Lecture 6: Bayesian Networks
CS486/686 Intro to Artificial Intelligence

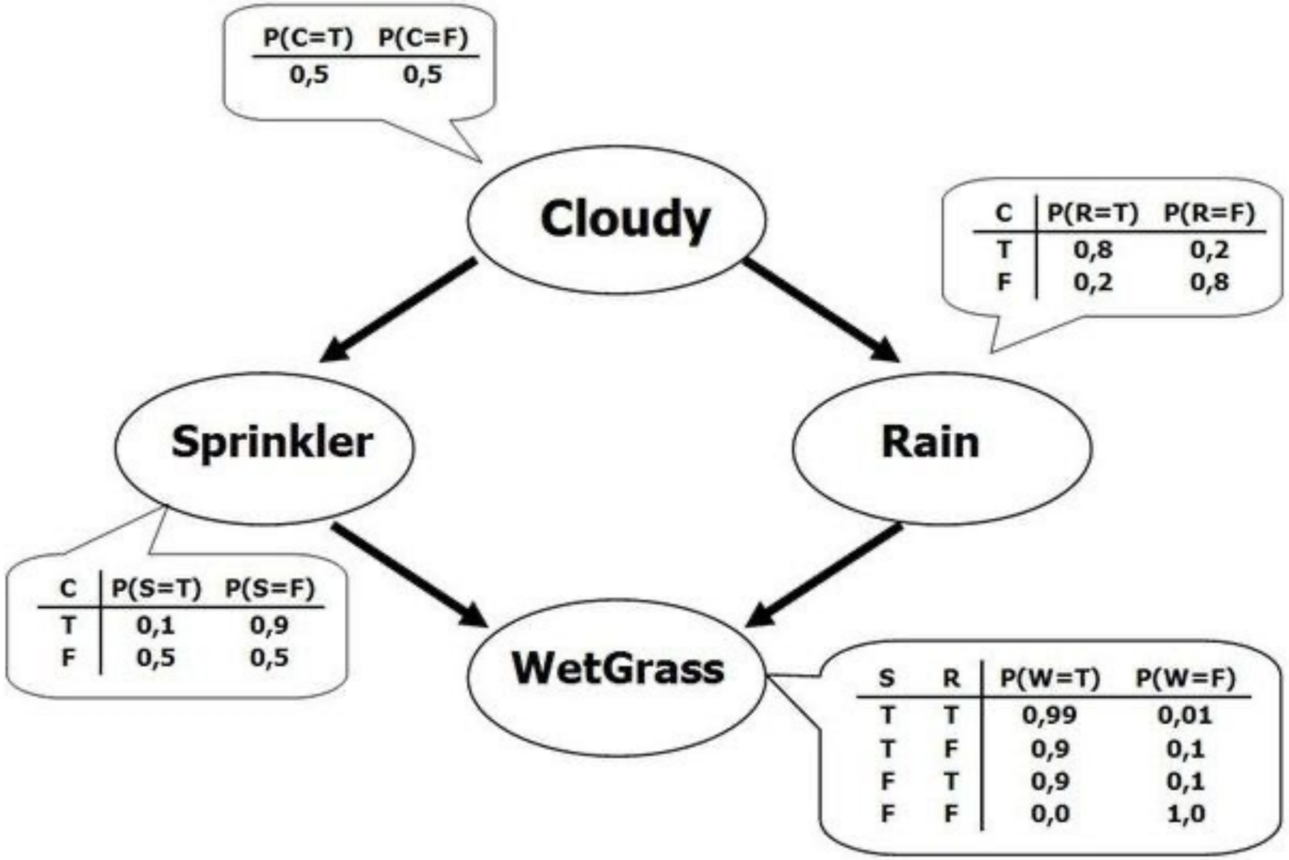
2026-1-22

Pascal Poupart
David R. Cheriton School of Computer Science

UNIVERSITY OF WATERLOO

Bayesian Networks (BN)

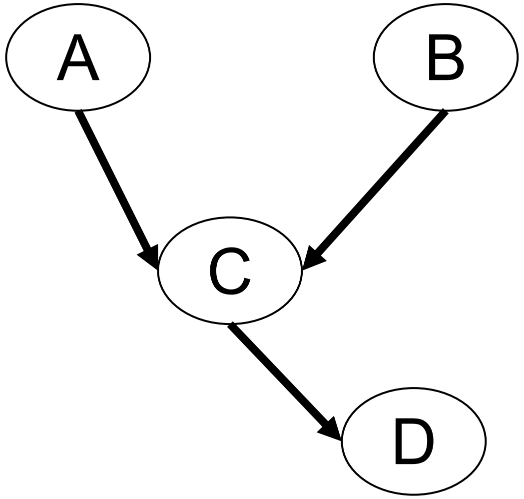
- *Graphical representation* of the direct dependencies over a set of variables + a set of *conditional probability tables (CPTs)* quantifying the strength of those influences.



