CMSC 671
Fall 2010

Tue 11/02/10

Probabilistic Reasoning
Chapter 14.1-14.5

Prof. Laura Zavala, laura.zavala@umbc.edu, ITE 373, 410-455-8775

Bayesian Networks

- Independence and conditional independence among variables can greatly reduce the full joint distribution
- Bayesian Networks
  - A structure used to represent the dependencies among variables

Bayesian Belief Networks (BNs)

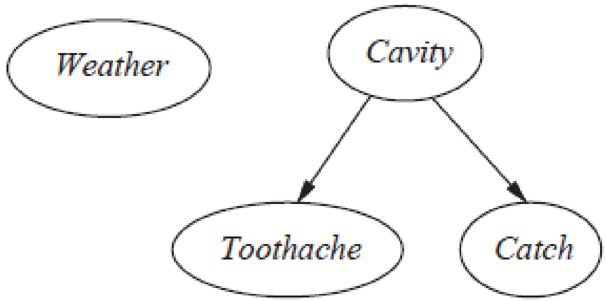
- Definition: **BN = (DAG, CPD)**
  - **DAG**: directed acyclic graph (BN's **structure**)
    - **Nodes**: random variables (typically binary or discrete, but methods also exist to handle continuous variables)
    - **Arcs**: indicate probabilistic dependencies between nodes (*lack* of link signifies conditional independence)
  - **CPD**: conditional probability distribution (BN's **parameters**)
    - Conditional probabilities at each node, usually stored as a table (conditional probability table, or **CPT**)

      $P(x_i \mid \pi_i)$ where π_i is the set of all parent nodes of x_i

  - Root nodes are a special case – no parents, so just use priors in CPD:

    $$\pi_i = \varnothing, \text{ so } P(x_i \mid \pi_i) = P(x_i)$$

Example BN

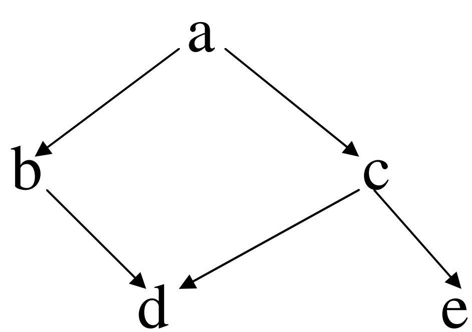


Toothache: boolean variable indicating whether the patient has a toothache
Cavity: boolean variable indicating whether the patient has a cavity
Catch: whether the dentist's probe catches in the cavity

- *Weather* is independent of all the other variables
- *Catch* is conditionally independent of *Toothache* given *Cavity*
  - P(Catch | Toothache, Cavity) = P(Catch | Cavity)
- Likewise, *Toothache* is conditionally independent of *Catch* given *Cavity*
  - P(Toothache | Catch, Cavity) = P(Toothache | Cavity)
- Equivalent statement:
  - P(Toothache, Catch | Cavity) = P(Toothache | Cavity) P(Catch | Cavity)
- Cavity is a direct cause of *Toothache* and *Catch*
- No direct causal relationship exists between *Toothache* and *Catch*

Example BN with CPTs

P(A) = 0.001



P(B|A) = 0.3
P(B|¬A) = 0.001

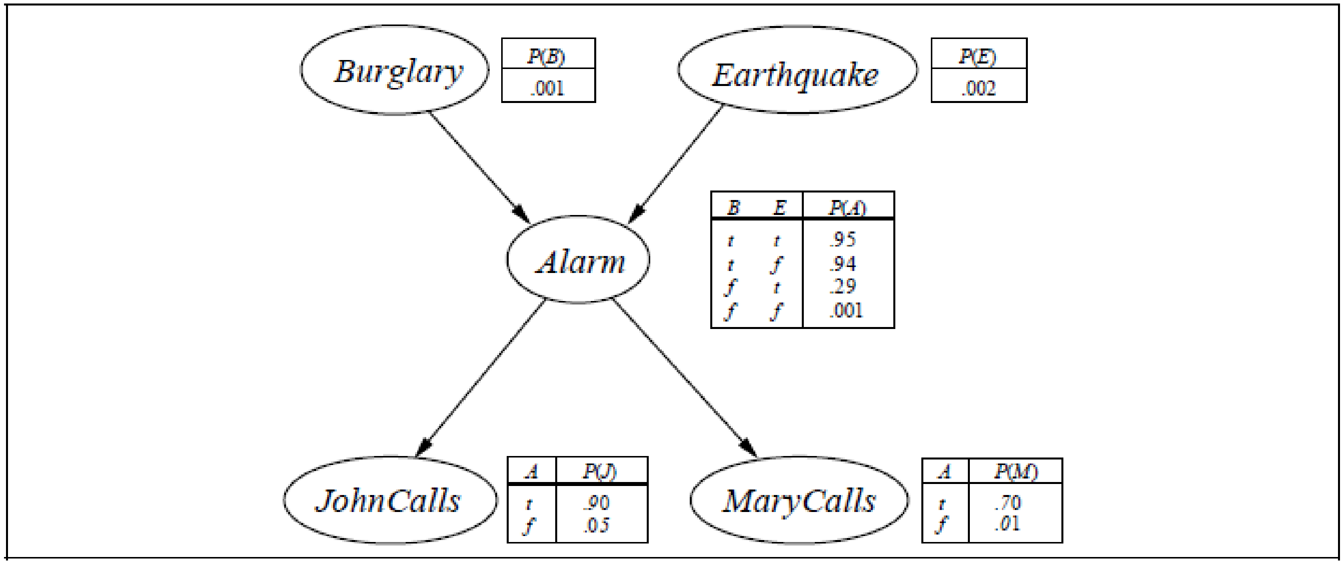
P(C|A) = 0.2
P(C|¬A) = 0.005

P(D|B,C) = 0.1
P(D|B,¬C) = 0.01
P(D|¬B,C) = 0.01
P(D|¬B,¬C) = 0.00001

P(E|C) = 0.4
P(E|¬C) = 0.002

Note that we only specify P(A) etc., not P(¬A), since they have to add to one

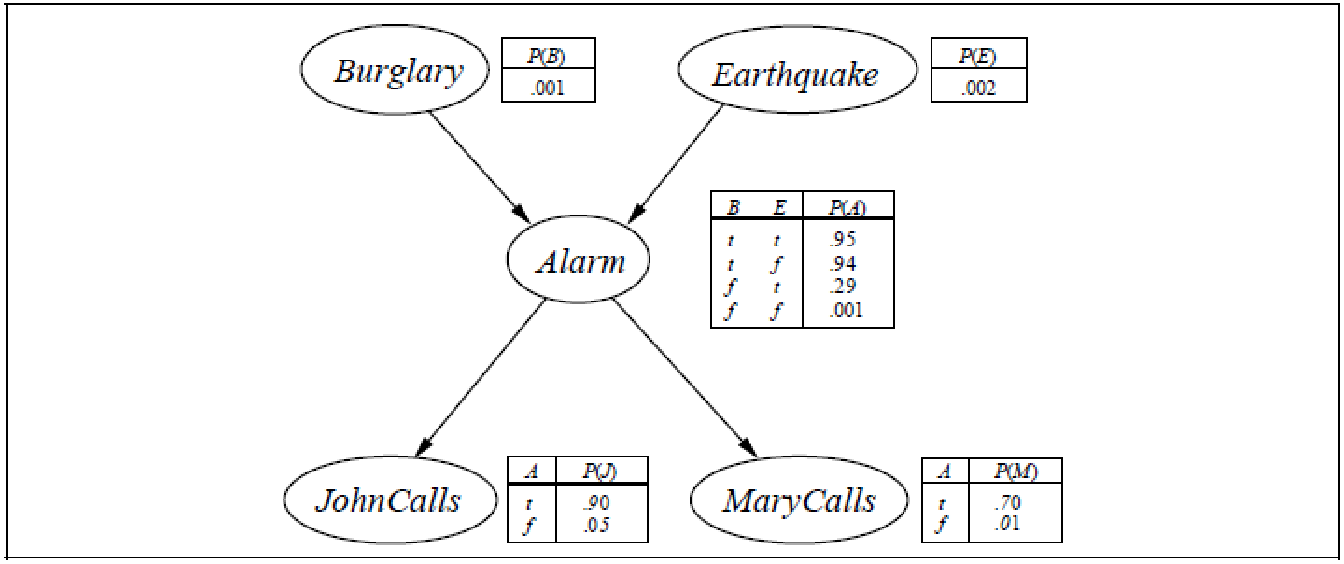
Example 2: BN with CPTs (1)



- Your neighbors Mary and John have promised to call you to work whenever they hear the alarm
- John sometimes confuses the phone ringing with the alarm
- Mary likes to hear loud music and sometimes fails to hear the alarm

