# Lecture 4: Bayes nets

- Bayesian networks
- Variable elimination algorithm

# Inference: Computational Bottleneck

- Semantically/conceptually, picture is clear; but several issues must be addressed
- Issue 1: How do we specify the full joint distribution over $X_1, X_2, \ldots, X_n$ ?
  - exponential number of possible worlds
  - e.g., if the $X_i$ are boolean, then $2^n$ numbers (or $2^n$ -1 parameters/degrees of freedom, since they sum to 1)
  - these numbers are not robust/stable
  - these numbers are not natural to assess (what is probability that "Craig wants coffee; it's raining in Coquitlam; robot charge level is low; …"?)

# Inference: Computational Bottleneck

- Issue 2: Inference in this rep’n frightfully slow
  - Must sum over exponential number of worlds to answer query $Pr(\alpha)$ or to condition on evidence e to determine $Pr_e(\alpha)$
- How do we avoid these two problems?
  - no solution in general
  - but in practice there is structure we can exploit
- We’ll use conditional independence

# Independence

- Recall that x and y are *independent* iff:
  - $Pr(x) = Pr(x|y)$ iff $Pr(y) = Pr(y|x)$ iff $Pr(xy) = Pr(x)Pr(y)$
  - intuitively, learning y doesn't influence beliefs about x
- x and y are *conditionally independent given* z iff:
  - $Pr(x|z) = Pr(x|yz)$ iff $Pr(y|z) = Pr(y|xz)$ iff

    $Pr(xy|z) = Pr(x|z)Pr(y|z)$ iff …
  - intuitively, learning y doesn't influence your beliefs about x *if you already know z*
  - e.g., learning someone's mark on 886 project can influence the probability you assign to a specific GPA; but if you already knew **final** 886 grade, learning the project mark would *not* influence GPA assessmnt

# What does independence buy us?

- Suppose (say, boolean) variables $X_1, X_2, \ldots, X_n$ are mutually independent
  - we can specify full joint distribution using only n parameters (linear) instead of $2^n$ -1 (exponential)
- How? Simply specify $Pr(x_1), \ldots\ Pr(x_n)$
  - from this I can recover probability of any world or any (conjunctive) query easily
  - e.g. $Pr(x_1 {\sim} x_2 x_3 x_4) = Pr(x_1)\ (1-Pr(x_2))\ Pr(x_3)\ Pr(x_4)$
  - we can condition on observed value $X_k = x_k$ trivially by changing Pr($x_k$) to 1, leaving Pr($x_i$) untouched for $i \neq k$

# The Value of Independence

- Complete independence reduces both *representation of joint* and *inference* from $O(2^n)$ to $O(n)$: pretty significant!
- Unfortunately, such complete mutual independence is very rare. Most realistic domains do not exhibit this property.
- Fortunately, most domains do exhibit a fair amount of conditional independence. And we can exploit conditional independence for representation and inference as well.
- **Bayesian networks** do just this

