Inference in Bayes Nets

Given a Bayes net

$P(X_1, X_2, \ldots, X_n) = P(X_n | P(Parents(X_n))) *$

$P(X_{n-1} | P(Parents(X_{n-1}))) * \ldots * P(X_1 | P(Parents(X_1)))$

And some evidence

E = {a set of values for some of the variables}

we want to compute the new probability distribution

$P(X_k | E)$

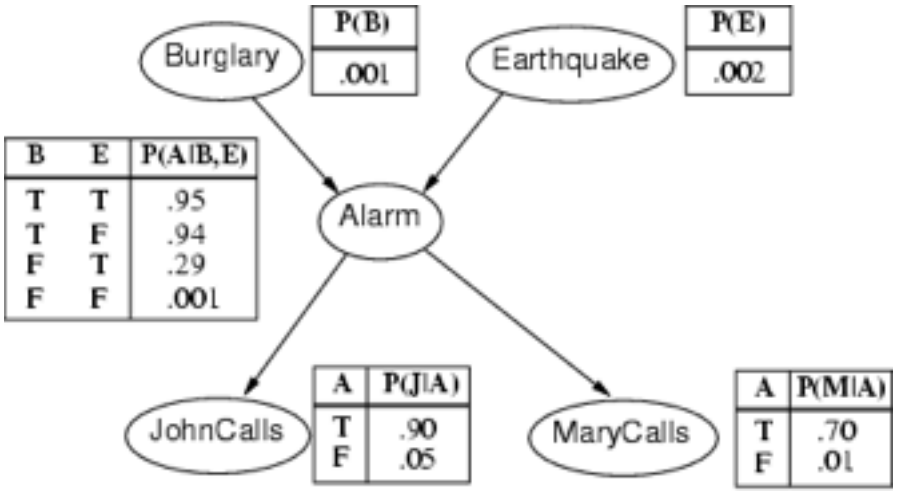
That is, we want to figure our

$P(X_k = d | E)$ for all $d \in Dom[X_k]$

Inference in Bayes Nets

- We might infer probability of different diseases given symptoms, or probability of hail storms given different metrological evidence, etc.
- In such cases getting a good estimate of the probability of the unknown event allows us to respond more effectively (gamble rationally)

Inference in Bayes Nets



In the Alarm example:

- P(Burglary,Earthquake, Alarm, JohnCalls, MaryCalls) =
  P(Earthquake) * P(Burglary) *
  P(Alarm|Earthquake,Burglary) *
  P(JohnCalls|Alarm) * P(MaryCalls|Alarm)
- We may want to infer things like
  P(Burglary=true| MaryCalls=false, JohnCalls=true)

Variable Elimination

If the network has a lot of variables, making exact inferences can be computationally hairy.

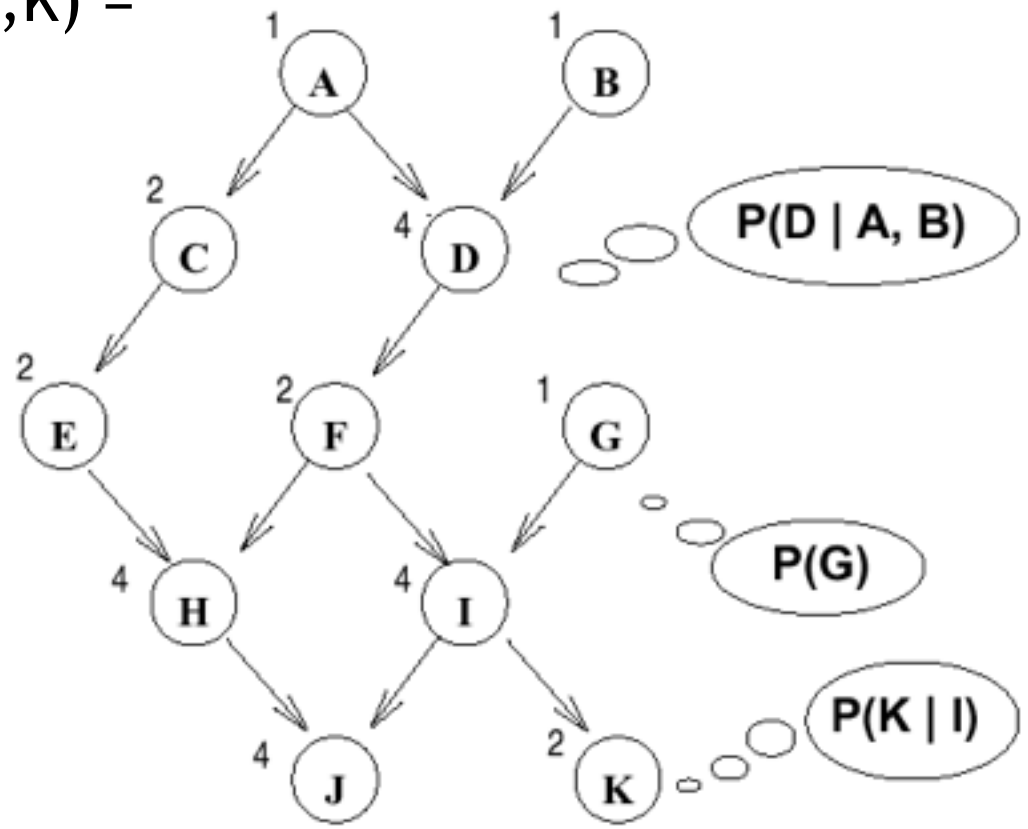
Variable elimination uses the product decomposition that defines a Bayes Net and the summing out rule to compute probabilities conditioned on evidence from information in the network (CPTs).

VE helps to reduce some of computation required to make exact inferences.

Example (Binary valued Variables)

P(A,B,C,D,E,F,G,H,I,J,K) =

P(A)
x P(B)
x P(C|A)
x P(D|A,B)
x P(E|C)
x P(F|D)
x P(G)
x P(H|E,F)
x P(I|F,G)
x P(J|H,I)
x P(K|I)



Example

P(A,B,C,D,E,F,G,H,I,J,K) =
P(A)P(B)P(C|A)P(D|A,B)P(E|C)P(F|D)P(G)P(H|E,F)
P(I|F,G)P(J|H,I)P(K|I)

Say that E (our evidence) = {H=true, I=false}, and we want to know

P(D|h,-i) **(h: H is true, -h: H is false)**

= P(d,h,-i)/N and P(-d,h,-i)/N where

N = P(h,-i) = P(d,h,-i) + P(-d,h,-i)

N can be considered a "normalizing" constant.

Example

First, we write as a sum for each value of D (i.e. D = d and D = -d)

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

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

We start by computing P(d,h,-i). Use Bayes Net product decomposition to rewrite the summation:

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

Example

Next, rearrange summations so that we are not summing over variables that do not depend on the summed variable.

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

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

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

Example

Now start computing from the last summation to the first.

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

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

$$\sum_A P(A) \sum_B P(B) P(d|A,B) \sum_C P(C|A) \sum_E P(E|C) \sum_F P(F|d) P(h|E,F) \sum_G P(G) P(-i|F,G) \sum_J P(J|h,-i)\, c_1$$

Example

$$\sum_A P(A) \sum_B P(B) P(d|A,B) \sum_C P(C|A) \sum_E P(E|C)$$
$$\sum_F P(F|d) P(h|E,F) \sum_G P(G) P(\text{-}i|F,G)$$
$$\sum_J P(J|h,\text{-}i)\ c_1$$

In this expression, variable K has been eliminated.
We can move c_1 to the front of the equation as it does not depend on other variables.

$$c_1 \sum_A P(A) \sum_B P(B) P(d|A,B) \sum_C P(C|A) \sum_E P(E|C)$$
$$\sum_F P(F|d) P(h|E,F) \sum_G P(G) P(\text{-}i|F,G)$$
$$\sum_J P(J|h,\text{-}i)$$

Example

$$c_1 \sum_A P(A) \sum_B P(B) P(d|A,B) \sum_C P(C|A) \sum_E P(E|C) \sum_F P(F|d) P(h|E,F) \sum_G P(G) P(-i|F,G) \sum_J P(J|h,-i)$$

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

Now we've eliminated variable J.

$$c_1 c_2 \sum_A P(A) \sum_B P(B) P(d|A,B) \sum_C P(C|A) \sum_E P(E|C) \sum_F P(F|d) P(h|E,F) \sum_G P(G) P(-i|F,G)$$

Example

$c_1\, c_2 \sum_A P(A) \sum_B P(B)\, P(d|A,B) \sum_C P(C|A) \sum_E P(E|C) \sum_F P(F|d)\, P(h|E,F) \sum_G P(G)\, P(-i|F,G)$

$\sum_G P(G)\, P(-i|F,G)$
$= (P(g)P(-i|F,g) + P(-g)P(-i|F,-g))$

P(-i|F,g) depends on the value of F, so this is not a single number.