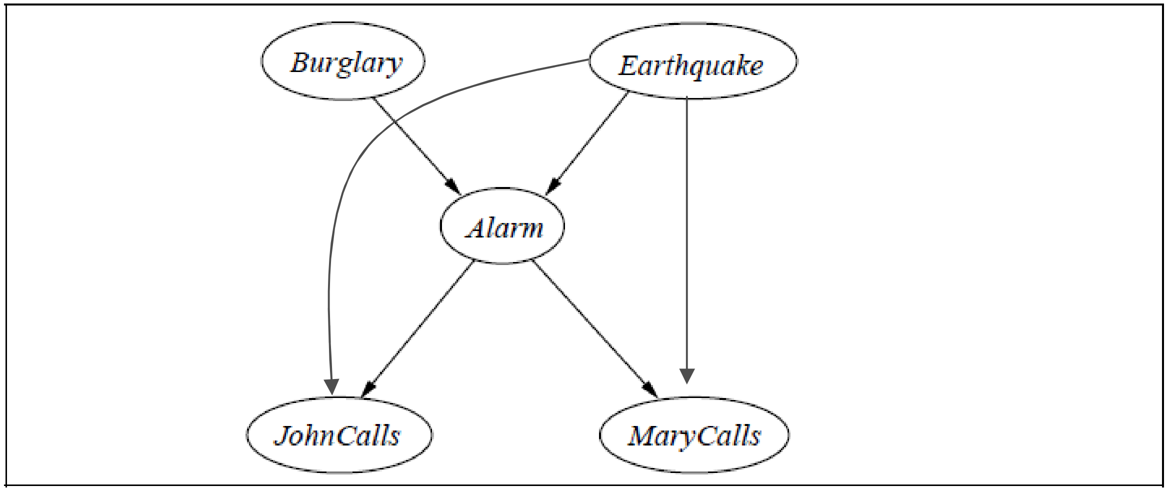
- Given the evidence of who has or has not called, we want to estimate *P(burglary)*

Example 2: BN with CPTs (2)



- The probabilities actually summarize a potentially infinite set of circumstances in which the alarm might fail to go off or John or Mary might fail to call and report it.
- In this way we can deal with a very large world, at least approximately.

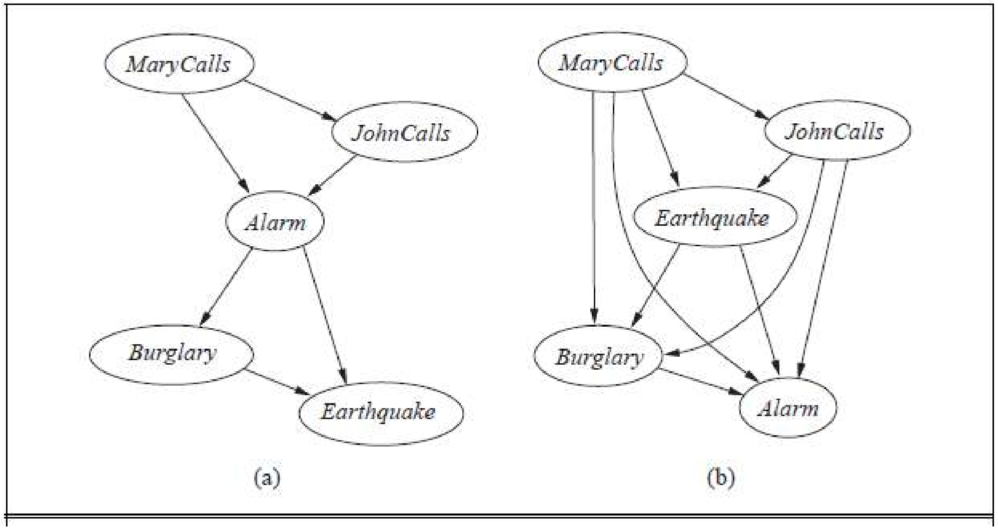
Tenuous dependencies



- If there is an earthquake, John and Mary may not call even if they heard the alarm …

- May not be worth adding the complexity in the network for the small gain in accuracy
  - As we come closer to a fully connected network, the conditional probability tables are the same as the joint distribution

Ordering Matters

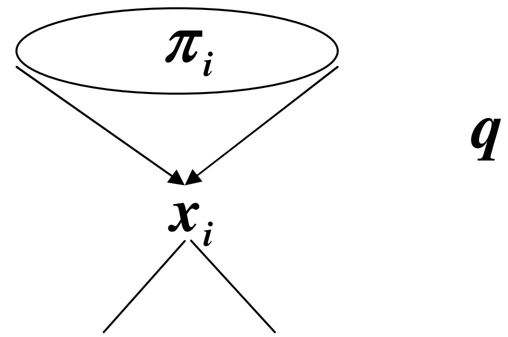


- Given an ordering, the parents of a variable is the subset of its predecessors that make it independent of all its other predecessors
- The ordering makes a big difference to the structure of the network
- (a) Order: Mary Calls, John Calls, Alarm, Burglary, Earthquake

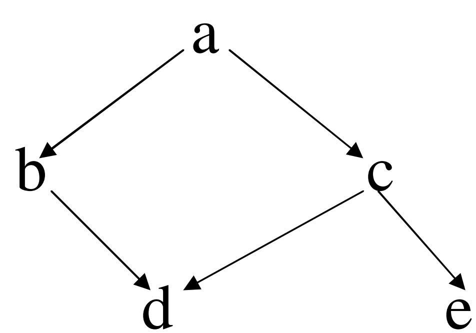
Conditional independence and chaining

- Conditional independence assumption
  - $P(x_i \mid \pi_i, q) = P(x_i \mid \pi_i)$
    where q is any set of variables (nodes) other than x_i and its successors
  - π_i **blocks influence** of other nodes on x_i and its successors (q influences x_i only through variables in π_i)
  - With this assumption, the complete joint probability distribution of all variables in the network can be represented by (recovered from) local CPDs by chaining these CPDs:

$$P(x_1, \ldots, x_n) = \Pi_{i=1}^{n} P(x_i \mid \pi_i)$$



Chaining: Example



Computing the joint probability for all variables is easy:

P(a, b, c, d, e)

= P(e | a, b, ***c***, d) P(a, b, c, d) by the product rule

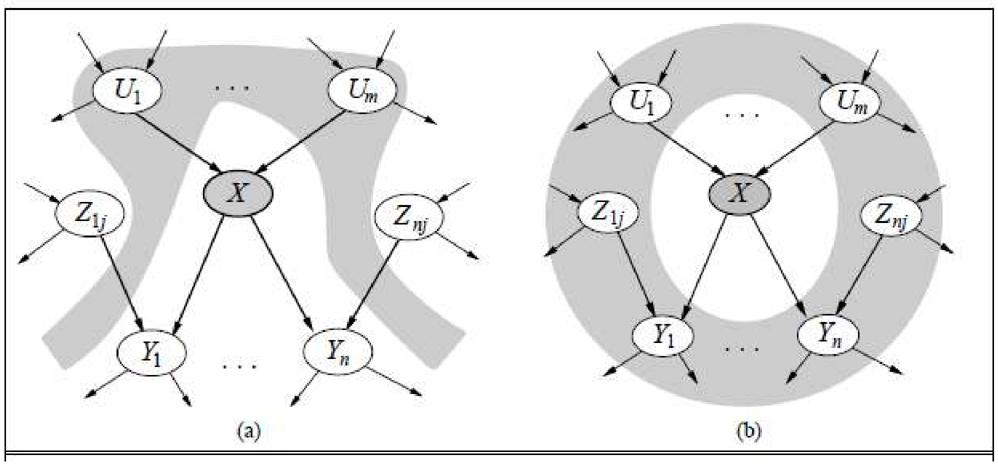
= P(e | c) P(a, b, c, d) by cond. indep. assumption

= P(e | c) P(d | a, ***b, c***) P(a, b, c)

= P(e | c) P(d | b, c) P(c | ***a,*** b) P(a, b)

= P(e | c) P(d | b, c) P(c | a) P(b | a) P(a)

Topological semantics



- A node is **conditionally independent** of its **non-descendants** given its **parents**
- A node is **conditionally independent** of all other nodes in the network given its parents, children, and children's parents (also known as its **Markov blanket**)

Representational extensions

- Even though they are more compact than the full joint distribution, CPTs for large networks can require a large number of parameters ($O(2^k)$ where k is the branching factor of the network)
- Compactly representing CPTs
  - Deterministic relationships
  - Noisy-OR
  - Noisy-MAX
- Adding continuous variables
  - Discretization
  - Use density functions (usually mixtures of Gaussians) to build hybrid Bayesian networks (with discrete *and* continuous variables)

Inference in Bayesian Networks

Inference tasks

- **Simple queries:** Compute posterior distribution $P(X_i \mid E=e)$
  - E.g., P(NoGas | Gauge=empty, Lights=on, Starts=false)
  - P(Burglary | JohnCalls=true, MaryCalls=true) = <0.284, 0.716>
- **Conjunctive queries:**
  - $P(X_i, X_j \mid E=e) = P(X_i \mid e=e)\, P(X_j \mid X_i, E=e)$
- **Optimal decisions:** *Decision networks* include utility information; probabilistic inference is required to find P(outcome | action, evidence)
- **Value of information:** Which evidence should we seek next?
- **Sensitivity analysis:** Which probability values are most critical?

