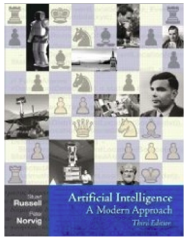
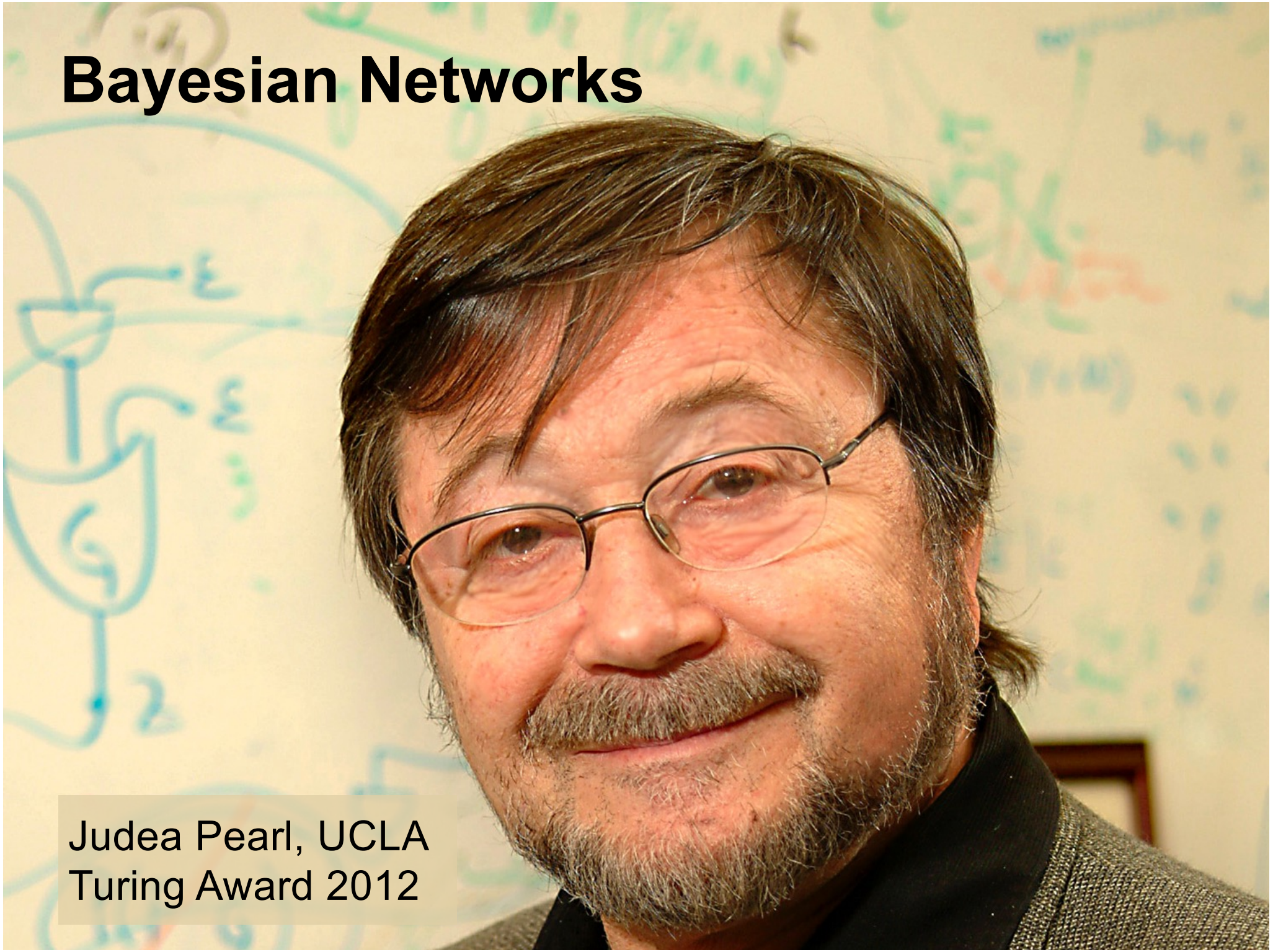
Bayesian Networks

- Syntax
- Semantics
- Inference
- Parameter Estimation



Many slides based on Russell & Norvig's slides Artificial Intelligence: A Modern Approach

