Reasoning under uncertainty: Part 2

- D-Separation for conditional independence from Bayes nets
- Variable elimination: examples, algorithm, and complexity

Next: machine learning

Reading: Chap 18, 20, 21

*Slides based on those of Sheila McIlraith

Variable Independence

Two variables X and Y are conditionally independent given variable Z if for all $x \in \text{Dom}(X)$, $y \in \text{Dom}(Y)$, $z \in \text{Dom}(Z)$, $X = x$ and $Y = y$ are conditionally independent given $Z = z$, *i.e.*, $Pr(X = x \wedge Y = y | Z = z) =$

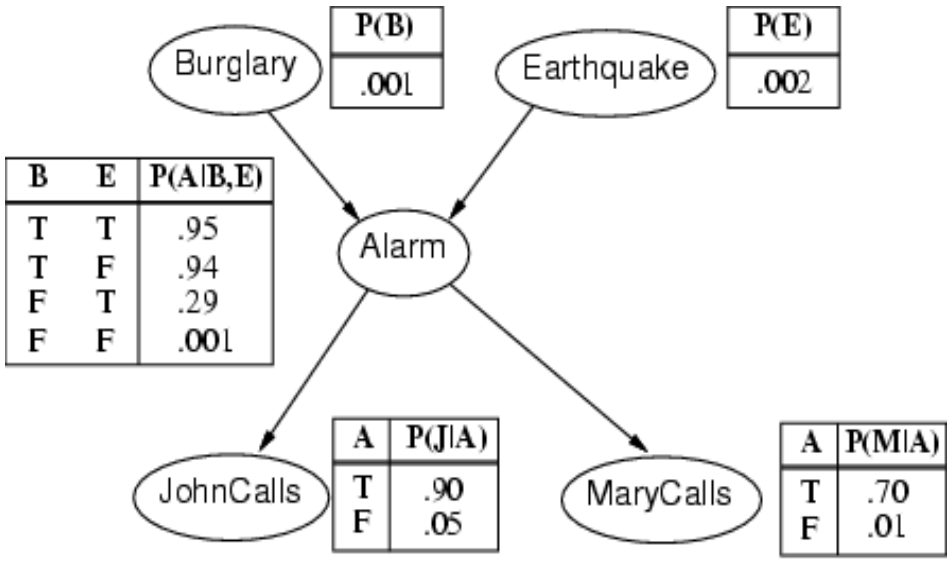
$$Pr(X = x | Z = z) \cdot Pr(Y = y | Z = z)$$

Bayesian Networks

A BN over variables $\{X_1, X_2, \ldots, X_n\}$ consists of:

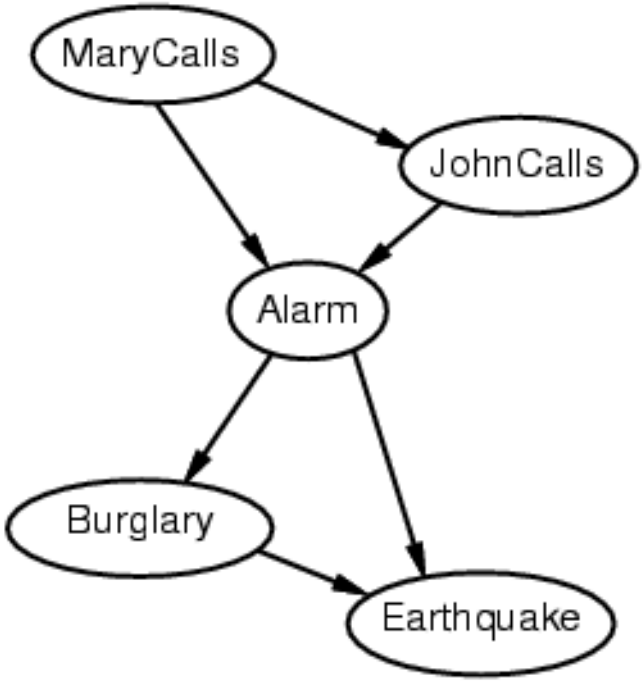
- a DAG (directed acyclic graph) whose nodes are the variables
- a set of CPTs (conditional probability tables) $Pr(X_i|Par(X_i))$ for each X_i

Inference: Given some evidence E, compute $Pr(X_i|E)$



Burglary example

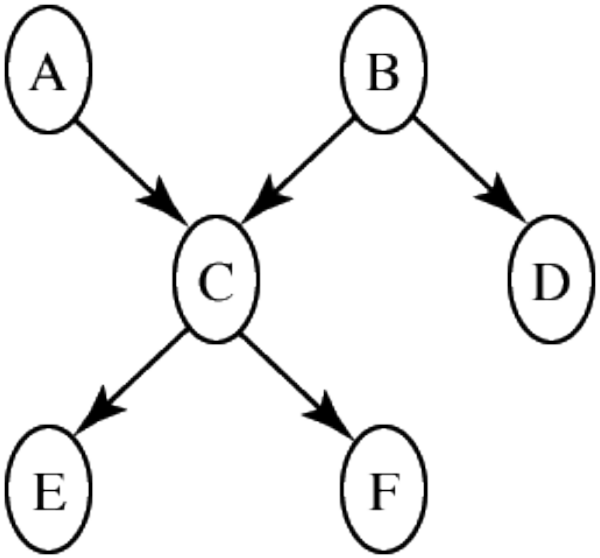
Use the order M,J,A,B,E



- Why $Pr(J|M) \neq Pr(J)$?
- Why $Pr(E|B, A, J, M) = Pr(E|B, A)$?
- Why $Pr(E|B, A) \neq Pr(E|A)$?

Exercise

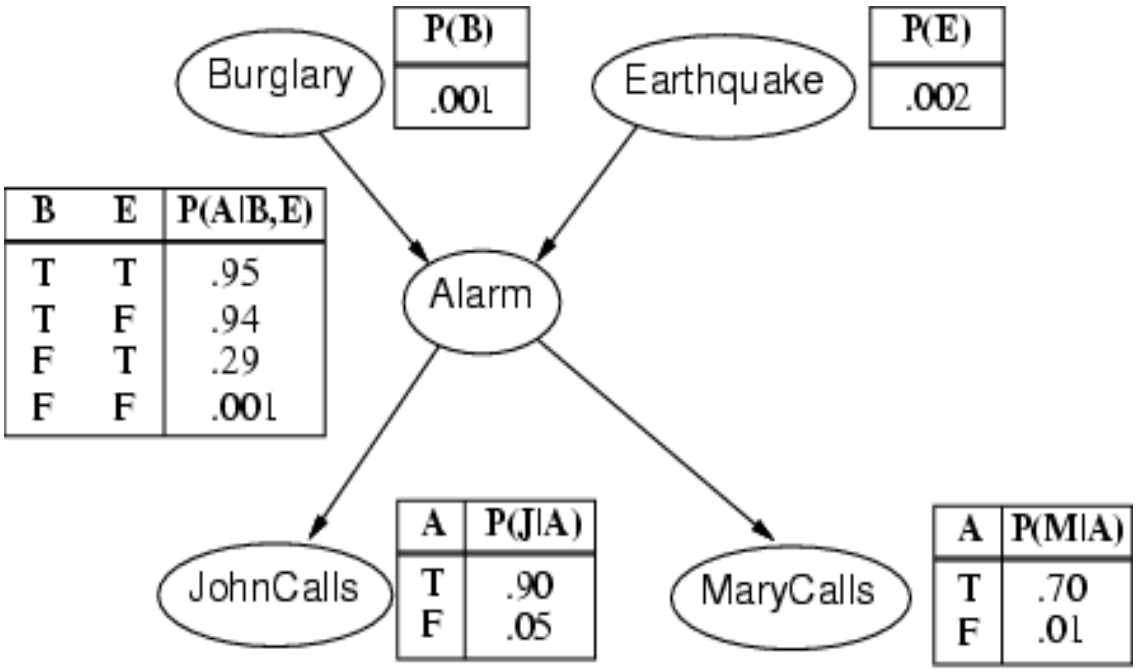
Compute $P(c|a, b, \neg d, \neg e, \neg f)$. Is it $= P(c|a, b)$?



$$
\begin{array}{rclrcl}
P(a) & = & 0.9 & P(d|b) & = & 0.1 \\
P(b) & = & 0.2 & P(d|\neg b) & = & 0.8 \\
P(c|a,b) & = & 0.1 & P(e|c) & = & 0.7 \\
P(c|a,\neg b) & = & 0.8 & P(e|\neg c) & = & 0.2 \\
P(c|\neg a,b) & = & 0.7 & P(f|c) & = & 0.2 \\
P(c|\neg a,\neg b) & = & 0.4 & P(f|\neg c) & = & 0.9
\end{array}
$$

Independence in a Bayes Net

- The structure of the BN means: every X_i is conditionally independent of all of its nondescendants given it parents:
  - $Pr(X_i|S \cup Par(Xi)) = Pr(Xi|Par(Xi))$ for any set S of non-descendents of X_i
- Given Alarm, JohnCalls and Earthquake are independent



More generally

- Many conditional independencies hold in a given BN.
- These independencies are useful in computation, explanation, etc.
- How do we determine if two variables X, Y are independent given a set of variables E?
- Answer: we use a (simple) graphical property

D-separation

- A set of variables E d-separates X and Y if it blocks every undirected path in the BN between X and Y.
- If evidence E d-separates X and Y, then X and Y are conditionally independent given evidence E
- So what is blocking?

Let P be an **undirected path** from X to Y in a BN.

Let **E** be a set of variables.

We say ***E blocks path P*** iff there is some node Z on the path such that:

Case 1: one arc on P ***goes into*** Z and one ***goes out*** of Z, and Z$\in$**E**; or

Case 2: both arcs on P leave Z, and Z$\in$**E**; or

Case 3: both arcs on P enter Z and *neither Z, nor any of its descendants*, are in **E**.

Blocking: Graphical View

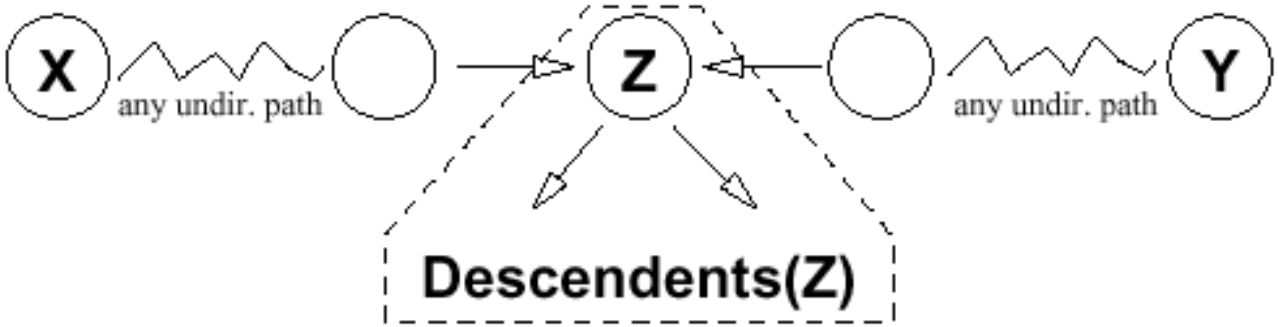
(1)

If Z in evidence, the path between X and Y blocked

(2)

If Z in evidence, the path between X and Y blocked

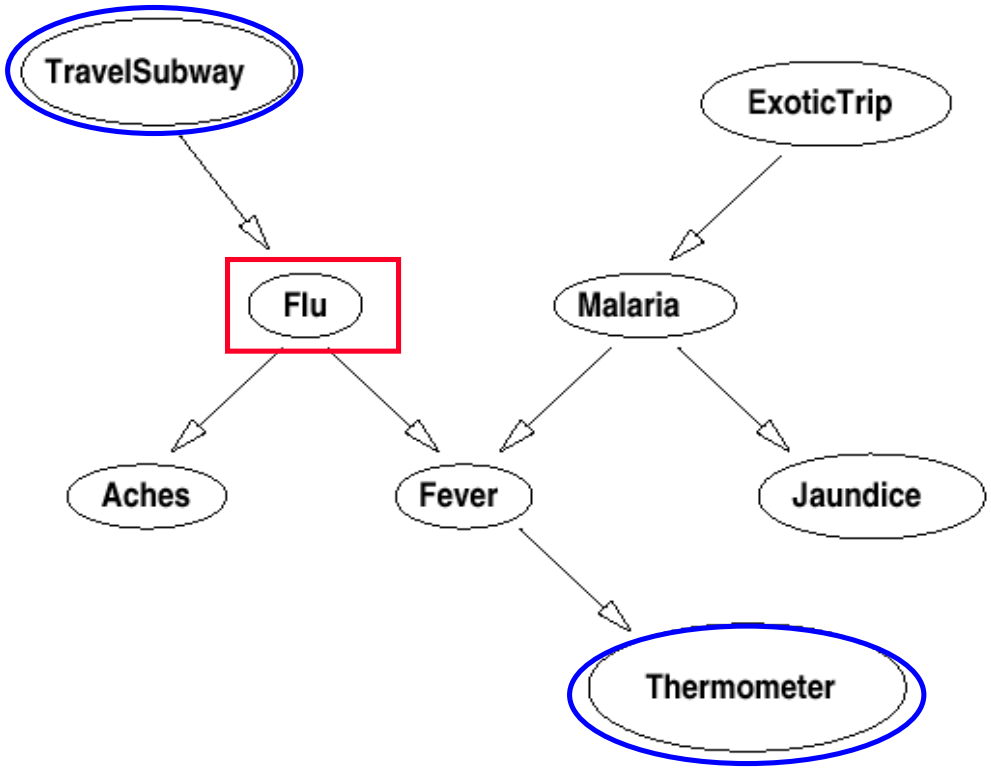
(3)

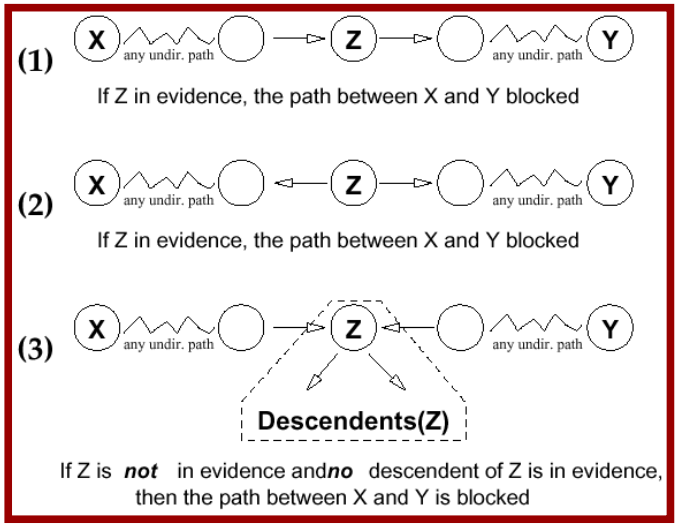


If Z is ***not*** in evidence and***no*** descendent of Z is in evidence, then the path between X and Y is blocked

D-Separation: Intuitions

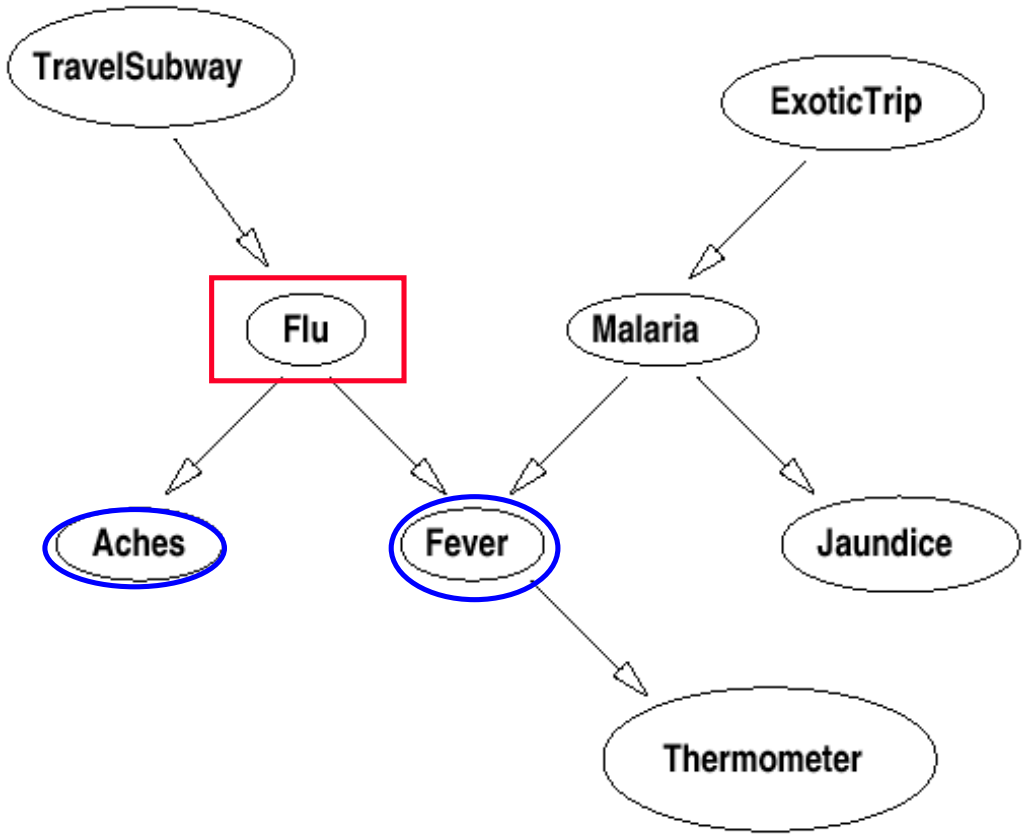
Subway and Thermometer are **dependent**; but are **independent given Flu** (since Flu blocks the only path (1))