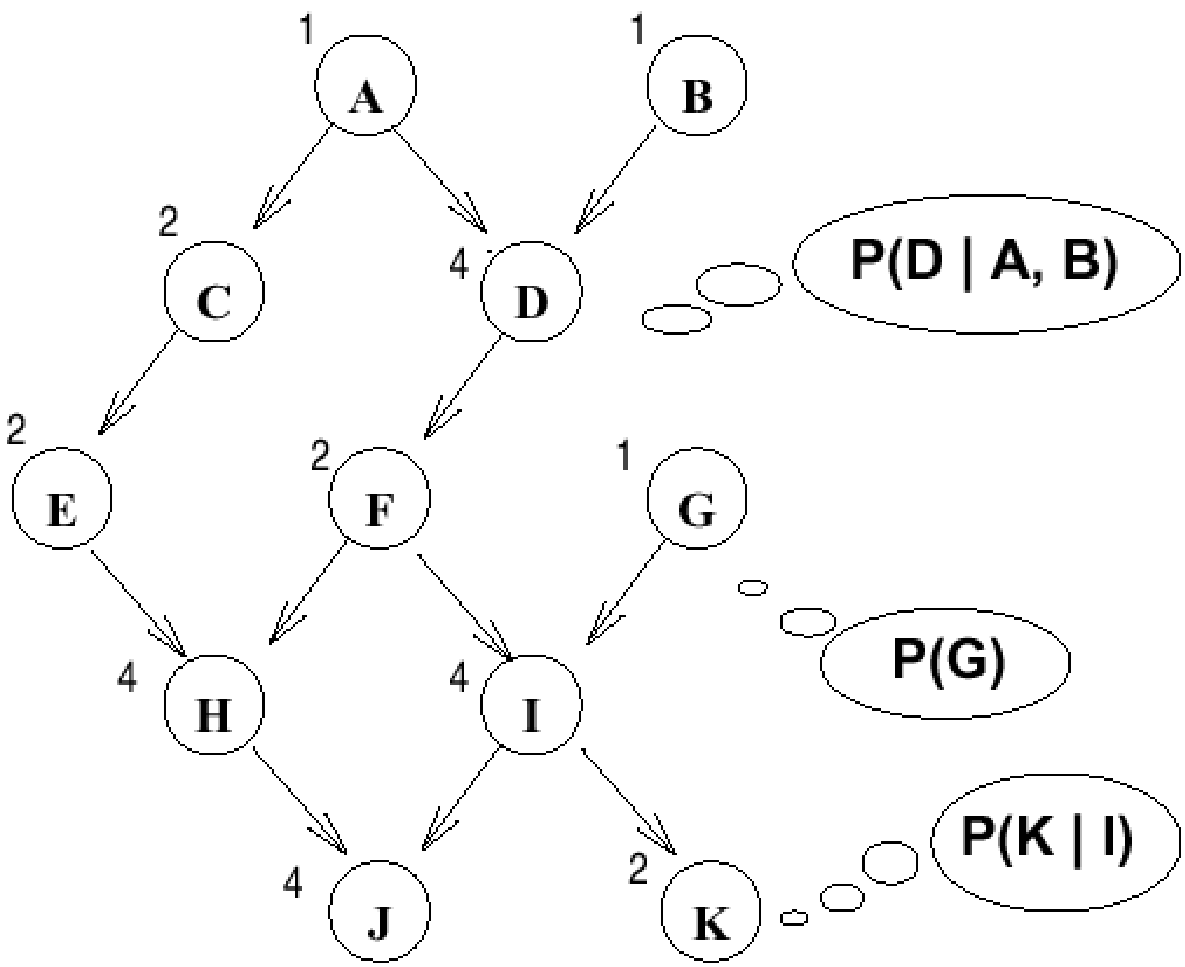
# Bayesian Networks

- A Bayesian Network is a *graphical representation* of the direct dependencies over a set of variables, together with a set of *conditional probability tables (CPTs)* quantifying the strength of those influences.
- Bayes nets exploit conditional independence in very interesting ways, leading to effective means of representation and inference under uncertainty.

# Bayesian Networks

- A BN over variables $\{X_1, X_2, \ldots, X_n\}$ consists of:
  - a DAG whose nodes are the variables
  - a set of CPTs $\Pr(X_i \mid \mathrm{Par}(X_i))$ for each $X_i$
- Key notions (see text for defn's, all are intuitive):
  - parents of a node: $\mathrm{Par}(X_i)$
  - children of node
  - descendents of a node
  - ancestors of a node
  - family: set of nodes consisting of $X_i$ and its parents
    - CPTs are defined over families in the BN