Approaches to inference

- Exact inference
  - **Enumeration**
  - **Variable elimination**
  - Clustering / join tree algorithms
- Approximate inference
  - **Stochastic simulation / sampling methods**
  - **Markov chain Monte Carlo methods**
  - Genetic algorithms
  - Neural networks
  - Simulated annealing
  - Mean field theory

Direct inference with BNs

- Instead of computing the joint, suppose we just want the probability for *one* variable
- Exact methods of computation:
  - **Enumeration**
  - **Variable elimination**
  - Join trees: get the probabilities associated with every query variable

Inference by enumeration

- Add all of the terms (atomic event probabilities) from the full joint distribution
- If **E** are the evidence (observed) variables and **Y** are the other (unobserved or hidden) variables, then:

  $$P(X|\mathbf{e}) = \alpha\ P(X, \mathbf{r}) = \alpha\ \Sigma_y\ P(X, \mathbf{e}, \mathbf{y})$$

- Each P(X, **E**, **Y**) term can be computed using the chain rule

- Computationally expensive!

Inference by enumeration

- P(Burglary | JohnCalls=true, MaryCalls=true)
- Hidden variables
  - *Earthquake* and *Alarm*
- $P(B|j,m) = \alpha\ P(B, \mathbf{j,m}) = \alpha \sum_e \sum_a P(B, \mathbf{j}, \mathbf{m,e,a})$

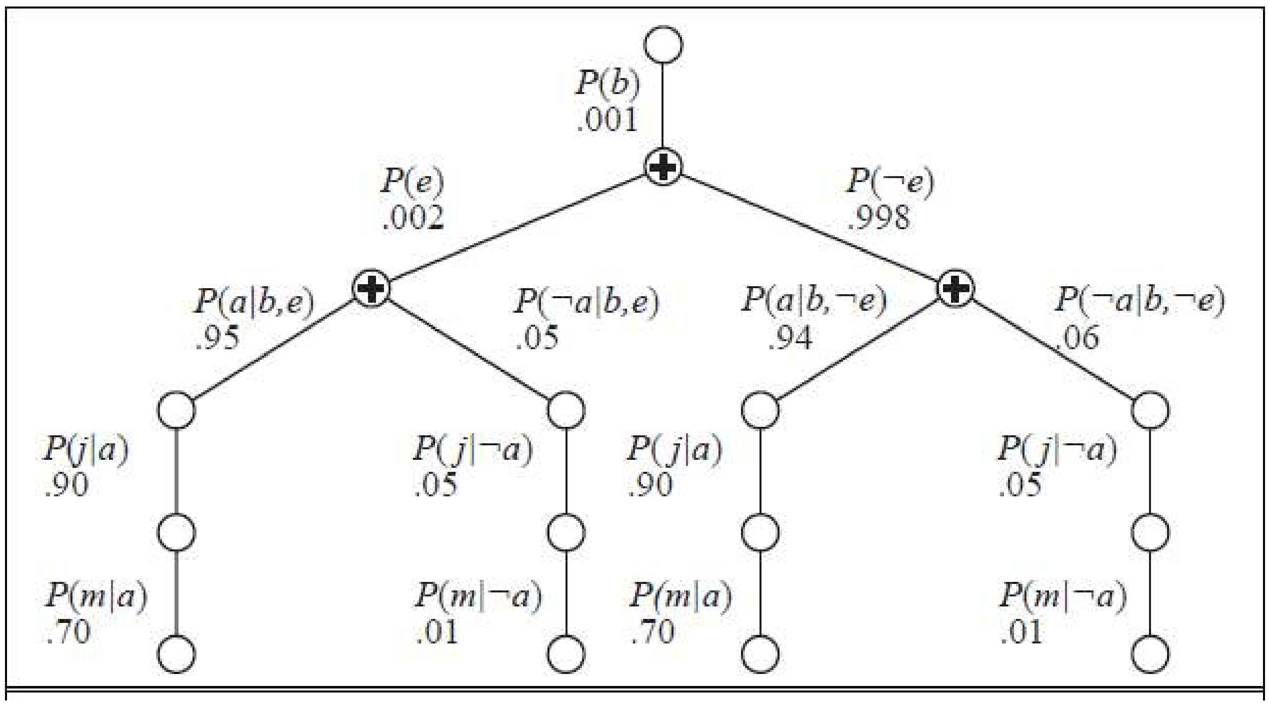
  $= \alpha \sum_e \sum_a P(b)P(e)P(a|b,e)P(j|a)P(m|a)$

  $= \alpha P(b) \sum_e P(e) \sum_a P(a|b,e)P(j|a)P(m|a)$
- We loop through the variables in order, multiplying CPT entries as we go

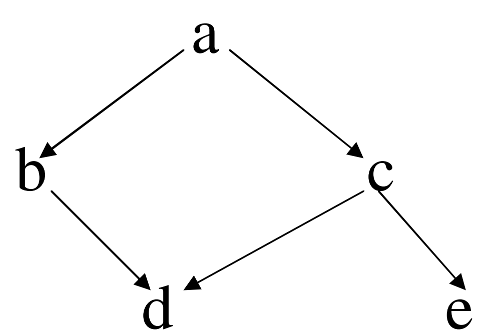
  $= <0.284, 0.716>$

Inference by enumeration

- P(Burglary | JohnCalls=true, MaryCalls=true)



Example: Enumeration



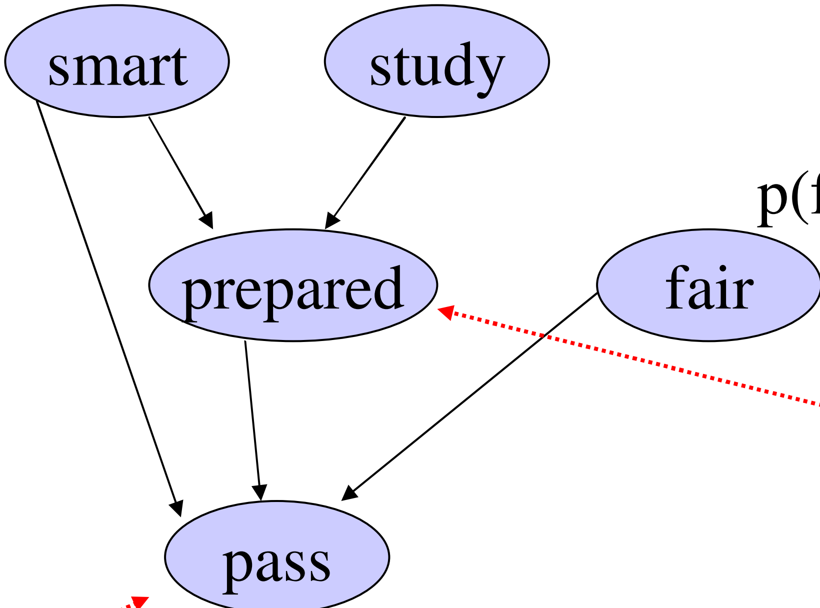
- $P(x_i) = \sum_{\pi_i} P(x_i \mid \pi_i)\, P(\pi_i)$
- Suppose we want P(D=true), and only the value of E is given as true
- $P(d|e) = \alpha \sum_{ABC} P(a, b, c, d, e)$
  $= \alpha \sum_{ABC} P(a)\, P(b|a)\, P(c|a)\, P(d|b,c)\, P(e|c)$
- With simple iteration to compute this expression, there's going to be a lot of repetition (e.g., P(e|c) has to be recomputed every time we iterate over C=true)

Exercise: Enumeration

p(smart)=.8

p(study)=.6

p(fair)=.9



p(prep|…)	smart	¬smart
study	.9	.7
¬study	.5	.1

p(pass|…)	smart		¬smart	
	prep	¬prep	prep	¬prep
fair	.9	.7	.7	.2
¬fair	.1	.1	.1	.1

Query: What is the probability that a student studied, given that they pass the exam?