Instead, its value depends on the assignment to F. But once F is fixed as f or -f, the value of P(-i|F,g) is also fixed.

Example

$c_1 c_2 \sum_A P(A) \sum_B P(B) P(d|A,B) \sum_C P(C|A) \sum_E P(E|C) \sum_F P(F|d) P(h|E,F) \sum_G P(G) P(-i|F,G)$

$\sum_G P(G) P(-i|F,G)$
$= (P(g)P(-i|F,g) + P(-g)P(-i|F,-g))$

To manage, we introduce a function to represent this sum:

$f1(F) = P(g)P(-i|F,g) + P(-g)P(-i|F,-g)$

We must store these fixed numbers to represent the function:

$f1(f) = P(g)P(-i|f,g) + P(-g)P(-i|f,-g)$
$f1(-f) = P(g)P(-i|-f,g) + P(-g)P(-i|-f,-g)$

Example

$$c_1 c_2 \sum_A P(A) \sum_B P(B) P(d|A,B) \sum_C P(C|A) \sum_E P(E|C) \sum_F P(F|d) P(h|E,F) f1(F)$$

$$\sum_F P(F|d) P(h|E,F) f1(F) = P(f|d) P(h|E,f) f1(f) + P(-f|d) P(h|E,-f) f1(-f)$$

This sum depends on E, so we introduce a function of E

$$f2(E) = P(f|d) P(h|E,f) f1(f) + P(-f|d) P(h|E,-f) f1(-f)$$

Again store two fixed values, f2(e) and f2(-e), to represent this function.

Variable Elimination (VE)

- We can continue this way, eliminating one variable after another by introducing functions. At the end of the process we will sum out A and be left with P(d,h,-i).
- We can repeat the process to compute P(-d,h,-i).
- Remember the normalizing constant (N) is P(d,h,-i) + P(-d,h,-i). If we normalize the numbers (ensure they sum to one) we will have P(d|h,-i) and P(d|h,i).
- Alternately, we could consider D to be the last variable in the elimination process, and end our process when by computing f(D). The fixed values, f(d) and f(-d), will be the values we want as they correspond to P(d|h,-i) and P(d|h,i).

Variable Elimination (VE)

- This process is called variable elimination (VE)
- VE computes intermediate functions and stores values that we reuse many times during computations.
- In this way variable elimination is a form of dynamic programming. Dynamic programming is a technique that stores solutions to sub-computations in order to avoid repeating them many times over.

Relevant (return to this later)

Note that in the sum

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

$\sum_K P(K|-i) = 1$ (Why?)

This means $\sum_J P(J|h,-i) \sum_K P(K|-i) = \sum_J P(J|h,-i)$

By the same reasoning, $\sum_J P(J|h,-i) = 1$.

So we can, in theory, drop these last two terms from the computation. The variables J and K are not relevant given our query D and evidence (–i and –h). But for now we will keep the terms; we will revisit relevance later.

Variable Elimination (VE)

- In general, each stage VE sums out the innermost variable, creating a function over any variables in that sum.
- Each function is represented as a table of numbers: one number for each different instantiation of the variables in the sum.
- The size of the tables is exponential in the number of variables that appear in the sum, e.g.,

$$\sum_F P(F|D)\ P(h|E,F)\ f1(F)$$

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 the tables of values computed by VE factors.
- Note that the original probabilities that appear in the summation, e.g., P(C|A), are also tables of values (one value for each instantiation of C and A).
- Thus we also call the original CPTs factors.
- Each factor is a function of some variables, e.g., P(C|A) = f(A,C): it maps each value of its arguments onto a number.
  - A tabular representation is exponential in the number of variables in the factor.

Operations on Factors

- If we examine the summation process we will see that various operations repeatedly occur on factors.

- Notation: $f(\underline{X},\underline{Y})$ denotes a factor over the variables $\underline{X}$ and $\underline{Y}$ (where $\underline{X}$ and $\underline{Y}$ are sets of variables)

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 x g (or sometimes just h = fg), is defined as:

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

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
- We can *sum out* variable X from f to produce a new factor h = Σ_X f, which is defined: **h(Y)** = $\Sigma_{x \in Dom(X)}$ **f(x,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		

Restricting a Factor

- Let **f(X,Y)** be a factor
- We can *restrict* factor f *to* X = a by setting X to the value x and "deleting" incompatible elements of f's domain .

Define $h = f_{X=a}$ as: h(Y) = f(a,Y)

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

Variable Elimination the Algorithm

Given query var **Q**, evidence vars **E** (variables observed to have values **e**), and remaining vars **Z**. Let **F** be factors in original CPTs.

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

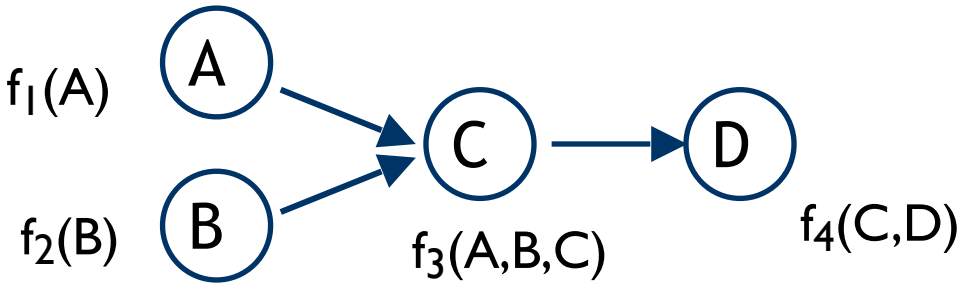
VE: Example

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

Query: P(A)?

Evidence: D = d

Elimination Order: C, B



Step 1 (Restriction): Replace $f_4(C,D)$ with $f_5(C) = f_4(C,d)$

Step 2 (Eliminate C): Compute & add $f_6(A,B) = \sum_C f_5(C)\, f_3(A,B,C)$ to list of factors.

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

Step 3 (Eliminate B): Compute & add $f_7(A) = \sum_B f_6(A,B)\, f_2(B)$ to list of factors.