- A BN over variables $\{X_1, X_2, \ldots, X_n\}$ consists of:
  - a DAG whose nodes are the variables
  - a set of CPTs ($\Pr(Xi \mid Parents(Xi))$) for each X_i

UNIVERSITY OF WATERLOO

Bayesian Networks



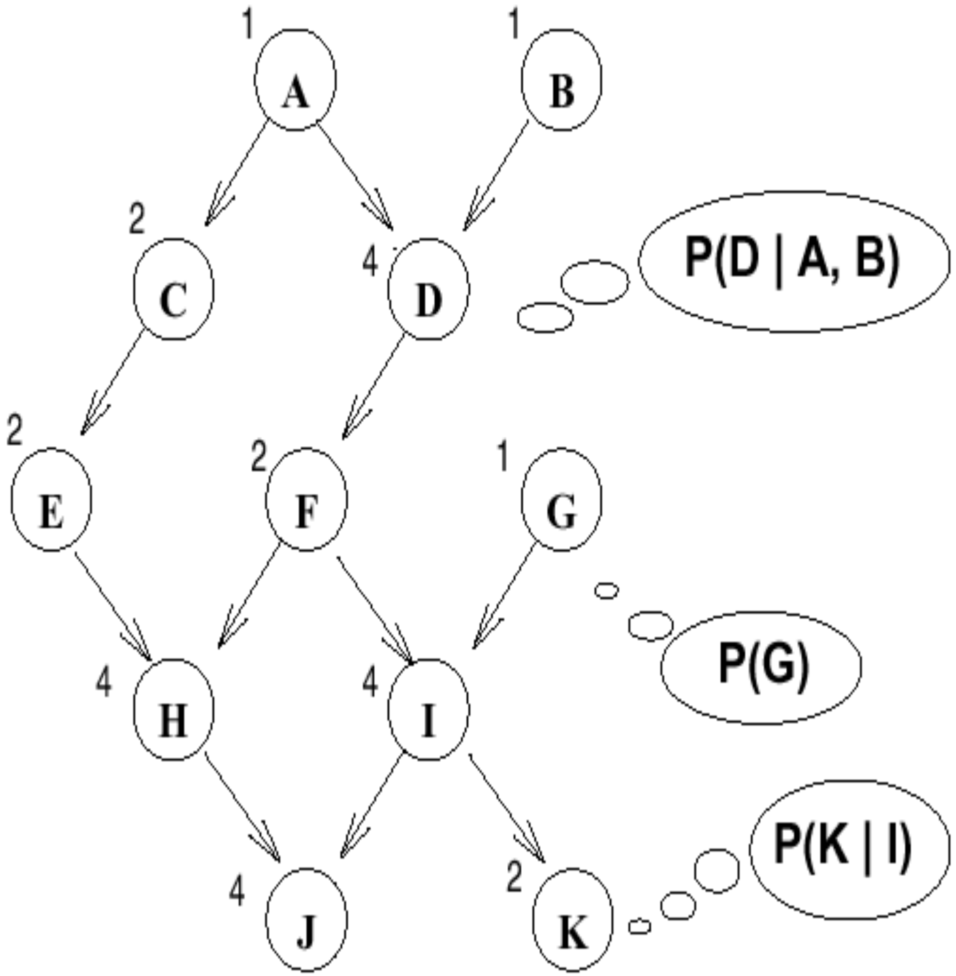
- Also known as
  - Belief networks
  - Probabilistic networks

- Key notions
  - parents of a node: $Par(X_i)$
  - children of node
  - descendants 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

$Parents(C) = \{A, B\}$
$Children(A) = \{C\}$
$Descendents(B) = \{C, D\}$
$Ancestors\{D\} = \{A, B, C\}$
$Family\{C\} = \{C, A, B\}$