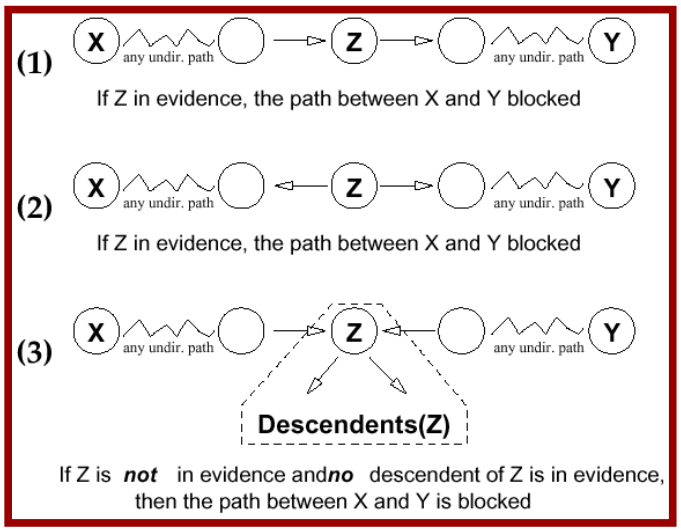




D-Separation: Intuitions

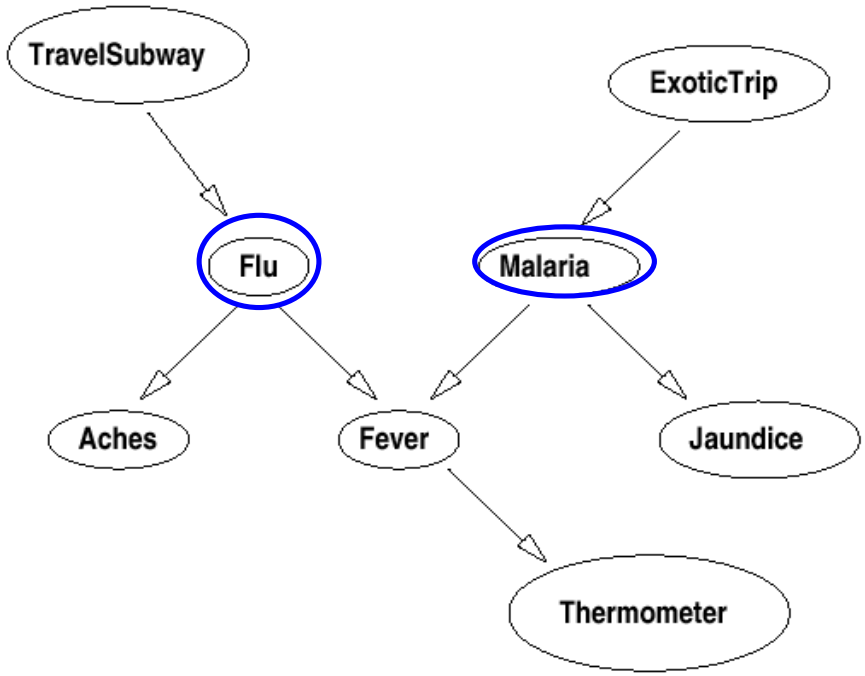
- Aches and Fever are **dependent**; but are **independent given Flu** (since Flu blocks the only path (via (2))).
- Similarly for Aches and Thermometer (dependent, but independent given Flu).

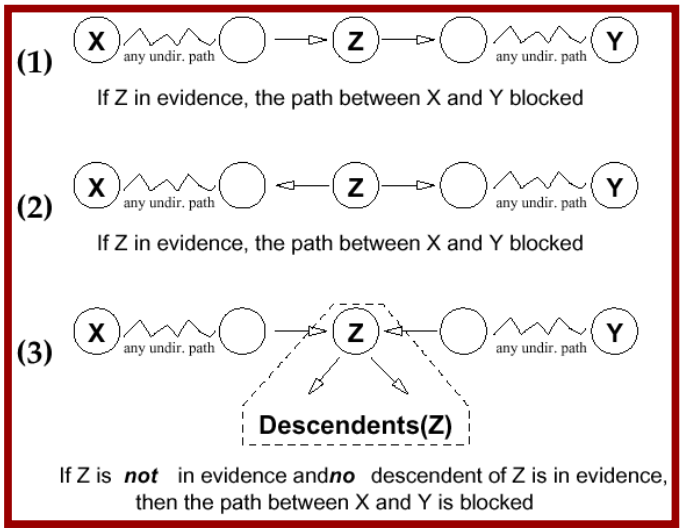




D-Separation: Intuitions

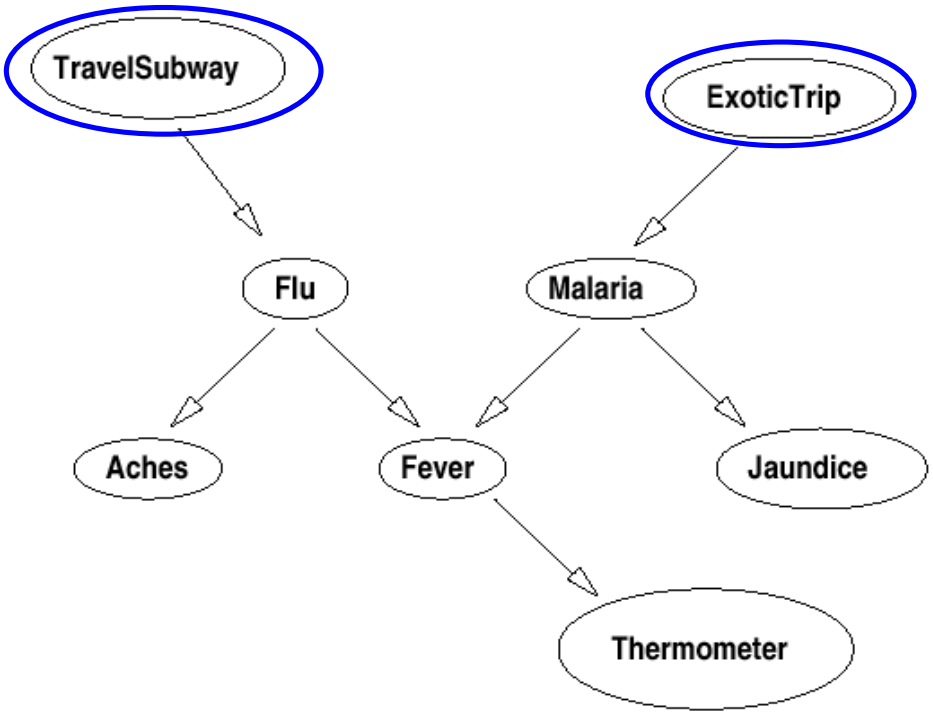
- Flu and Malaria are **independent (given no evidence):** Fever blocks the path, since it is *not in evidence*, nor is its descendant Thermometer (by (3))
- Flu and Malaria are **dependent** given Fever (or given Thermometer): nothing blocks path now. **What's the intuition?** Explaining Away: If you know John has Flu, it's explains the Fever, making Malaria less likely (and vice versa).

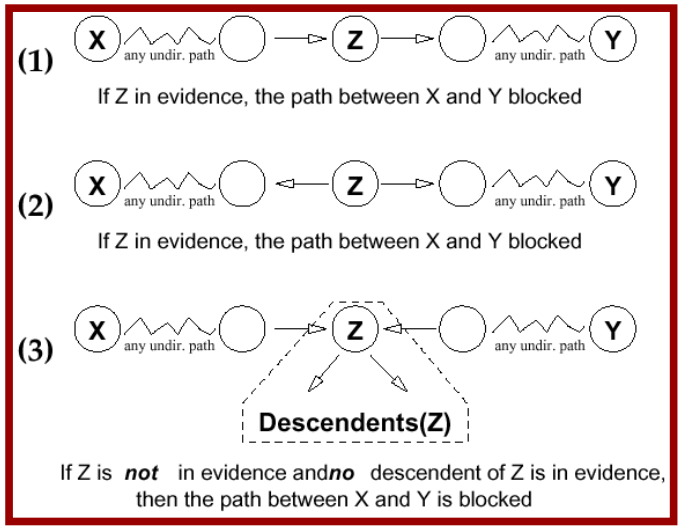




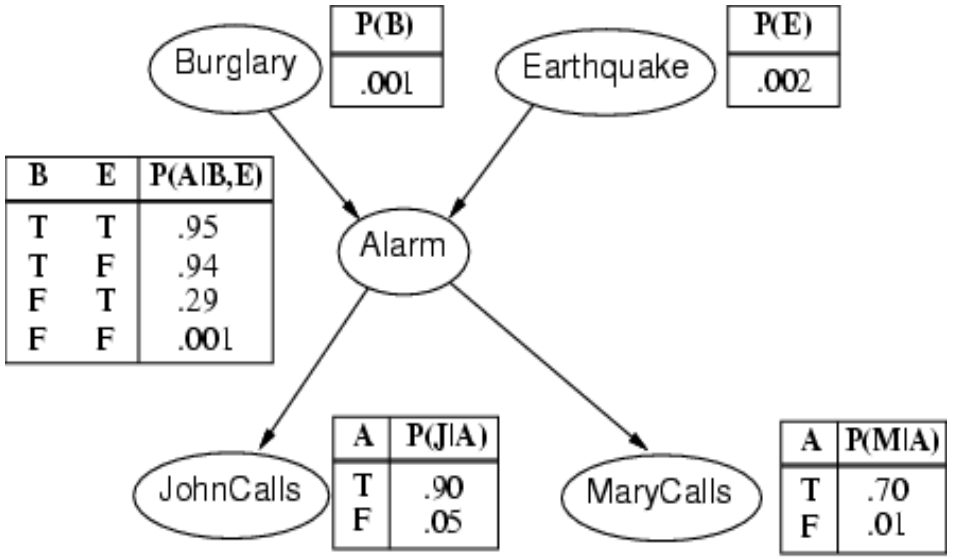
D-Separation: Intuitions

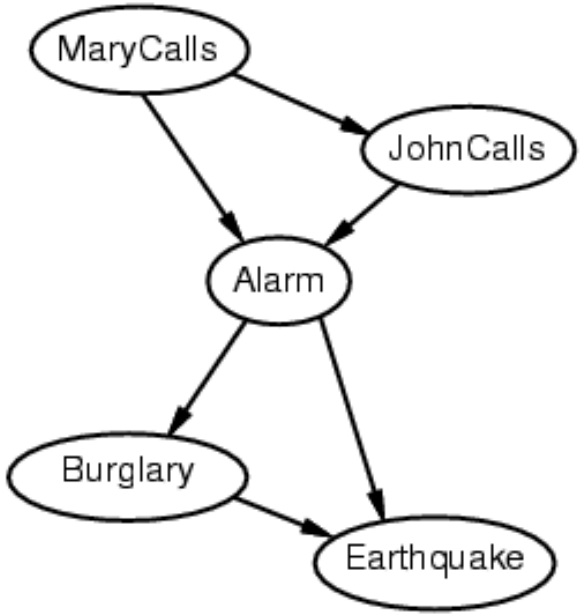
Subway and ExoticTrip are **independent** (by (3));
They are **dependent given Thermometer** ((3) is now violated by Thermometer);
They are **independent given Thermometer and Malaria**. This for exactly the same reasons for Flu/Malaria above.





Burglary example

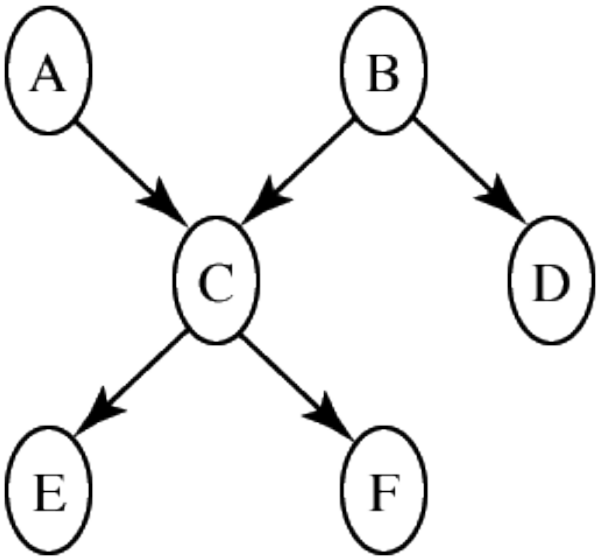




- A and M are dependent given J
- B and M are independent, given A
- J and M are dependent, but independent given A
- B and E are independent
- B and E are dependent, given A, J, or M

Exercise

Compute $P(c|a, b, \neg d, \neg e, \neg f)$. Note that it $\neq P(c|a, b)$



$$
\begin{array}{rclrcl}
P(a) & = & 0.9 & P(d|b) & = & 0.1 \\
P(b) & = & 0.2 & P(d|\neg b) & = & 0.8 \\
P(c|a, b) & = & 0.1 & P(e|c) & = & 0.7 \\
P(c|a, \neg b) & = & 0.8 & P(e|\neg c) & = & 0.2 \\
P(c|\neg a, b) & = & 0.7 & P(f|c) & = & 0.2 \\
P(c|\neg a, \neg b) & = & 0.4 & P(f|\neg c) & = & 0.9
\end{array}
$$

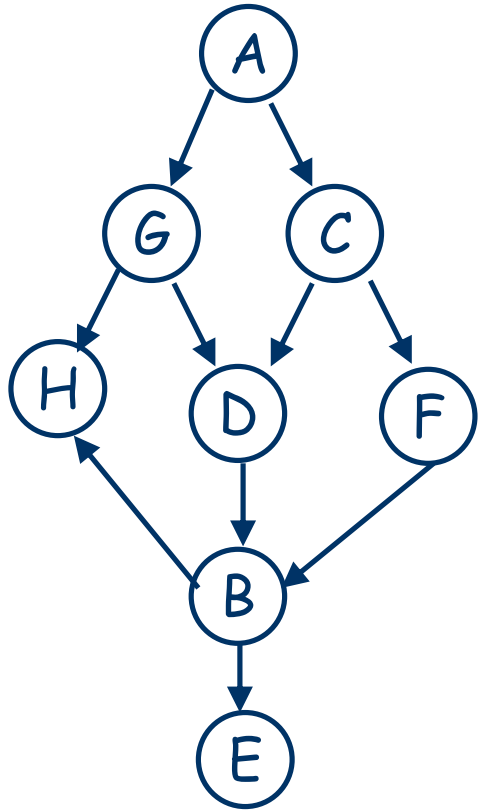
Given A and B, C and E are dependent

D-Separation Example

In the following network,

determine if **A and E are independent** given the evidence:

1. A and E given no evidence? **No**
2. A and E given {C}? **No**
3. A and E given {G,C}? Y
4. A and E given {G,C,H}? Y
5. A and E given {G,F}? **No**
6. A and E given {F,D}? Y
7. A and E given {F,D,H}? **No**
8. A and E given {B}? Y
9. A and E given {H,B}? Y
10. A and E given {G,C,D,H,D,F,B}? Y



Why the answer to 7 is No?

Note: A set of variables E d-separates X and Y if it blocks every undirected path in the BN between X and Y.