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

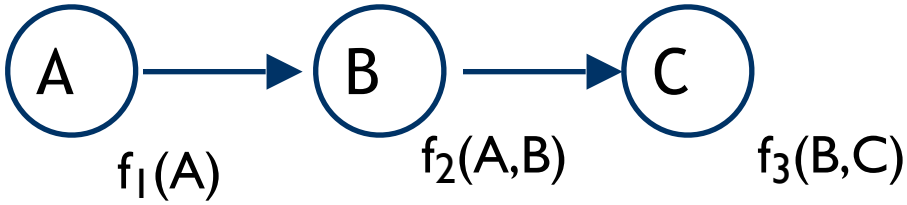
Step 4 (Normalize Final Factors): $f_7(A)$, $f_1(A)$. The product $f_1(A) \times f_7(A)$ is (un-normalized) posterior for A. So we normalize:

$P(A|d) = \alpha\, f_1(A) \times f_7(A)$

where $\alpha = 1/\sum_A f_1(A) f_7(A)$

Numeric Example

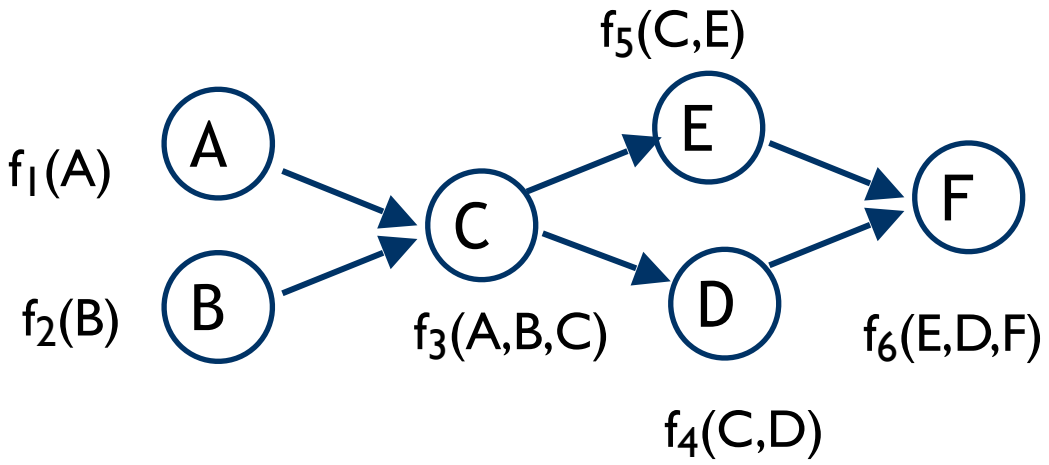
Here's an example with some . We want P(C) given no evidence.



$f_1(A)$		$f_2(A,B)$		$f_3(B,C)$		$f_4(B)$ Σ_A $f_2(A,B)f_1(A)$		$f_5(C)$ Σ_B $f_3(B,C)$ $f_4(B)$	
a	0.9	ab	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
		~ab	0.4	~bc	0.2				
		~a~b	0.6	~b~c	0.8				

VE: Buckets as a Notational Device

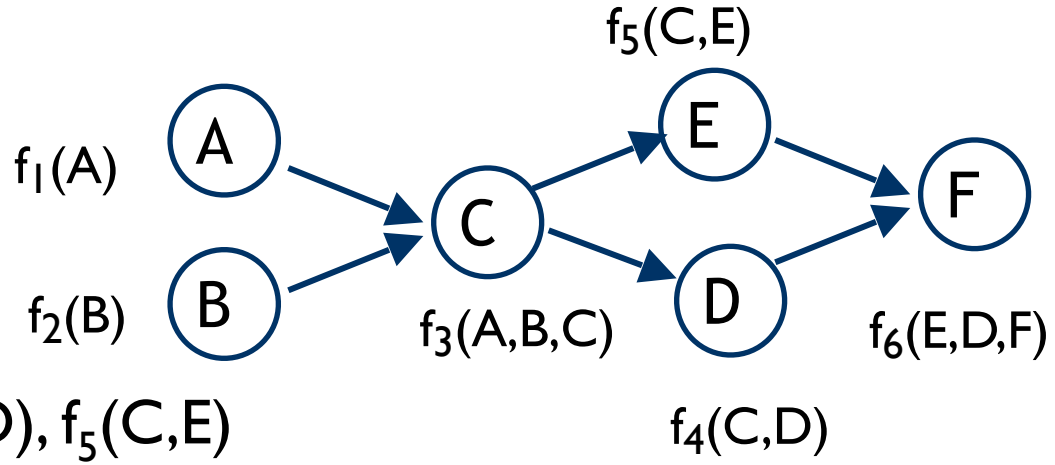
Ordering:
C,F,A,B,E,D



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

VE: Buckets—Place Original Factors in first applicable bucket.

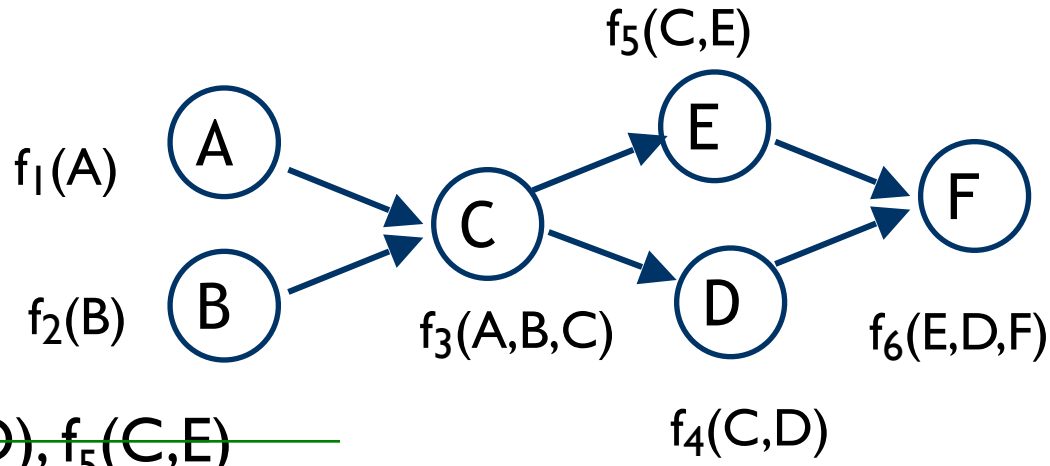
Ordering:
C,F,A,B,E,D



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:

VE: Eliminate the variables in order, placing new factor in first applicable bucket.

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

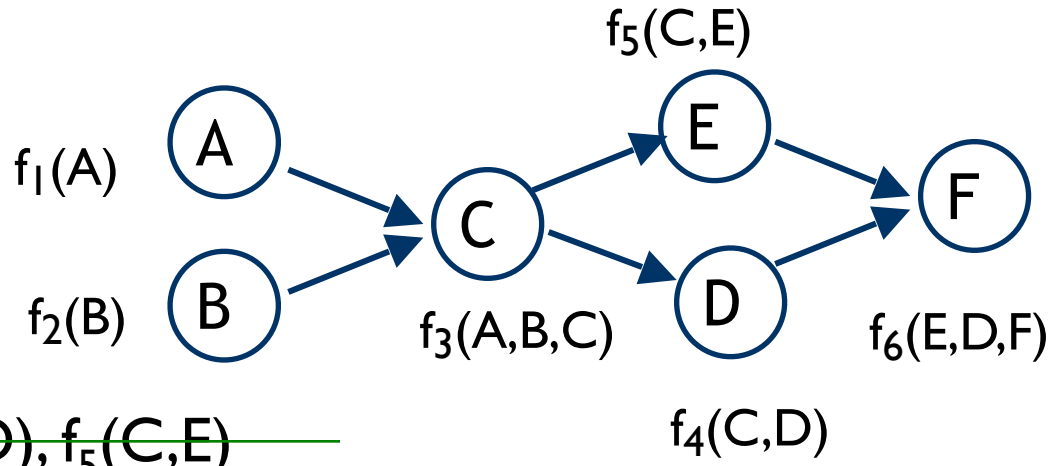


1. ~~C: $f_3(A,B,C), f_4(C,D), f_5(C,E)$~~
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E:
6. D:

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

VE: Eliminate the variables in order, placing new factor in first applicable bucket.