UNIVERSITY OF WATERLOO

An Example Bayes Net

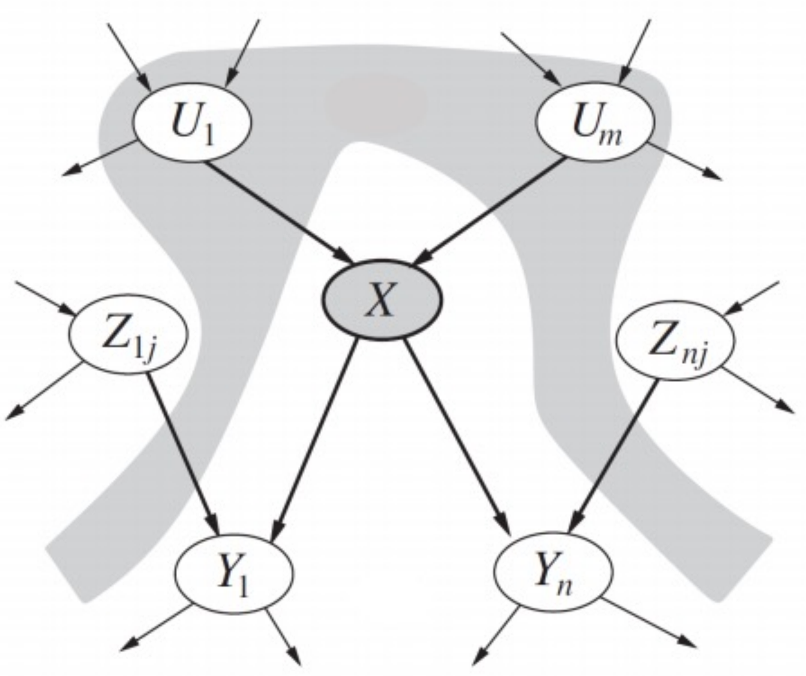


- A few CPTs are "shown"
- Explicit joint requires $2^{11} - 1 = 2047$ parameters
- BN requires only 27 params (the number of entries for each CPT is listed)

UNIVERSITY OF WATERLOO

Semantics of a Bayes Net

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



$$\Pr(X_i \mid S \cup Par(X_i)) = \Pr(X_i \mid Par(X_i))$$

for any subset $S \subseteq NonDescendants(X_i)$

Semantics of Bayes Nets

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

- By the chain rule, we have:

$$\Pr(x_1, x_2, \ldots, x_n)$$

$$= \Pr(x_n | x_{n-1}, \ldots, x_1) \Pr(x_{n-1} | x_{n-2}, \ldots, x_1) \ldots \Pr(x_1)$$

$$= \Pr(x_n | Par(x_n)) \Pr(x_{n-1} | Par(x_{n-1})) \ldots \Pr(x_1)$$

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

Constructing a Bayes Net

- Given any distribution over variables $X_1, X_2, \dots, X_n$, we can construct a Bayes net that faithfully represents that distribution.

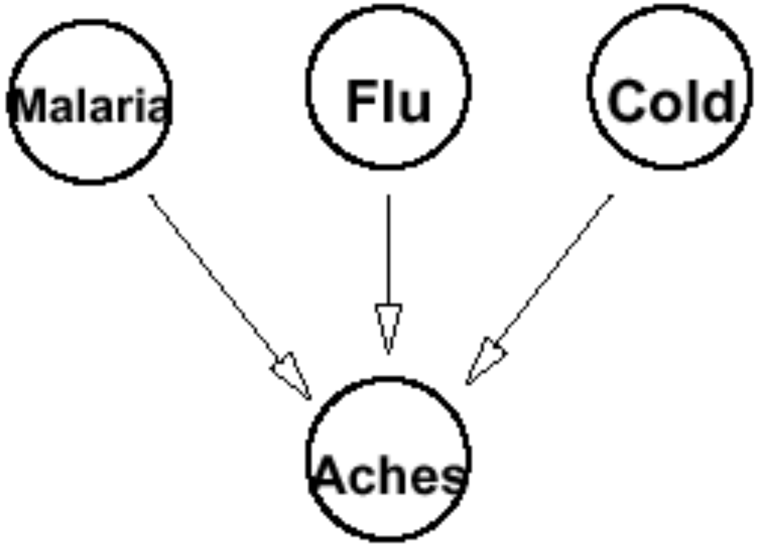
Take any ordering of the variables (say, the order given), and go through the following procedure for X_n down to X_1.

- Let $Par(X_n)$ be any subset $S \subseteq \{X_1, \dots, X_{n-1}\}$ such that X_n is independent of $\{X_1, \dots, X_{n-1}\} - S$ given S. Such a subset must exist (convince yourself).
- Then determine the parents of X_{n-1} in the same way, finding a similar $S \subseteq \{X_1, \dots, X_{n-2}\}$, and so on.

In the end, a DAG is produced and the BN semantics must hold by construction.

Causal Intuitions

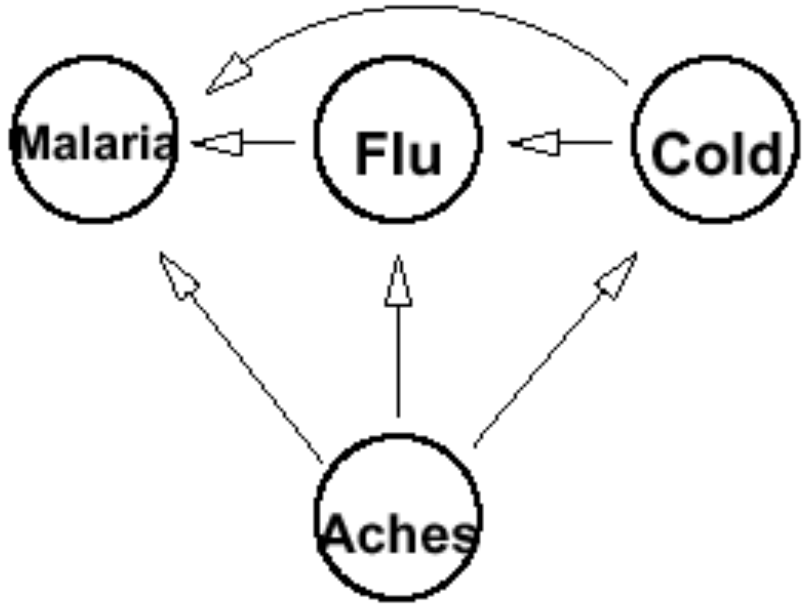
- The construction of a BN is simple
  - works with arbitrary orderings of variable set
  - but some orderings are much better than others!
  - generally, if ordering/dependence structure reflects causal intuitions, a more natural, compact BN results