Bayesian Networks



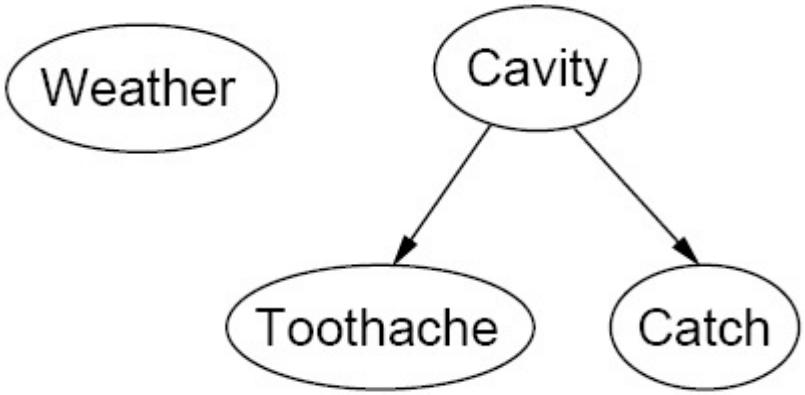
Judea Pearl, UCLA
Turing Award 2012

Bayesian Networks - Structure

- Are a simple, graphical notation for **conditional independence** assertions, hence for **compact specifications of full joint distributions**

A BN is a directed acyclic graph with the following components:

- **Nodes**: one node for each variable
- **Edges**: a directed edge from node N_i to node N_j indicates that variable X_i has a direct influence upon variable X_j

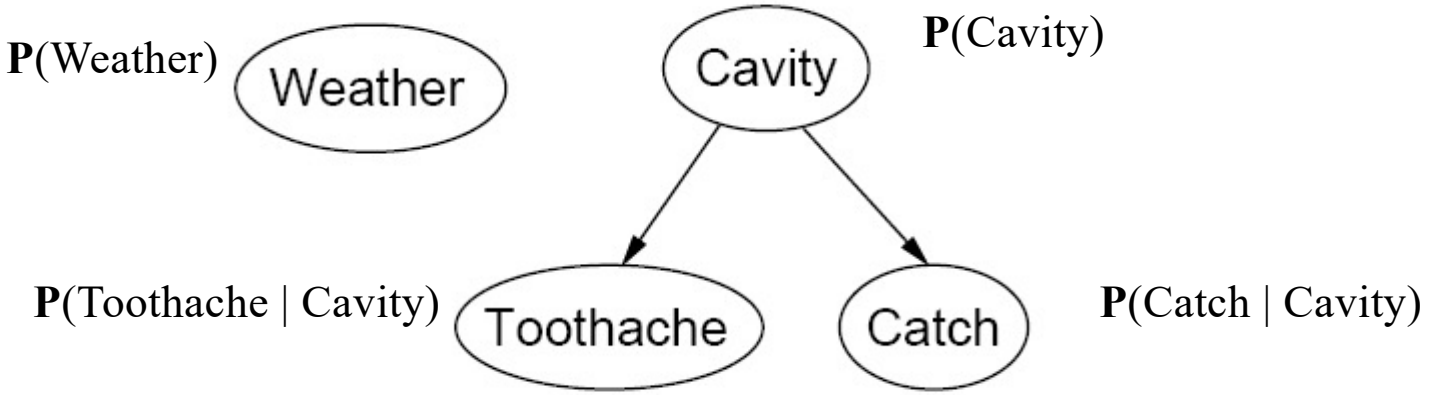


Bayesian Networks - Probabilities

In addition to the structure, we need a **conditional probability distribution** for the random variable of each node given the random variables of its parents.

- i.e. we need $\mathbf{P}(X_i \mid \text{Parents}(X_i))$
- nodes/variables that are not connected are (conditionally) independent:

Example: Weather is independent of Cavity, Toothache is independent of Catch given Cavity



Running Example: Alarm

Situation:

- I'm at work
- John calls to say that in my house the alarm went off but Mary (my neighbor) did not call
- The alarm will usually be set off by burglars but sometimes it may also go off because of minor earthquakes

How do you model this?