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

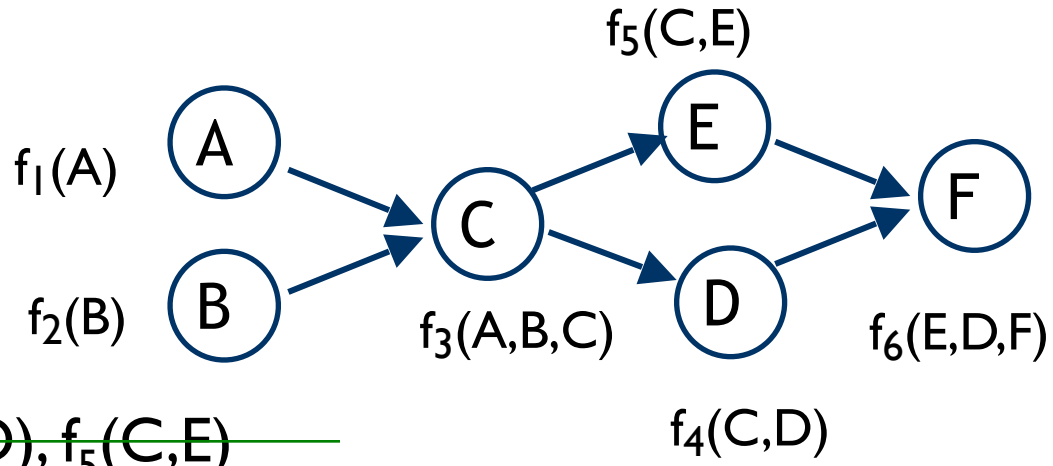


1. ~~C: $f_3(A,B,C), f_4(C,D), f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. A: $f_1(A), f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E: $f_8(E,D)$
6. D:

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

VE: Eliminate the variables in order, placing new factor in first applicable bucket.

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

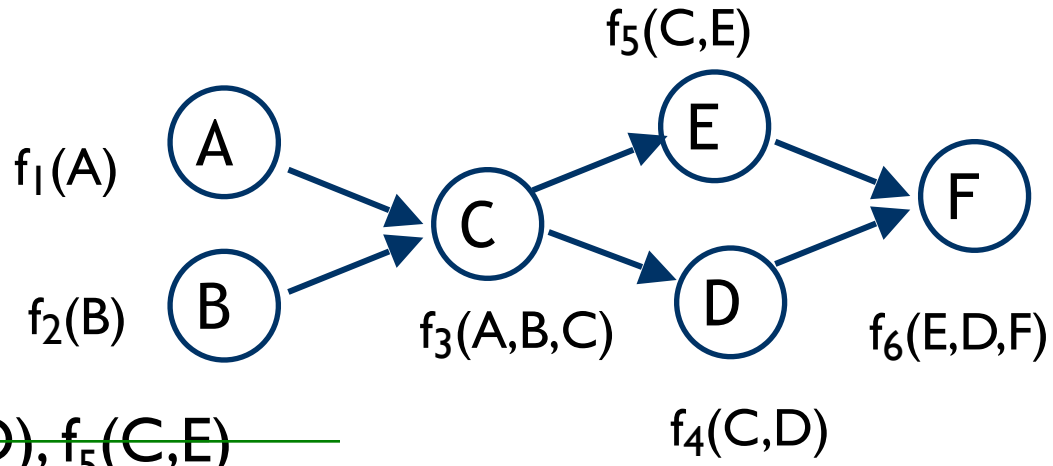


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. B: $f_2(B)$, $f_9(B,D,E)$
5. E: $f_8(E,D)$
6. D:

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

VE: Eliminate the variables in order, placing new factor in first applicable bucket.

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

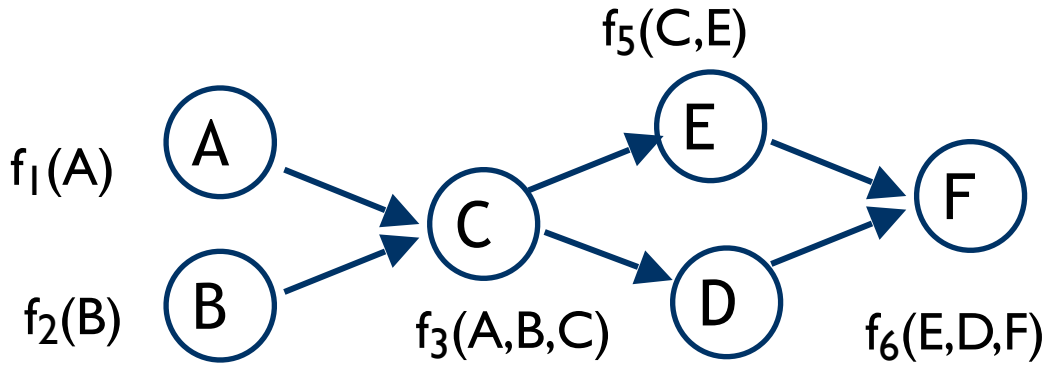


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. E: $f_8(E,D)$, $f_{10}(D,E)$
6. D:

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

VE: Eliminate the variables in order, placing new factor in first applicable bucket.

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: $f_{11}(D)$

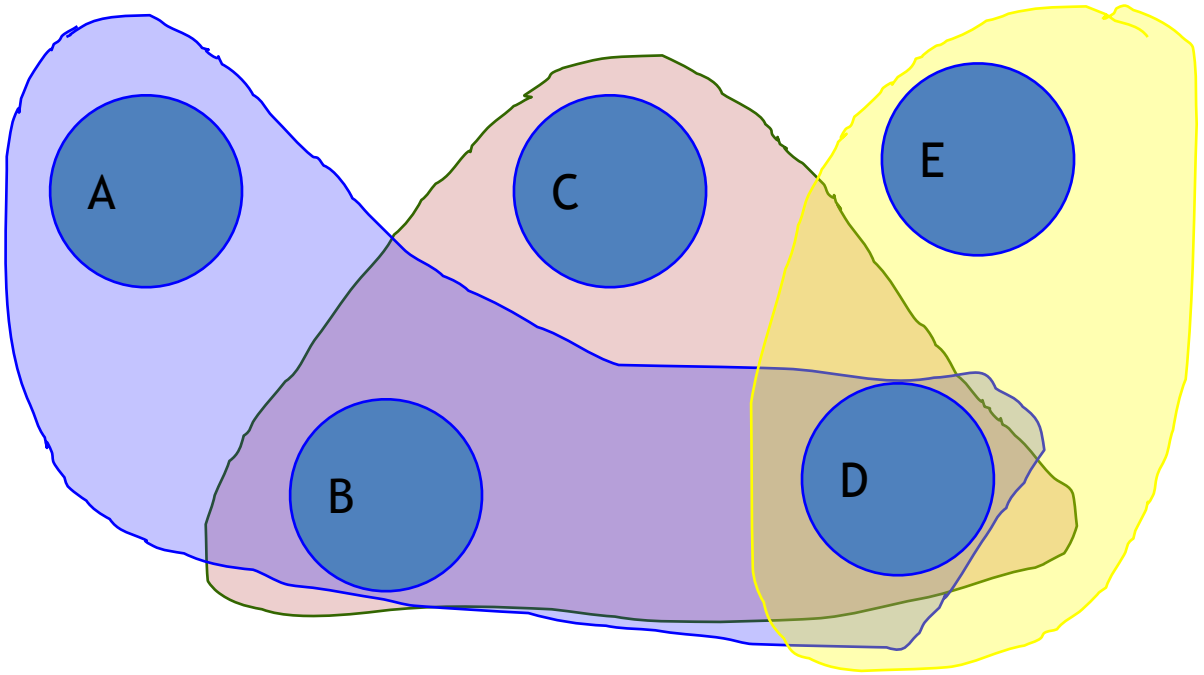
5. $\Sigma_E f_8(E,D) \times f_{10}(D,E) = f_{11}(D)$

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

Complexity of Variable Elimination

A **hypergraph** has vertices just like an ordinary graph, but instead of edges between two vertices X↔Y it contains **hyperedges**.