Variable elimination

- Basically just enumeration, but with caching of local calculations
- Linear for polytrees (singly connected BNs)
- Potentially exponential for multiply connected BNs

$\Rightarrow$**Exact inference in Bayesian networks is NP-hard!**

Variable elimination

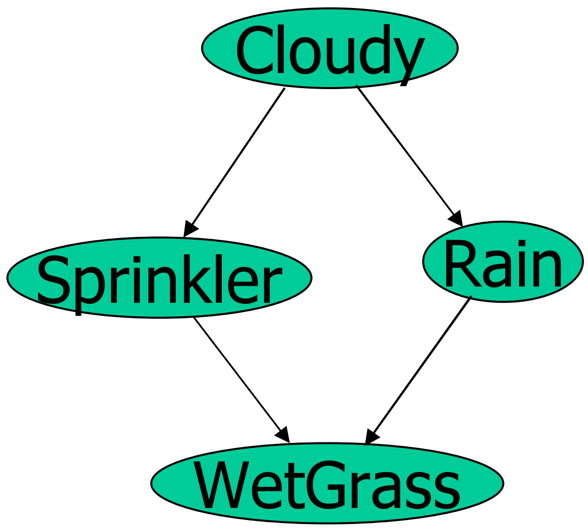
General idea:

- Write query in the form

$$P(X_n, \boldsymbol{e}) = \sum_{x_k} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i \mid pa_i)$$

- Iteratively
  - Move all irrelevant terms outside of innermost sum
  - Perform innermost sum, getting a new term
  - Insert the new term into the product

Variable elimination: Example



$$P(w) = \sum_{r,s,c} P(w \mid r,s) P(r \mid c) P(s \mid c) P(c)$$

$$= \sum_{r,s} P(w \mid r,s) \sum_{c} P(r \mid c) P(s \mid c) P(c)$$

$$f_1(r,s)$$

$$= \sum_{r,s} P(w \mid r,s) f_1(r,s)$$

Computing factors

R	S	C	P(R|C)	P(S|C)	P(C)	P(R|C) P(S|C) P(C)
T	T	T				
T	T	F				
T	F	T				
T	F	F				
F	T	T				
F	T	F				
F	F	T				
F	F	F				

R	S	$f_1(R,S) = \Sigma_c P(R\|C) P(S\|C) P(C)$
T	T	
T	F	
F	T	
F	F	

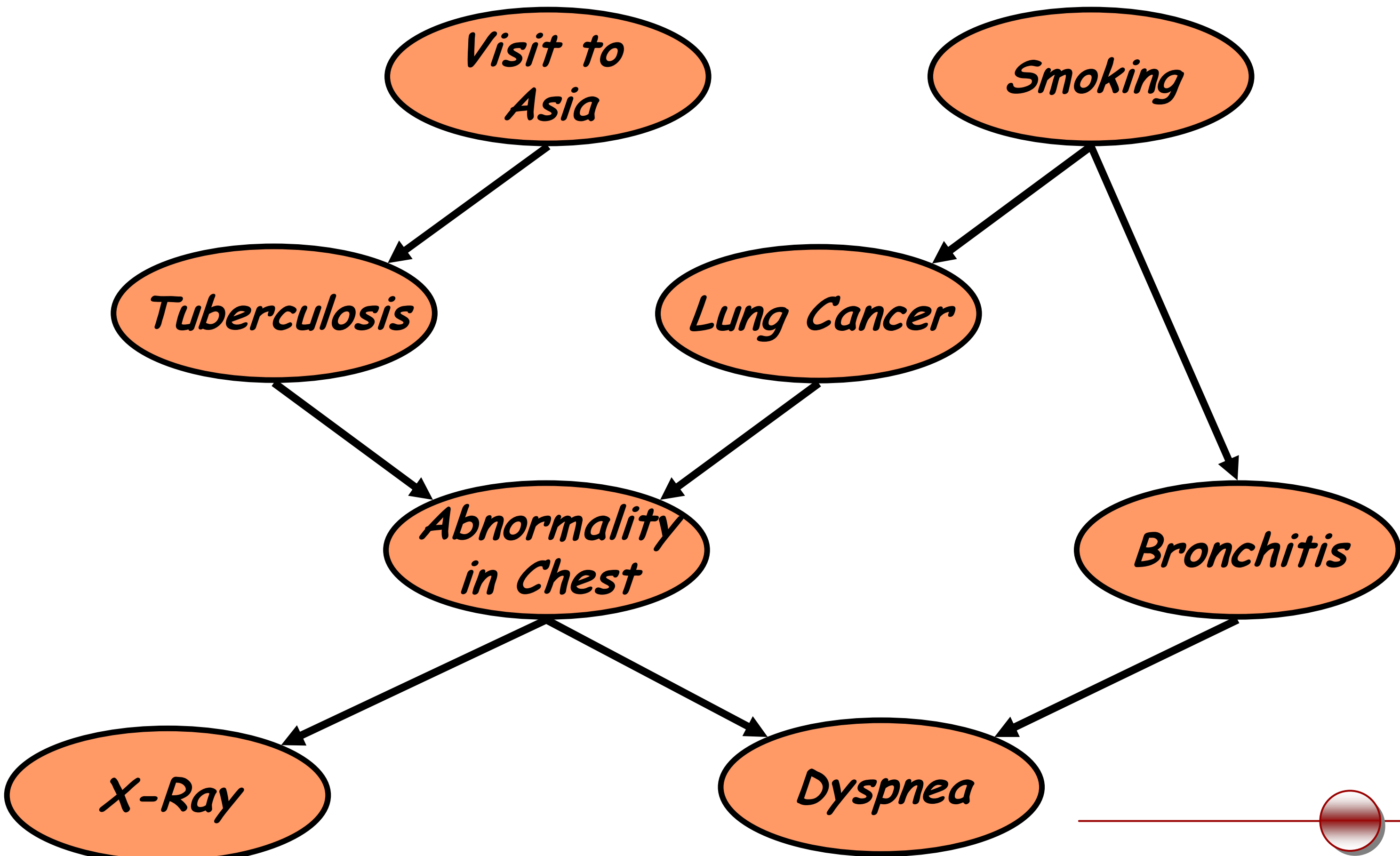
Variable elimination: Example 2

- P(Burglary | JohnCalls=true, MaryCalls=true)

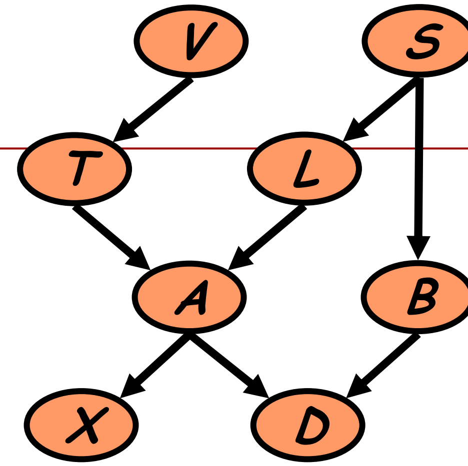
- $P(B|j,m) = \alpha \underbrace{P(b)}_{f1(B)} \sum_e \underbrace{P(e)}_{f2(E)} \sum_a \underbrace{P(a|b,e)}_{f3(A,B,E)} \underbrace{P(j|a)}_{f4(A)} \underbrace{P(m|a)}_{f5(A)}$

A more complex example

- “Asia” network:



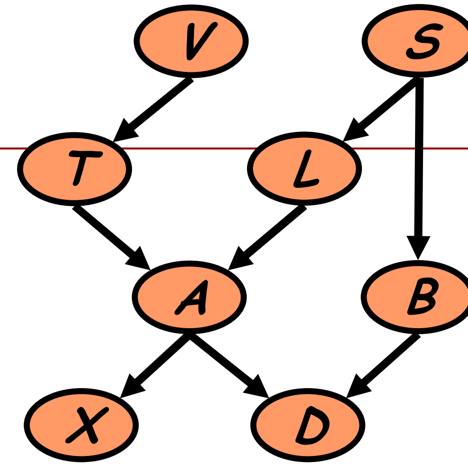
- We want to compute *P(d)*
- Need to eliminate: *v,s,x,t,l,a,b*



Initial factors

$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

- We want to compute $P(d)$
- Need to elimininate: v,s,x,t,l,a,b



Initial factors

$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

Eliminate: v

Compute: $$f_v(t) = \sum_v P(v)P(t|v)$$

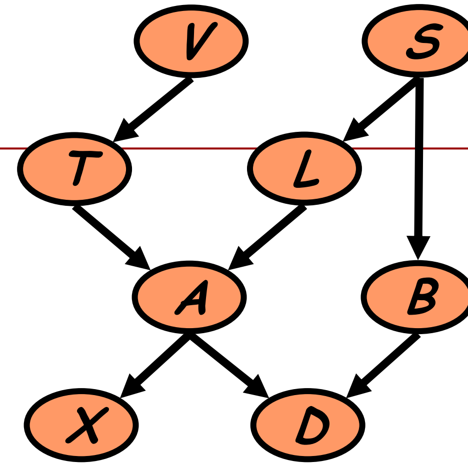
$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

