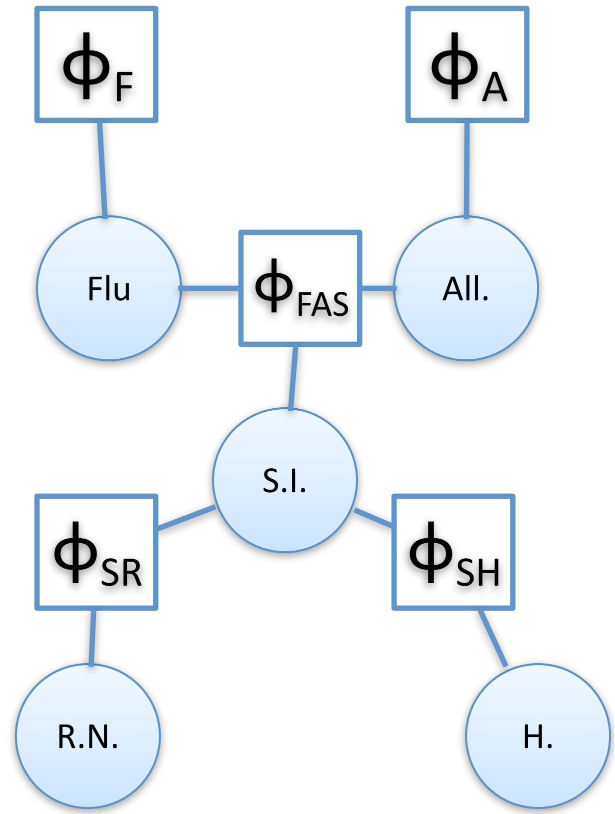
- Eliminate S.



Example

- Eliminate S.

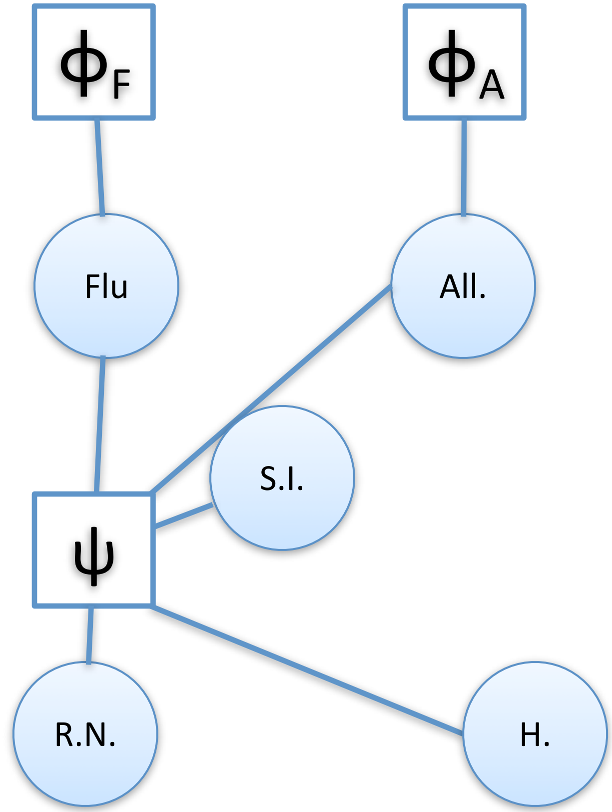
- $\psi = \phi_{FAS} \cdot \phi_{SR} \cdot \phi_{SH}$



Example

- Eliminate S.

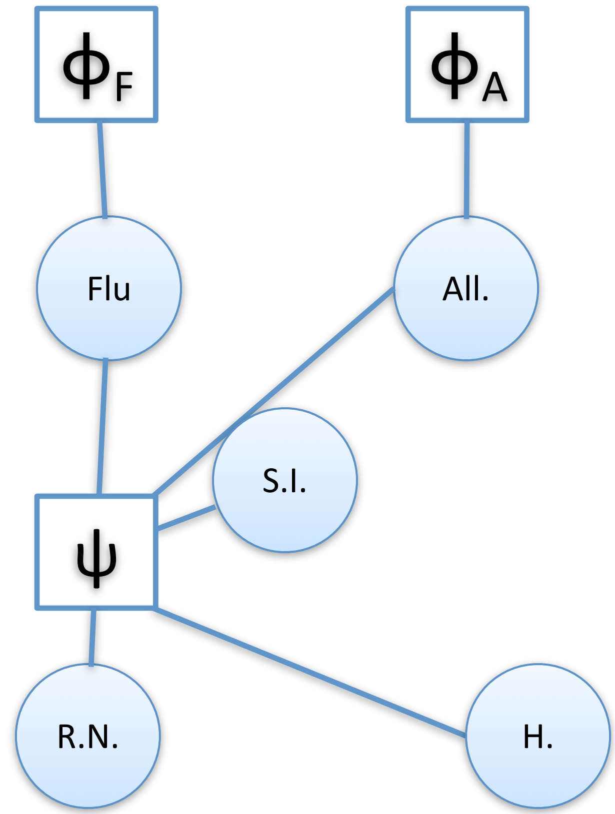
- $\psi = \phi_{FAS} \cdot \phi_{SR} \cdot \phi_{SH}$



Example

- Eliminate S.

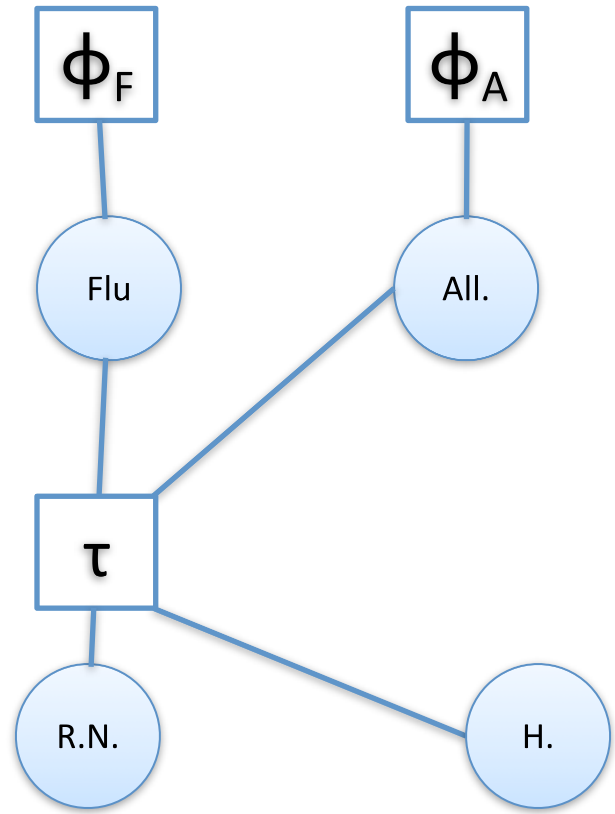
- $\psi = \phi_{FAS} \cdot \phi_{SR} \cdot \phi_{SH}$
- $\tau = \sum_S \psi$



Example

- Eliminate S.