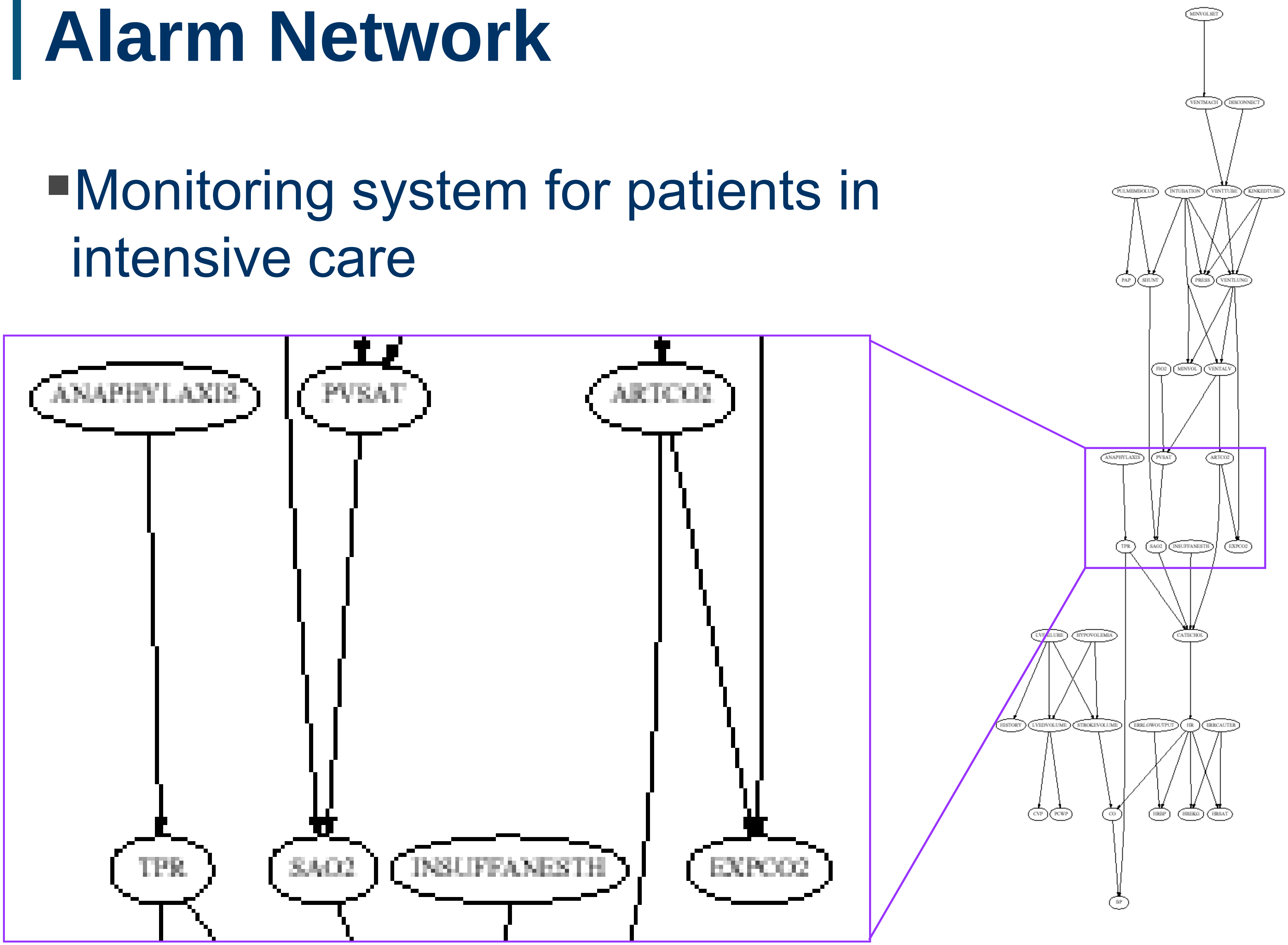
# An Example Bayes Net




- A couple CPTS are "shown"
- Explicit joint requires $2^{11}$ -1 =2047 parmtrs
- BN requires only 27 parmtrs (the number of entries for each CPT is listed)