The path between A and E via H is not blocked

An example

Let's look a little closer at **how we compute the two sums** in (1)

Consider

$$\Pr(d,h,-i) = \sum_{A,B,C,E,F,G,J,K} \Pr(A,B,C,d,E,F,h,-i,J,K)$$

Use Bayes Net **product decomposition** to rewrite summation:

$$\sum_{A,B,C,E,F,G,J,K} \Pr(A,B,C,d,E,F,h,-i,J,K)$$

$$= \sum_{A,B,C,E,F,G,J,K} \Pr(A)\Pr(B)\Pr(C|A)\Pr(d|A,B)\Pr(E|C)$$
$$\Pr(F|d)\Pr(G)\Pr(h|E,F)\Pr(-i|F,G)\Pr(J|h,-i)$$
$$\Pr(K|-i)$$

An example

1) Move product terms out so they are scoped appropriately

$$= \sum_A \sum_B \sum_C \sum_E \sum_F \sum_G \sum_J \sum_K \Pr(A)\Pr(B)\Pr(C|A)\Pr(d|A,B)\Pr(E|C)$$
$$\Pr(F|d)\Pr(G)\Pr(h|E,F)\Pr(-i|F,G)\Pr(J|h,-i)$$
$$\Pr(K|-i)$$

$$= \sum_A \Pr(A) \sum_B \Pr(B) \sum_C \Pr(C|A) \Pr(d|A,B) \sum_E \Pr(E|C)$$
$$\sum_F \Pr(F|d) \sum_G \Pr(G)\Pr(h|E,F)\Pr(-i|F,G) \sum_J \Pr(J|h,-i)$$
$$\sum_K \Pr(K|-i)$$

$$= \sum_A \Pr(A) \sum_B \Pr(B) \Pr(d|A,B) \sum_C \Pr(C|A) \sum_E \Pr(E|C)$$
$$\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G) \sum_J \Pr(J|h,-i)$$
$$\sum_K \Pr(K|-i)$$

An example

Now start computing.

$$\sum_A \Pr(A) \sum_B \Pr(B) \Pr(d|A,B) \sum_C \Pr(C|A) \sum_E \Pr(E|C)$$
$$\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G)$$
$$\sum_J \Pr(J|h,-i)$$
$$\sum_K \Pr(K|-i)$$

2) Compute the things that are numbers/constants

$$\sum_K \Pr(K|-i) = \Pr(k|-i) + \Pr(-k|-i) = c_1$$

$$\sum_J \Pr(J|h,-i)\ c_1 = c_1 \sum_J \Pr(J|h,-i)$$
$$= c_1\ (\Pr(j|h,-i) + \Pr(-j|h,-i))$$
$$= c_1 c_2$$

An example

$$\sum_A \Pr(A) \sum_B \Pr(B) \Pr(d|A,B)$$
$$\sum_C \Pr(C|A) \sum_E \Pr(E|C) \sum_F \Pr(F|d) \Pr(h|E,F)$$
$$\sum_G \Pr(G) \Pr(-i|F,G) \sum_J \Pr(J|h,-i) \sum_K \Pr(K|-i)$$

$$c_1 c_2 \sum_G \Pr(G) \Pr(-i|F,G)$$
$$= c_1 c_2 (\Pr(g)\Pr(-i|F,g) + \Pr(-g)\Pr(-i|F,-g))$$

!!But $\Pr(-i|F,g)$ depends on the value of F, so this is not a single number.

An example

So, Let's try eliminate in outside->inside order:

$$\sum_A \Pr(A) \sum_B \Pr(B) \Pr(d|A,B) \sum_C \Pr(C|A) \sum_E \Pr(E|C) \sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G) \sum_J \Pr(J|h,-i) \sum_K \Pr(K|-i)$$

=

$$\Pr(a) \sum_B \Pr(B) \Pr(d|a,B) \sum_C \Pr(C|a) \sum_E \Pr(E|C) \sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G) \sum_J \Pr(J|h,-i) \sum_K \Pr(K|-i)$$

+

$$\Pr(-a) \sum_B \Pr(B) \Pr(d|-a,B) \sum_C \Pr(C|-a) \sum_E \Pr(E|C) \sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G) \sum_J \Pr(J|h,-i) \sum_K \Pr(K|-i)$$

An example

=

Pr(a)Pr(b) $\Pr(d|a,b) \sum_C \Pr(C|a) \sum_E \Pr(E|C)$
$\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G)$
$\sum_J \Pr(J|h,-i)$
$\sum_K \Pr(K|-i)$

+

Pr(a)Pr(-b) $\Pr(d|a,-b) \sum_C \Pr(C|a) \sum_E \Pr(E|C)$
$\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G)$
$\sum_J \Pr(J|h,-i)$
$\sum_K \Pr(K|-i)$

+

Pr(-a)Pr(b) $\Pr(d|-a,b) \sum_C \Pr(C|-a) \sum_E \Pr(E|C)$
$\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G)$
$\sum_J \Pr(J|h,-i)$
$\sum_K \Pr(K|-i)$

+

Pr(-a)Pr(-b) $\Pr(d|-a,-b) \sum_C \Pr(C|-a) \sum_E \Pr(E|C)$
$\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G)$
$\sum_J \Pr(J|h,-i)$
$\sum_K \Pr(K|-i)$

Problem: The size of the sum is doubling as we expand each variable (into -v and v). This approach has exponential complexity. But let's look a bit closer.

An example

$$
\begin{aligned}
&= \\
&\Pr(a)\Pr(b)\ \Pr(d|a,b) \sum_C \Pr(C|a) \sum_E \Pr(E|C) \\
&\qquad \sum_F \Pr(F|d)\ \Pr(h|E,F) \sum_G \Pr(G)\ \Pr(-i|F,G) \\
&\qquad \sum_J \Pr(J|h,-i) \\
&\qquad \sum_K \Pr(K|-i) \\
&+ \\
&\Pr(a)\Pr(-b)\ \Pr(d|a,-b) \sum_C \Pr(C|a) \sum_E \Pr(E|C) \\
&\qquad \sum_F \Pr(F|d)\ \Pr(h|E,F) \sum_G \Pr(G)\ \Pr(-i|F,G) \\
&\qquad \sum_J \Pr(J|h,-i) \\
&\qquad \sum_K \Pr(K|-i) \\
&+ \\
&\Pr(-a)\Pr(b)\ \Pr(d|-a,b) \sum_C \Pr(C|-a) \sum_E \Pr(E|C) \\
&\qquad \sum_F \Pr(F|d)\ \Pr(h|E,F) \sum_G \Pr(G)\ \Pr(-i|F,G) \\
&\qquad \sum_J \Pr(J|h,-i) \\
&\qquad \sum_K \Pr(K|-i) \\
&+ \\
&\Pr(-a)\Pr(-b)\ \Pr(d|-a,-b) \sum_C \Pr(C|-a) \sum_E \Pr(E|C) \\
&\qquad \sum_F \Pr(F|d)\ \Pr(h|E,F) \sum_G \Pr(G)\ \Pr(-i|F,G) \\
&\qquad \sum_J \Pr(J|h,-i) \\
&\qquad \sum_K \Pr(K|-i)
\end{aligned}
$$

Repeated subterm

Repeated subterm

Solution: If we store the value of the subterms, we need only compute them once.

Dynamic Programming

=

$\Pr(a)\Pr(b)\ \Pr(d|a,b) \sum_C \Pr(C|a) \sum_E \Pr(E|C)$
$\sum_F \Pr(F|d)\ \Pr(h|E,F) \sum_G \Pr(G)\ \Pr(-i|F,G)$
$\sum_J \Pr(J|h,-i)$
$\sum_K \Pr(K|-i)$

+

$\Pr(a)\Pr(-b)\ \Pr(d|a,-b) \sum_C \Pr(C|a) \sum_E \Pr(E|C)$
$\sum_F \Pr(F|d)\ \Pr(h|E,F) \sum_G \Pr(G)\ \Pr(-i|F,G)$
$\sum_J \Pr(J|h,-i)$
$\sum_K \Pr(K|-i)$

f_1

+

$\Pr(-a)\Pr(b)\ \Pr(d|-a,b) \sum_C \Pr(C|-a) \sum_E \Pr(E|C)$
$\sum_F \Pr(F|d)\ \Pr(h|E,F) \sum_G \Pr(G)\ \Pr(-i|F,G)$
$\sum_J \Pr(J|h,-i)$
$\sum_K \Pr(K|-i)$

+

$\Pr(-a)\Pr(-b)\ \Pr(d|-a,-b) \sum_C \Pr(C|-a) \sum_E \Pr(E|C)$
$\sum_F \Pr(F|d)\ \Pr(h|E,F) \sum_G \Pr(G)\ \Pr(-i|F,G)$
$\sum_J \Pr(J|h,-i)$
$\sum_K \Pr(K|-i)$

f_2

$$= c_1 f_1 + c_2 f_1 + c_3 f_2 + c_4 f_2$$

$$c_1 = \Pr(a)\Pr(b)\ \Pr(d|a,b)$$

$$c_2 = \Pr(a)\Pr(-b)\ \Pr(d|a,-b)$$

$$c_3 = \Pr(-a)\Pr(b)\ \Pr(d|-a,b)$$

$$c_4 = \Pr(-a)\Pr(-b)\ \Pr(d|-a,-b)$$

Dynamic Programming

$$f_1 = \sum_C \Pr(C|a) \sum_E \Pr(E|C) \sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G) \sum_J \Pr(J|h,-i) \sum_K \Pr(K|-i)$$

$$= \Pr(c|a) \sum_E \Pr(E|c) \sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G) \sum_J \Pr(J|h,-i) \sum_K \Pr(K|-i)$$

$$+ \Pr(-c|a) \sum_E \Pr(E|-c) \sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G) \sum_J \Pr(J|h,-i) \sum_K \Pr(K|-i)$$

Repeated subterm

Dynamic Programming

- So within the computation of the subterms we obtain more repeated smaller subterms.
- he core idea of dynamic programming is to remember all "smaller" computations, so that they can be reused.
- This can convert an exponential computation into one that takes only polynomial time.
- Variable elimination is a dynamic programming technique that computes the sum from the bottom up (starting with the smaller subterms and working its way up to the bigger terms).

Relevant

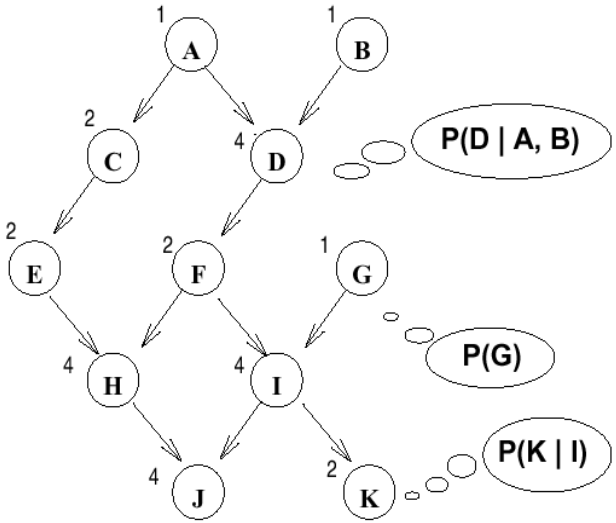
A brief aside .... note that in the sum

$$\sum_A \Pr(A) \sum_B \Pr(B) \Pr(d|A,B) \sum_C \Pr(C|A) \sum_E \Pr(E|C)$$
$$\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G)$$
$$\sum_J \Pr(J|h,-i)$$
$$\sum_K \Pr(K|-i)$$

we have that $\sum_K \Pr(K|-i) = 1$ (Why?), thus

$$\sum_J \Pr(J|h,-i) \sum_K \Pr(K|-i) = \sum_J \Pr(J|h,-i)$$

Furthermore $\sum_J \Pr(J|h,-i) = 1$.



So we could drop these last two terms from the computation:
J and K are irrelevant given our query D and evidence -i and -h.
For now we keep these terms.

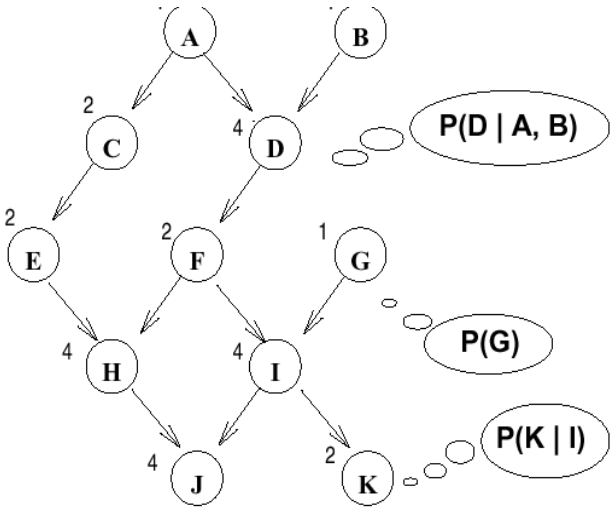
Variable Elimination (VE)

VE works from the inside out, summing out K, then J, G, ...

- Recall, when we tried to sum out G

$$\sum_A \Pr(A) \sum_B \Pr(B) \Pr(d|A,B) \sum_C \Pr(C|A) \sum_E \Pr(E|C)$$
$$\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G)$$
$$\sum_J \Pr(J|h,-i)$$
$$\sum_K \Pr(K|-i)$$

$$c_1 c_2 \sum_G \Pr(G) \Pr(-i|F,G)$$
$$= c_1 c_2 (\Pr(g)\Pr(-i|F,g) + \Pr(-g)\Pr(-i|F,-g))$$



we found that Pr(-i|F,-g) depends on the value of F, it wasn't a single number.

- However, we can still continue with the computation by computing two different numbers, one for each value of F

Variable Elimination (VE)

$$\sum_A \Pr(A) \sum_B \Pr(B) \Pr(d|A,B) \sum_C \Pr(C|A) \sum_E \Pr(E|C)$$
$$\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G)$$
$$\sum_J \Pr(J|h,-i)$$
$$\sum_K \Pr(K|-i)$$

- $t(-f) = c_1 c_2 \sum_G \Pr(G) \Pr(-i|-f,G)$
  $t(f) = c_1 c_2 (\sum_G \Pr(G) \Pr(-i|f,G)$
  - $t(-f) = c_1 c_2 (\Pr(g)\Pr(-i|-f,g) + \Pr(-g)\Pr(-i|-f,-g))$
  - $t(f) = c_1 c_2 (\Pr(g)\Pr(-i|f,g) + \Pr(-g)\Pr(-i|f,-g))$
- Now we sum out F

Variable Elimination (VE)

- $\sum_A \Pr(A) \sum_B \Pr(B) \Pr(d|A,B) \sum_C \Pr(C|A) \sum_E \Pr(E|C)$
  $\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G)$
  $\sum_J \Pr(J|h,-i)$
  $\sum_K \Pr(K|-i)$

$c_1 c_2 \sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G)$

$= c_1 c_2 (\Pr(f|d) \Pr(h|E,f)(\sum_G \Pr(G) \Pr(-i|f,G)) + \Pr(-f|d) \Pr(h|E,-f)(\sum_G \Pr(G) \Pr(-i|-f,G)))$

$= c_1 c_2 \sum_F \Pr(F|d) \Pr(h|E,F) t(F)$

$t(f), t(-f)$

Variable Elimination (VE)

- $c_1 c_2$ (Pr(f|d) Pr(h|E,f) t(f)
  + Pr(-f|d)Pr(h|E,-f) t(-f)

This is a function of E, so we obtain <u>two new numbers</u>

s(e) = $c_1 c_2$ (Pr(f|d) Pr(h|e,f)t(f)
  + Pr(-f|d)Pr(h|e,-f) t(-f)

s(-e) = $c_1 c_2$(Pr(f|d) Pr(h|-e,f)t(f)
  + Pr(-f|d)Pr(h|-e,-f) t(-f)

Variable Elimination (VE)

$$\sum_A \Pr(A) \sum_B \Pr(B) \Pr(d|A,B) \sum_C \Pr(C|A) \sum_E \Pr(E|C)$$
$$\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(\text{-}i|F,G)$$
$$\sum_J \Pr(J|h,\text{-}i)$$
$$\sum_K \Pr(K|\text{-}i)$$

- On summing out E we obtain two numbers, or a function of C.
- Then a function of B, then a function of A.
- On finally summing out A we obtain the single number we wanted to compute which is Pr(d,h,-i).
- Now we can repeat the process to compute Pr(-d,h,-i).
- Instead, we can regard D as a variable in the computation.
- This way, summing out A will yield a function of D.

Variable Elimination (VE)

- In general, at each stage VE will compute a table of numbers: one for each different instantiation of the variables in the sum.
- The size of these tables is exponential in the number of variables appearing in the sum, *e.g.*

$$\sum_F Pr(F|D)Pr(h|E,F)t(F)$$

depends on the value of D and E, thus we will obtain $|Dom[D]||Dom[E]|$ different numbers in the resulting table.

Factors

- We call these tables of values computed by VE factors.
- Note that the original CPTs are also table of values.
  Thus we also all them factors
- Each factor is a function of some variables, *e.g.*, $P(C|A) = f(A, C)$: it maps each value of its arguments to a number.
- A tabular representation is exponential in the number of variables in the factor.
- Notation: $f(\mathbf{X}, \mathbf{Y})$ denotes a factor over the variables $\mathbf{X} \cup \mathbf{Y}$ (where $\mathbf{X}$ and $\mathbf{Y}$ are sets of variables)
- If we examine the inside-out summation process we see that various operations occur on factors.

The Product of Two Factors

- Let f(**X**,**Y**) & g(**Y**,**Z**) be two factors with variables **Y** in common
- The ***product*** of f and g, denoted h = f × g (or sometimes just h = fg), is defined:

$$h(\mathbf{X},\mathbf{Y},\mathbf{Z}) = f(\mathbf{X},\mathbf{Y}) \times g(\mathbf{Y},\mathbf{Z})$$

f(A,B)		g(B,C)		h(A,B,C)			
ab	0.9	bc	0.7	abc	0.63	ab~c	0.27
a~b	0.1	b~c	0.3	a~bc	0.08	a~b~c	0.02
~ab	0.4	~bc	0.8	~abc	0.28	~ab~c	0.12
~a~b	0.6	~b~c	0.2	~a~bc	0.48	~a~b~c	0.12

Summing a Variable Out of a Factor

- Let f(X,**Y**) be a factor with variable X (**Y** is a set)
- We ***sum out*** variable X from f to produce a new factor h = Σ_X f, which is defined:

$$h(\mathbf{Y}) = \Sigma_{x \in Dom(X)} f(x, \mathbf{Y})$$

f(A,B)		h(B)	
ab	0.9	b	1.3
a~b	0.1	~b	0.7
~ab	0.4		
~a~b	0.6		

No error in the table. Here $f(A, B)$ is not $P(AB)$, but $P(B|A)$.