- $\psi = \phi_{FAS} \cdot \phi_{SR} \cdot \phi_{SH}$
- $\tau = \sum_S \psi$

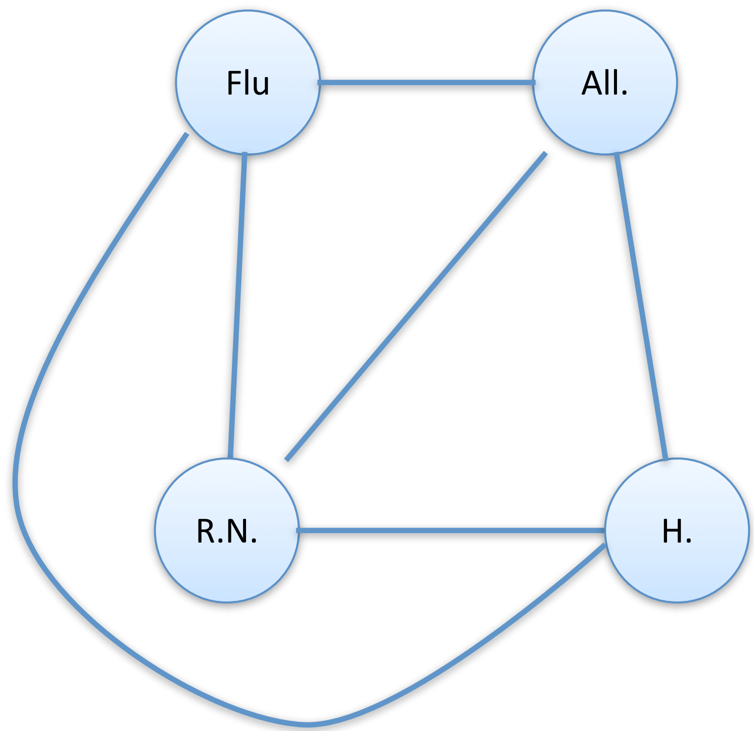


Example

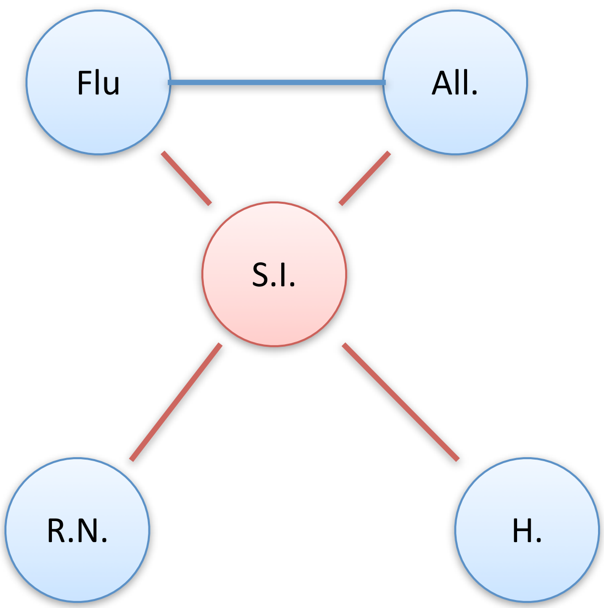
- Eliminate S.

- $\psi = \phi_{FAS} \cdot \phi_{SR} \cdot \phi_{SH}$
- $\tau = \sum_{S} \psi$

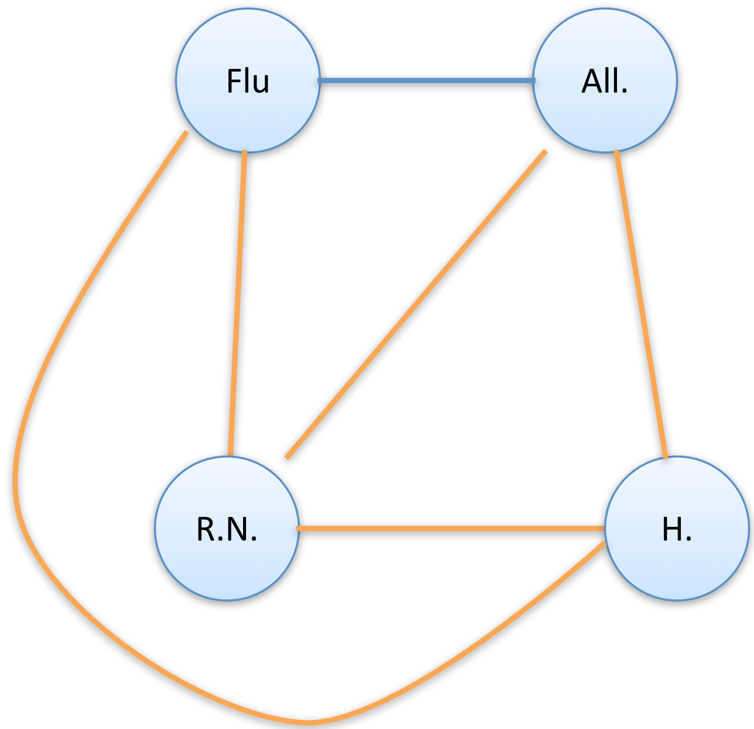
- Back to Markov net?



Example



removed stuff

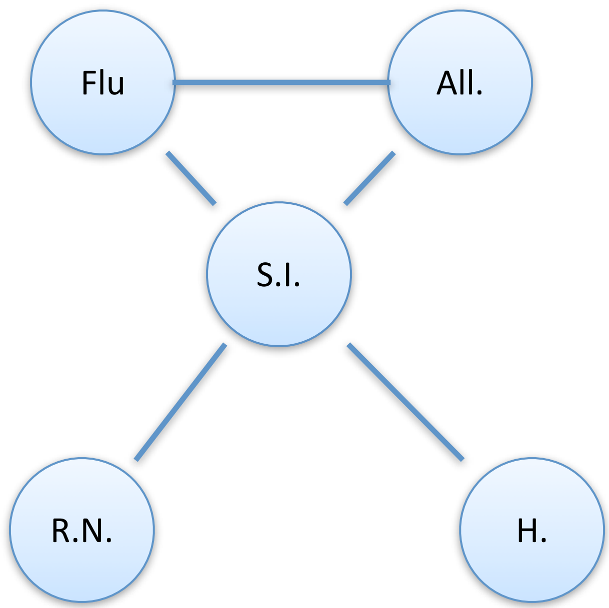


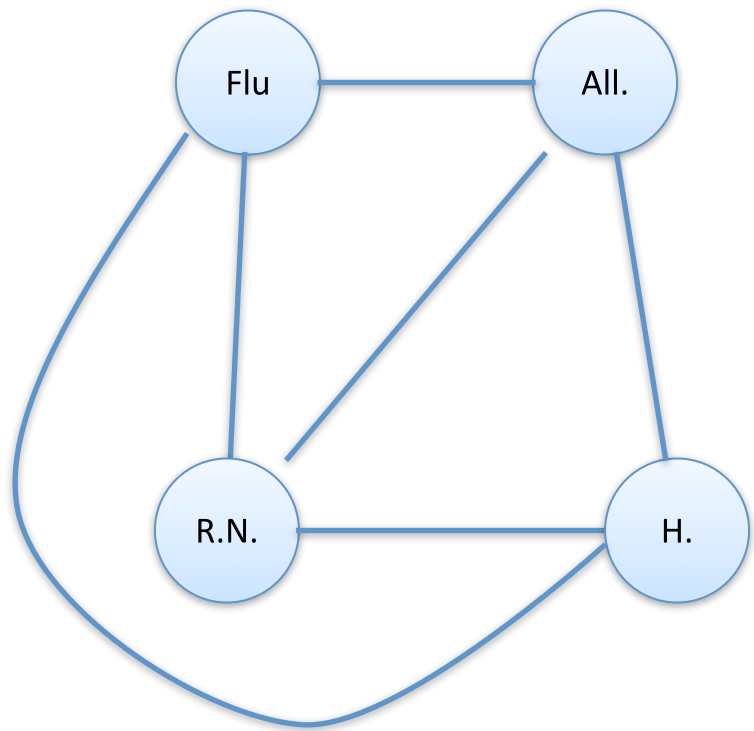
“fill edges”

Insight

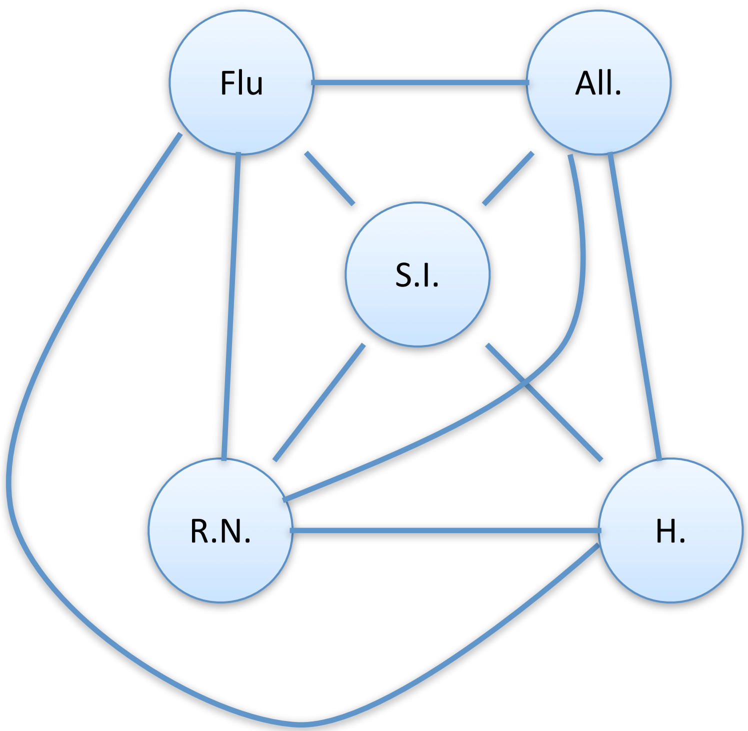
- Each VE step is a transformation on the graph.
  - We’ve been drawing it on slides this way all along!
- We can put the full sequence of graphs together into a single structure.