Note: $f_v(t) = P(t)$

In general, result of elimination is not necessarily a probability term

- We want to compute *P(d)*
- Need to elimінate: *s,x,t,l,a,b*

- Initial factors



$$P(v)P(s)P(t \mid v)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$

$$\Rightarrow f_v(t)P(s)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$

Eliminate: *s*

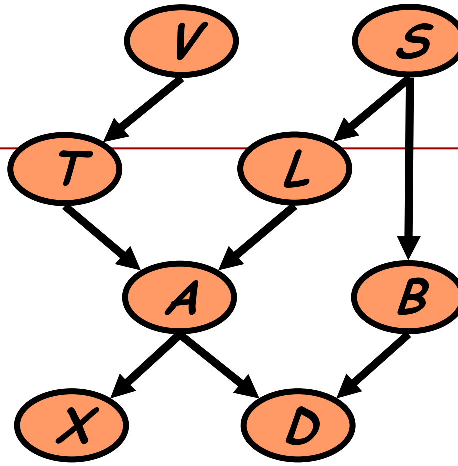
Compute: $$f_s(b,l) = \sum_s P(s)P(b \mid s)P(l \mid s)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$

Summing on *s* results in a factor with two arguments $f_s(b,l)$
In general, result of elimination may be a function of several variables

- We want to compute $P(d)$
- Need to eliminate: x,t,l,a,b

- Initial factors



$$P(v)P(s)P(t \mid v)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$
$$\Rightarrow f_v(t)P(s)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$
$$\Rightarrow f_v(t)f_s(b,l)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$

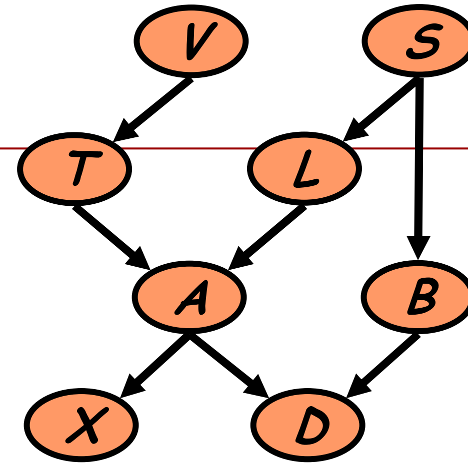
Eliminate: x

Compute: $$f_x(a) = \sum_x P(x \mid a)$$

$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a \mid t,l)P(d \mid a,b)$$

Note: $f_x(a) = 1$ for all values of a !!

- We want to compute *P(d)*
- Need to eliminate: *t,l,a,b*



- Initial factors

$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

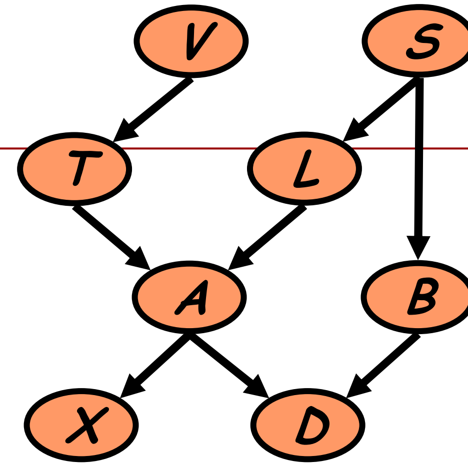
$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a|t,l)P(d|a,b)$$

Eliminate: *t*

Compute: $$f_t(a,l) = \sum_t f_v(t)P(a|t,l)$$

$$\Rightarrow f_s(b,l)f_x(a)f_t(a,l)P(d|a,b)$$

- We want to compute $P(d)$
- Need to eliminate: l,a,b
- Initial factors



$$P(v)P(s)P(t \mid v)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$

$$\Rightarrow f_v(t)P(s)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a \mid t,l)P(d \mid a,b)$$

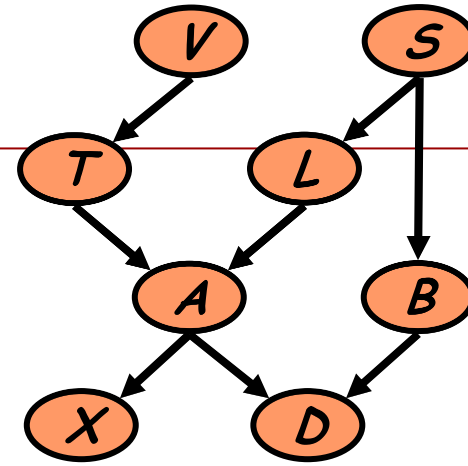
$$\Rightarrow f_s(b,l)f_x(a)f_t(a,l)P(d \mid a,b)$$

Eliminate: l

Compute: $$f_l(a,b) = \sum_l f_s(b,l)f_t(a,l)$$

$$\Rightarrow f_l(a,b)f_x(a)P(d \mid a,b)$$

- We want to compute *P(d)*
- Need to eliminate: *b*
- Initial factors



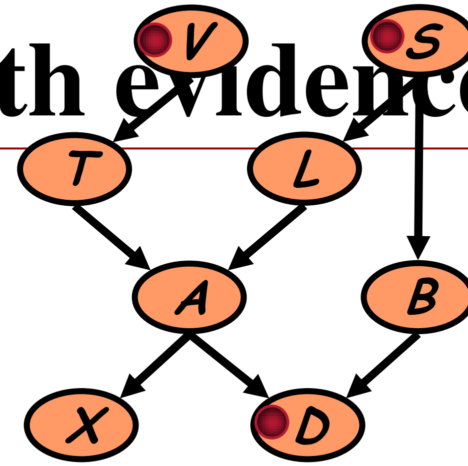
$$P(v)P(s)P(t \mid v)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$
$$\Rightarrow f_v(t)P(s)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$
$$\Rightarrow f_v(t)f_s(b,l)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$
$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a \mid t,l)P(d \mid a,b)$$
$$\Rightarrow f_s(b,l)f_x(a)f_t(a,l)P(d \mid a,b)$$
$$\Rightarrow f_l(a,b)f_x(a)P(d \mid a,b) \Rightarrow f_a(b,d) \Rightarrow f_b(d)$$

Eliminate: *a,b*

Compute:

$$f_a(b,d) = \sum_a f_l(a,b)f_x(a)p(d \mid a,b) \qquad f_b(d) = \sum_b f_a(b,d)$$

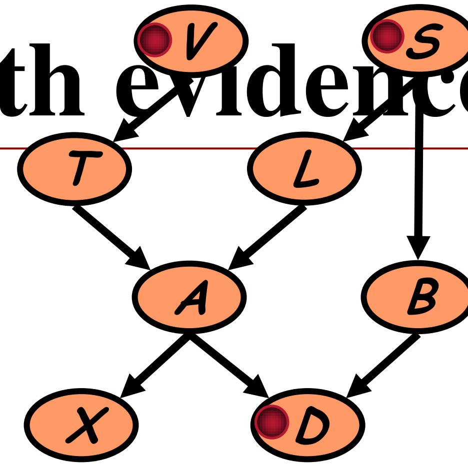
Dealing with evidence



- How do we deal with evidence?

- Suppose we are give evidence $V = t, S = f, D = t$
- We want to compute $P(L, V = t, S = f, D = t)$

Dealing with evidence



- We start by writing the factors:

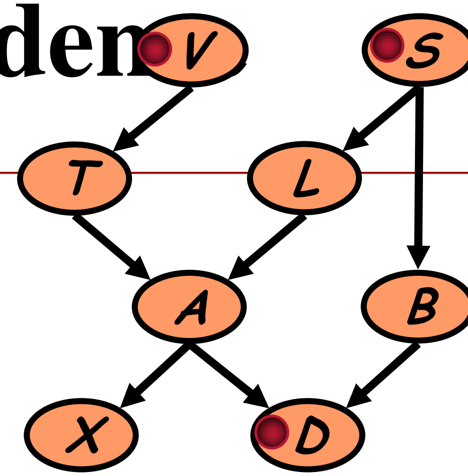
$$P(v)P(s)P(t \mid v)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$

- Since we know that $V = t$, we don't need to eliminate V
- Instead, we can replace the factors $P(V)$ and $P(T|V)$ with

$$f_{P(V)} = P(V = t) \quad f_{p(T|V)}(T) = P(T \mid V = t)$$

- These "select" the appropriate parts of the original factors given the evidence
- Note that $f_{p(V)}$ is a constant, and thus does not appear in elimination of other variables