Restricting a Factor

- Let f(X,**Y**) be a factor with variable X (**Y** is a set)
- We ***restrict*** factor f *to* X=*a* by setting X to the value *a* and "deleting" incompatible elements of f's domain . Define $h = f_{X=a}$ as: h(**Y**) = f(a,**Y**)

f(A,B)		$h(B) = f_{A=a}$	
ab	0.9	b	0.9
a~b	0.1	~b	0.1
~ab	0.4		
~a~b	0.6		

The VE Algorithm

Given a Bayes Net with CPTs F, query variable Q, evidence variables **E** (observed to have values e), and remaining variables **Z**. Compute Pr(Q|**E**)

1. Replace each factor $f \in F$ that mentions a variable(s) in **E** with its restriction $f_{\mathbf{E}=e}$ (this might yield a "constant" factor)
2. For each Z_j– in the order given –eliminate $Z_j \in \mathbf{Z}$ as follows:
   1. Let $f_1, f_2, \ldots, f_k$ be the factors in F that include Z_j
   2. Compute new factor $g_j = \sum_{Z_j} f_1 \times f_2 \times \ldots \times f_k$
   3. Remove the factors f_i from F and add new factor g_j to F
3. The remaining factors refer only to the query variable Q. Take their product and normalize to produce Pr(Q|**E**).

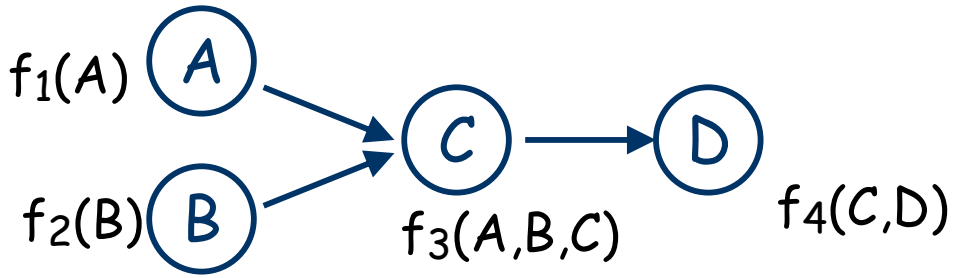
VE: Example

Factors: $f_1(A)$ $f_2(B)$ $f_3(A,B,C)$ $f_4(C,D)$

Query: P(A)?

Evidence: D = d

Elim. Order: C, B



Restriction: replace $f_4(C,D)$ with $f_5(C) = f_4(C,d)$

Step 1: Eliminating C: Compute & Add $f_6(A,B) = \Sigma_C f_5(C) f_3(A,B,C)$

Remove: $f_3(A,B,C)$, $f_5(C)$

Step 2: Eliminating B: Compute & Add $f_7(A) = \Sigma_B f_6(A,B) f_2(B)$

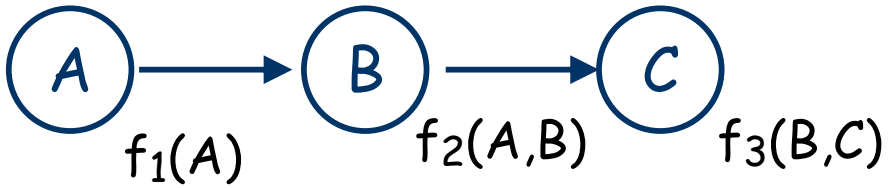
Remove: $f_6(A,B)$, $f_2(B)$

Last factors: $f_7(A)$, $f_1(A)$. The product $f_1(A) \times f_7(A)$ is (unnormalized) posterior. So… $P(A|d) = \alpha f_1(A) \times f_7(A)$ where $\alpha = 1/\sum_A f_1(A)f_7(A)$ ← **Note the Normalization Constant!**

Numeric example

Here's an example with some numbers

Eliminate A then B



0.85=0.9*0.9 + 0.1*0.4
0.15=0.9*0.1 + 0.1*0.6

Eliminate A Eliminate B

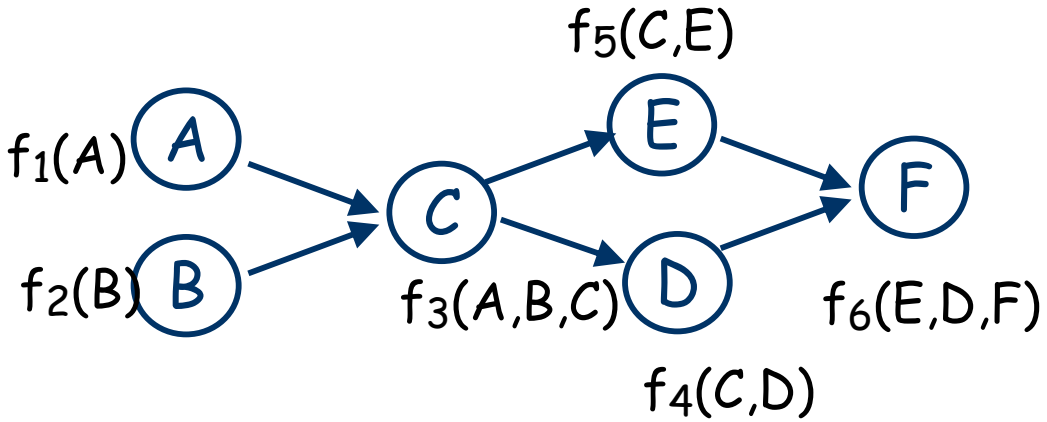
$f_1(A)$		$f_2(A,B)$		$f_3(B,C)$		$f_4(B)$ $\Sigma_A f_2(A,B)f_1(A)$		$f_5(C)$ $\Sigma_B f_3(B,C) f_4(B)$	
a	**0.9**	a**b**	**0.9**	bc	0.7	**b**	**0.85**	c	0.625
~a	**0.1**	a**~b**	**0.1**	b~c	0.3	**~b**	**0.15**	~c	0.375
		~a**b**	**0.4**	~bc	0.2				
		~a**~b**	**0.6**	~b~c	0.8				

VE as Buckets Elimintation

The bucket elimination framework is a unifying algorithmic framework that generalizes dynamic programming to accommodate algorithms for many complex problem-solving and reasoning activities $\cdots$including the variable elimination algorithm for probabilistic inference

We will use buckets as a notational device to do Variable Elimination

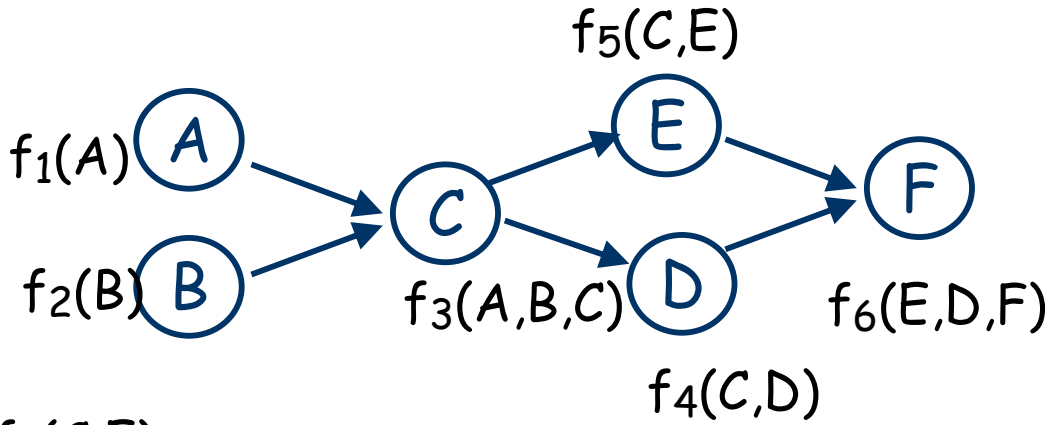
VE Ordering:
C,F,A,B,E,D



1. C:
2. F:
3. A:
4. B:
5. E:
6. D:

STEP 1: Place Original Factors in first applicable bucket.

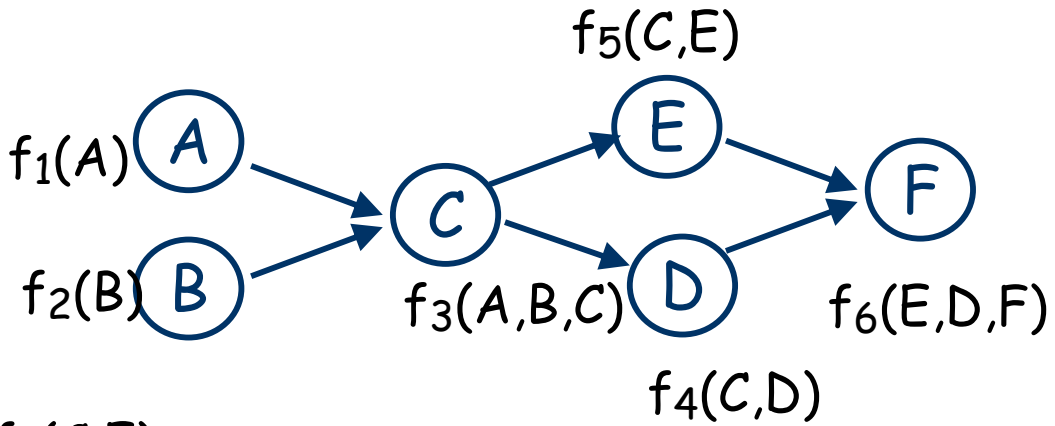
VE Ordering:
C,F,A,B,E,D



1. C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$
4. B: $f_2(B)$
5. E:
6. D:

STEP 2: Eliminate variables in order, placing new factor in 1st applicable bucket

VE Ordering:
C,F,A,B,E,D



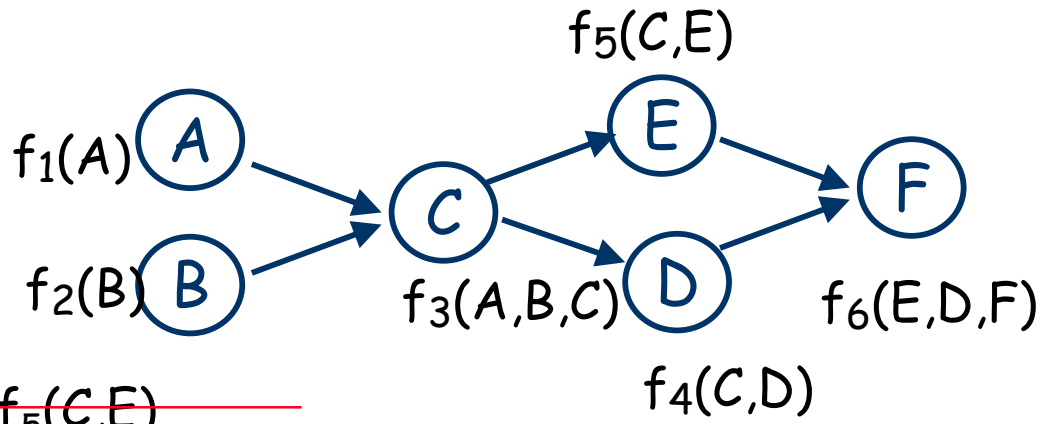
1. ~~C: $f_3(A,B,C), f_4(C,D), f_5(C,E)$~~
2. F: $f_6(E,D,F)$
3. A: $f_1(A), f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E:
6. D:

1. Eliminating C:

$$\Sigma_C f_3(A,B,C), f_4(C,D), f_5(C,E) = f_7(A,B,D,E)$$

Eliminate F, placing new factor f8 in first applicable bucket.

VE Ordering:
C,F,A,B,E,D



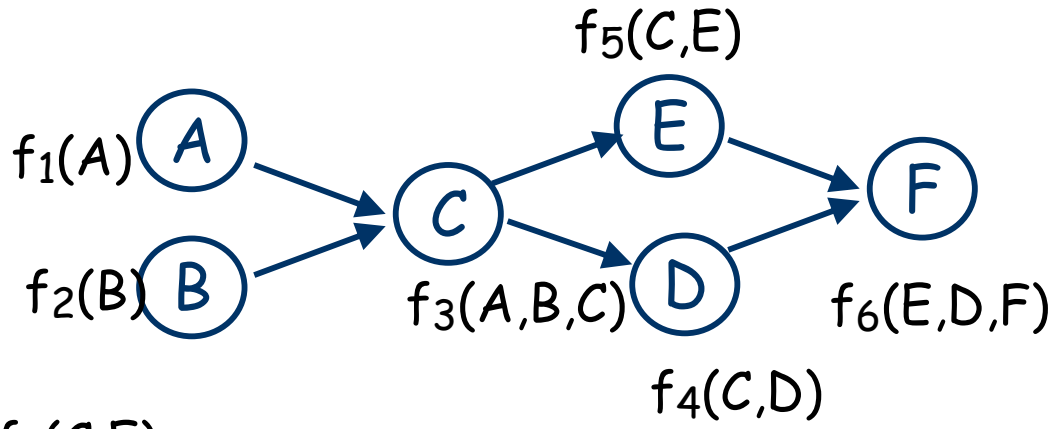
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E: $f_8(E,D)$
6. D:

2. Eliminating F:

$\Sigma_F f_6(E,D,F) = f_8(E,D)$

Eliminate A, placing new factor f9 in first applicable bucket.

VE Ordering:
C,F,A,B,E,D



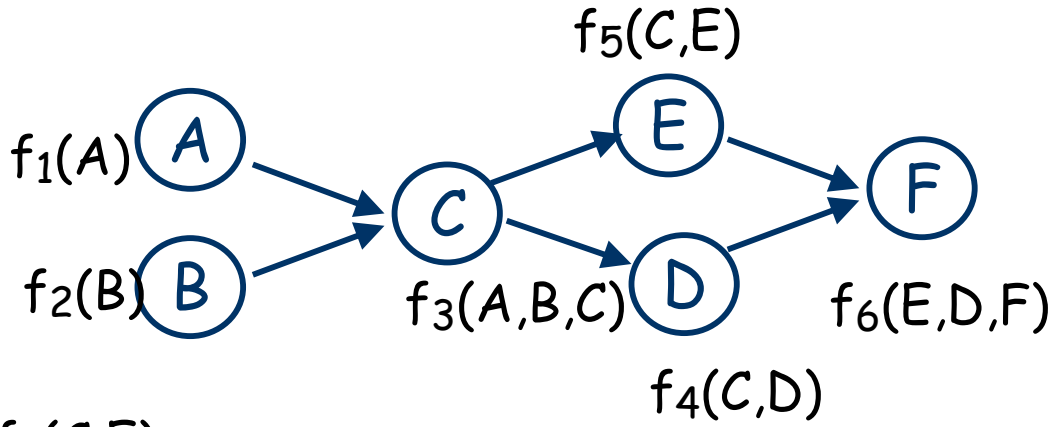
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. B: $f_2(B)$, $f_9(B,D,E)$
5. E: $f_8(E,D)$
6. D:

3. Eliminating A:

$\Sigma_A f_1(A), f_7(A,B,D,E) = f_9(B,D,E)$

Eliminate B, placing new factor f10 in first applicable bucket.

VE Ordering:
C,F,A,B,E,D



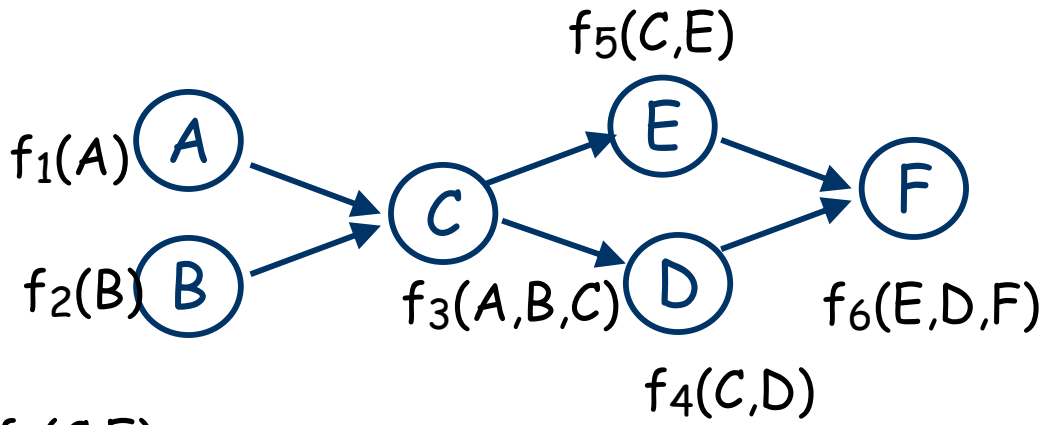
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. E: $f_8(E,D)$, $f_{10}(D,E)$
6. D:

4. Eliminating B:

$\Sigma_B f_2(B), f_9(B,D,E)$
$= f_{10}(D,E)$

Eliminate E, placing new factor f11 in first applicable bucket.