- In this BN, we used the ordering Mal, Cold, Flu, Aches to build BN for joint distribution P
  - Variable can only have parents that come earlier in the ordering

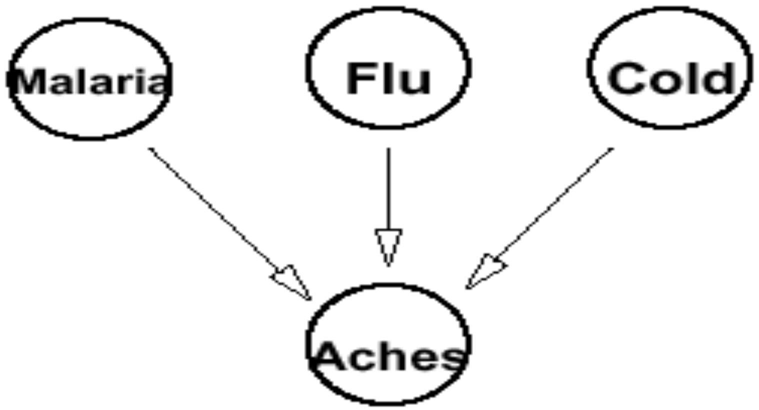
Causal Intuitions

- Suppose we build the BN for distribution P using the opposite ordering
  - i.e., we use ordering Aches, Cold, Flu, Malaria
  - resulting network is more complicated!

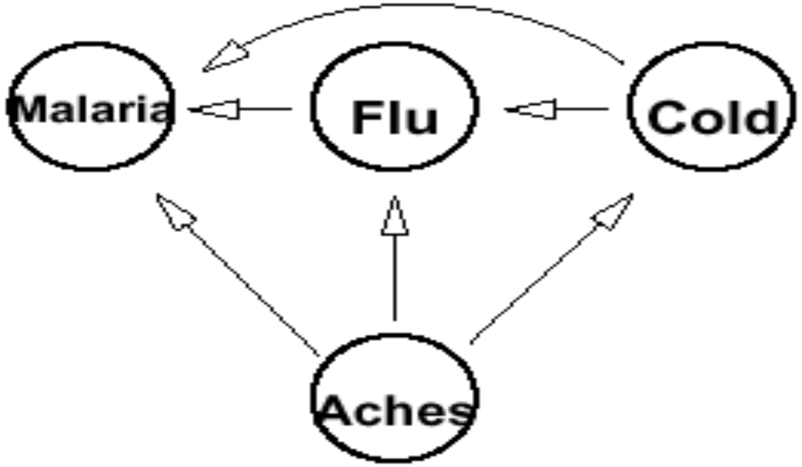


- Mal depends on Aches; but it also depends on Cold, Flu *given* Aches
  - Cold, Flu explain away Mal given Aches
- Flu depends on Aches; but also on Cold *given* Aches
- Cold depends on Aches

Compactness



1+1+1+8=11 numbers



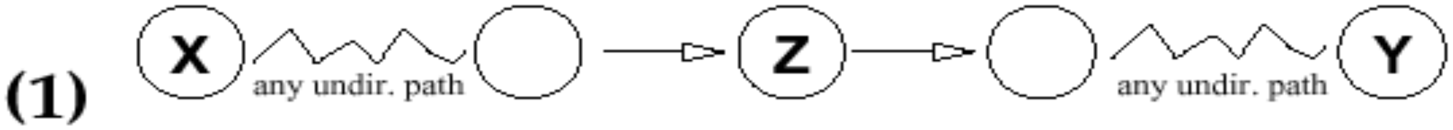
1+2+4+8=15 numbers

In general, if each random variable is directly influenced by at most k others, then each CPT will be at most 2^k. Thus, the entire network of n variables is specified by $n2^k$.