Running Example: Alarm

Situation:

- I'm at work
- John calls to say that in my house the alarm went off but Mary (my neighbor) did not call
- The alarm will usually be set off by burglars but sometimes it may also go off because of minor earthquakes

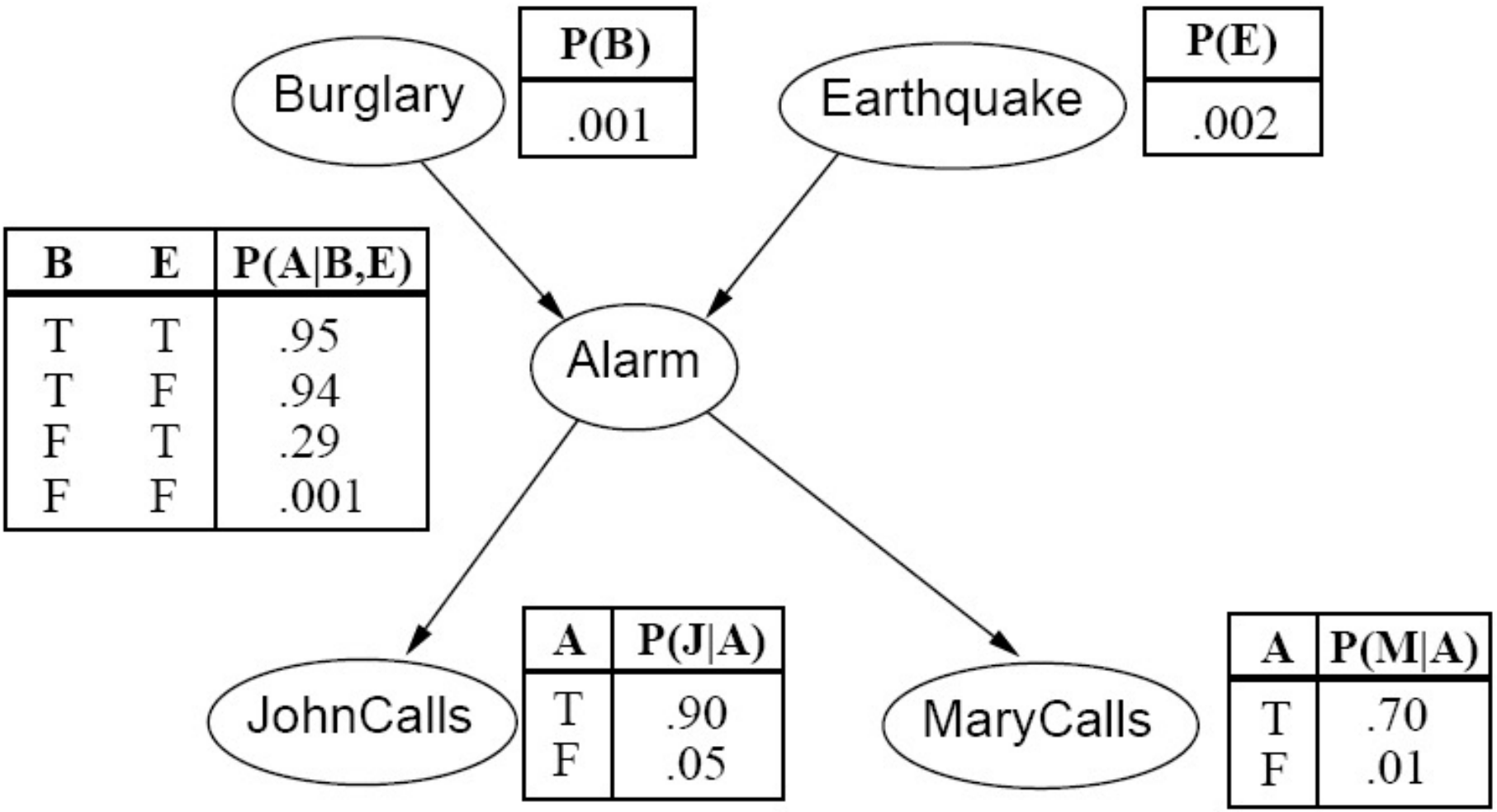
Random Variables:

- *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*

Network topology reflects causal knowledge:

- A burglar can set the alarm off
- An earthquake can set the alarm off
- The alarm can cause Mary to call
- The alarm can cause John to call