VE Ordering:
C,F,A,B,E,D



1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. ~~E: $f_8(E,D)$, $f_{10}(D,E)$~~
6. D: $f_{11}(D)$

5. Eliminating E:

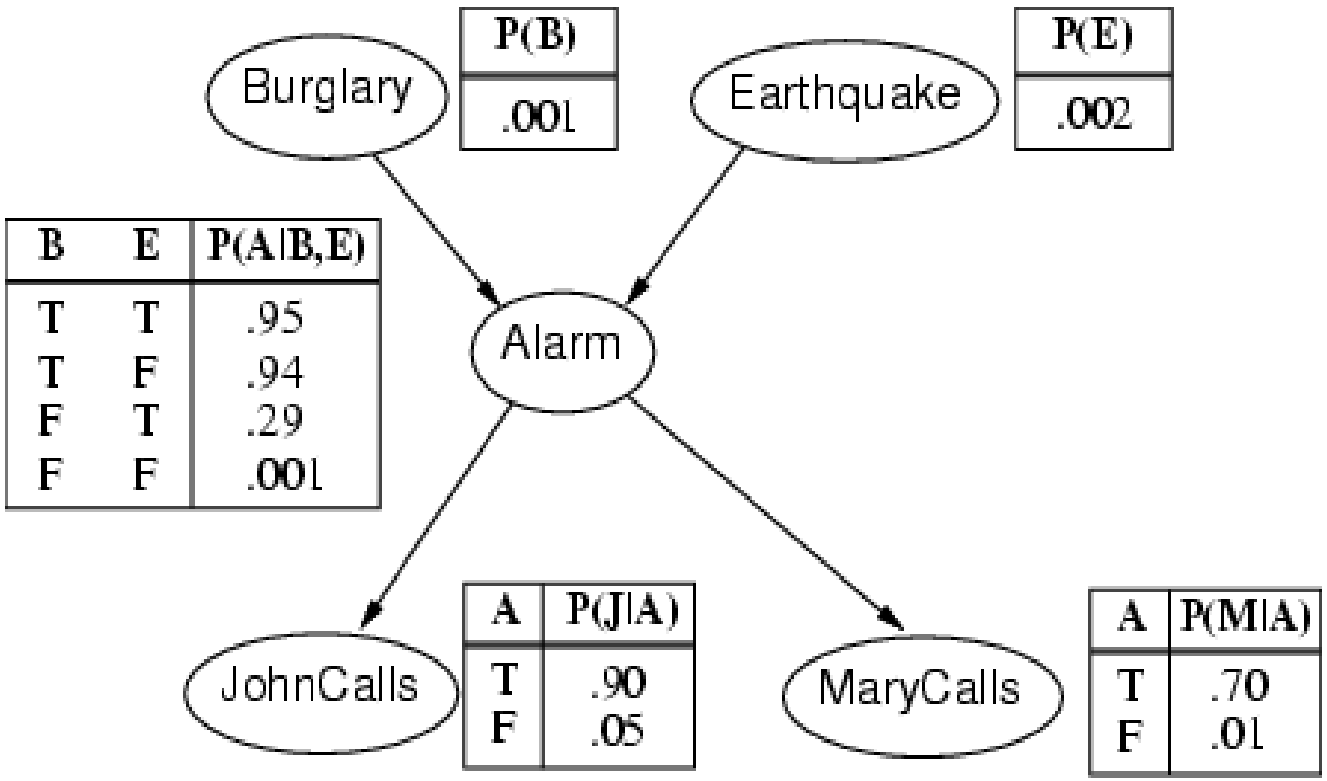
$\Sigma_E f_8(E,D), f_{10}(D,E)$
$= f_{11}(D)$

f_{11} is the final answer, once we normalize it.

Exercise

Compute $P(M|J)$ use ordering E, B, A, M

- First run of VE: evidence is J = j
- Second run of VE: evidence is J = -j
- Some factors can be reused between the two runs

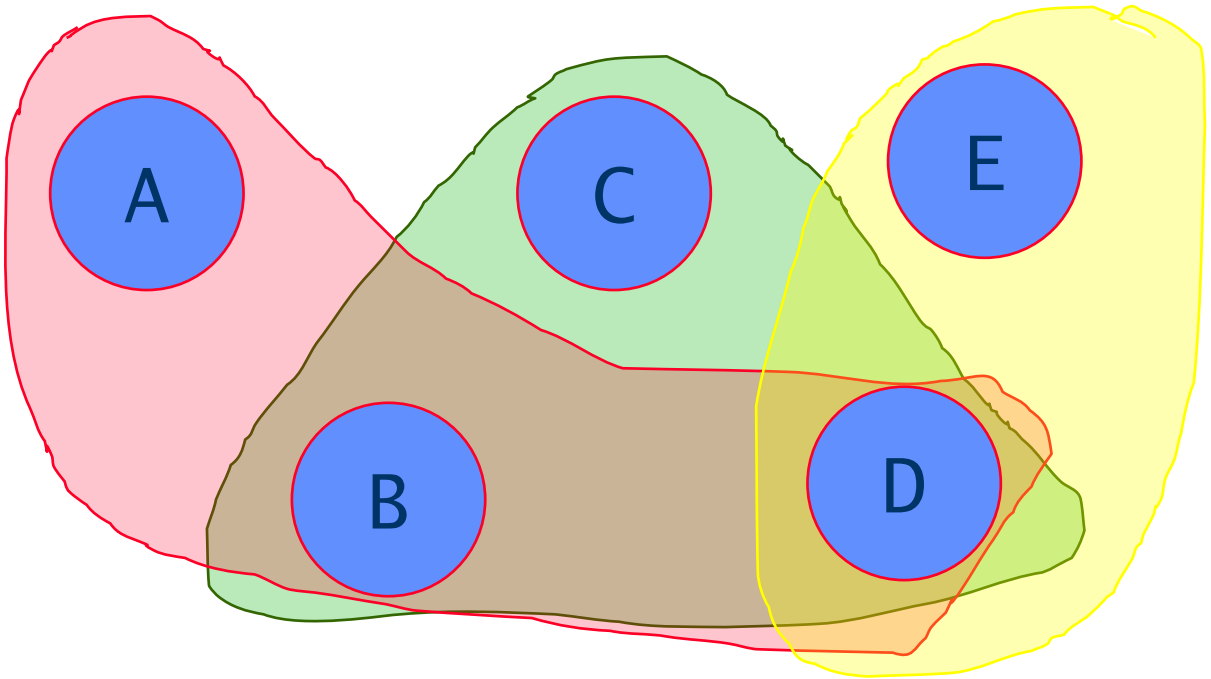


Complexity of VE

- Variable elimination (VE) starts with the set of CPTs (tables) from the original Bayes Net (BN).
- As it eliminates variables it produces new intermediate tables (factors).
- The complexity of VE is determined by the size of the largest such intermediate table.
- We would like to find a variable elimination ordering that minimizes this.
- A table can be viewed as a hyperedge in a hypergraph.

Hypergraphs

- Hypergraph has vertices just like an ordinary graph, but instead of edges between two vertices, it contains hyperedges.
- A hyperedge is a set of vertices (potentially more than one)

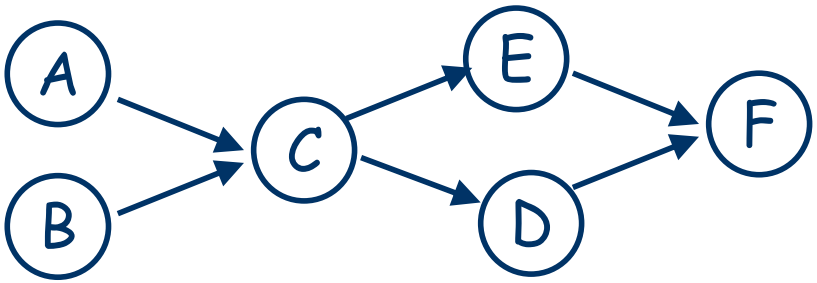


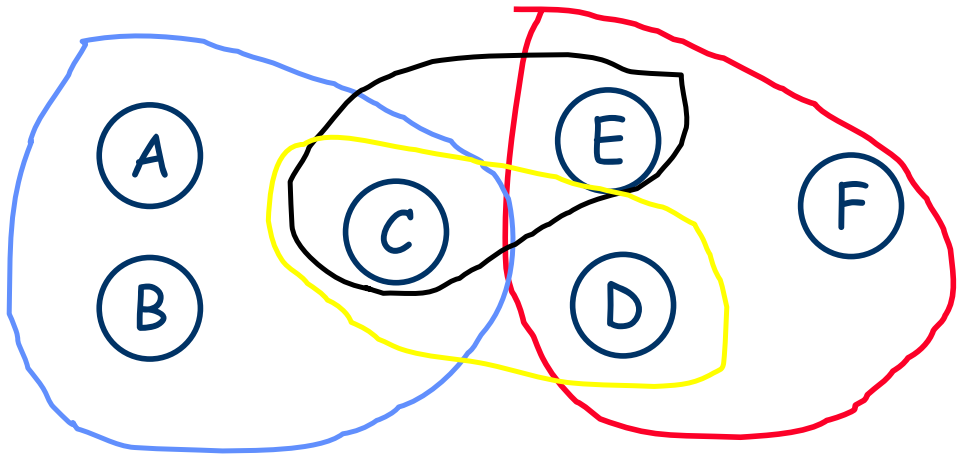
{A,B,D}
{B,C,D}
{E,D}

Hypergraph of Bayes Net

- The set of vertices are precisely the nodes of the Bayes net.
- The hyperedges are the variables appearing in each CPT, *i.e.*, $\{X_i\} \cup Par(X_i)$

Pr(A,B,C,D,E,F) =
Pr(A)Pr(B)
X Pr(C|A,B)
X Pr(E|C)
X Pr(D|C)
X Pr(F|E,D).

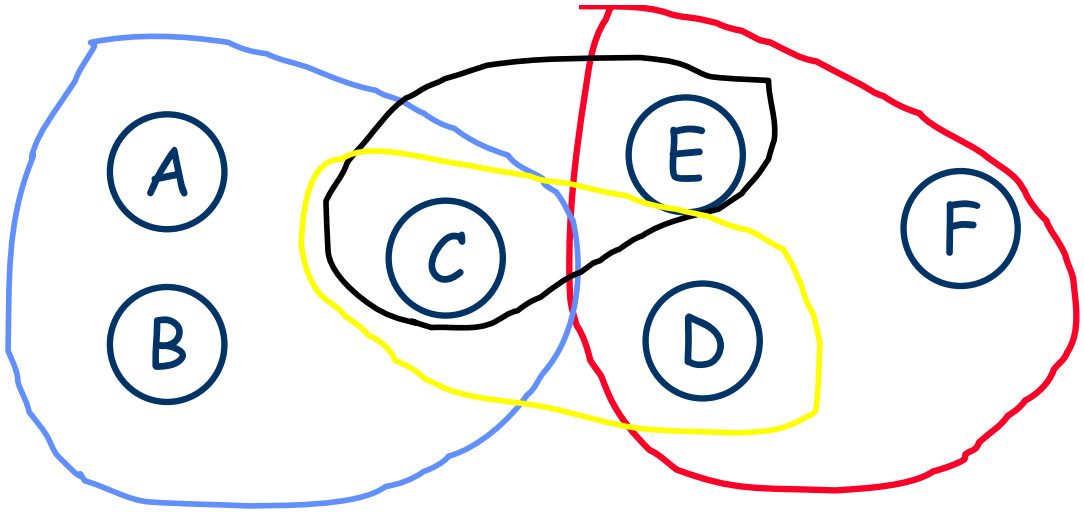




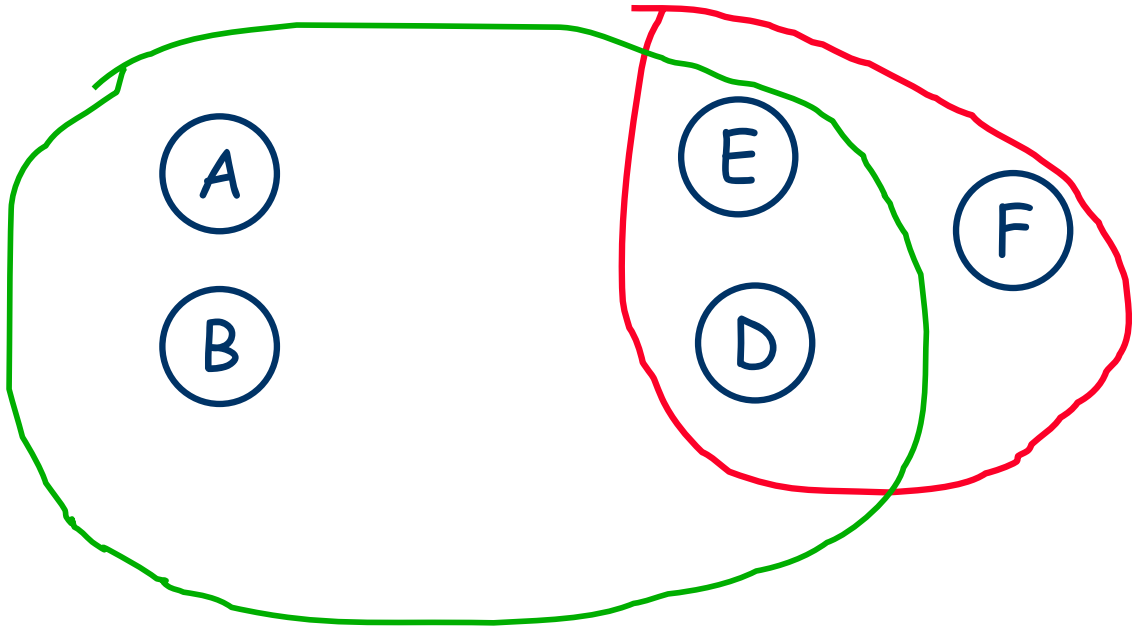
Variable Elimination in the HyperGraph

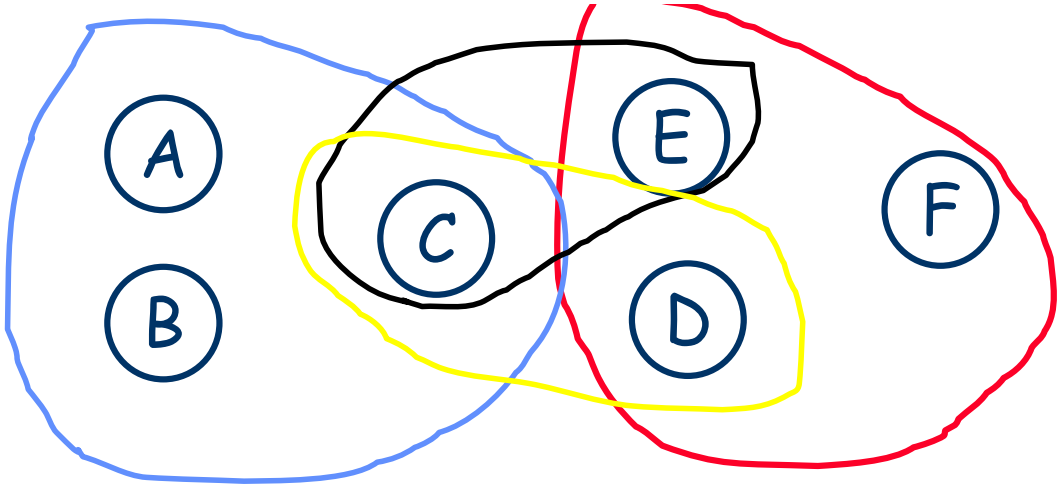
To eliminate variable X_i in the hypergraph we

- Remove the vertex X_i
- Create a new hyperedge H_i equal to the union of all of the hyperedges that contain X_i minus X_i
- Remove all hyperedges containing X_i from the hypergraph.
- Add the new hyperedge H_i to the hypergraph.