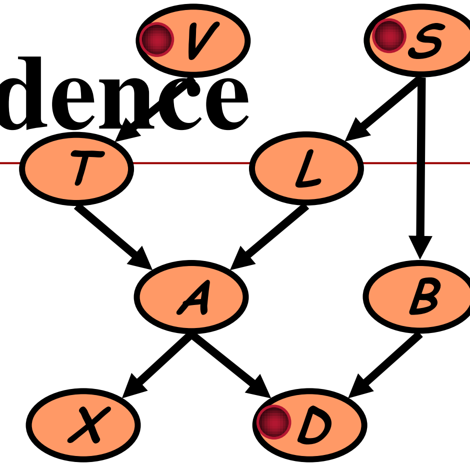
Dealing with eviden



- Given evidence $V = t, S = f, D = t$
- Compute $P(L, V = t, S = f, D = t)$
- Initial factors, after setting evidence:

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a \mid t, l) P(x \mid a) f_{P(d|a,b)}(a, b)$$

Dealing with evidence



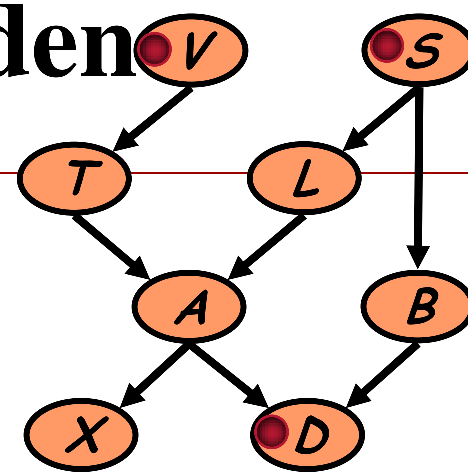
- Given evidence $V = t, S = f, D = t$
- Compute $P(L, V = t, S = f, D = t)$
- Initial factors, after setting evidence:

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a \mid t, l) P(x \mid a) f_{P(d|a,b)}(a,b)$$

- Eliminating x, we get

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a \mid t, l) f_x(a) f_{P(d|a,b)}(a,b)$$

Dealing with evidence



- Given evidence $V = t$, $S = f$, $D = t$
- Compute $P(L, V = t, S = f, D = t)$
- Initial factors, after setting evidence:

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a \mid t, l) P(x \mid a) f_{P(d|a,b)}(a,b)$$

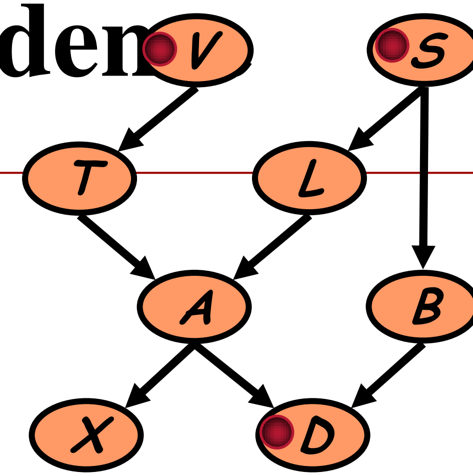
- Eliminating x, we get

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a \mid t, l) f_x(a) f_{P(d|a,b)}(a,b)$$

- Eliminating t, we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_{P(b|s)}(b) f_t(a,l) f_x(a) f_{P(d|a,b)}(a,b)$$

Dealing with eviden



- Given evidence $V = t$, $S = f$, $D = t$
- Compute $P(L, V = t, S = f, D = t)$
- Initial factors, after setting evidence:

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a \mid t, l) P(x \mid a) f_{P(d|a,b)}(a,b)$$

- Eliminating x, we get

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a \mid t, l) f_x(a) f_{P(d|a,b)}(a,b)$$

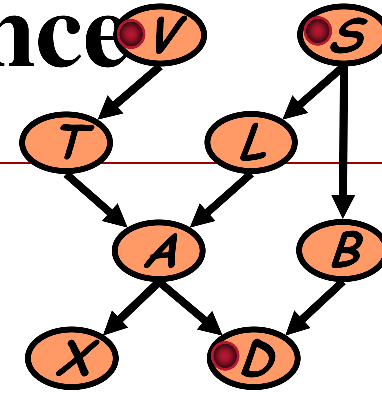
- Eliminating t, we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_{P(b|s)}(b) f_t(a,l) f_x(a) f_{P(d|a,b)}(a,b)$$

- Eliminating a, we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_{P(b|s)}(b) f_a(b,l)$$

Dealing with evidence



- Given evidence $V = t, S = f, D = t$
- Compute $P(L, V = t, S = f, D = t)$
- Initial factors, after setting evidence:

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a \mid t,l) P(x \mid a) f_{P(d|a,b)}(a,b)$$

- Eliminating x, we get

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a \mid t,l) f_x(a) f_{P(d|a,b)}(a,b)$$

- Eliminating t, we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_{P(b|s)}(b) f_t(a,l) f_x(a) f_{P(d|a,b)}(a,b)$$

- Eliminating a, we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_{P(b|s)}(b) f_a(b,l)$$

- Eliminating b, we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_b(l)$$

Variable elimination algorithm

- Let $X_1,\ldots, X_m$ be an ordering on the non-query variables
- For i = m, …, 1 $\sum_{X_1} \sum_{X_2} \cdots \sum_{X_m} \prod_j P(X_j \mid Parents(X_j))$
  - Leave in the summation for X_i only factors mentioning X_i
  - Multiply the factors, getting a factor that contains a number for each value of the variables mentioned, including X_i
  - Sum out X_i, getting a factor f that contains a number for each value of the variables mentioned, not including X_i
  - Replace the multiplied factor in the summation

Complexity of variable elimination

Suppose in one elimination step we compute

$$f_x(y_1,\ldots,y_k) = \sum_x f'_x(x, y_1,\ldots,y_k)$$

$$f'_x(x, y_1,\ldots,y_k) = \prod_{i=1}^{m} f_i(x, y_{1,1},\ldots,y_{1,l_i})$$

This requires

$$m \cdot |Val(X)| \cdot \prod_i |Val(Y_i)|$$

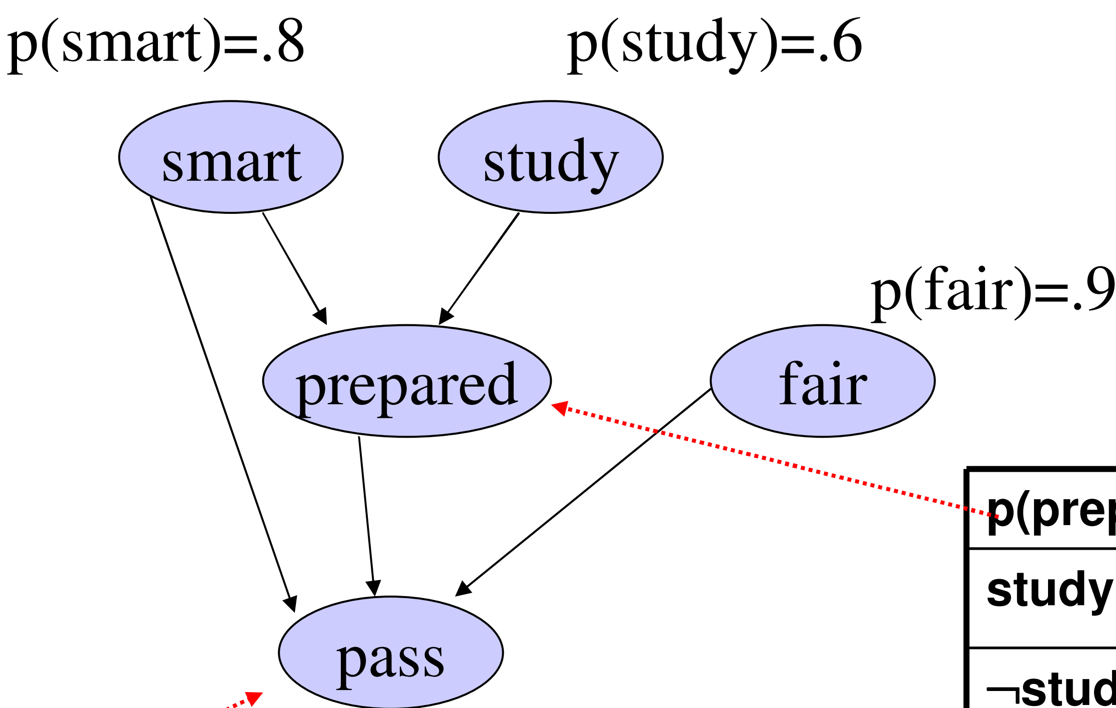
multiplications (for each value for $x, y_1, \ldots, y_k$, we do m multiplications) and

$$|Val(X)| \cdot \prod_i |Val(Y_i)|$$

additions (for each value of $y_1, \ldots, y_k$, we do $|Val(X)|$ additions)

- **Complexity is exponential in the number of variables in the intermediate factors**
- **Finding an optimal ordering is NP-hard**

Exercise: Variable elimination

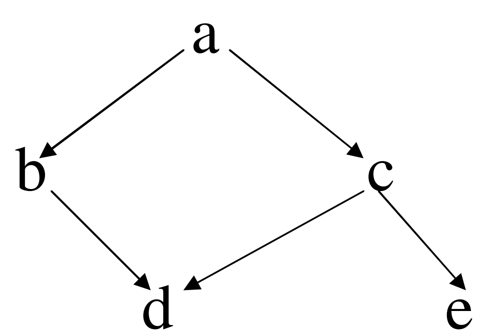


p(prep|…)	smart	¬smart
study	.9	.7
¬study	.5	.1

p(pass|…)	smart		¬smart	
	prep	¬prep	prep	¬prep
fair	.9	.7	.7	.2
¬fair	.1	.1	.1	.1

Query: What is the probability that a student is smart, given that they pass the exam?