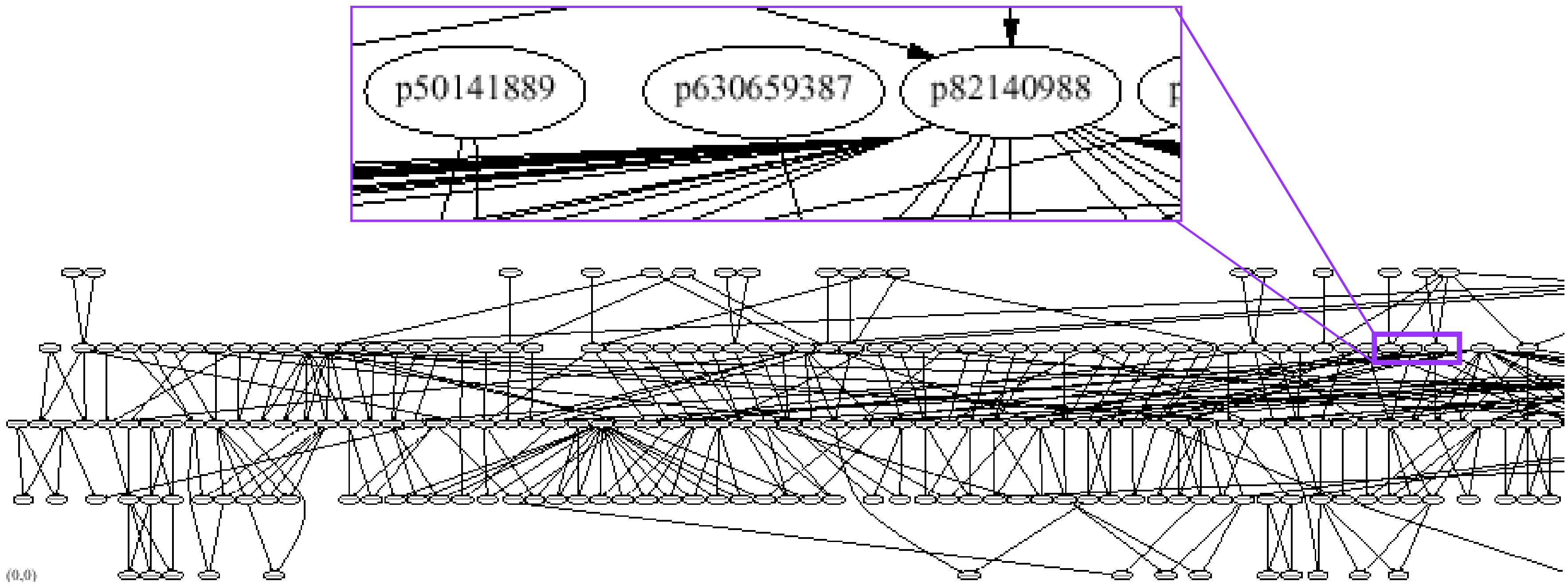
# Alarm Network

- Monitoring system for patients in intensive care



# Pigs Network

- Determines pedigree of breeding pigs
  - used to diagnose PSE disease
  - half of the network show here



# Semantics of a Bayes Net

- The structure of the BN means: every $X_i$ is *conditionally independent of all of its nondescendants given its parents*:

$$\Pr(X_i \mid S \cup \mathrm{Par}(X_i)) = \Pr(X_i \mid \mathrm{Par}(X_i))$$

for any subset $S \subseteq \mathrm{NonDescendants}(X_i)$

# Semantics of Bayes Nets (2)

- If we ask for $Pr(x_1, x_2, \ldots, x_n)$ we obtain
  - assuming an ordering consistent with network

- By the chain rule, we have:

$$
\begin{aligned}
& Pr(x_1, x_2, \ldots, x_n) \\
& \quad = Pr(x_n \mid x_{n-1}, \ldots, x_1)\, Pr(x_{n-1} \mid x_{n-2}, \ldots, x_1) \ldots Pr(x_1) \\
& \quad = Pr(x_n \mid Par(X_n))\, Pr(x_{n-1} \mid Par(x_{n-1})) \ldots Pr(x_1)
\end{aligned}
$$

- Thus, the joint is recoverable using the parameters (CPTs) specified in an arbitrary BN

# Bayes net queries

- Example Query: Pr(**X**|**Y=y**)?
- Intuitively, want to know value of X given some information about the value of Y
- Concrete examples:
  - Doctor: Pr(Disease|Symptoms)?
  - Car: Pr(condition|mechanicsReport)?
  - Fault diag.: Pr(pieceMalfunctioning|systemStatistics)?
- Use Bayes net structure to quickly compute Pr(X|Y=y)