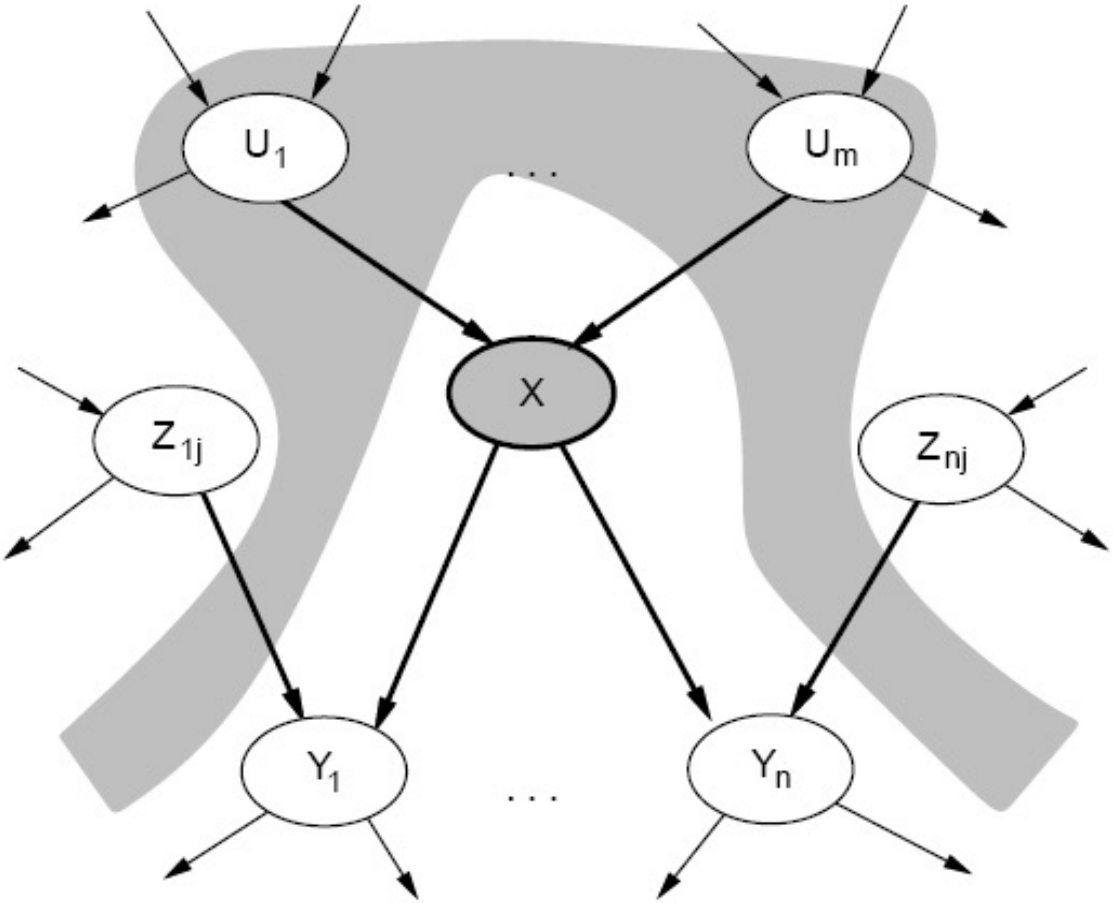
Alarm Example



Local Semantics of a BN

- Each node is is conditionally independent of its nondescendants given its parents