Union of the Graphs …





Union of the Graphs



Induced Graph

- Take the union over all of the undirected graphs from each step: **induced graph**.
  (1) The scope of every intermediate factor is a clique in this graph.
  (2) Every maximal clique in the graph is the scope of some intermediate factor.

- Important:
  different ordering → different induced graph …

Proof (1)

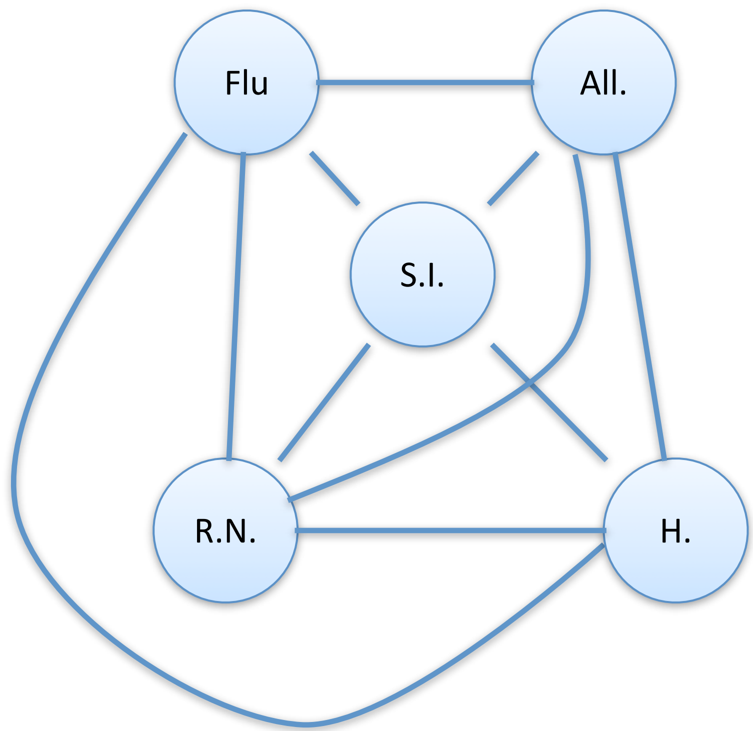
- The scope of every intermediate factor is a clique in the induced graph.
  - Consider $\psi(X_1, \ldots, X_k)$, an intermediate factor.
  - In the corresponding Markov network,
    all of the X_i are connected (they share a factor).
  - Hence they form a clique.

Proof (2)

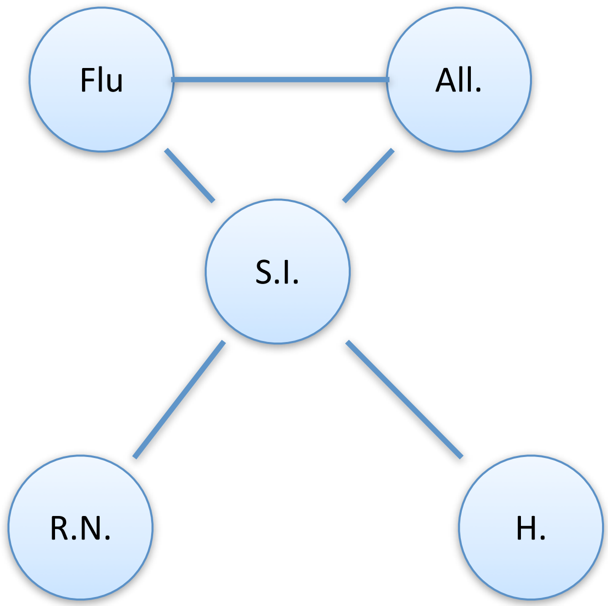
- Every maximal clique in the induced graph is the scope of some intermediate factor.
  - Consider maximal clique **Y** = $\{Y_1, \ldots, Y_k\}$
  - Let Y_1 be the first one eliminated, with resulting product-of-factors ψ.
  - All edges relating to Y_1 are introduced *before* it is eliminated.
  - Y_1 and Y_i share an edge, so they share a factor that gets multiplied into ψ; so ψ includes all of **Y**.
  - Any other variable X can't be in the scope of ψ, because it would have to be linked to all of **Y**, so that **Y** wouldn't be a maximal clique.