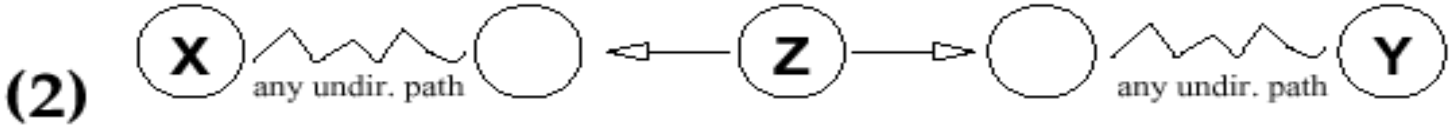
Testing Independence

- Given BN, how do we determine if two variables X, Y are independent (given evidence E)?
  - we use a (simple) graphical property
- **D-separation**: A set of variables $\boldsymbol{E}$ *d-separates* X and Y if it *blocks every undirected path* in the BN between X and Y.
- X and Y are conditionally independent given evidence $\boldsymbol{E}$ if $\boldsymbol{E}$ d-separates X and Y
  - Thus, BN gives us an easy way to tell if two variables are independent (set $E = \emptyset$) or cond. independent

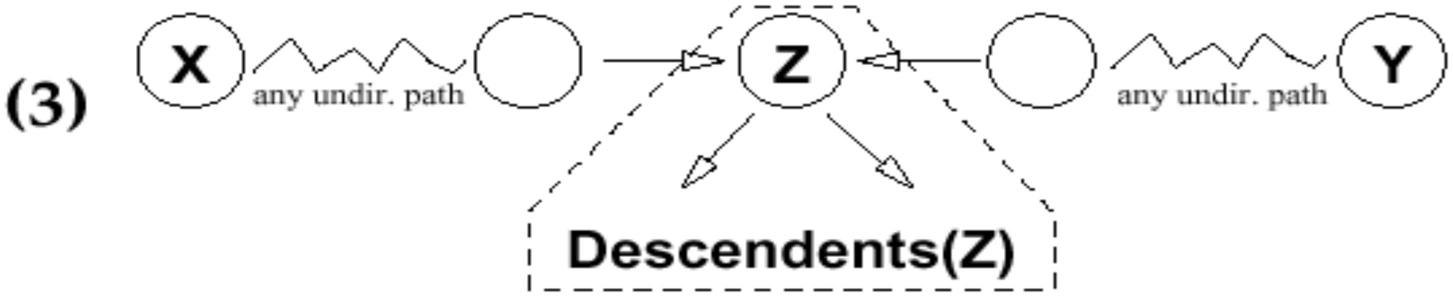
Blocking: Graphical View



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



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



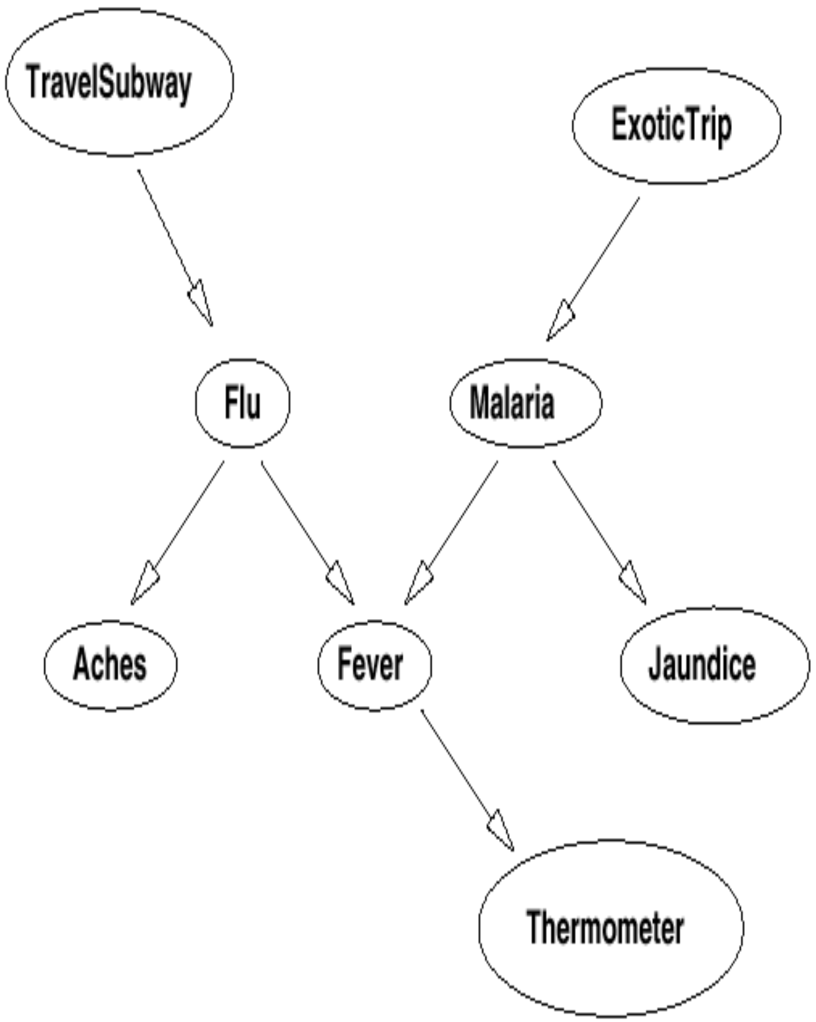
If Z is ***not*** in evidence and ***no*** descendent of Z is in evidence, then the path between X and Y is blocked

UNIVERSITY OF WATERLOO

Blocking in D-Separation

- Let P be an undirected path from X to Y in a BN. Let $\boldsymbol{E}$ be an evidence set. We say $\boldsymbol{E}$ *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 $Z \in \boldsymbol{E}$; or
  - **Case 2:** both arcs on P leave Z, and $Z \in \boldsymbol{E}$; or
  - **Case 3:** both arcs on P enter Z and *neither Z, nor any of its descendants*, are in $\boldsymbol{E}$.