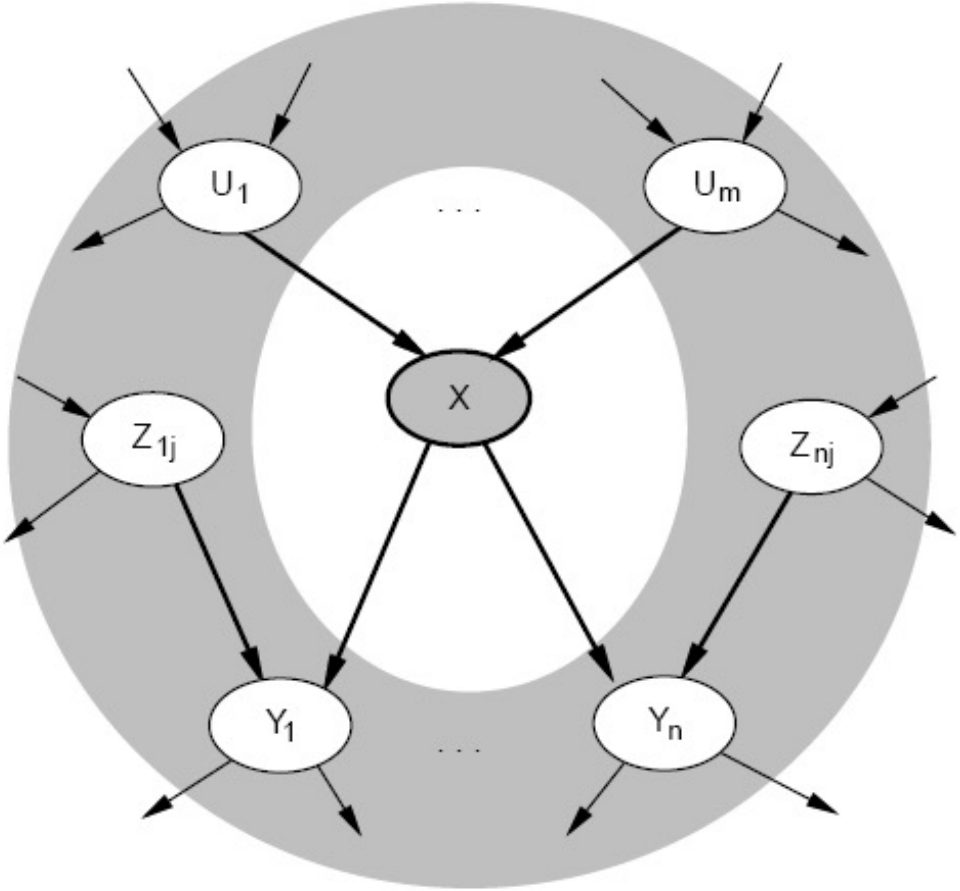
  (sometime called the **local Markov assumption**)



$$\mathbf{P}\left(X \mid U_1,\ldots,U_m,Z_{1j},\ldots,Z_{nj}\right) = \\ = \boldsymbol{P}\left(X \mid U_1,\ldots,U_m\right)$$

Markov Blanket

- **Markov Blanket = parents + children + children's parents**



Each node is conditionally independent of all other nodes given its Markov blanket

$$\mathbf{P}(X \mid U_1, \ldots, U_m, Y_1, \ldots, Y_n, Z_{1j}, \ldots, Z_{nj}) = \\ = \boldsymbol{P}(X \mid allvariables)$$

Global Semantics of a BN

- The conditional probability distributions define (**factorizes**) the joint probability distribution of the variables of the network

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid \text{Parents}(X_i))$$

- Example:
  - What is the probability that the alarm goes off and both John and Mary call, but there is neither a burglary nor an earthquake?

$$P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) =$$
$$= P(j \mid a)P(m \mid a)P(a \mid \neg b, \neg e)P(\neg b)P(\neg e)$$
$$= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 \approx 0.00063$$

Constructing Bayesian Networks

Need a method such that a series of locally testable assertions of conditional independence guarantees the required global semantics

1. Choose an ordering of variables $X_1, \ldots, X_n$
2. For $i = 1$ to n
   add X_i to the network
   select parents from $X_1, \ldots, X_{i-1}$ such that
   $$\mathbf{P}(X_i|Parents(X_i)) = \mathbf{P}(X_i|X_1, \ldots, X_{i-1})$$

This choice of parents guarantees the global semantics:

$$\begin{aligned}\mathbf{P}(X_1, \ldots, X_n) &= \prod_{i=1}^{n} \mathbf{P}(X_i|X_1, \ldots, X_{i-1}) \quad \text{(chain rule)} \\ &= \prod_{i=1}^{n} \mathbf{P}(X_i|Parents(X_i)) \quad \text{(by construction)}\end{aligned}$$

Example

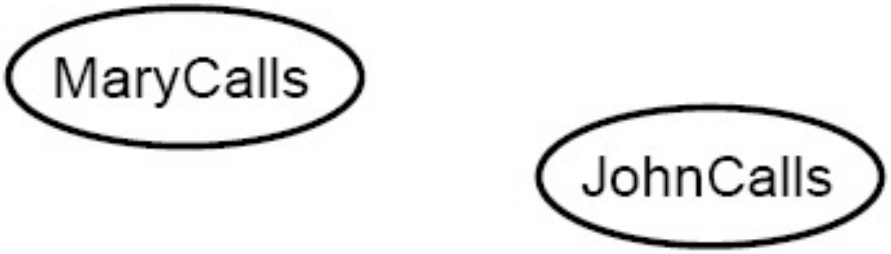
- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