# Algorithms to answer Bayes net queries

- There are many…
  - Variable elimination (very simple!)
  - Clique tree propagation (quite popular!)
  - Cut-set conditioning
  - Arc reversal node reduction
  - Symbolic probabilistic inference

- They all exploit conditional independence to speed up computation

# Potentials

- A function $f(X_1, X_2, \ldots, X_k)$ is also called a *potential*. We can view this as table of numbers, one for each instantiation of the variables $X_1, X_2, \ldots, X_k$.
- A tabular rep'n of a potential is exponential in k
- Each CPT in a Bayes net is a potential:
  - e.g., Pr(C|A,B) is a function of three variables, A, B, C
- Notation: f(**X**,**Y**) denotes a potential over the variables **X** ∪ **Y**. (Here **X**, **Y** are *sets* of variables.)

# The Product of Two Potentials

- Let f(**X**,**Y**) & g(**Y**,**Z**) be two potentials with variables **Y** in common
- The *product* of f and g, denoted h = f x 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 Potential

- Let $f(X,\mathbf{Y})$ be a factor with variable X ($\mathbf{Y}$ is a set)
- We *sum out* variable X from f to produce a new potential $h = \Sigma_X f$, 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 | | |

# Restricting a Potential

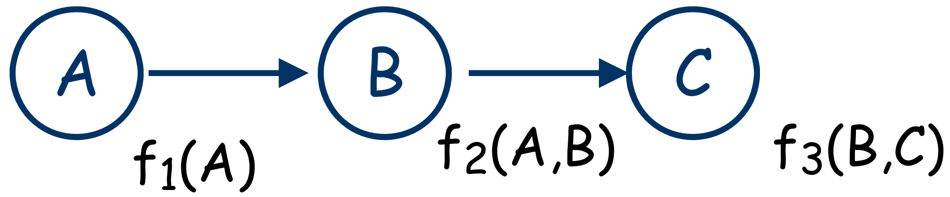
- Let $f(X,\mathbf{Y})$ be a potential with var. $X$ ($\mathbf{Y}$ is a set)
- We *restrict* potential $f$ *to* $X=x$ by setting $X$ to the value $x$ and "deleting". Define $h = f_{X=x}$ as:

$$h(\mathbf{Y}) = f(x,\mathbf{Y})$$

| 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 | | |

# Variable Elimination: No Evidence

- Compute prior probability of var C

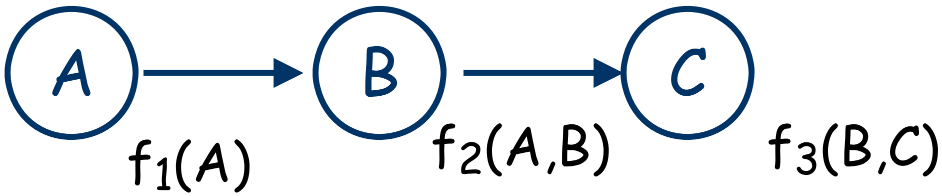



$$
\begin{aligned}
P(C) &= \Sigma_{A,B}\ P(A,B,C) \\
&= \Sigma_{A,B}\ P(C|B)\ P(B|A)\ P(A) \\
&= \Sigma_B\ P(C|B)\ \Sigma_A\ P(B|A)\ P(A) \\
&= \Sigma_B\ f_3(B,C)\ \Sigma_A\ f_2(A,B)\ f_1(A) \\
&= \Sigma_B\ f_3(B,C)\ f_4(B) \\
&= f_5(C)
\end{aligned}
$$

Define new potentials: $f_4(B) = \Sigma_A\ f_2(A,B)\ f_1(A)$ and $f_5(C) = \Sigma_B\ f_3(B,C)\ f_4(B)$

# Variable Elimination: No Evidence

- Here's the example with some numbers




| $f_1(A)$ | | $f_2(A,B)$ | | $f_3(B,C)$ | | $f_4(B)$ | | $f_5(C)$ | |
|---|---|---|---|---|---|---|---|---|---|
| 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 | | | | |

# Variable Elimination: One View

- One way to think of variable elimination:
  - write out desired computation using the chain rule, exploiting the independence relations in the network
  - arrange the terms in a convenient fashion
  - distribute each sum (over each variable) in as far as it will go
    - i.e., the sum over variable X can be "pushed in" as far as the "first" potential mentioning X
  - apply operations "inside out", repeatedly eliminating and creating new potentials (note that each step/removal of a sum eliminates one variable)

# Variable Elimination Algorithm

- Given query var Q, remaining vars **Z**. Let F be set of potentials corresponding to CPTs for $\{Q\} \cup \mathbf{Z}$.