Conditioning



- **Conditioning**: Find the network's smallest **cutset** S (a set of nodes whose removal renders the network singly connected)
  - In this network, S = {A} or {B} or {C} or {D}
- For each instantiation of S, compute the belief update with the polytree algorithm
- Combine the results from all instantiations of S
- Computationally expensive (finding the smallest cutset is in general NP-hard, and the total number of possible instantiations of S is $O(2^{|S|})$)

Approximate Inference

Approaches to inference

- Exact inference
  - **Enumeration**
  - **Variable elimination**
  - Clustering / join tree algorithms
- Approximate inference
  - **Stochastic simulation / sampling methods**
  - **Markov chain Monte Carlo methods**

Approximate inference: Direct sampling

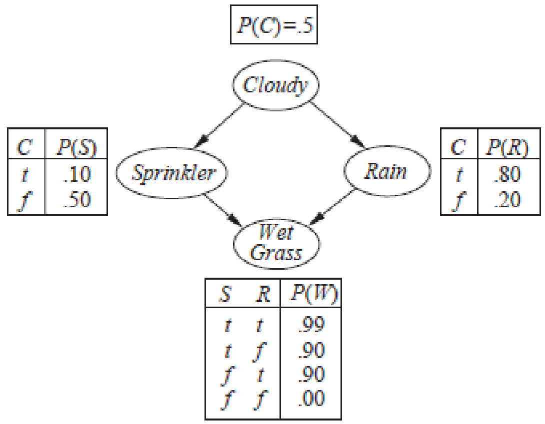
- Generates events from a network that has no evidence associated with it
- Randomly generate a very large number of instantiations from the BN
  - Generate instantiations for **all** variables – start at root variables and work your way “forward” in topological order
  - Probability distribution conditioned on values assigned to parents
- Use the frequency of values for Z to get estimated probabilities
- Accuracy of the results depends on the size of the sample (asymptotically approaches exact results)

Direct sampling algorithm

```
function PRIOR-SAMPLE(bn) returns an event sampled from the prior specified by bn
   inputs: bn, a Bayesian network specifying joint distribution P(X_1, ..., X_n)

   x ← an event with n elements
   foreach variable X_i in X_1, ..., X_n do
       x[i] ← a random sample from P(X_i | parents(X_i))
   return x
```

Direct sampling example



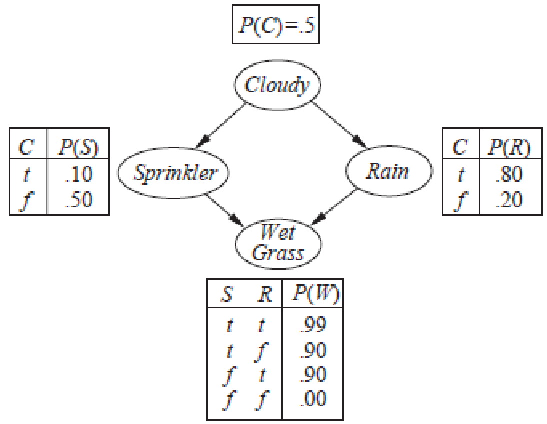
- Sample from P(Cloudy) = <0.5, 0.5>, value is true
- Sample from P(Sprinkler|cloudy) = <0.1, 0.9>, value is false
- Sample from P(Rain|cloudy) = <0.8, 0.2>, value is true
- Sample from P(WetGrass|~sprinkler, rain) = <0.9, 0.1>, value is true

- [true, false, true, true]

Approximate inference: Rejection sampling

- Suppose you are given values for some subset of the variables, E, and want to infer values for unknown variables, Z
- Used to compute conditional probabilities, i.e. P(X|e)
- Randomly generate a very large number of instantiations from the BN
  - Generate instantiations for **all** variables
  - Rejection sampling: Only keep those instantiations that are consistent with the values for E
- Use the frequency of values for Z to get estimated probabilities
- Accuracy of the results depends on the size of the sample (asymptotically approaches exact results)

Rejection sampling example



- Query P(Rain|sprinkler), using 100 samples
  - Out of the 100, 73 have Sprinkler=false
  - We reject them
  - From the 27 left, 8 have Rain=true
  - P(Rain|Sprinkler) = <0.296, 0.704>

Likelihood weighting

- Idea: Don't generate samples that need to be rejected in the first place!
- Sample only from the unknown variables Z
- Weight each sample according to the likelihood that it would occur, given the evidence E

Markov Chain Monte Carlo algorithm

- So called because
  - Markov chain – each instance generated in the sample is dependent on the previous instance
  - Monte Carlo – statistical sampling method
- Works different from rejection sample and likelihood weighting
  - MCMC generates each sample by making a random change to the preceding example
  - *Current state*: a value for every variable
  - *Next state*: Make random changes to the current state

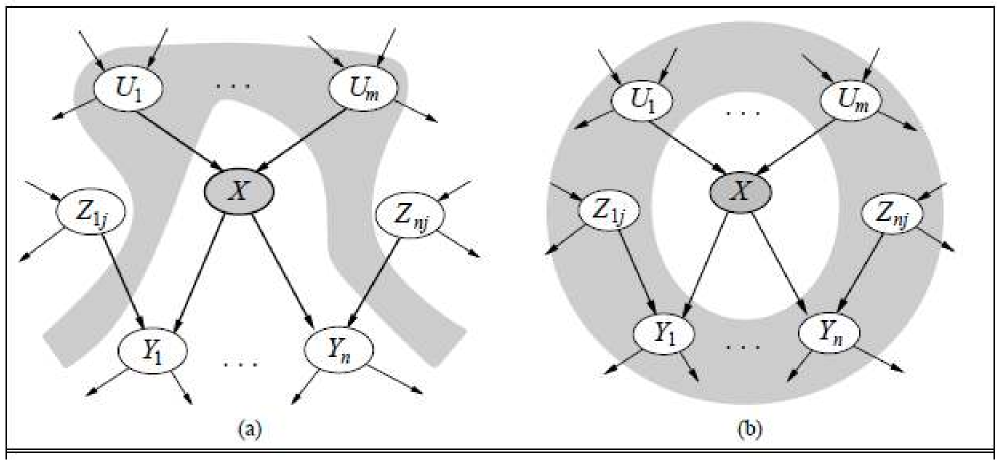
Markov chain Monte Carlo algorithm

- So called because
  - Markov chain – each instance generated in the sample is dependent on the previous instance
  - Monte Carlo – statistical sampling method
- Perform a random walk through variable assignment space, collecting statistics as you go
  - Start with a random instantiation, consistent with evidence variables
  - At each step, for some nonevidence variable, randomly sample its value, consistent with the other current assignments
- Given enough samples, MCMC gives an accurate estimate of the true distribution of values

Gibbs sampling

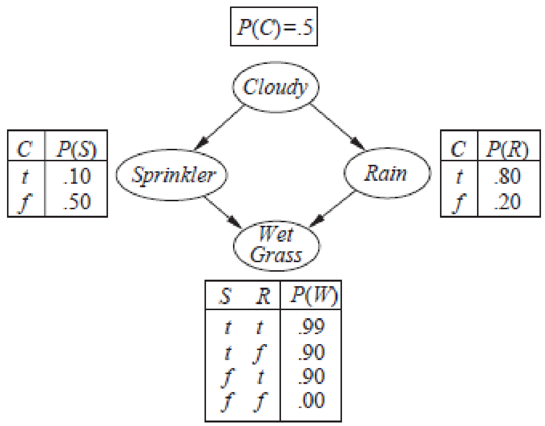
- A particular form of MCMC
  - Start with a random instantiation, consistent with evidence variables
  - Generate next state by randomly sample a value for some nonevidence variable X
    - The sampling for X is done conditioned on the current values of the variables in the Markov blanket of X
- Wanders randomly around the space of possible complete assignments, flipping one variable at a time, but keeping the evidence variables fixed