Example

- Suppose we first select the ordering

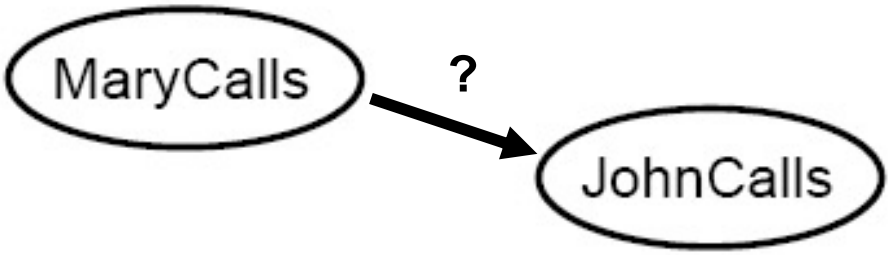
MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,



Example

- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

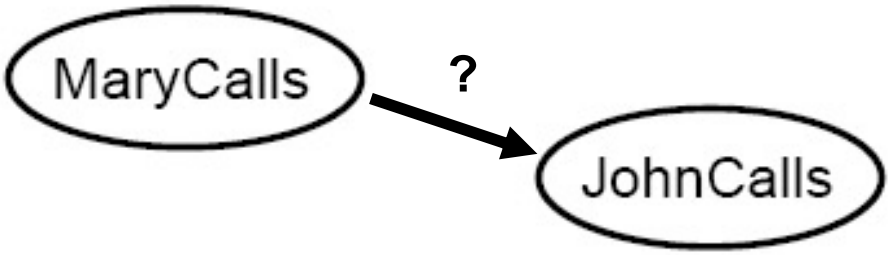


Example

- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

$$P(J \mid M) = P(J)?$$



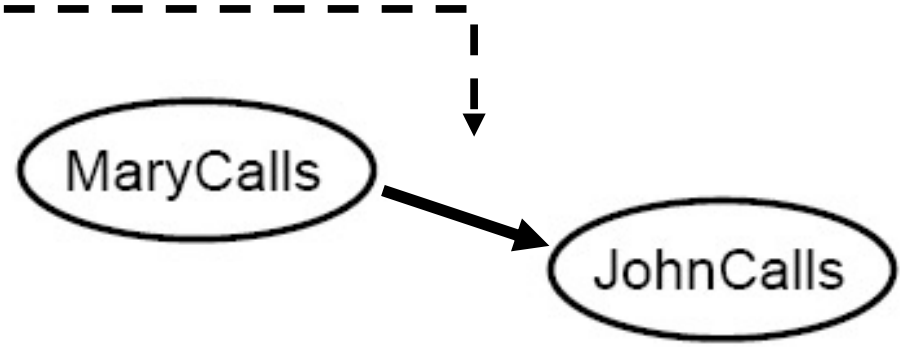
Example

- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

$P(J \mid M) = P(J)?$ ✗

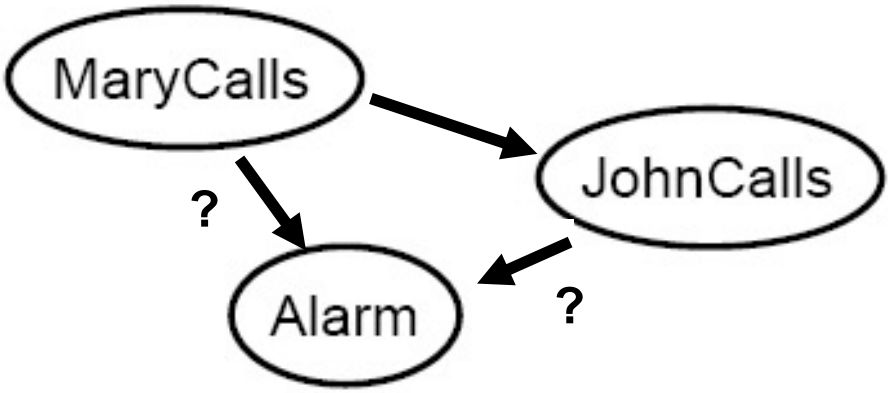
If Mary calls, it is more likely that John calls as well.