KF p 309

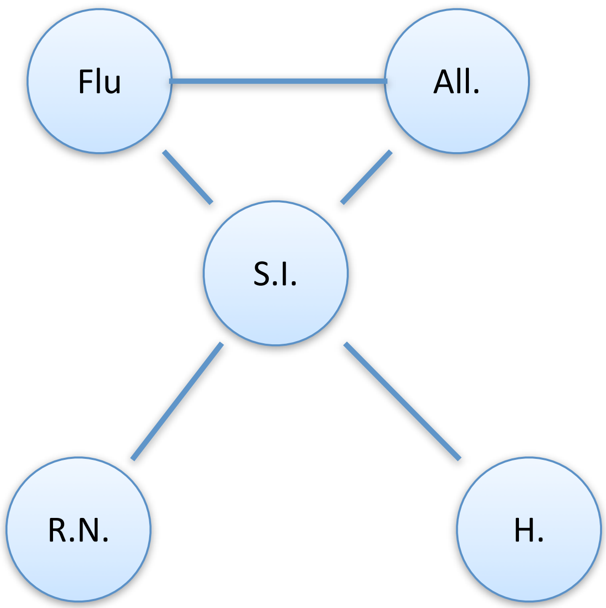
Ordering: {S, …}

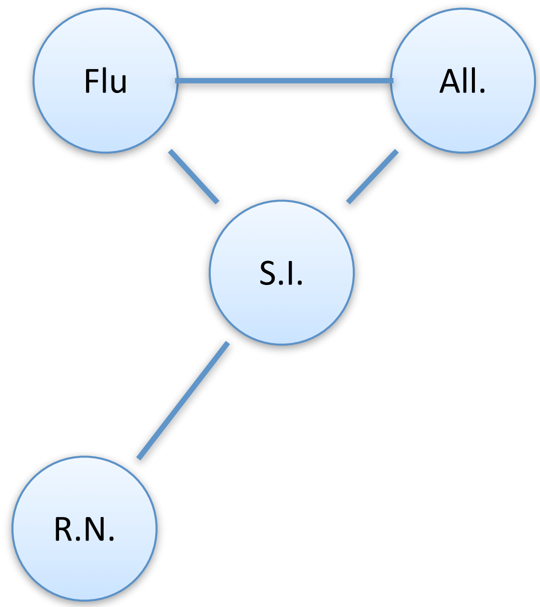


Ordering: {H, R, S, A, F}

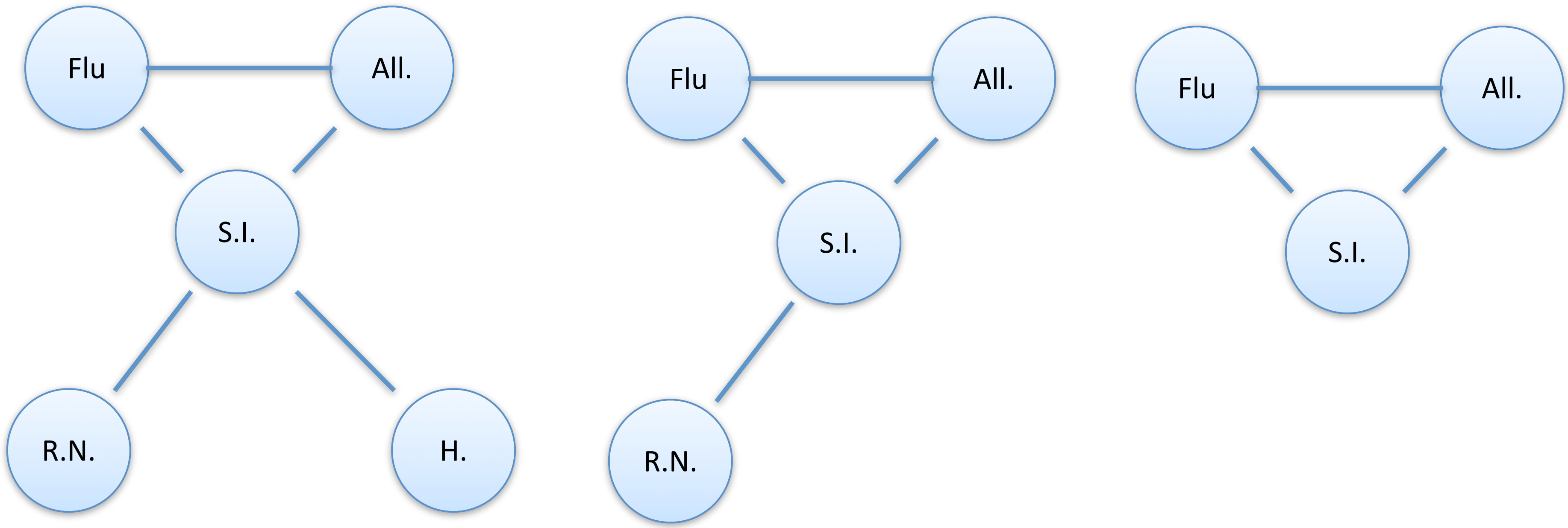


Ordering: {H, R, S, A, F}

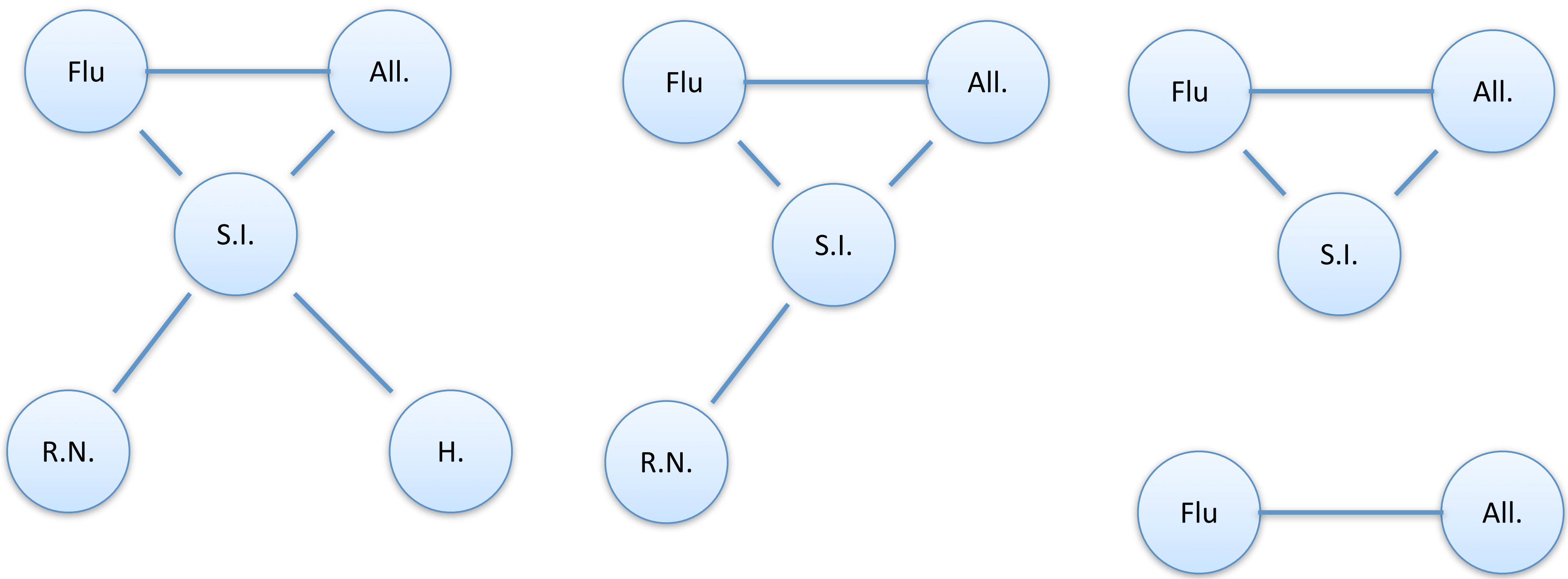




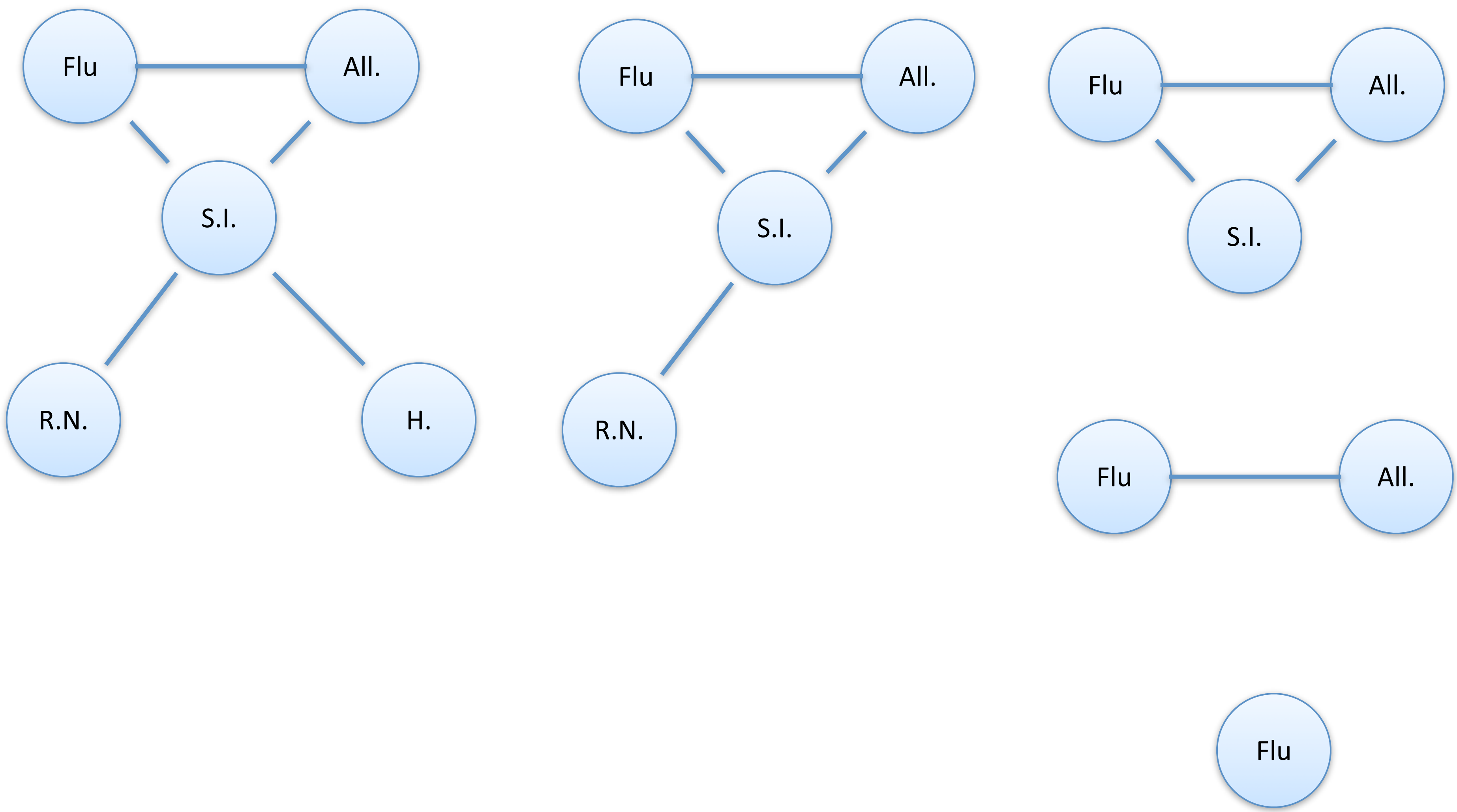
Ordering: {H, R, S, A, F}



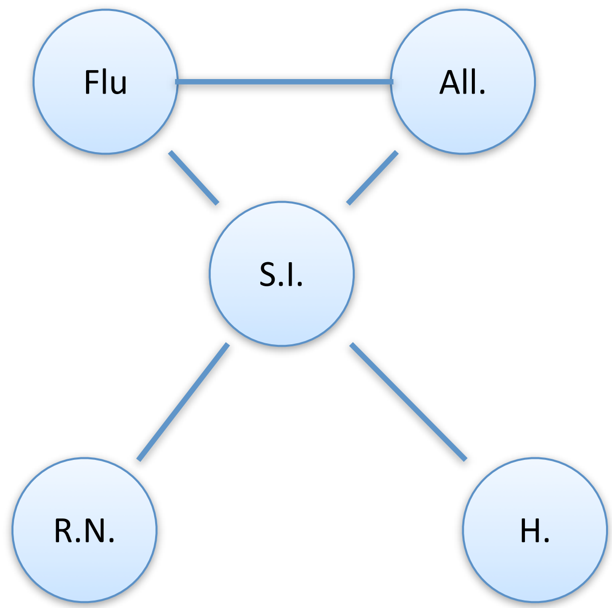
Ordering: {H, R, S, A, F}



Ordering: {H, R, S, A, F}



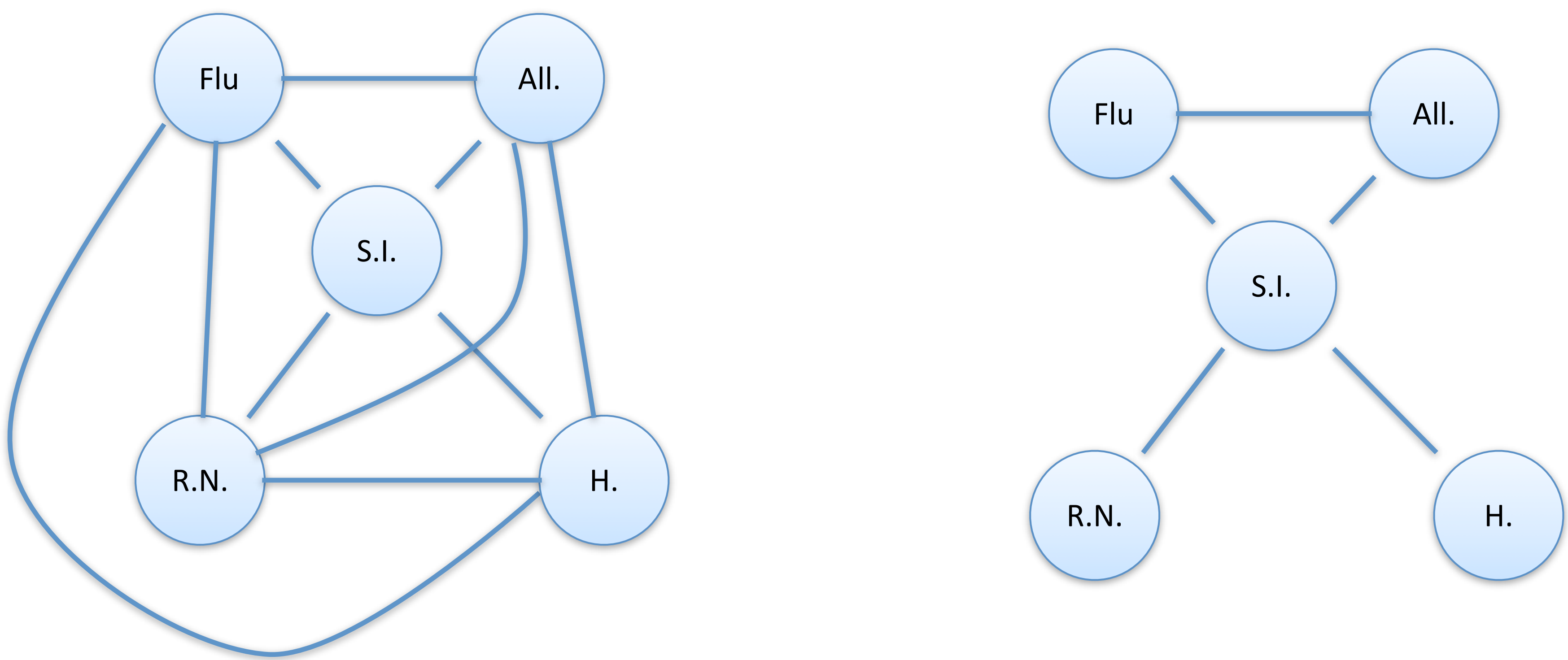
Ordering: {H, R, S, A, F}



induced graph = original graph