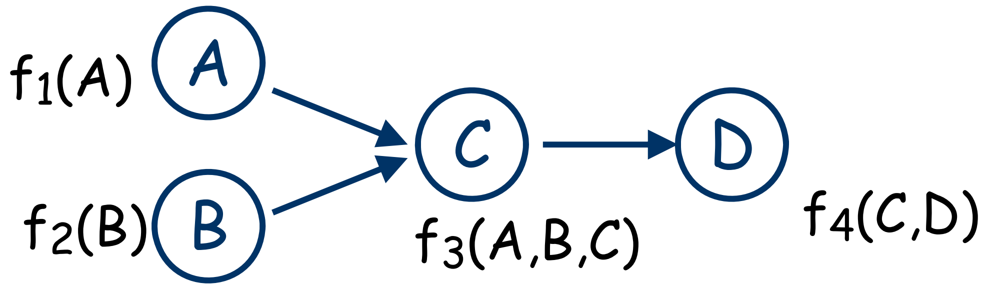
1. Choose an elimination ordering $Z_1, \ldots, Z_n$ of variables in **Z**.
2. For each $Z_j$ -- in the order given -- eliminate $Z_j \in \mathbf{Z}$ as follows:
   (a) Compute new potential $g_j = \Sigma_{Z_j} f_1 \times f_2 \times \ldots \times f_k$, where the $f_i$ are the potentials in F that include $Z_j$
   (b) Remove the potentials $f_i$ (that mention $Z_j$ ) from F and add new potential $g_j$ to F
3. The remaining potentials refer only to the query variable Q. Take their product and normalize to produce P(Q)

# VE: Example 2

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

**Query:** P(D)?

**Elim. Order:** A, B, C




Step 1: Add $f_5(B,C) = \Sigma_A f_3(A,B,C)\, f_1(A)$

Remove: $f_1(A)$, $f_3(A,B,C)$

Step 2: Add $f_6(C) = \Sigma_B f_2(B)\, f_5(B,C)$

Remove: $f_2(B)$ , $f_5(B,C)$

Step 3: Add $f_7(D) = \Sigma_C f_4(C,D)\, f_6(C)$

Remove: $f_4(C,D)$, $f_6(C)$

Last factor $f_7(D)$ is (possibly unnormalized) probability P(D)

# Variable Elimination with Evidence

Given query var Q, evidence vars **E** (observed to be **e**), remaining vars **Z**. Let F be set of factors involving CPTs for {Q} ∪ **Z**.

1. Replace each potential $f \in F$ that mentions variable(s) in **E** with its restriction $f_{E=e}$ (somewhat abusing notation)
2. Choose an elimination ordering $Z_1, \ldots, Z_n$ of variables in **Z**.
3. Run variable elimination as above.
4. The remaining potentials refer only to query variable Q. Take their product and normalize to produce P(Q)

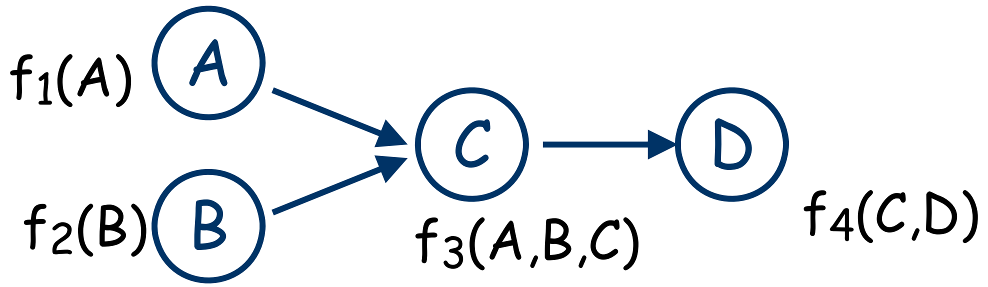
# VE: Example 2 again with Evidence

**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: 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: Add $f_7(A) = \Sigma_B\, f_6(A,B)\, f_2(B)$