- A hyperedge is a set of vertices (i.e., potentially more than one)



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

Complexity of Variable Elimination

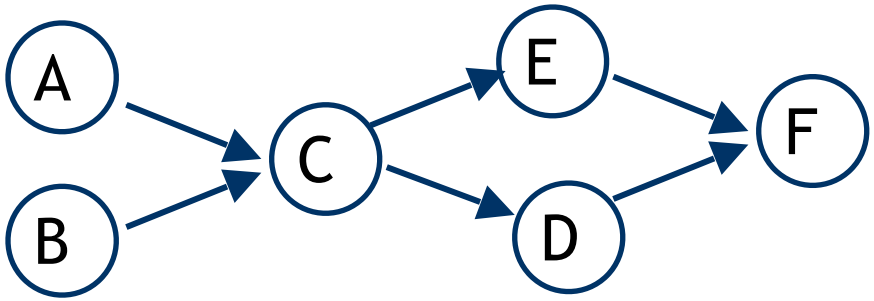
Hypergraph of Bayes Net

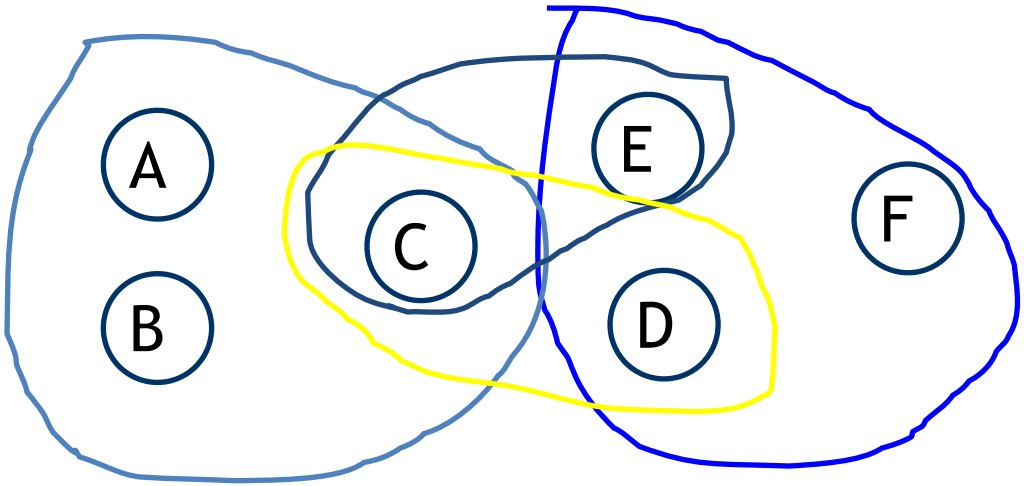
- The set of vertices are the nodes of the Bayes net.
- The hyperedges are the families appearing in each CPT.

  $\{X_i\} \cup \text{Parents}(X_i)$

Complexity of Variable Elimination

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



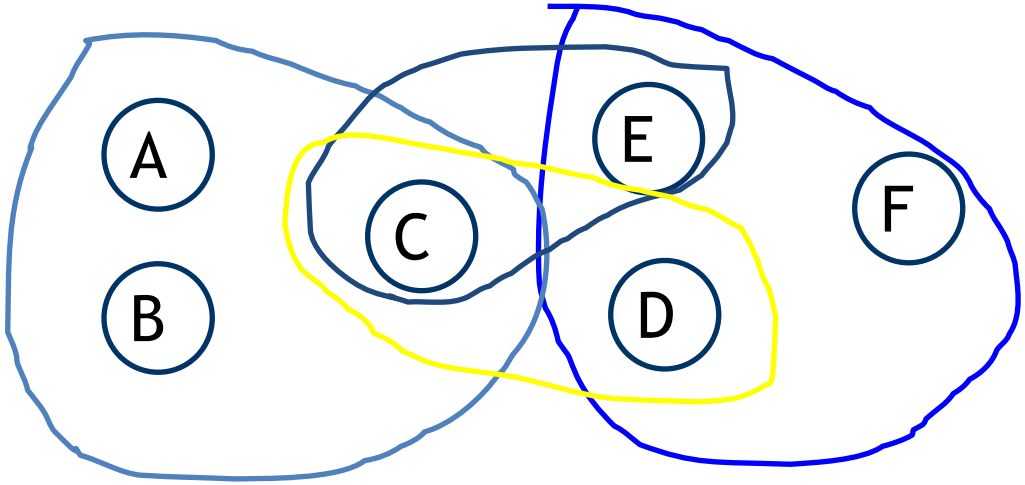


Variable Elimination in the HyperGraph

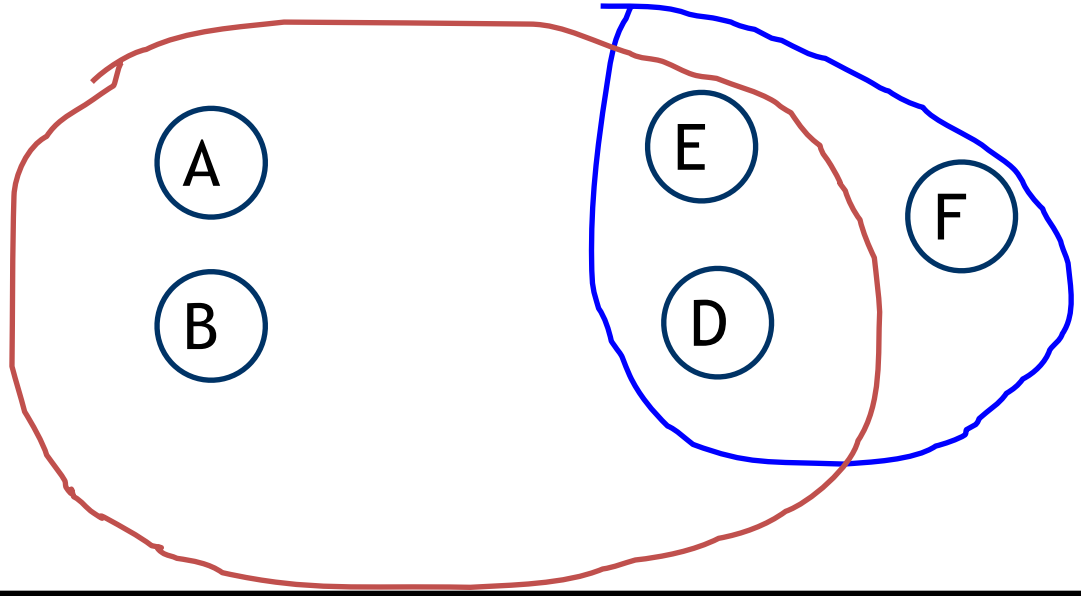
To eliminate variable X_i in the hypergraph we

- 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 of the hyperedges containing X from the hypergraph.
- Add the new hyperedge H_i to the hypergraph.