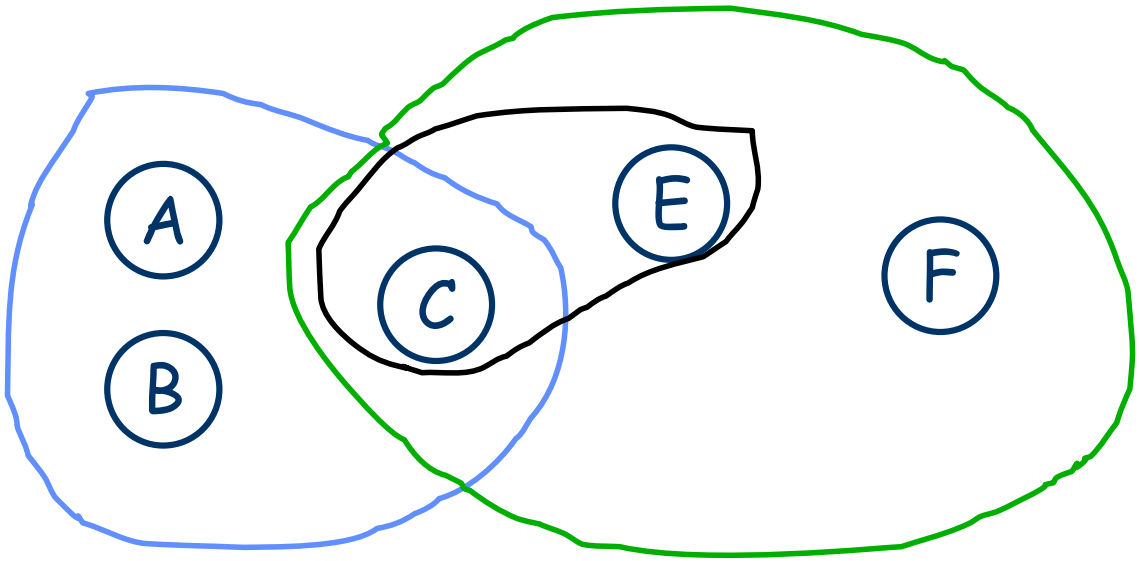


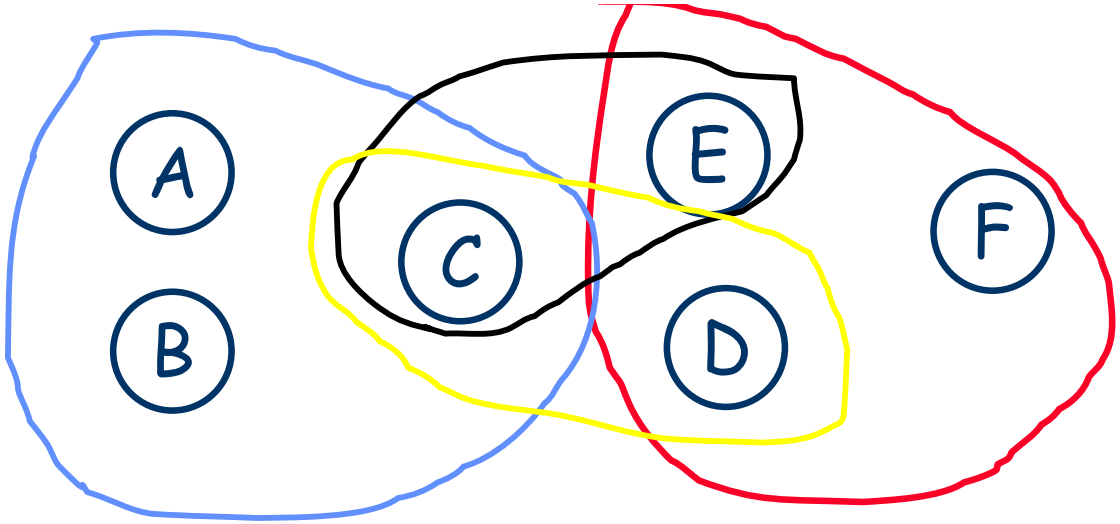
From above
Eliminate C



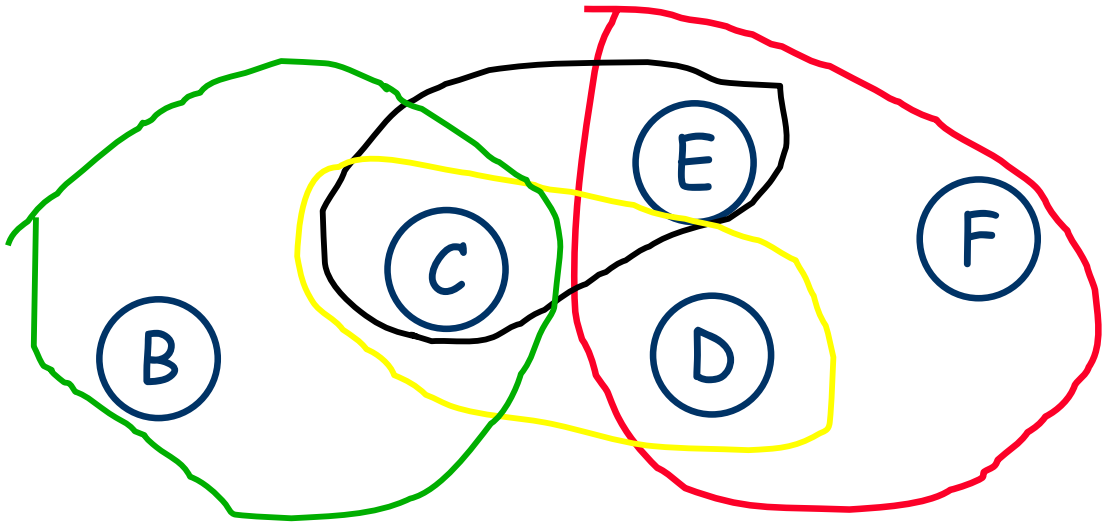


From above
Eliminate D



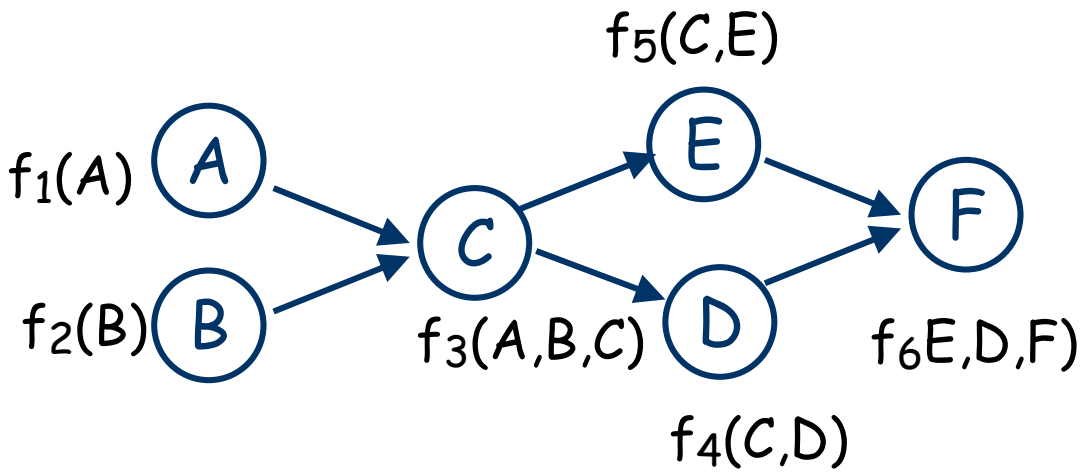


From above
Eliminate A

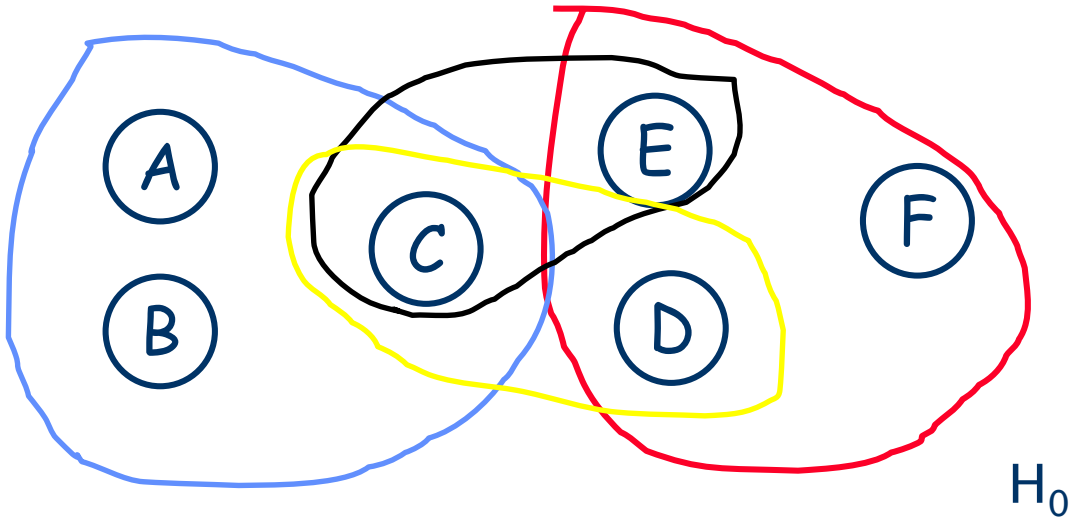


Variable Elimination – looking at the hypergraphs

VE Ordering:
C,F,A,B,E,D

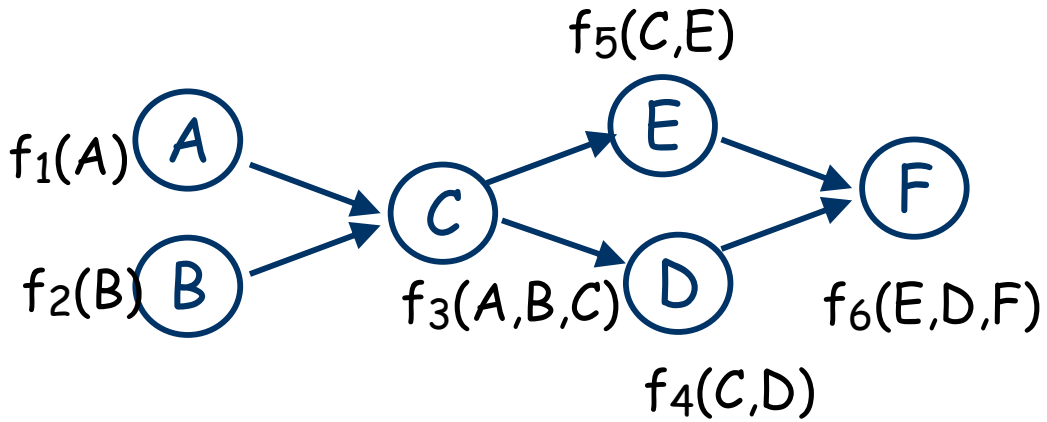


1. C:
2. F:
3. A:
4. B:
5. E:
6. D:



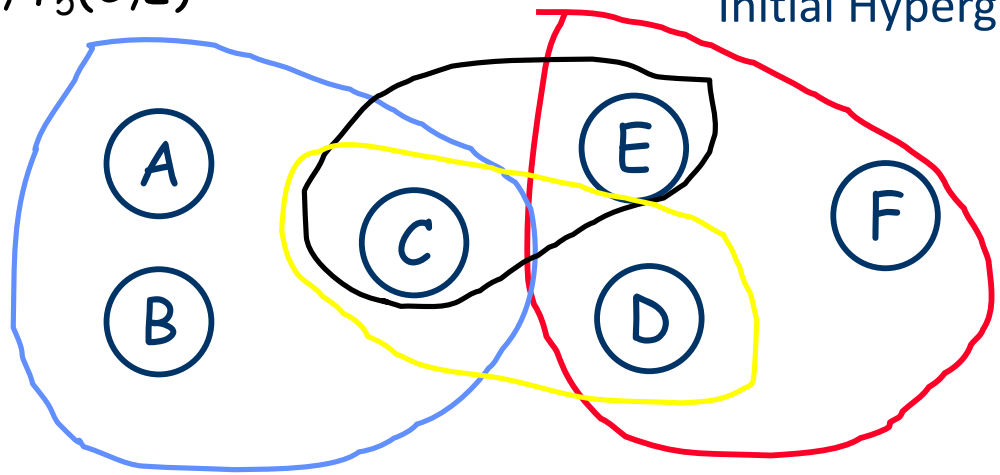
Place Original Factors in first applicable bucket.

VE Ordering:
C,F,A,B,E,D



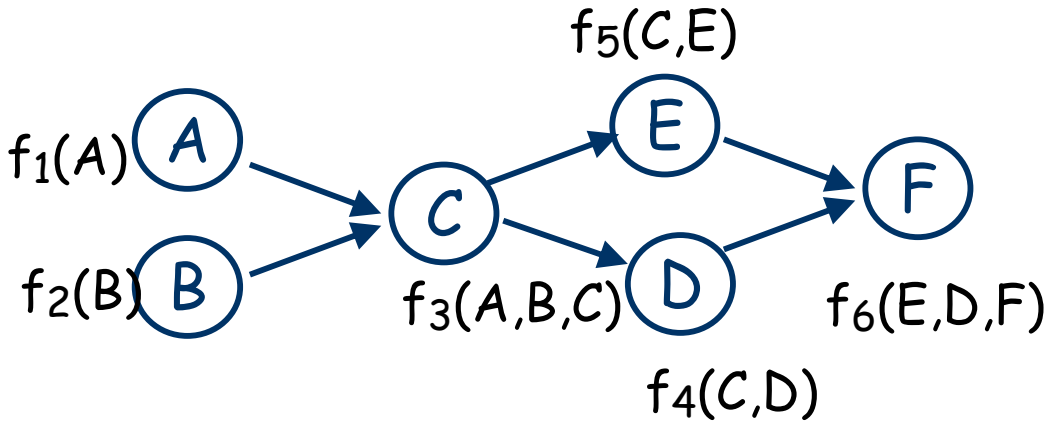
1. C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$
4. B: $f_2(B)$
5. E:
6. D:

Initial Hypergraph H_0

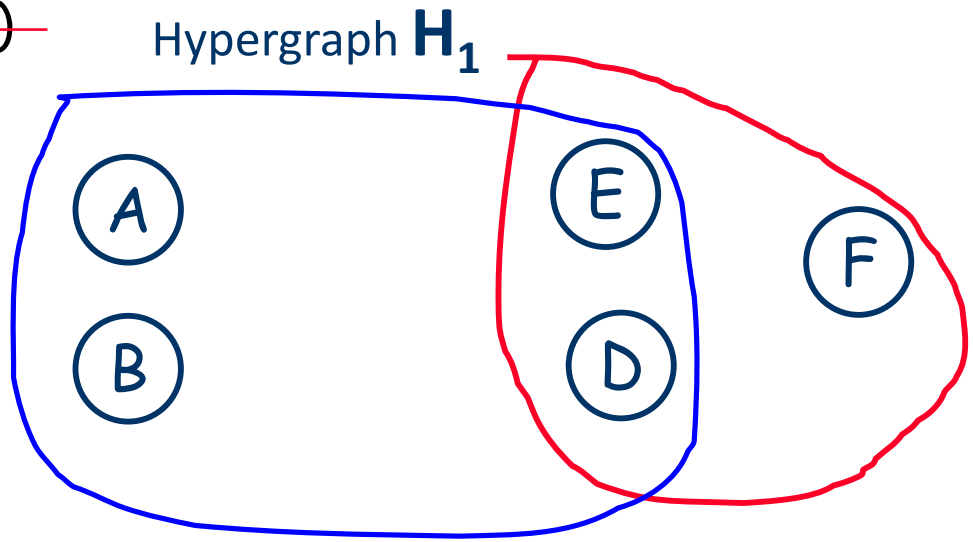


Eliminate C, placing new factor f7 in first applicable bucket.

VE Ordering:
C,F,A,B,E,D

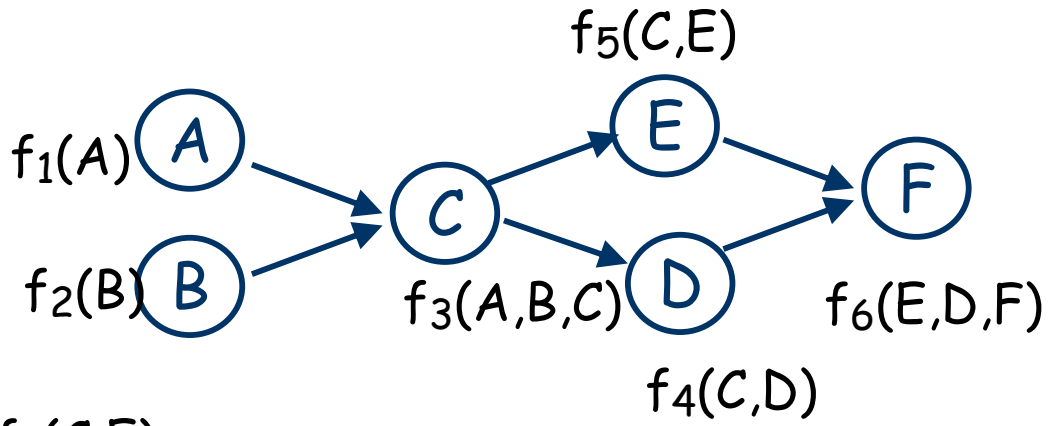


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E:
6. D:



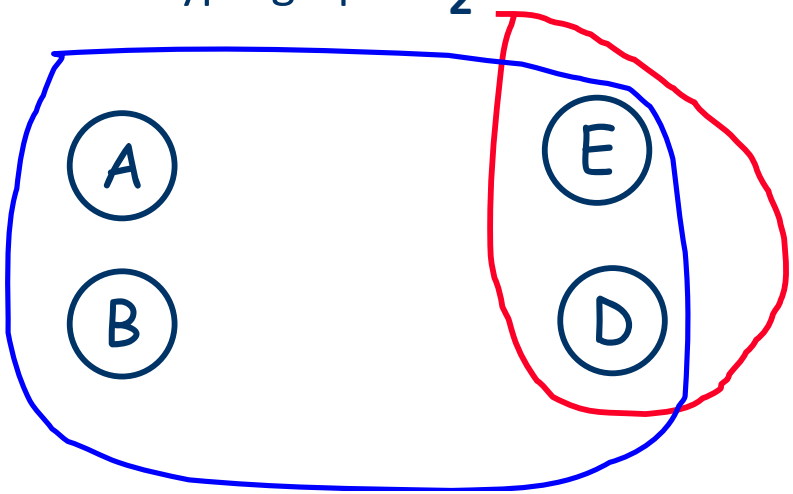
Eliminate F, placing new factor f8 in first applicable bucket.