D-Separation: Intuitions



1. Subway and Thermometer?
2. Aches and Fever?
3. Aches and Thermometer?
4. Flu and Malaria?
5. Subway and ExoticTrip?

D-Separation: Intuitions

- Subway and Thermometer are dependent; but are independent given Flu (since Flu blocks the only path)
- Aches and Fever are dependent; but are independent given Flu (since Flu blocks the only path). Similarly for Aches and Thermometer (dependent, but independent given Flu).
- Flu and Mal are independent (given no evidence): Fever blocks the path, since it is *not in evidence*, nor is its descendant Thermometer. Flu, Malaria are dependent given Fever (or given Thermometer): nothing blocks path now.
- Subway, ExoticTrip are independent; they are dependent given Thermometer; they are independent given Thermometer and Malaria. This for exactly the same reasons for Flu/Malaria above.

UNIVERSITY OF WATERLOO

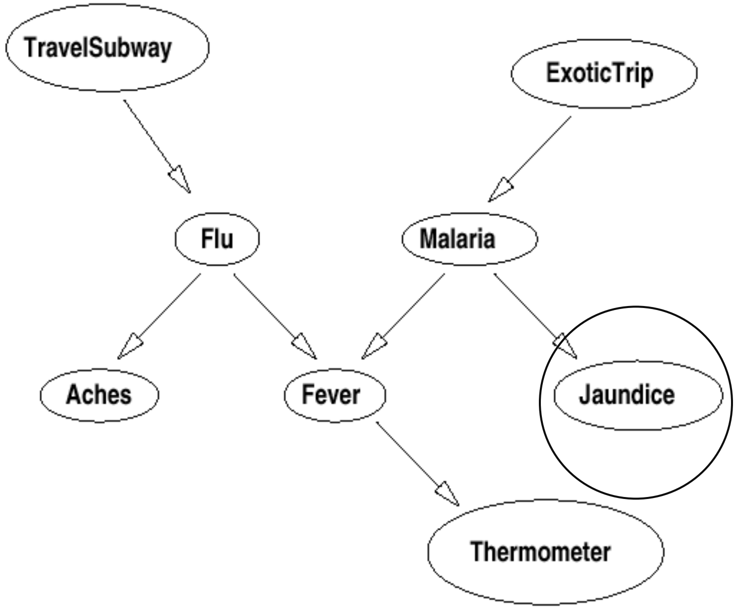
Inference in Bayes Nets

- The independence sanctioned by D-separation (and other methods) allows us to compute prior and posterior probabilities quite effectively.

- We'll look at a few simple examples to illustrate. We'll focus on networks without *loops*. (A loop is a cycle in the underlying *undirected* graph. Recall the directed graph has no cycles.)

Simple Forward Inference (Chain)

- Computing marginal requires simple forward "propagation" of probabilities



$P(J)=\Sigma_{M,ET}\ P(J,M,ET)$
(marginalization)

$P(J)=\Sigma_{M,ET}\ P(J|M,ET)P(M|ET)P(ET)$
(chain rule)

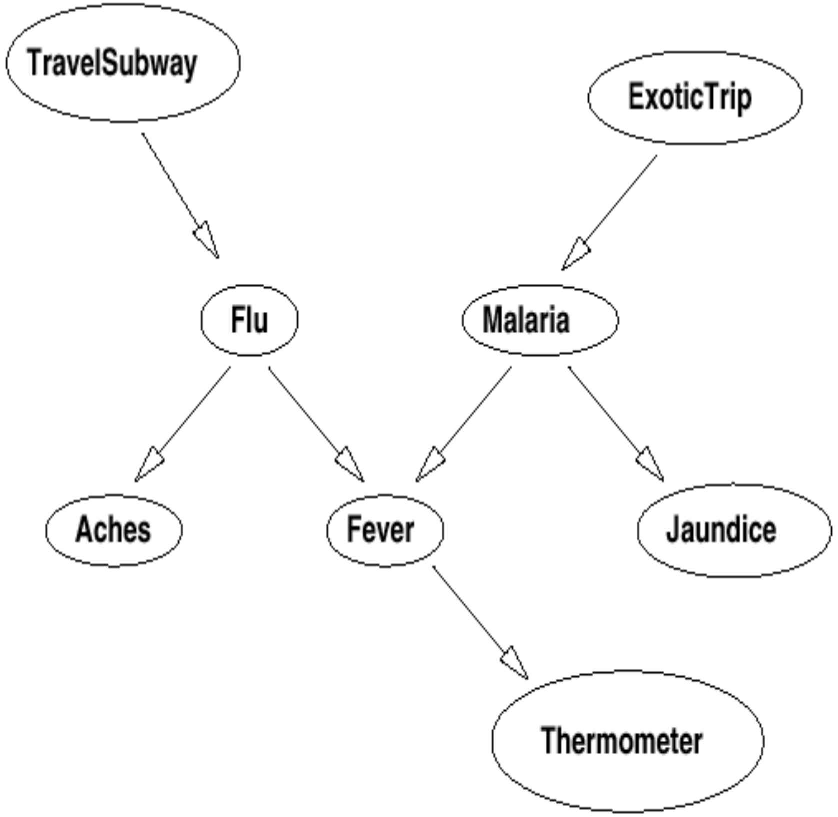
$P(J)=\Sigma_{M,ET}\ P(J|M)P(M|ET)P(ET)$
(conditional independence)