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

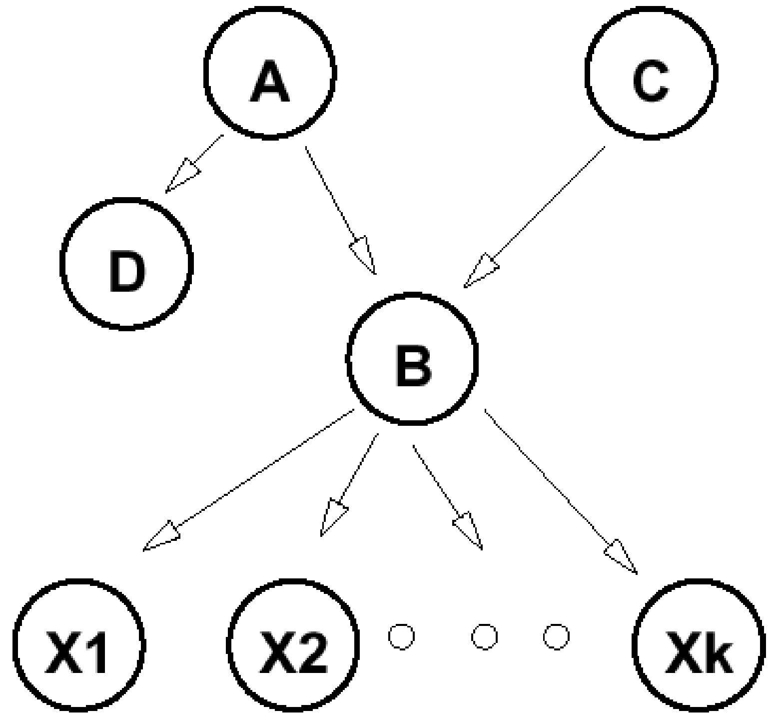
Last potent.: $f_7(A)$, $f_1(A)$. The product $f_1(A) \times f_7(A)$ is (possibly unnormalized) posterior. So… $P(A|d) = \alpha\, f_1(A) \times f_7(A)$.

# Some Notes on the VE Algorithm

- The size of the resulting factors is determined by elimination ordering!
- For *polytrees*, easy to find good ordering (e.g., work outside in).
- For general BNs, sometimes good orderings exist, sometimes they don't (then inference is exponential in number of vars).
  - Simply *finding* the optimal elimination ordering for general BNs is NP-hard.
  - Inference in general is NP-hard in general BNs