"Induced Width"

- Number of nodes in the largest clique of the induced graph, minus one.
  - Relative to an ordering!



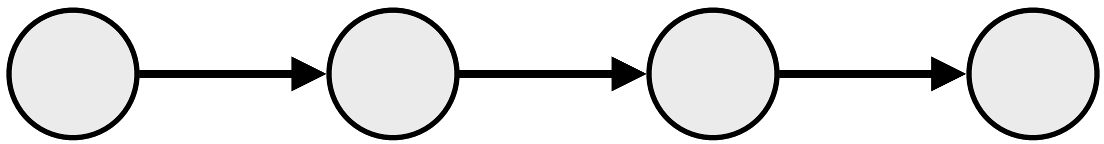
“Induced Width”

- Number of nodes in the largest clique of the induced graph, minus one.
  - Relative to an ordering!

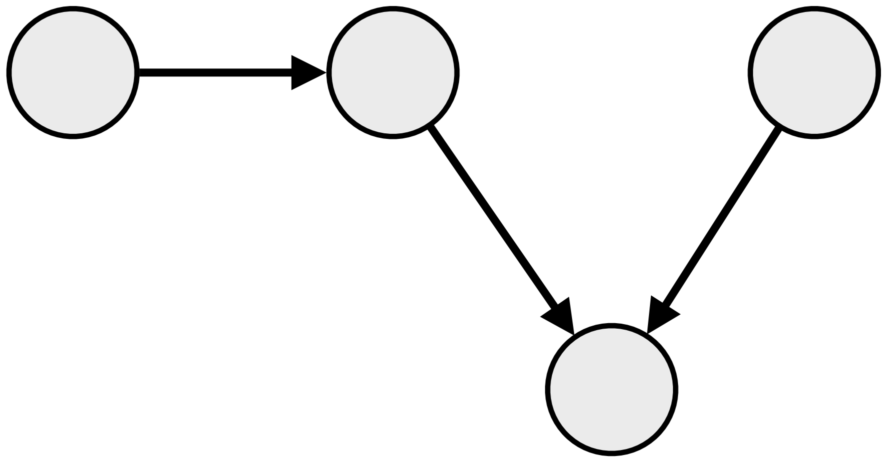
- “**Tree width”** = minimum width over all possible orderings.
  - Bound on the best performance we can hope for … VE runtime is *exponential* in treewidth!

Why minus one?