$P(J)=\Sigma_{M}P(J|M)\Sigma_{ET}P(M|ET)P(ET)$
(distribution of sum)

Note: all (final) terms are CPTs in the BN
Note: only ancestors of J considered

Simple Forward Inference (Chain)

- Same idea applies when we have "upstream" evidence



$$P(J|ET) = \Sigma_M P(J,M|ET)$$

(marginalisation)

$$P(J|ET) = \Sigma_M P(J|M,ET)\ P(M|ET)$$

(chain rule)

$$P(J|ET) = \Sigma_M P(J|M)\ P(M|ET)$$

(conditional independence)

Simple Backward Inference

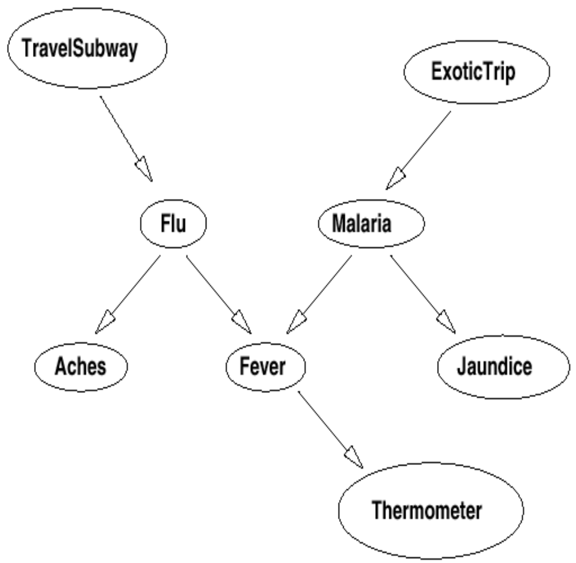
- When evidence is downstream of query variable, we must reason "backwards." This requires the use of Bayes rule:

$P(ET \mid j) = \alpha P(j \mid ET) P(ET)$

$= \alpha \Sigma_M P(j,M|ET) P(ET)$

$= \alpha \Sigma_M P(j|M,ET) P(M|ET) P(ET)$

$= \alpha \Sigma_M P(j|M) P(M|ET) P(ET)$



- First step is just Bayes rule
  - normalizing constant α is 1/P(j); but we needn't compute it explicitly if we compute P(ET | j) for each value of ET: we just add up terms P(j | ET) P(ET) for all values of ET (they sum to P(j))

UNIVERSITY OF WATERLOO

Variable Elimination

- The intuitions in the above examples give us a simple inference algorithm for networks without loops: the *polytree* algorithm.
- Instead, we'll look at a more general algorithm that works for general BNs; but the polytree algorithm will be a special case.
- The algorithm, ***variable elimination***, simply applies the summing out rule repeatedly.
  - To keep computation simple, it exploits the independence in the network and the ability to distribute sums inward

Factors

- A function $f(X_1, X_2,..., X_k)$ is also called a *factor*. We can view this as a table of numbers, one for each instantiation of the variables $X_1, X_2,..., X_k$.
  - A tabular representation of a factor is exponential in k
- Each CPT in a Bayes net is a factor:
  - e.g., Pr(C|A,B) is a function of three variables, A, B, C
- Notation: f(**X**,**Y**) denotes a factor over the variables $\mathbf{X} \cup \mathbf{Y}$. (Here **X**, **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:

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

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	0.02	a~b~c	0.08
~ab	0.4	~bc	0.2	~abc	0.28	~ab~c	0.12
~a~b	0.6	~b~c	0.8	~a~bc	0.12	~a~b~c	0.48

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(**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		

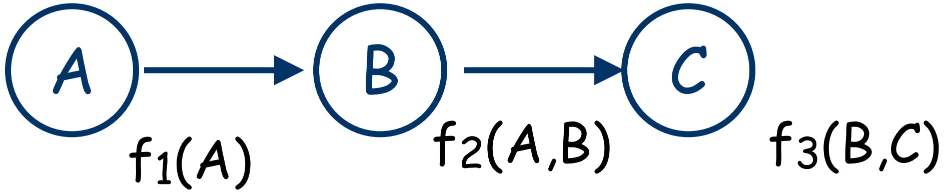
Restricting a Factor

- Let $f(X,\mathbf{Y})$ be a factor with variable X ($\mathbf{Y}$ is a set)
- We *restrict* factor 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