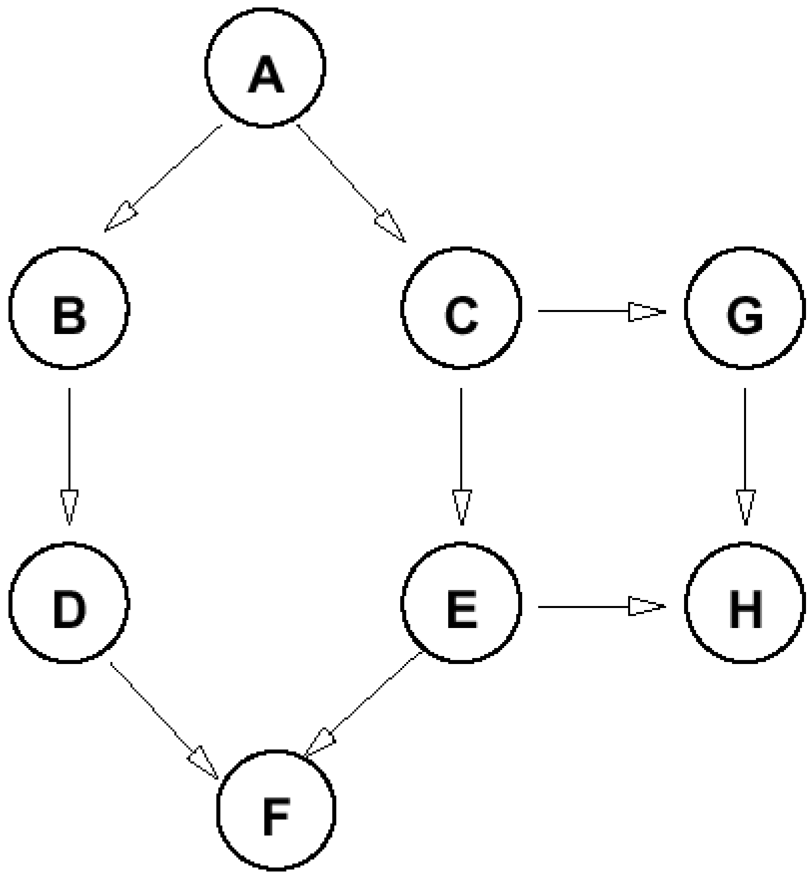
# Elimination Ordering: Polytrees

- Inference is linear in size of network
  - ordering: eliminate only “singly-connected” nodes
  - 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
  - eliminating B before these gives factors that include all of A,C, X1,… $X_k$ !!!

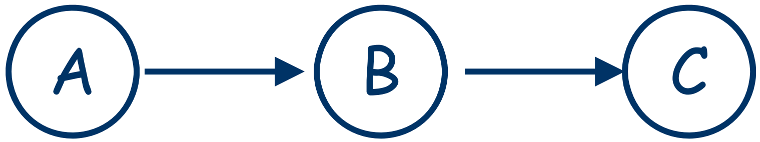


# Effect of Different Orderings

- Suppose query variable is D. Consider different orderings for this network
  - A,F,H,G,B,C,E:
    - good: why?
  - E,C,A,B,G,H,F:
    - bad: why?
- Which ordering creates smallest factors?
  - either max size or total
- which creates largest factors?



# Relevance

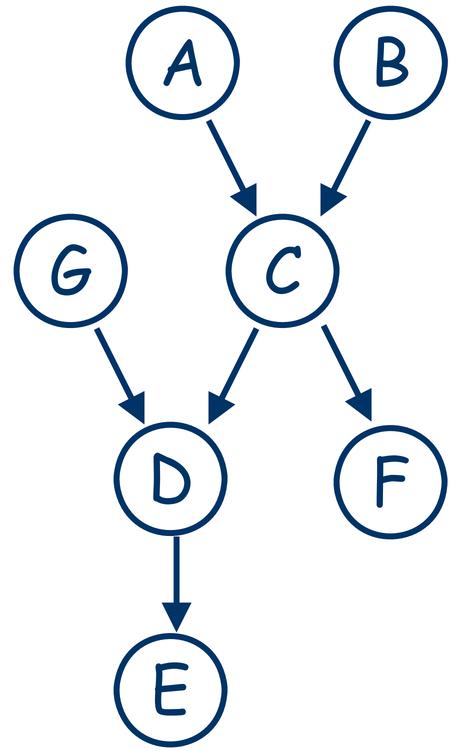



- Certain variables have no impact on the query. In ABC network, computing Pr(A) with no evidence requires elimination of B and C.
  - But when you sum out these vars, you compute a trivial potential (all values are ones); for example:
  - eliminating C: $f_4(B) = \Sigma_C f_3(B,C) = \Sigma_C \Pr(C|B)$
  - 1 for any value of B (e.g., $\Pr(c|b) + \Pr(\sim c|b) = 1$)
- No need to think about B or C for this query

# Pruning irrelevant variables

- Can restrict attention to *relevant* variables. Given query Q, evidence **E**:
  - Q is relevant
  - if any node Z is relevant, its parents are relevant
  - if E∈**E** is a descendent of a relevant node, then E is relevant
- We can restrict our attention to the *subnetwork comprising only relevant variables* when evaluating a query Q

# 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)=α P(C) P(E|C) to accurately compute P(F|E)
- Query: P(F|E,C)
  - algorithm says all vars relevant; but really none except C, F (since C cuts of all influence of others)
  - algorithm is overestimating relevant set