VE Ordering:
C,F,A,B,E,D



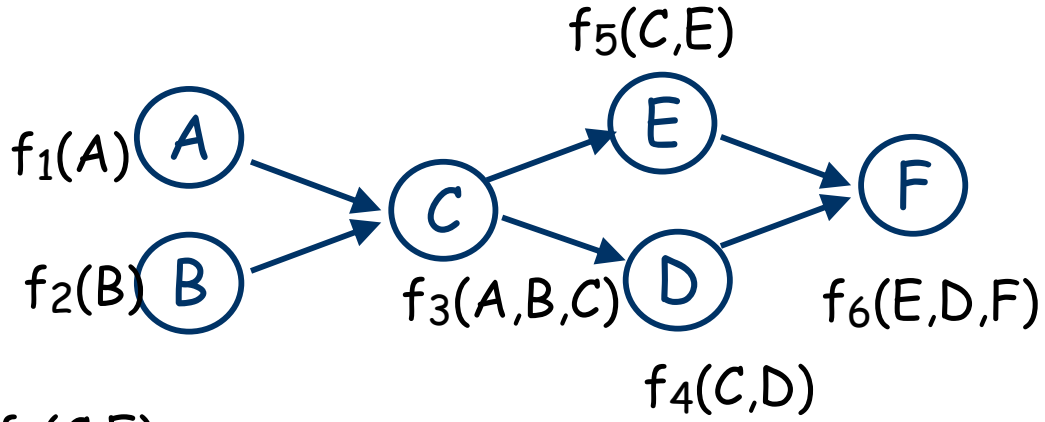
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E: $f_8(E,D)$
6. D:

Hypergraph $\mathbf{H_2}$

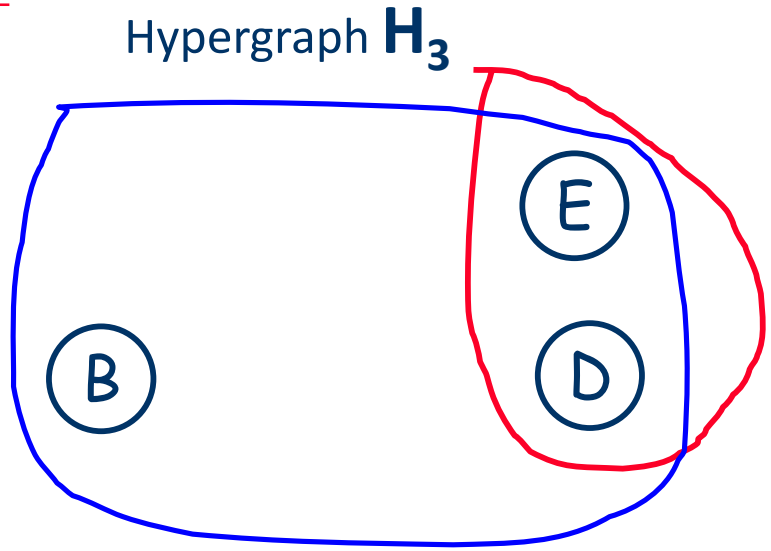


Eliminate A, placing new factor f9 in first applicable bucket.

VE Ordering:
C,F,A,B,E,D

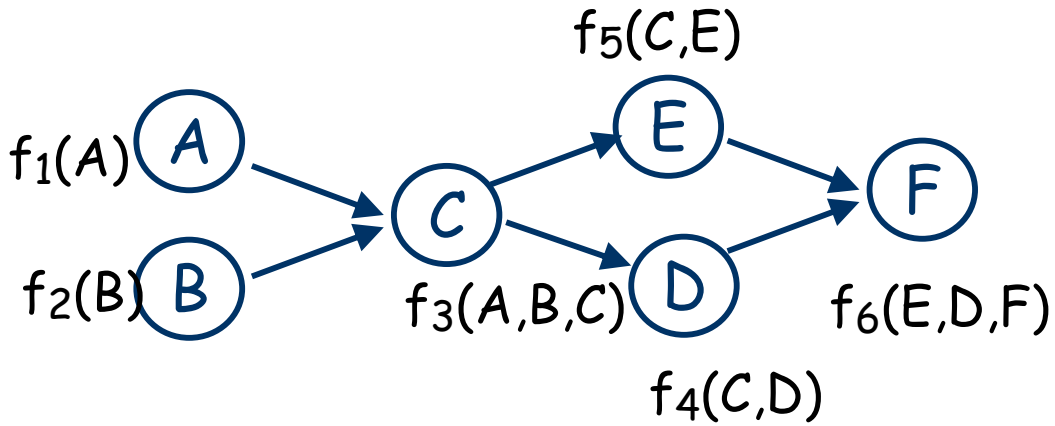


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. B: $f_2(B)$, $f_9(B,D,E)$
5. E: $f_8(E,D)$
6. D:



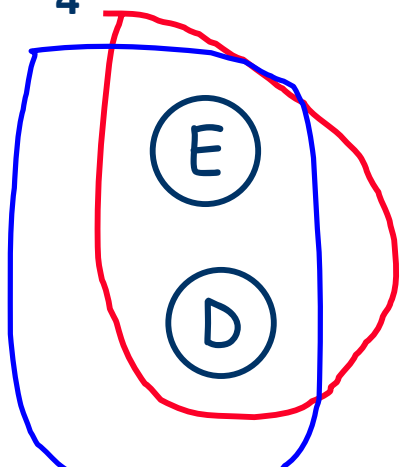
Eliminate B, placing new factor f10 in first applicable bucket.

VE Ordering:
C,F,A,B,E,D



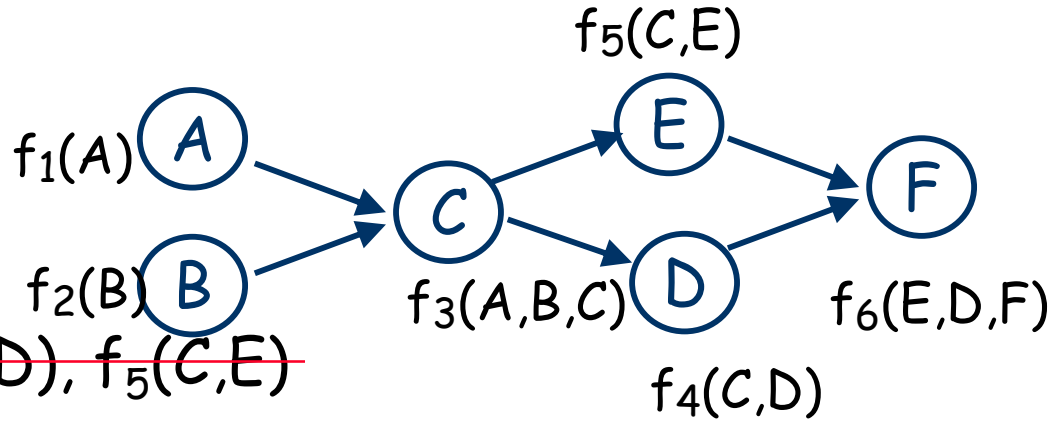
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. E: $f_8(E,D)$, $f_{10}(D,E)$
6. D:

Hypergraph $\mathbf{H_4}$



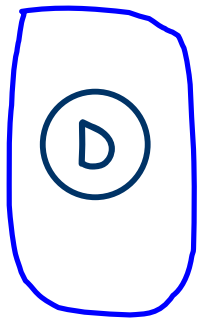
Eliminate E, placing new factor f11 in first applicable bucket.

VE Ordering:
C,F,A,B,E,D



1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. ~~E: $f_8(E,D)$, $f_{10}(D,E)$~~
6. D: $f_{11}(D)$

Hypergraph $\mathbf{H_5}$



Elimination width

- Given an ordering of the variables and an initial hypergraph H. Eliminating these variables yields a sequence of hypergraphs $H_0, H_1, H_2, \ldots, H_n$,
  where H_n contains only one vertex (the query variable).
- The elimination width is the maximum size (number of variables) of any hyperedge in any of the hypergraphs $H_0, H_1, H_2, \ldots, H_n$.
- The elimination width of the previous example was 4 ($\{A, B, E, D\}$ in H1 and H2).

Tree width

- Given a hypergraph H with vertices $\{X_1, X_2, \ldots, X_n\}$, the tree width of H is $k-1$, where k is the minimum elimination width of any of the $n!$ different orderings of the variables
- Thus VE has best case complexity of $2^{O(w)}$, where w is the tree width of the initial Bayes Net.
- In the worst case, the tree width can be equal to the number of variables.

Tree width

Exponential in the tree width is the best that VE can do.

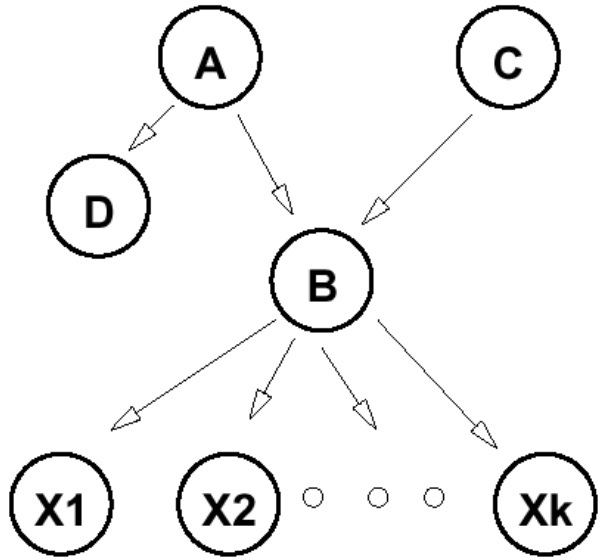
- Finding an ordering with elimination width equal to tree width is NP-Hard.
- So in practice there is no point in trying to speed up VE by finding the best possible elimination ordering.
- Heuristics are used to find orderings with good (low) elimination widths.
- In practice, this can be very successful. Elimination widths can often be relatively small, 8-10 even when the network has 1000s of variables.
- Thus VE can be much!! more efficient than simply summing the probability of all possible events (which is exponential in the number of variables).

Finding Good Orderings (sometimes)

- A polytree is a singly connected Bayes Net: in particular there is only one path between any two nodes.
- A node can have multiple parents, but there are no cycles.
- Good orderings are easy to find for polytrees
  - At each stage eliminate a singly connected node.
  - Because we have a polytree we are assured that a singly connected node will exist at each elimination stage.
  - The size of the factors in the tree never increase.

Elimination Ordering: Polytrees

Treewidth of a polytree = maximum number of parents among all nodes.



Eliminating singly connected nodes

allows VE to run in **time linear in size of network**

E.g., in this network,

- eliminate D, A, C, X1,…; or
- eliminate X1,… Xk, D, A, C; or
- mix up…

Result: no factor ever larger than original CPTs

BUT E.g.,

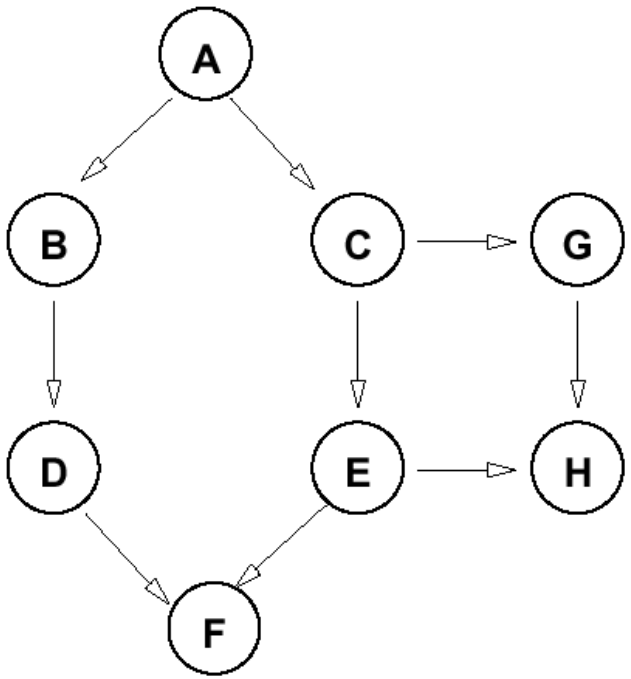
- eliminating B before these

Result: factors that include all of A,C, X1,… Xk !!!

Effect of Different Orderings

Given the following BN with **query variable** is **D**.

This BN is **not a polytree!**



Consider different **variable elimination orderings**

A,F,H,G,B,C,E:

- Good ordering

E,C,A,B,G,H,F:

- Bad ordering

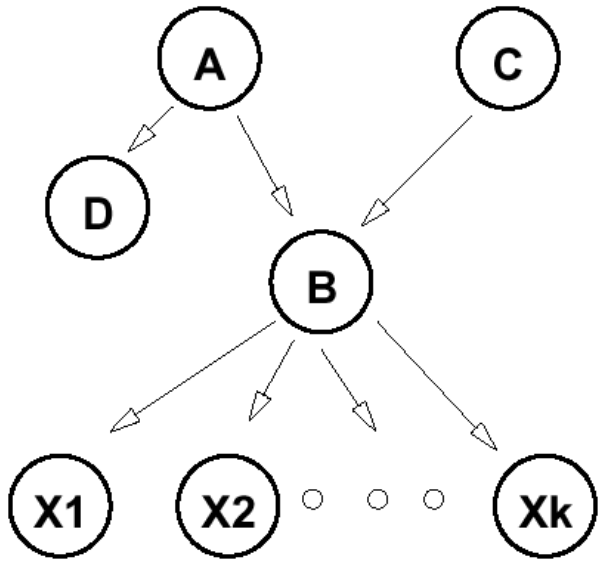
Elimination Order: Min Fill Heuristic

Min-fill Heuristic:

"always eliminate next the variable that creates the smallest size factor."

This is a reasonably effective heuristic for determining an elimination order for VE

- B creates a factor of size k+2
- A creates a factor of size 2
- D creates a factor of size 1



The heuristic **always solves polytrees in linear time**.

Relevant

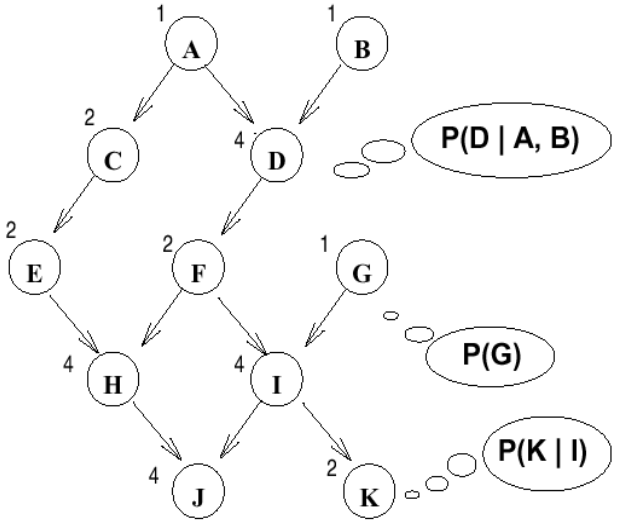
A brief aside .... note that in the sum

$$\sum_A \Pr(A) \sum_B \Pr(B) \Pr(d|A,B) \sum_C \Pr(C|A) \sum_E \Pr(E|C)$$
$$\sum_F \Pr(F|d) \Pr(h|E,F) \sum_G \Pr(G) \Pr(-i|F,G)$$
$$\sum_J \Pr(J|h,-i)$$
$$\sum_K \Pr(K|-i)$$

we have that $\sum_K \Pr(K|-i) = 1$ (Why?), thus

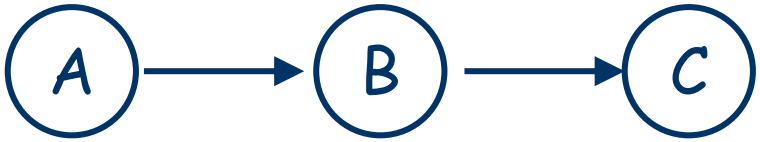
$$\sum_J \Pr(J|h,-i) \sum_K \Pr(K|-i) = \sum_J \Pr(J|h,-i)$$

Furthermore $\sum_J \Pr(J|h,-i) = 1$.