- Computing prior probability of query var X can be seen as applying these operations on factors



- $P(C) = \Sigma_{A,B}\, P(C|B)\, P(B|A)\, P(A)$

$$= \Sigma_B\, P(C|B)\, \Sigma_A\, P(B|A)\, P(A)$$

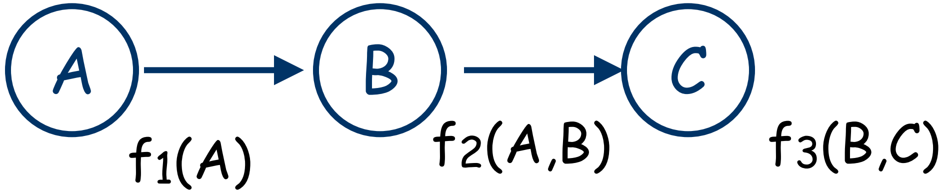
$$= \Sigma_B\, f_3(B,C)\, \Sigma_A\, f_2(A,B)\, f_1(A)$$

$$= \Sigma_B\, f_3(B,C)\, f_4(B) = f_5(C)$$

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

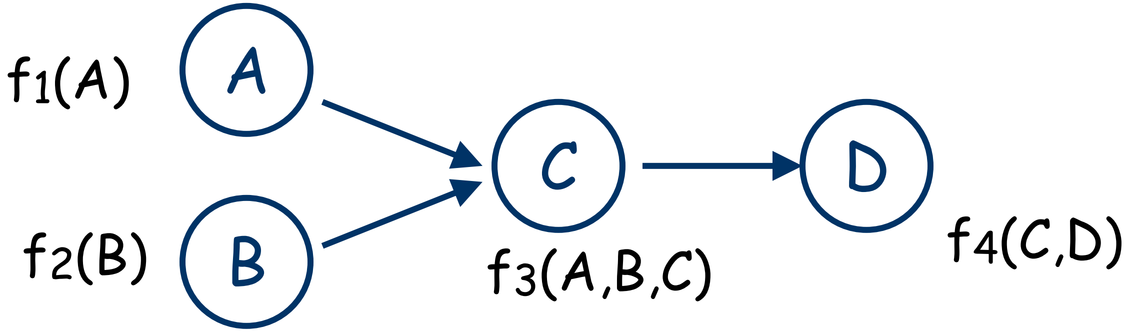
Variable Elimination: No Evidence

- 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				

VE: No Evidence (Example 2)



$$P(D) = \Sigma_{A,B,C}\ P(D|C)\ P(C|B,A)\ P(B)\ P(A)$$
$$= \Sigma_C\ P(D|C)\ \Sigma_B\ P(B)\ \Sigma_A\ P(C|B,A)\ P(A)$$
$$= \Sigma_C\ f_4(C,D)\ \Sigma_B\ f_2(B)\ \Sigma_A\ f_3(A,B,C)\ f_1(A)$$
$$= \Sigma_C\ f_4(C,D)\ \Sigma_B\ f_2(B)\ f_5(B,C)$$
$$= \Sigma_C\ f_4(C,D)\ f_6(C)$$
$$= f_7(D)$$

Define new factors: $f_5(B,C)$, $f_6(C)$, $f_7(D)$, in the obvious way

UNIVERSITY OF WATERLOO

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” factor mentioning X
  - apply operations “inside out”, repeatedly eliminating and creating new factors (note that each step/removal of a sum eliminates one variable)

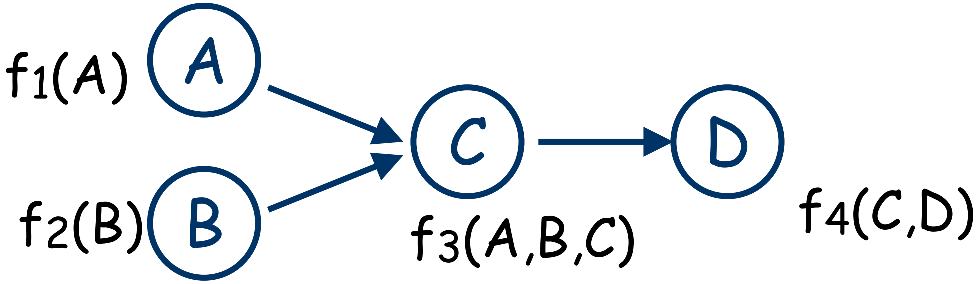
Variable Elimination Algorithm

- Given query var Q, remaining vars **Z**. Let F be the set of factors corresponding to CPTs for {Q} ∪ **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 factor $g_j = \Sigma_{Z_j} f_1 \times f_2 \times \ldots \times f_k$,
   where the f_i are the factors in F that include Z_j
   (b) Remove the factors f_i (that mention Z_j) 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 P(Q)

VE: Example 2 again

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 $P(D)$