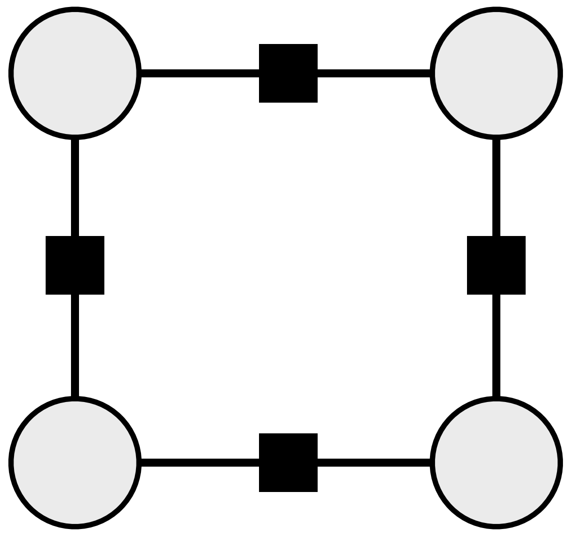
Treewidth Example



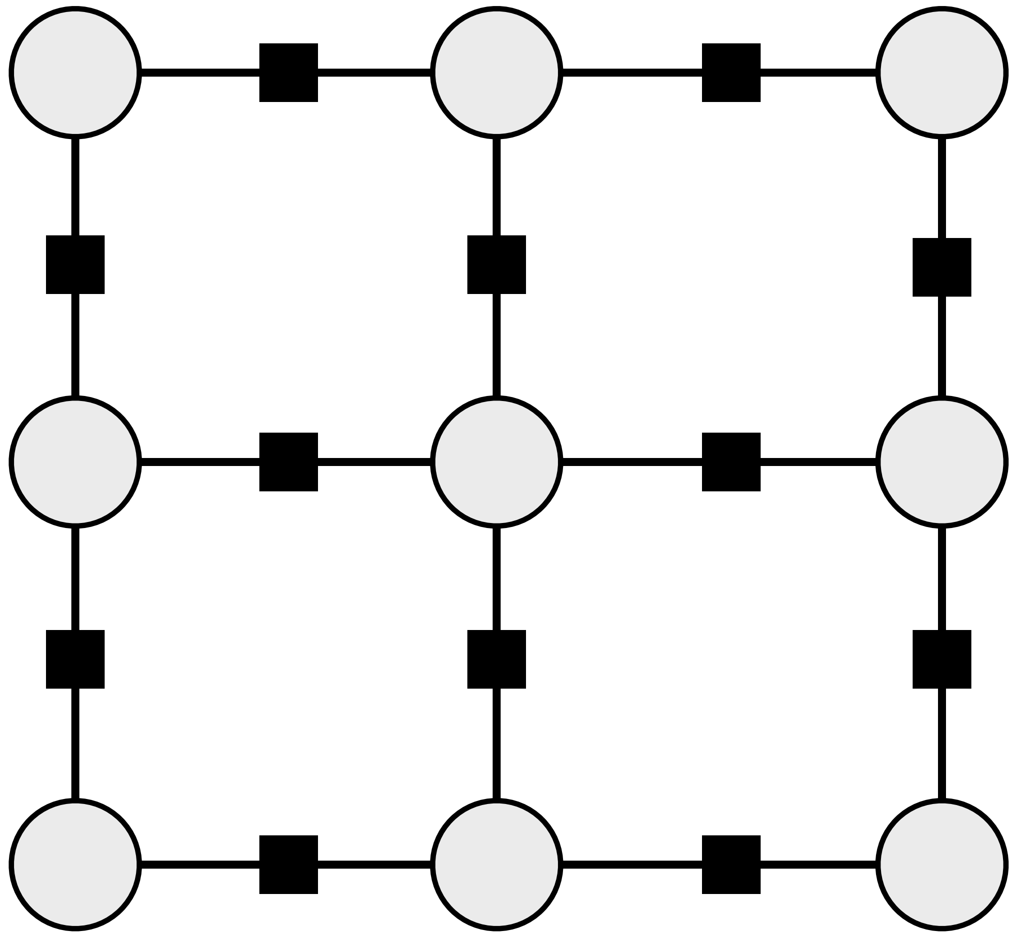
Treewidth Example



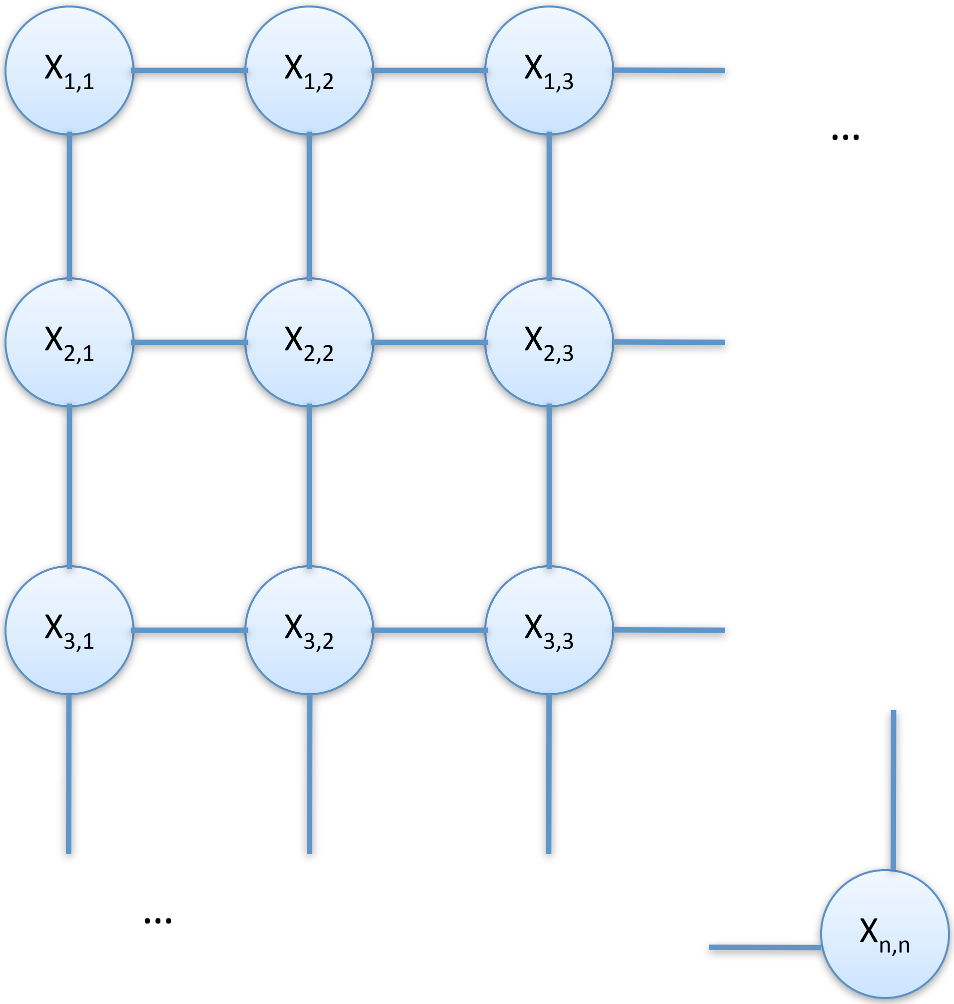
Treewidth Example



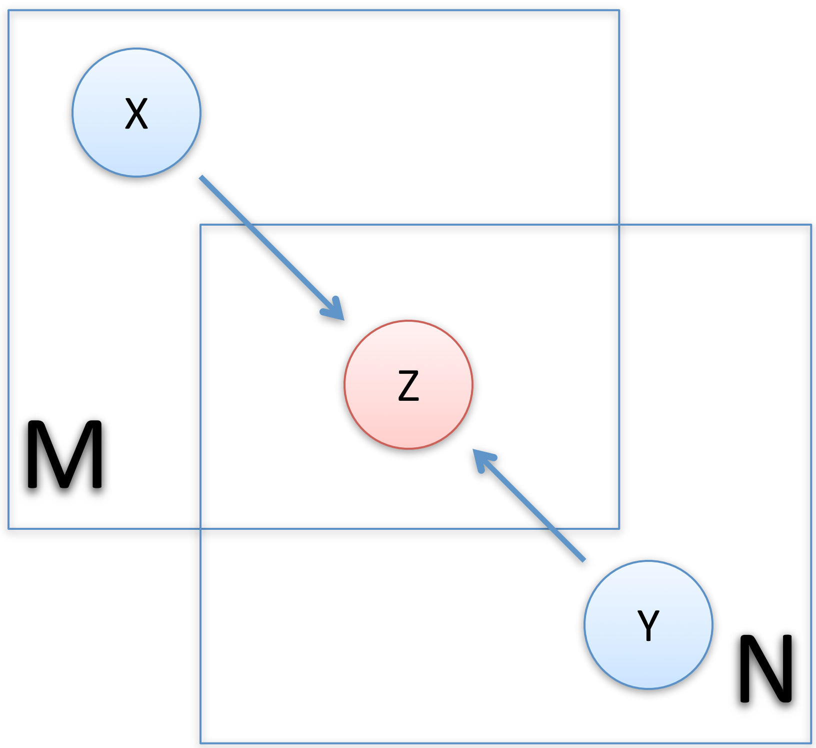
Treewidth Example



Treewidth Example



Treewidth Example



Finding Elimination Orderings

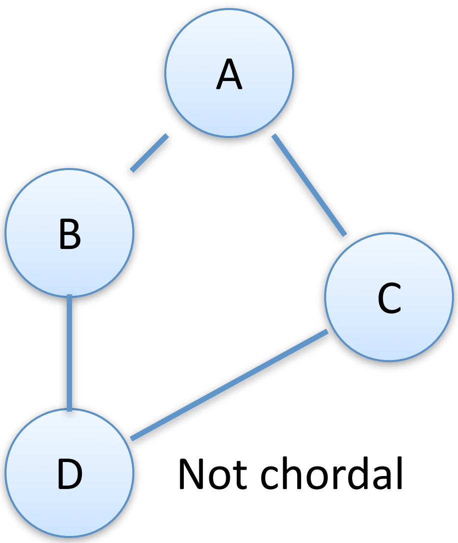
- NP-complete:
  "Is there an elimination ordering such that induced width $\leq$ K?"

- Nonetheless, some convenient cases arise.

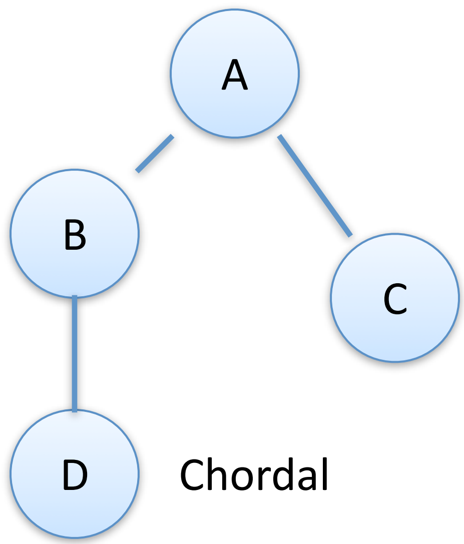
You'd like to be able to look at the *original* graph and easily say something about the difficulty of inference

Chordal Graphs

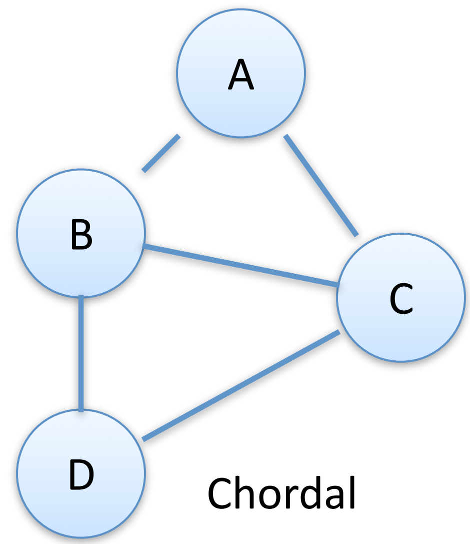