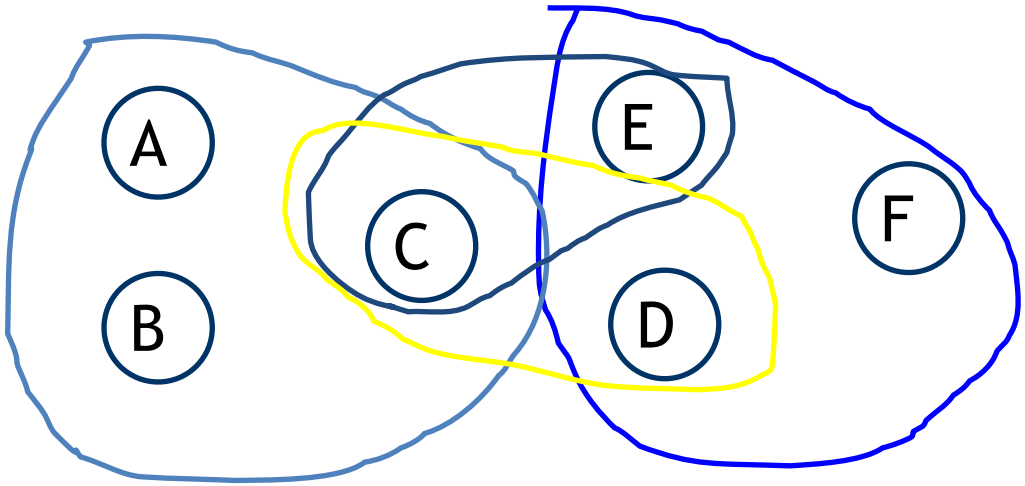
Complexity of Variable Elimination



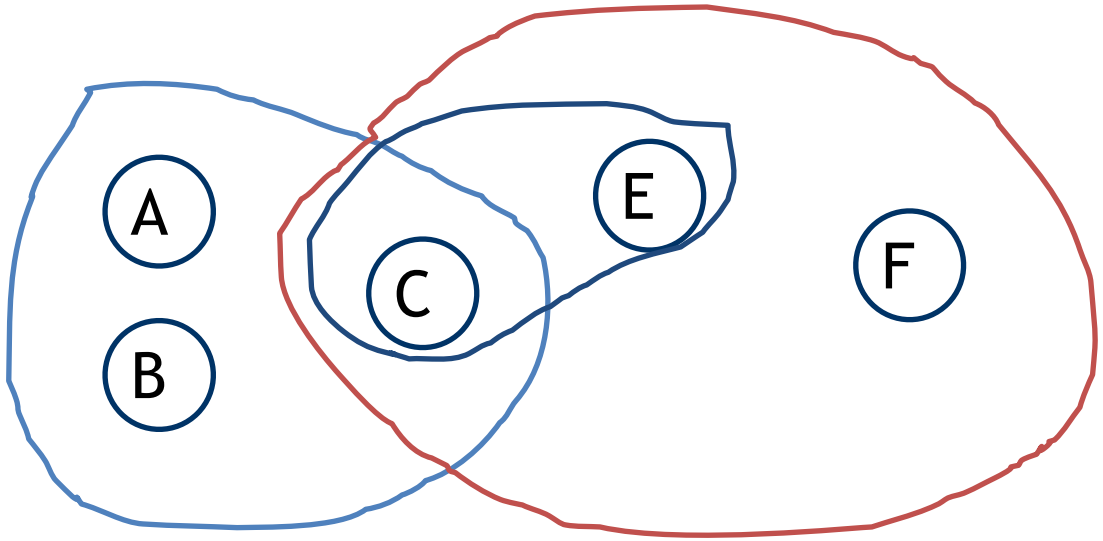
- Eliminate C



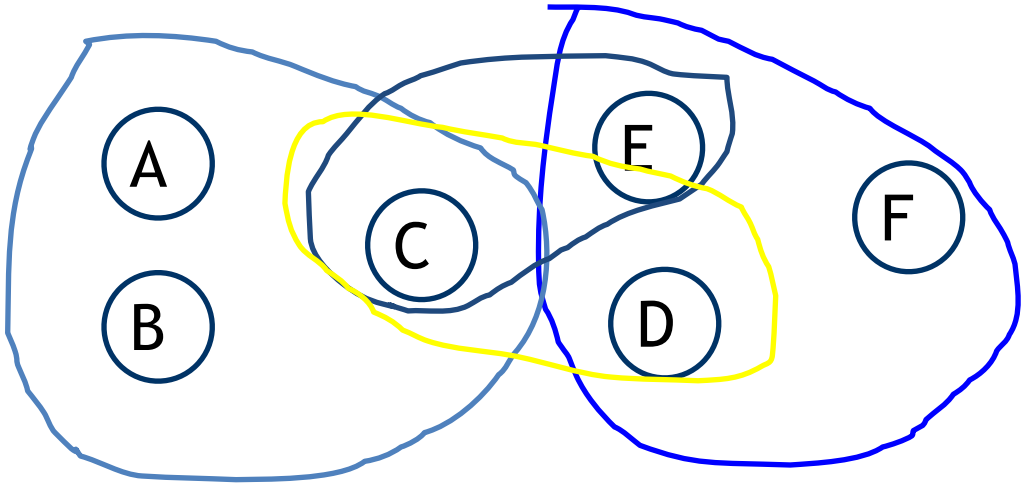
Complexity of Variable Elimination



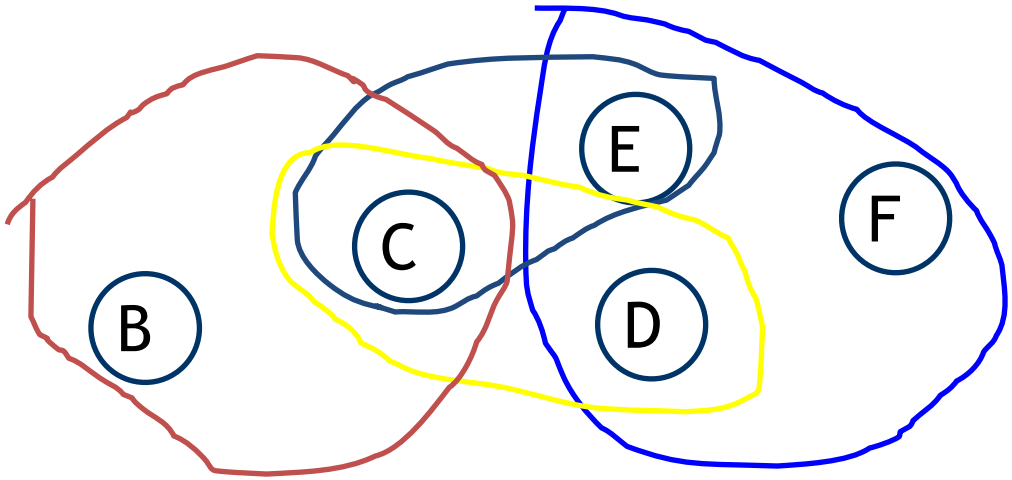
- Eliminate D



Complexity of Variable Elimination



- Eliminate A



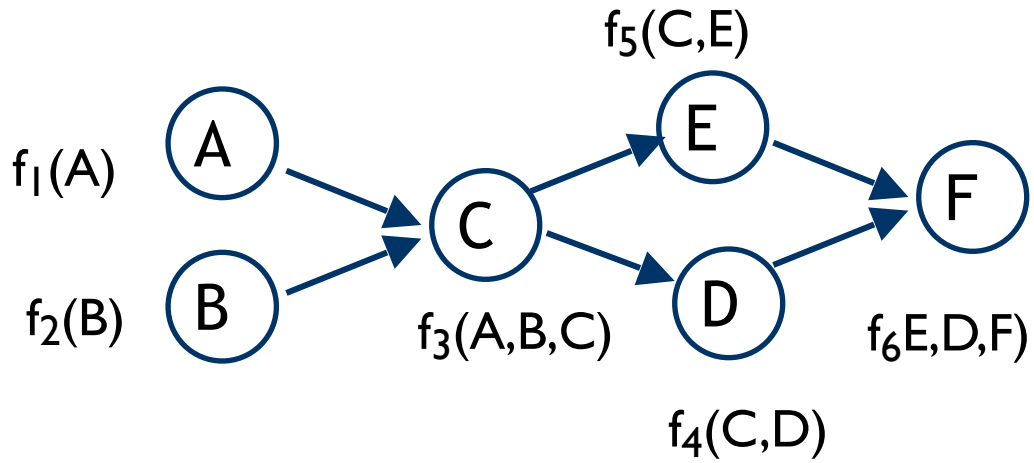
Variable Elimination

- Notice that when at the outset of VE we have a set of factors consisting of the reduced (or restricted) CPTs. The unassigned variables for the vertices and the set of variables each factor depends on forms the hyperedges of a hypergraph H_1.