So we could drop these last two terms from the computation:
J and K are irrelevant given our query D and evidence -i and -h.

Relevance



Certain variables have no impact on the query.

In network ABC, computing Pr(A) with no evidence requires elimination of B and C.

- But when you sum out these vars, you compute a trivial factor (whose value are all ones); for example:
- eliminating C: $f_4(B) = \Sigma_C f_3(B,C) = \Sigma_C \Pr(C|B)$
- 1 for any value of B (e.g., $\Pr(c|b) + \Pr(\sim c|b) = 1$)

Observation: B or C *are irrelevant* to the query. No need to think about them.

Relevance

Observation: We can restrict attention to ***relevant*** variables.

When is a variable relevant to a query?

Given

- query **Q**,
- evidence **E**

The following variables are **relevant** to the evaluation of Q given E:

- Q itself is relevant
- if any node **Z** is relevant, its parents are relevant
- if e$\in$**E** is a descendent of a relevant node, then E is relevant

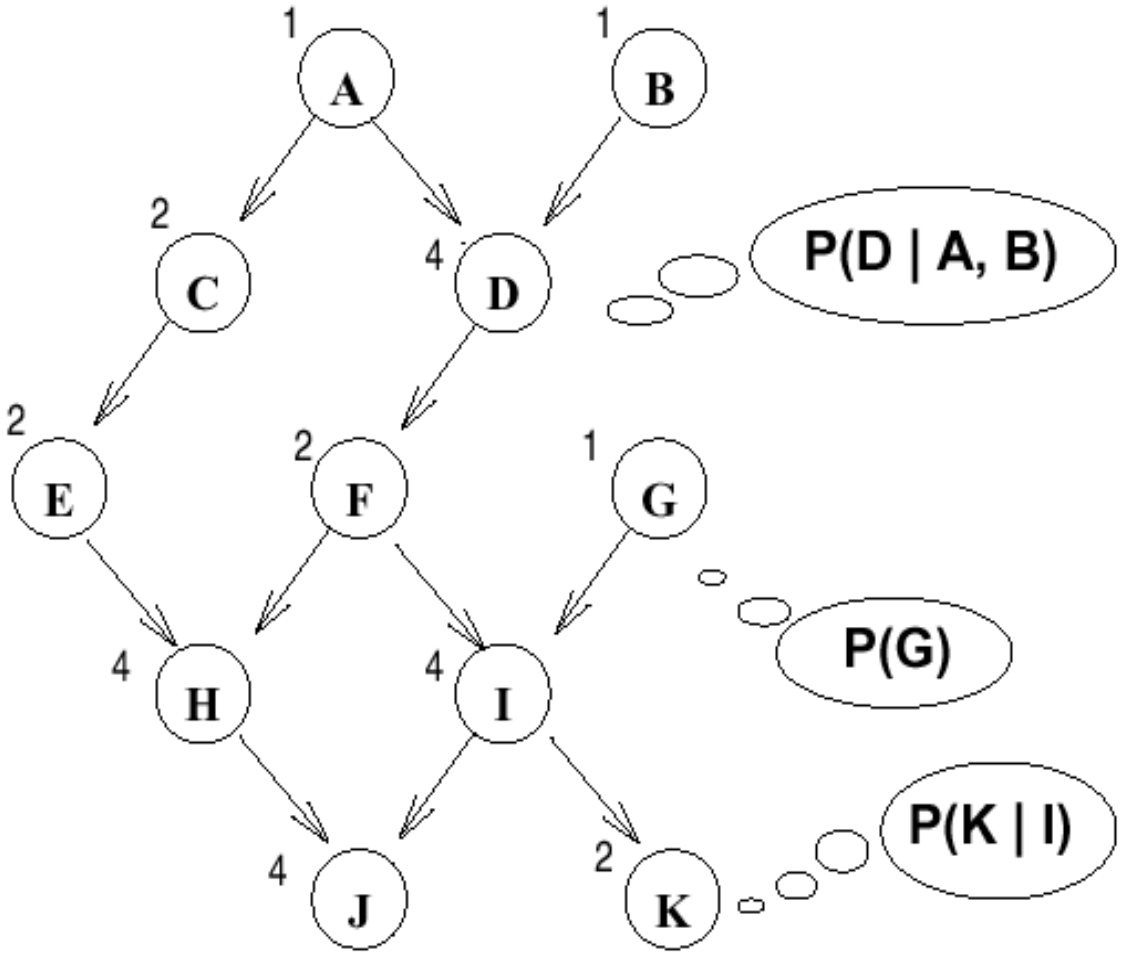
---

When evaluating query Q, we can restrict our attention to the *subnetwork comprising only relevant variables*

An example

Query: $P(D|h, -i)$

What variables are relevant?



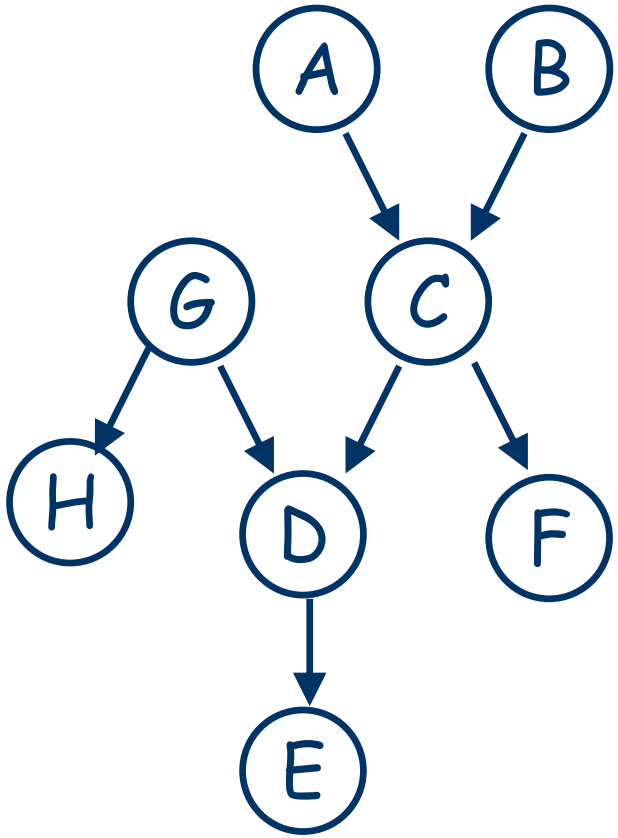
Relevance: Examples

Query: P(F)

- relevant: F, C, B, A

Query: P(F|E)

- relevant: F, C, B, A
- **also: E, hence D, G**
- intuitively, we need to compute P(C|E) to compute P(F|E)



Relevance: Examples

Query: P(F|H)

- relevant F,C,A,B.

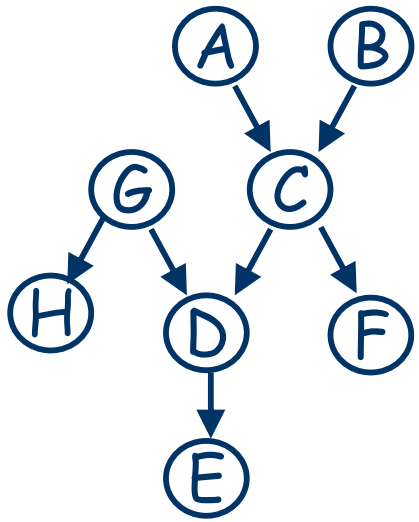
Pr(A,B,C,D,E,F,G,H)

= Pr(A)Pr(B)Pr(C|A,B)Pr(F|C) Pr(G)Pr(h|G) Pr(D|G,C) Pr(E|D)

= ... Pr(G)Pr(h|G)Pr(D|G,C) $\sum_E$Pr(E|D) = a table of 1's

= ... Pr(G)Pr(h|G)$\sum_D$ Pr(D|G,C) = a table of 1's

= [Pr(A)Pr(B)Pr(C|A,B)Pr(F|C)] [Pr(G)Pr(h|G)]



[Pr(G)Pr(h|G)] $\neq 1$

but irrelevant once we normalize, multiplies each value of F equally

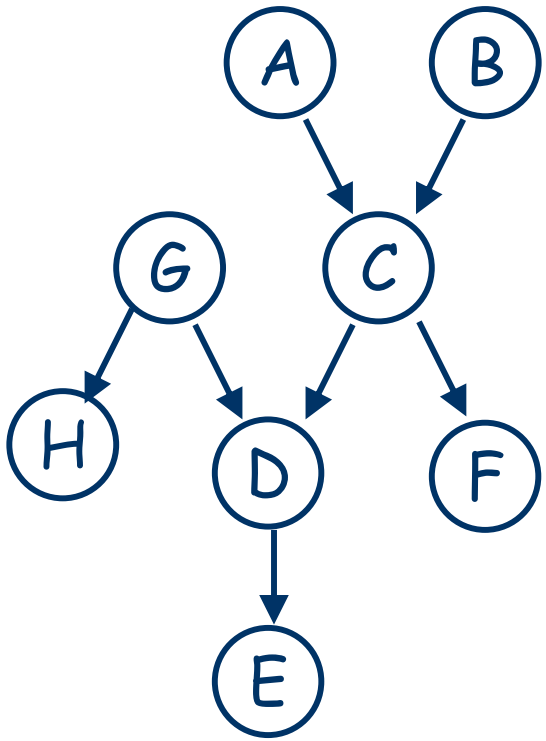
So D, E, G are actually irrelevant

Relevance: Examples

Query: $P(F|E, C)$

- algorithm says all variables are relevant, except H;
- but really none except C, F (since C cuts of all influence of others)

The algorithm is overestimating relevant set



Exercise

- Which of the following are asserted by the network structure?
  - $P(B, I, M) = P(B)P(I)P(M)$.
  - $P(J|G) = P(J|G, I)$.
  - $P(M|G, B, I) = P(M|G, B, I, J)$.

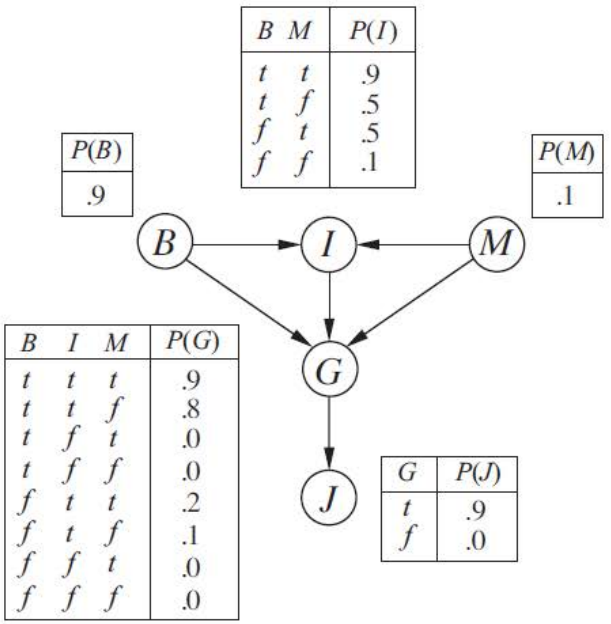


Figure 14.23 A simple Bayes net with Boolean variables $B = BrokeElectionLaw$, $I = Indicted$, $M = PoliticallyMotivatedProsecutor$, $G = FoundGuilty$, $J = Jailed$.

Exercise

- Calculate the value of $P(b, i, \neg m, g, j)$.
- Calculate the probability that someone goes to jail given that they broke the law, have been indicted, and face a politically motivated prosecutor.

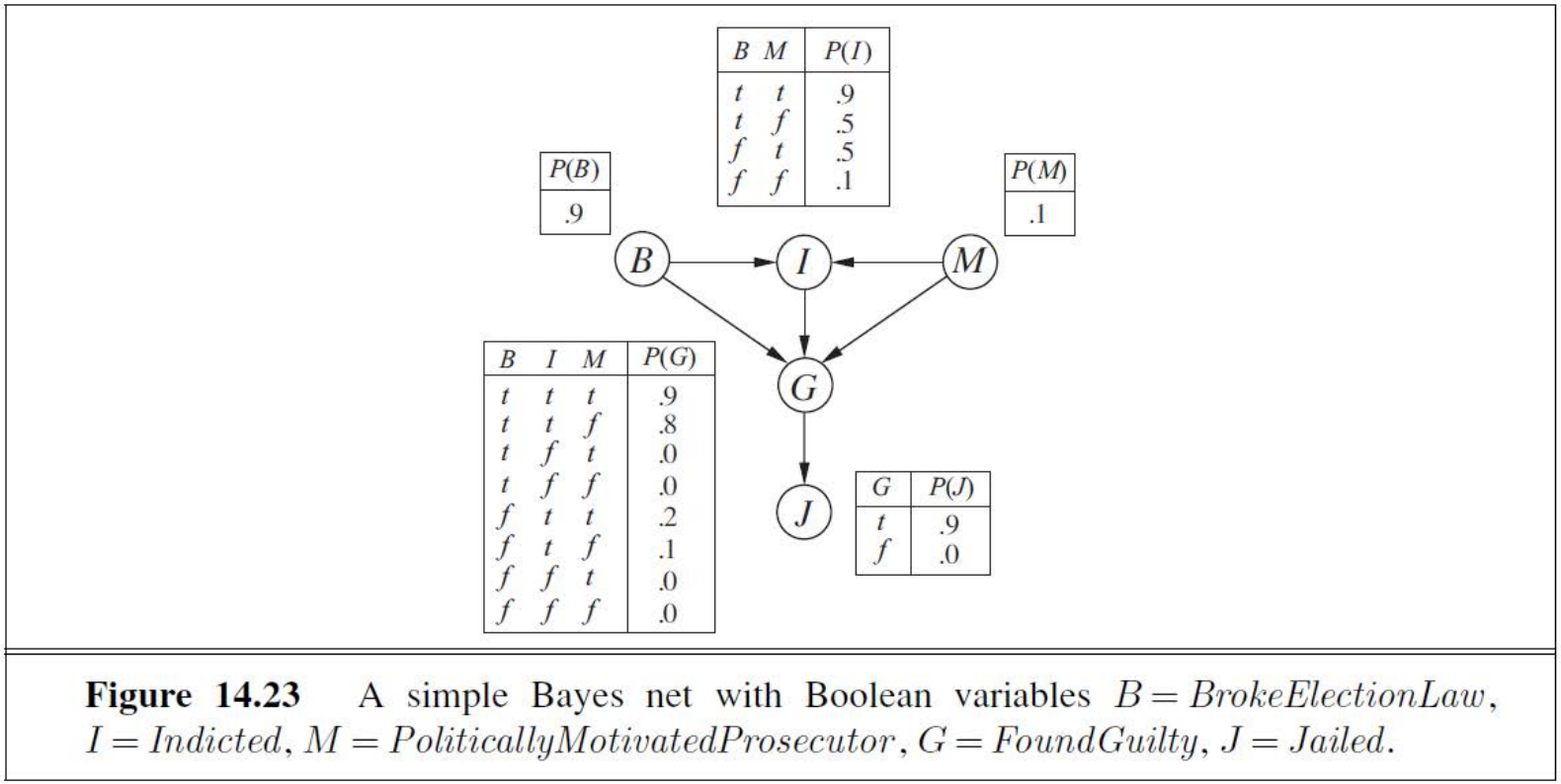


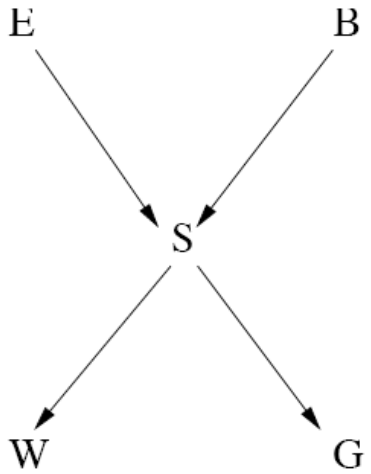
Figure 14.23 A simple Bayes net with Boolean variables $B = BrokeElectionLaw$, $I = Indicted$, $M = PoliticallyMotivatedProsecutor$, $G = FoundGuilty$, $J = Jailed$.

Exercise

E – Earthquake, B – Burglary

S – Sound of alarm heard

W – Dr. Watson Calls G – Mrs Gibbons Calls



P(E)	e	-e
	1/10	9/10

P(B)	b	-b
	1/10	9/10

P(S|E,B)	s	-s
e ∧ b	9/10	1/10
e ∧ -b	2/10	8/10
-e ∧ b	8/10	2/10
-e ∧ -b	0	1

P(W|S)	w	-w
s	8/10	2/10
-s	2/10	8/10

P(G|S)	g	-g
s	1/2	1/2
-s	0	1

- What is $P(G|W)$? (i.e., the four probability values $P(g|w)$, $P(-g|w)$, $P(g|-w)$, and $P(-g|-w)$).
- Query variable is G.
- First run of VE, evidence is $W = w$.
- Second run of VE, evidence is $W = -w$.
- Use same ordering for both runs of VE: E, B, S, G.