- Undirected graph whose minimal cycles are not longer than 3.



Not chordal



Chordal



Chordal

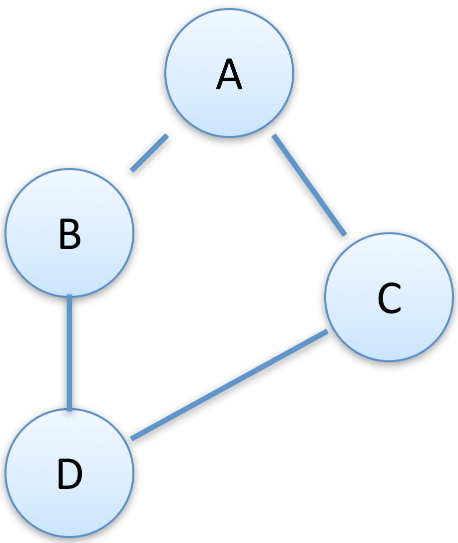
Lecture 5
(MN → BN)

Chordal Graphs

- Induced graphs are always chordal!

Chordal Graphs

- Induced graphs are always chordal!



induced graph? not chordal

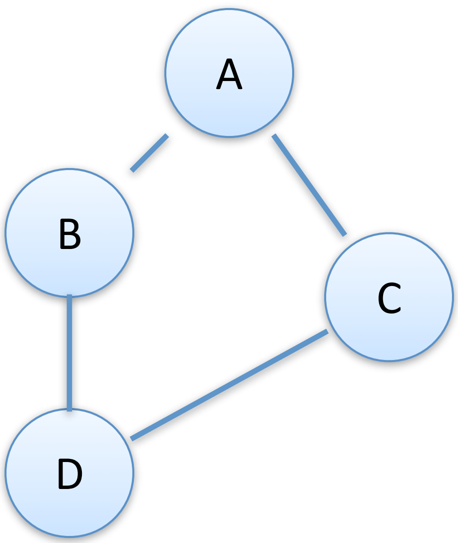
Lemma: cannot add any edges incident on X_i after it is eliminated.

When we eliminate C, edges A-C and C-D must exist.

After elimination A-D will exist.

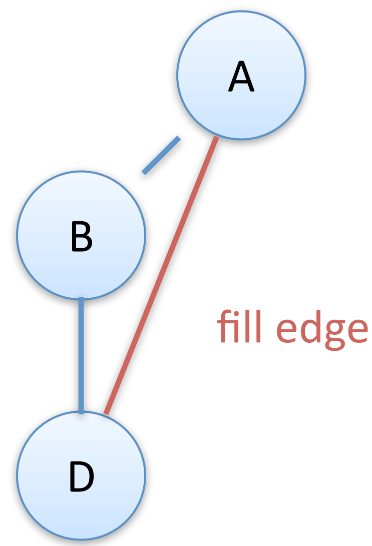
Chordal Graphs

- Induced graphs are always chordal!



induced graph? not chordal

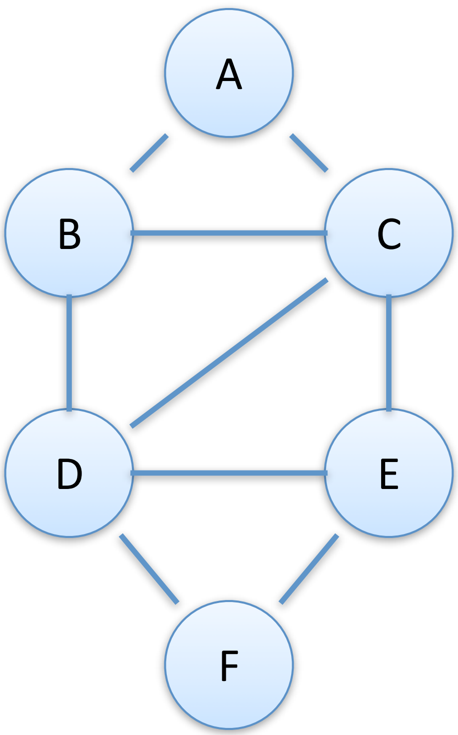
Lemma: cannot add any edges incident on X_i after it is eliminated.