(from slide 13) Topological semantics



- A node is **conditionally independent** of its **non-descendants** given its **parents**
- A node is **conditionally independent** of all other nodes in the network given its parents, children, and children's parents (also known as its **Markov blanket**)

MCMC Gibbs sampling example



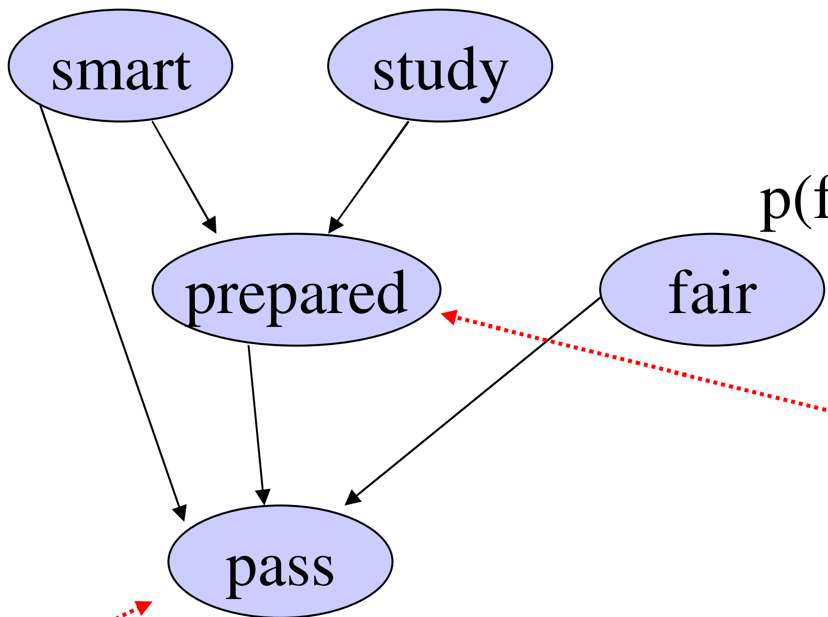
- Query P(Rain|sprinkler, wetgrass)
- Initial state [true, true, false, true]
  - Cloudy is sampled P(Cloudy|sprinkler, ~rain)
  - Suppose result is Cloudy=false
  - New state is [false, true, false, true]
  - Rain is sampled P(Rain|~cloudy, sprinkler, wetgrass)
  - Suppose result is Rain=true
- Continue sampling, and normalize frequencies to get result at the end

Exercise: MCMC sampling

p(smart)=.8

p(study)=.6

p(fair)=.9



p(prep|…)	smart	¬smart
study	.9	.7
¬study	.5	.1

p(pass|…)	smart		¬smart	
	prep	¬prep	prep	¬prep
fair	.9	.7	.7	.2
¬fair	.1	.1	.1	.1

Topological order = …?

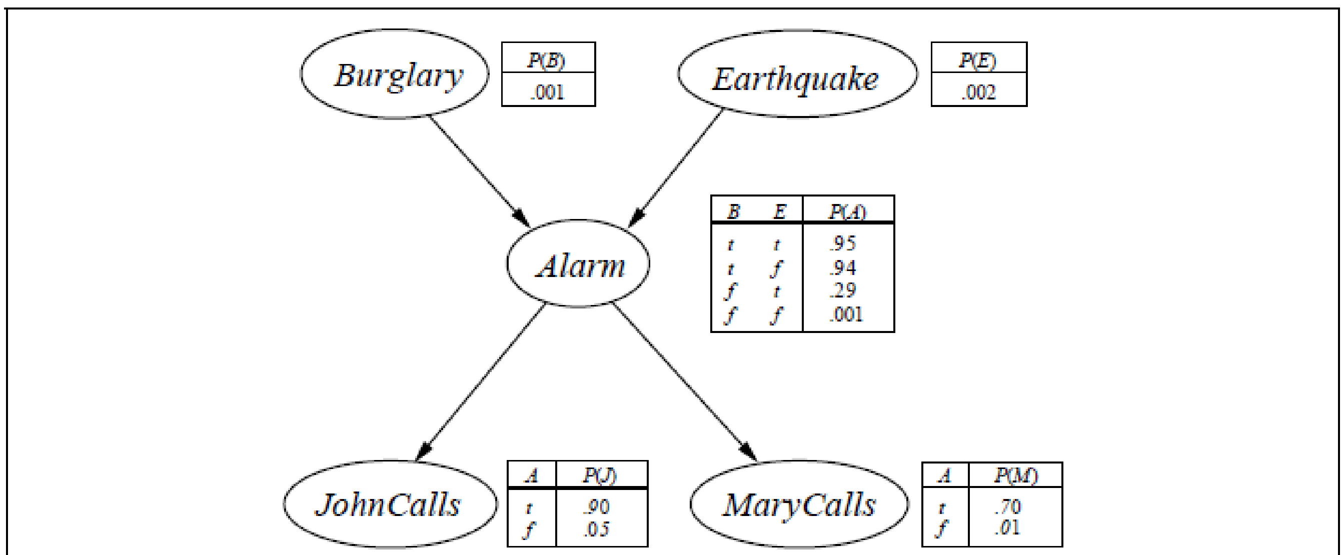
Random number generator: .35, .76, .51, .44, .08, .28, .03, .92, .02, .42

Summary

- Bayes nets
  - Structure
  - Parameters
  - Conditional independence
  - Chaining
- BN inference
  - Enumeration
  - Variable elimination
  - Sampling methods

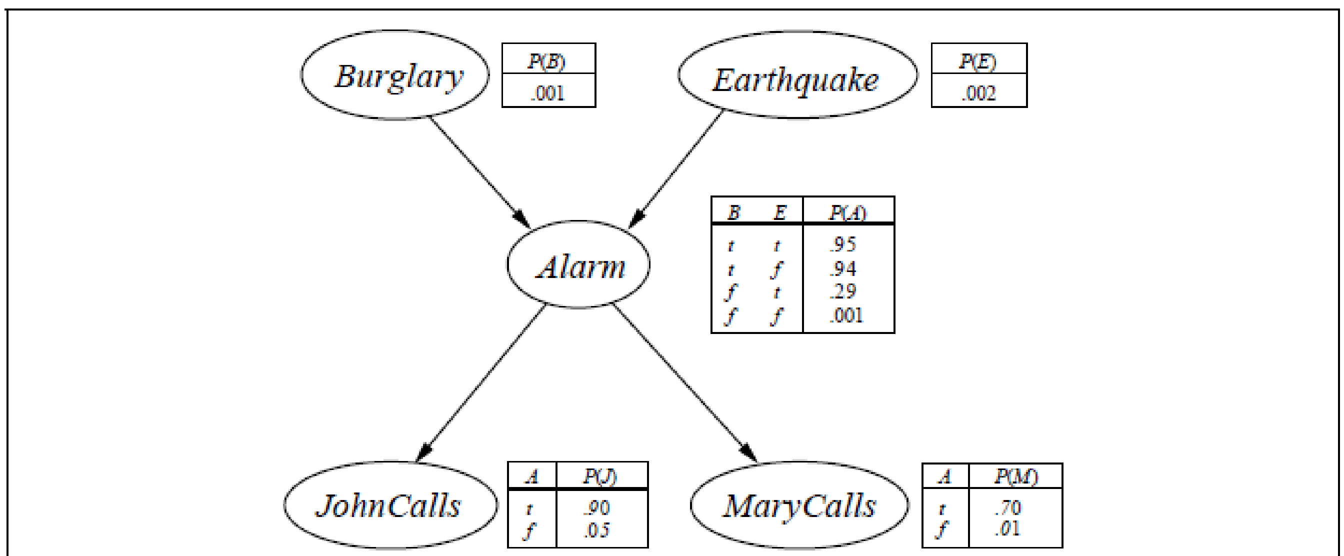
Summary

- **Bayesian Networks**
- Independence and conditional independence among variables can greatly reduce the full joint distribution
- Bayesian Networks
  - A structure used to represent the dependencies among variables



Summary

- **Conditional Independence and Chaining**
  - With this assumption, the complete joint probability distribution of all variables in the network can be represented by (recovered from) local CPDs by chaining these CPDs



Summary

- Inference tasks
  - **Simple queries:** Compute posterior distribution $P(X_i \mid E{=}e)$
  - E.g., P(NoGas | Gauge=empty, Lights=on, Starts=false)
  - P(Burglary | JohnCalls=true, MaryCalls=true) = <0.284, 0.716>
  - **Conjunctive queries:**
  - $P(X_i, X_j \mid E{=}e) = P(X_i \mid e{=}e)\, P(X_j \mid X_i, E{=}e)$
- Exact inference
  - **Enumeration**
  - **Variable elimination**
  - Clustering / join tree algorithms
- Approximate inference
  - **Stochastic simulation / sampling methods**
  - **Markov chain Monte Carlo methods**