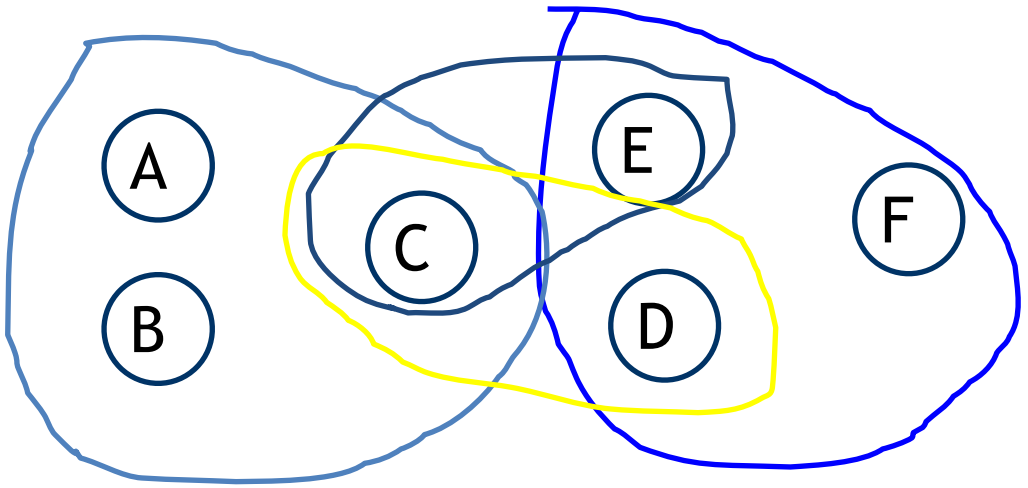
- If the first variable we eliminate is X, then we remove all factors containing X (all hyperedges) and add a new factor that has as variables the union of the variables in the factors containing X (we add a hyperdege that is the union of the removed hyperedges minus X).

VE Factors

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

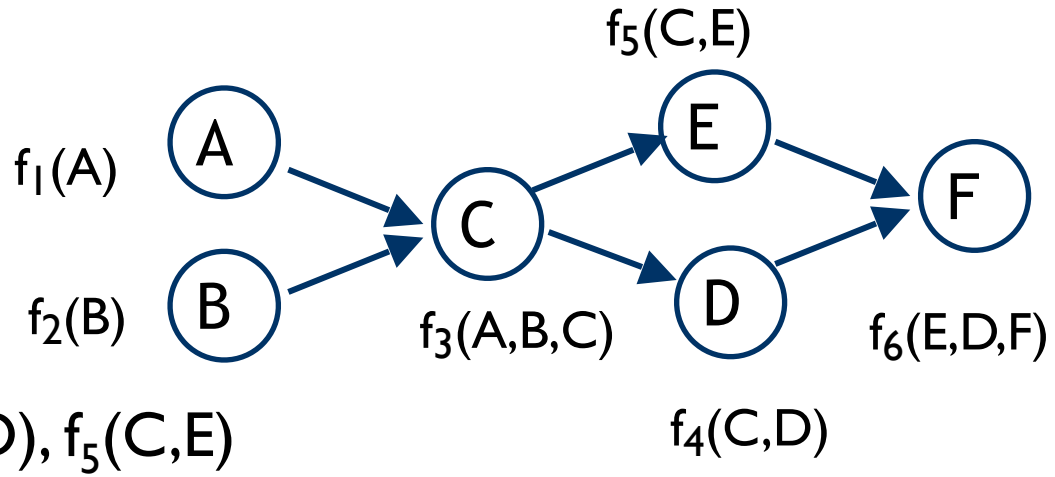


1. C:
2. F:
3. A:
4. B:
5. E:
6. D:



VE: Place Original Factors in first applicable bucket.

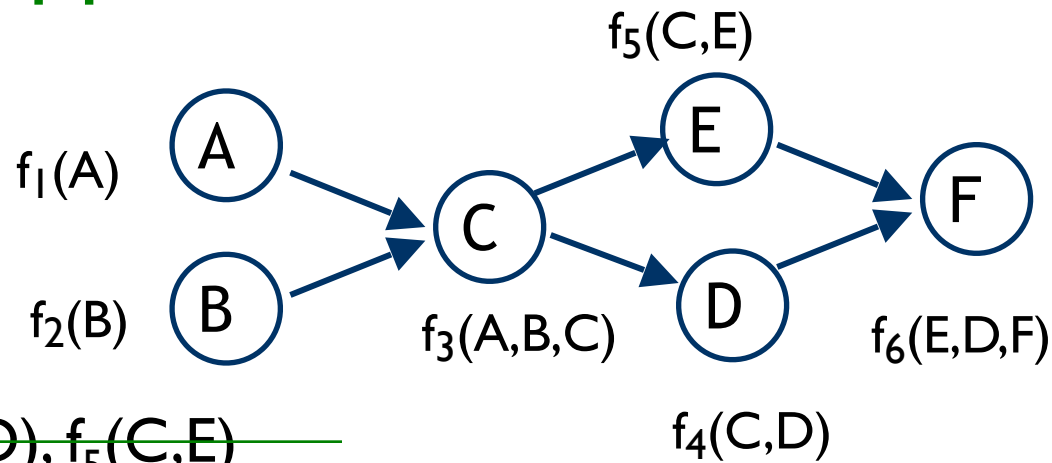
Ordering:
C,F,A,B,E,D



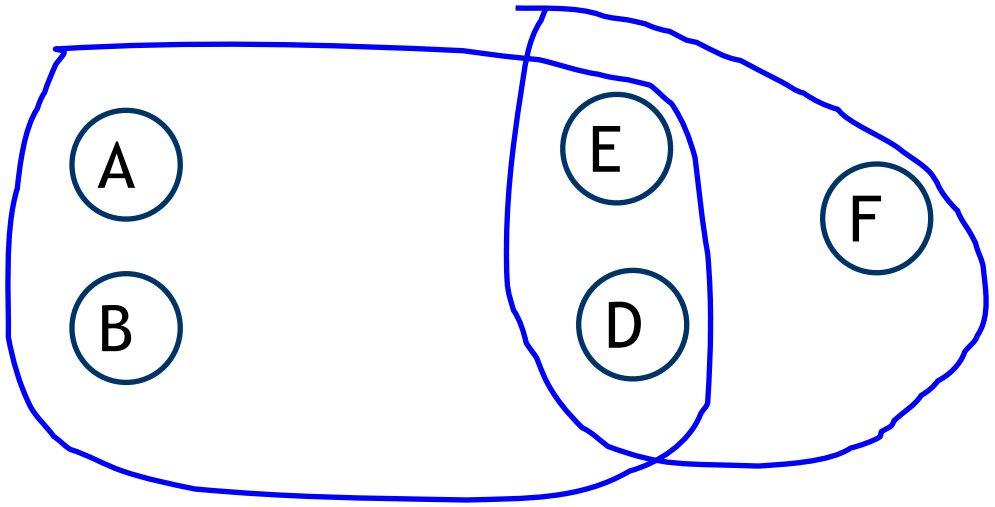
1. C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$
4. B: $f_2(B)$
5. E:
6. D:

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

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

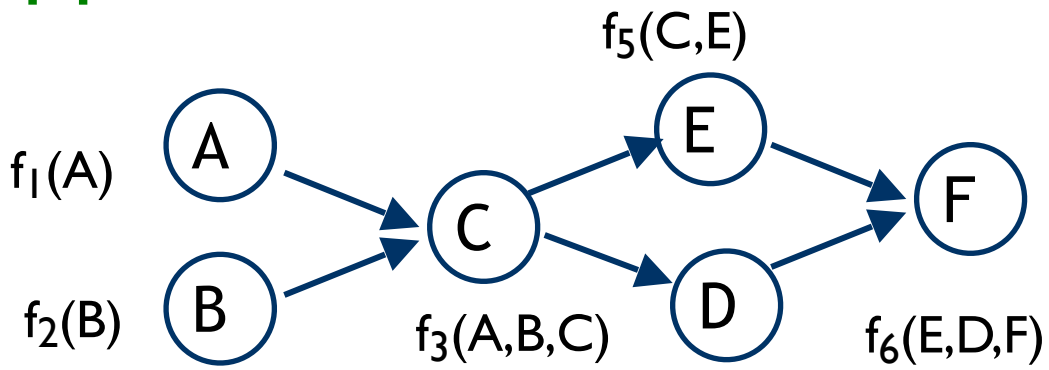


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E:
6. D:

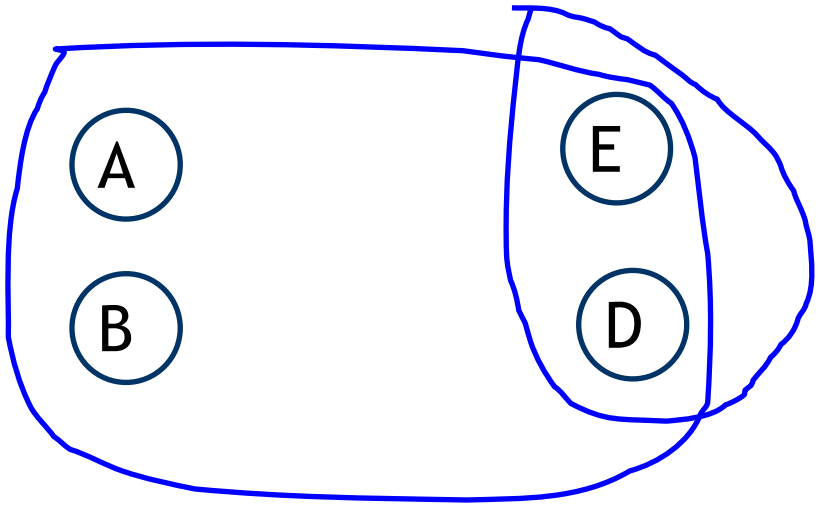


VE: Eliminate F, placing new factor f8 in first applicable bucket.

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

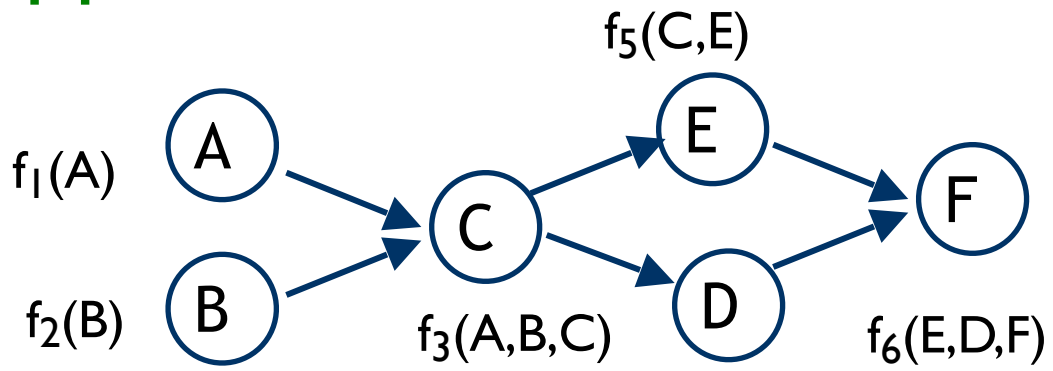


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E: $f_8(E,D)$
6. D:

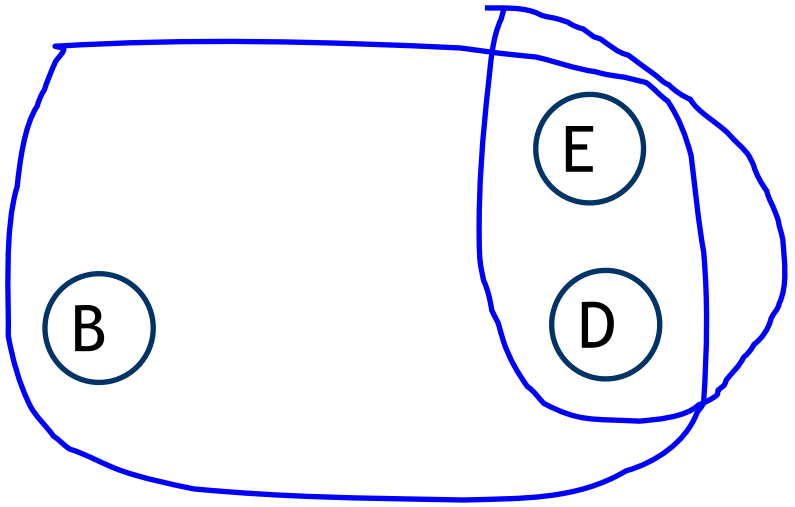


VE: Eliminate A, placing new factor f9 in first applicable bucket.

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

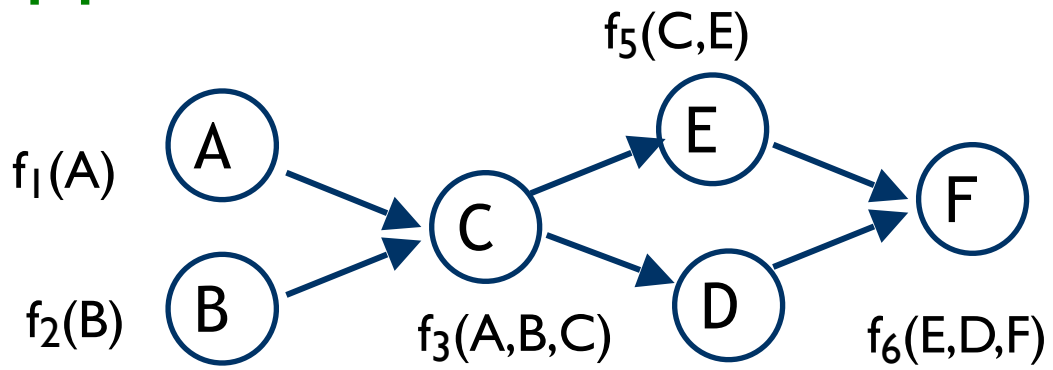


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. B: $f_2(B)$, $f_9(B,D,E)$
5. E: $f_8(E,D)$
6. D:

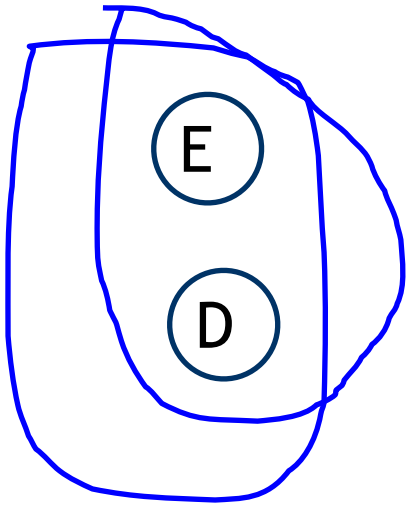


VE: Eliminate B, placing new factor f10 in first applicable bucket.

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

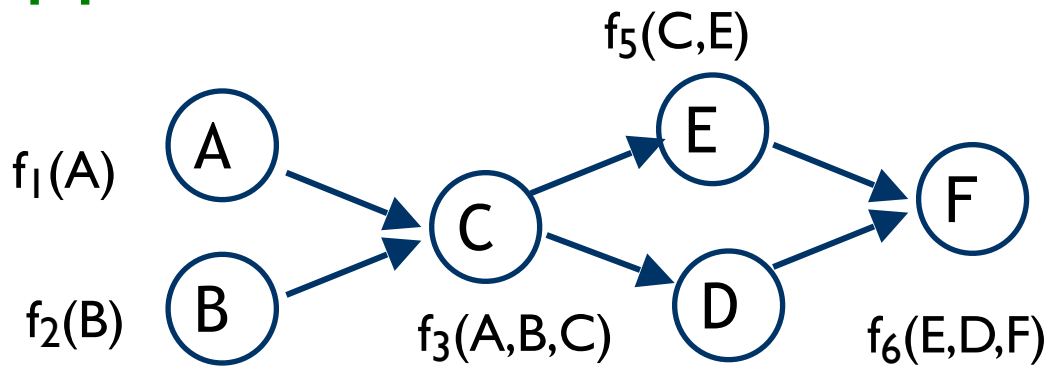


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. E: $f_8(E,D)$, $f_{10}(D,E)$
6. D:



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

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: $f_{11}(D)$