Theorem

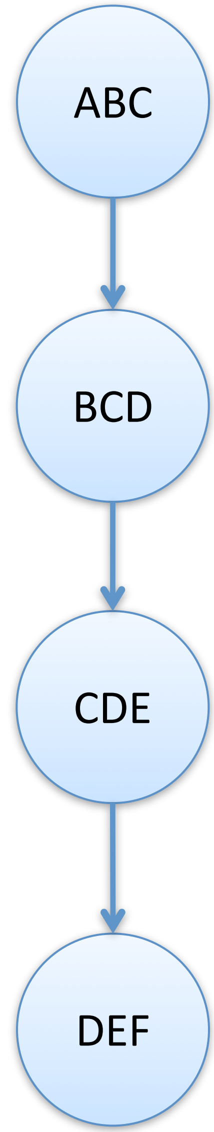
- Chordal graphs always admit an elimination ordering that doesn’t introduce any fill edges into the graph.
  - No fill edges: no blowup.
  - Inference becomes *linear* in size of the factors already present!

Clique Tree



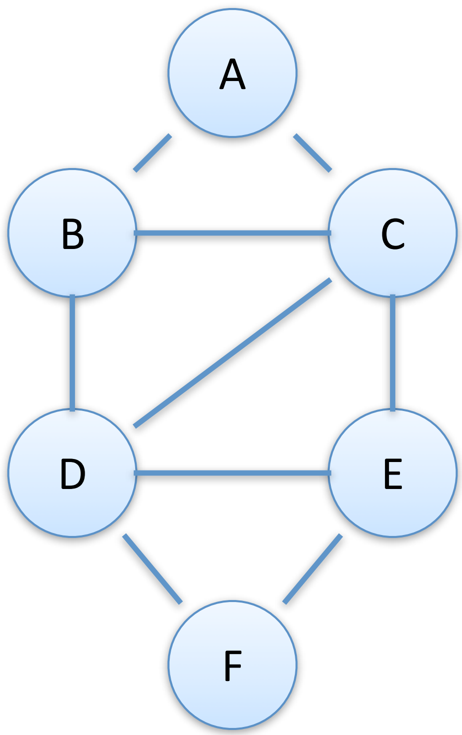
Every maximal clique becomes a vertex.

Tree structure.



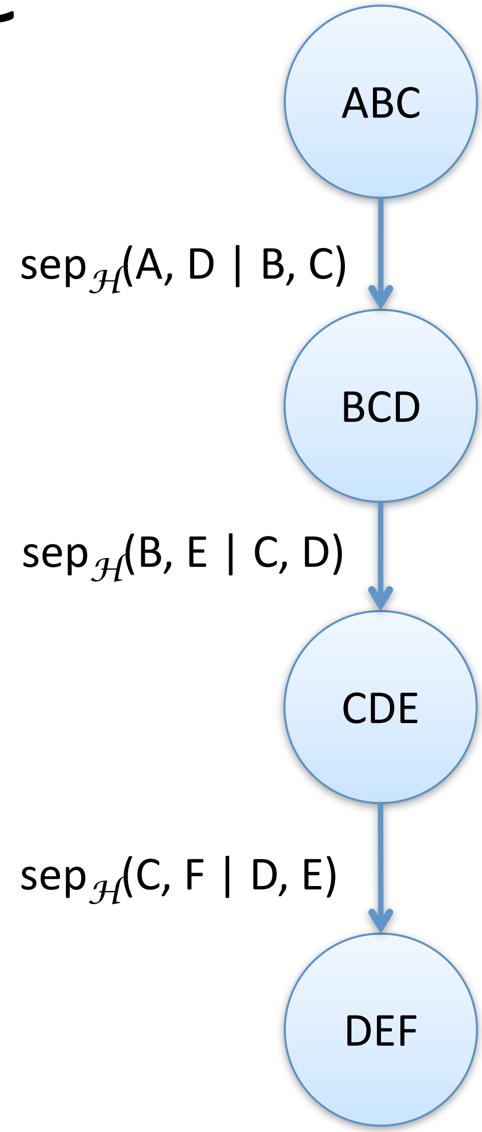
Lecture 5

Clique Tree



Lecture 5

For each edge, intersection of r.v.s *separates* the rest in $\mathcal{H}$.



Clique Tree

- Does a clique tree exist?
  - Yes, if the undirected graph $\mathcal{H}$ is chordal!

Lecture 5

Theorem

- Chordal graphs always admit an elimination ordering that doesn't introduce any fill edges into the graph.
- Proof by induction on the number of nodes in the tree:
  - Take a leaf $\mathbf{C}_i$ in the clique tree.
  - Eliminate a variable in $\mathbf{C}_i$ (but not $\mathbf{C}_i$'s neighbor).
    - No fill edges.
    - Still chordal.

Heuristics for Variable Elimination Ordering

other than using a clique tree

Alternative Ordering Heuristic: Maximum Cardinality

- Start with undirected graph on **X**, all nodes unmarked.
- For i = |**X**| to 1:
  - Let Y be the unmarked variable in **X** with the largest number of *marked* neighbors
  - $\pi(Y) = i$
  - Mark Y.
- Eliminate using permutation π.
  - i.e. π maps each variable to an integer;
    eliminate the variables in order of those integers 1, 2, 3...

Alternative Ordering Heuristic: Maximum Cardinality

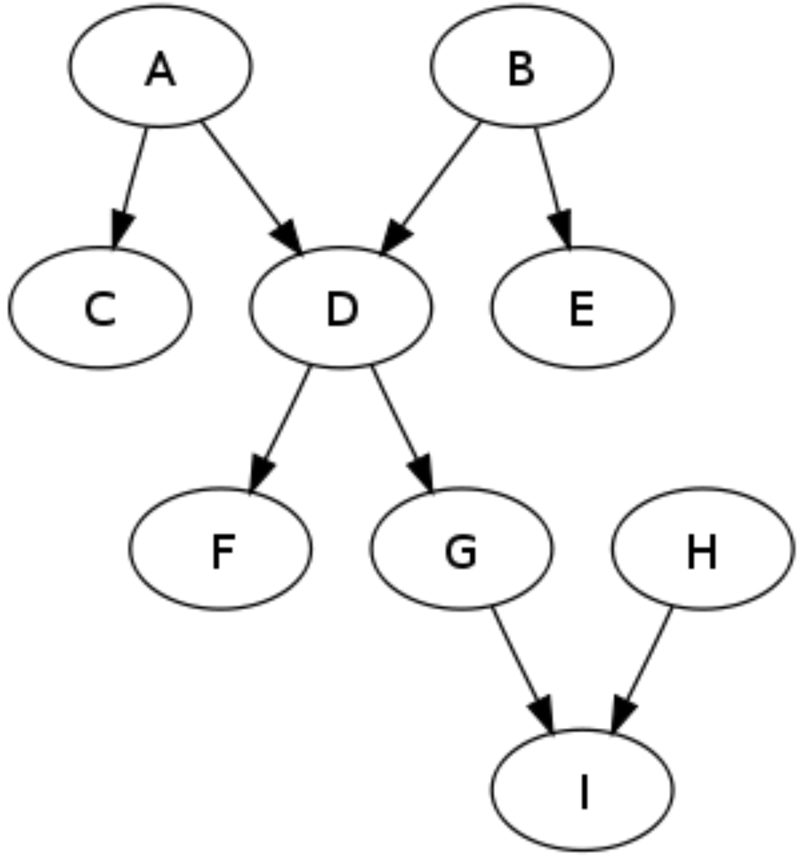
- “Maximum Cardinality” permutation will not introduce any fill edges for chordal graphs.
- Don’t need the clique tree.
- Can also use it on non-chordal graphs!
  – Better ideas exist, though;
    greedy algorithms that try to add a small number of edges at a time.

Bayesian Networks Again

- Recall: If undirected graph $\mathcal{H}$ is chordal, then there is a Bayesian network structure $\mathcal{G}$ that is a P-map for $\mathcal{H}$.
- Chordal graphs correspond to Bayesian networks with a **polytree** structure.
  - At most one trail between any pair of nodes.

"polytree" = directed graph with at most one undirected path between any two vertices; equiv: directed acyclic graph (DAG) for which there are no undirected cycles either.

Example Polytree



From Wikipedia :-)

Inference in Polytrees

- Linear in conditional probability table sizes!
  - Consider the skeleton.
  - Pick a root; form a tree.
  - Work in from "leaves."
- In the corresponding undirected graph, no fill edges are introduced.

Variable Elimination Summary

- 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 fill edges (or "big" intermediate factors), you can make inference linear in the size of the original factors.
  - Chordal graphs
  - Polytrees