Example

- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

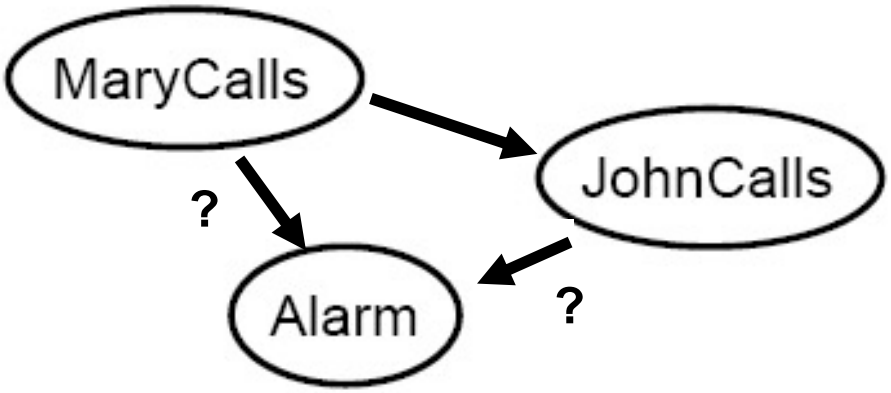


Example

- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

$P(A \mid J, M) = P(A)?$ ✗



Example

- Suppose we first select the ordering

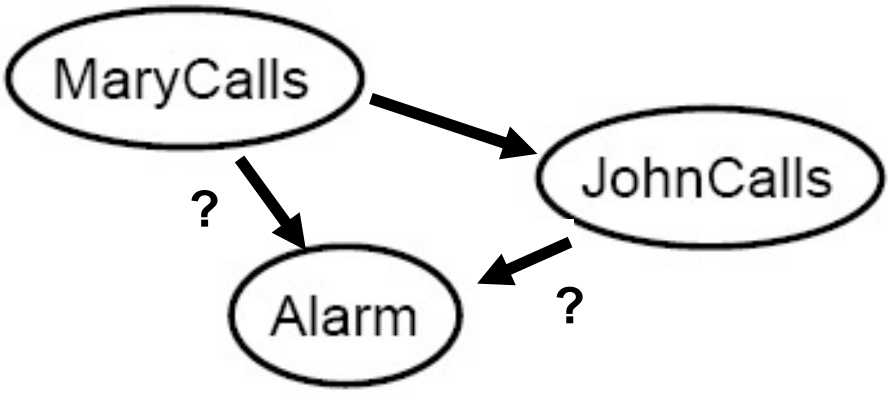
MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

$P(A \mid J, M) = P(A)?$ ✗

If Mary and John call, the probability that the alarm has gone off is larger than if they don't call.

↓

Node A needs parents J or M

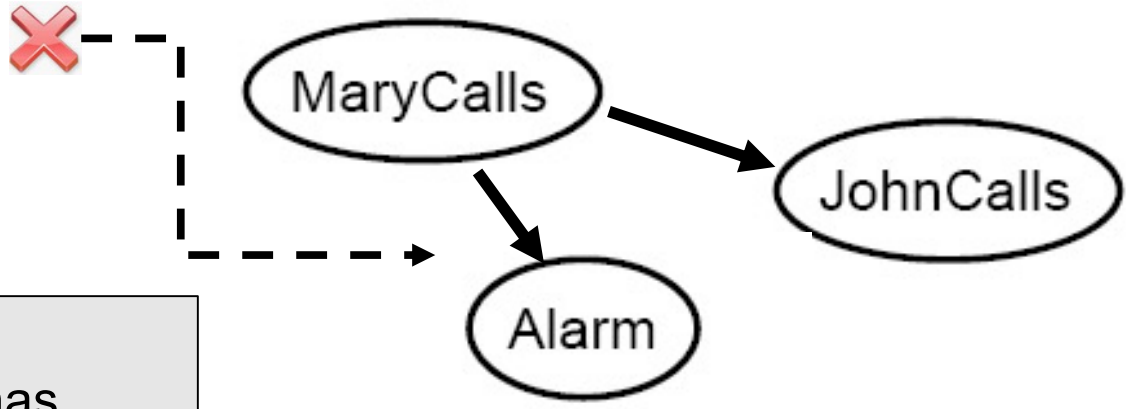


Example

- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

$P(A \mid J, M) = P(A \mid J)$?



If John and Mary call, the probability that the alarm has gone off is higher than if only John calls.

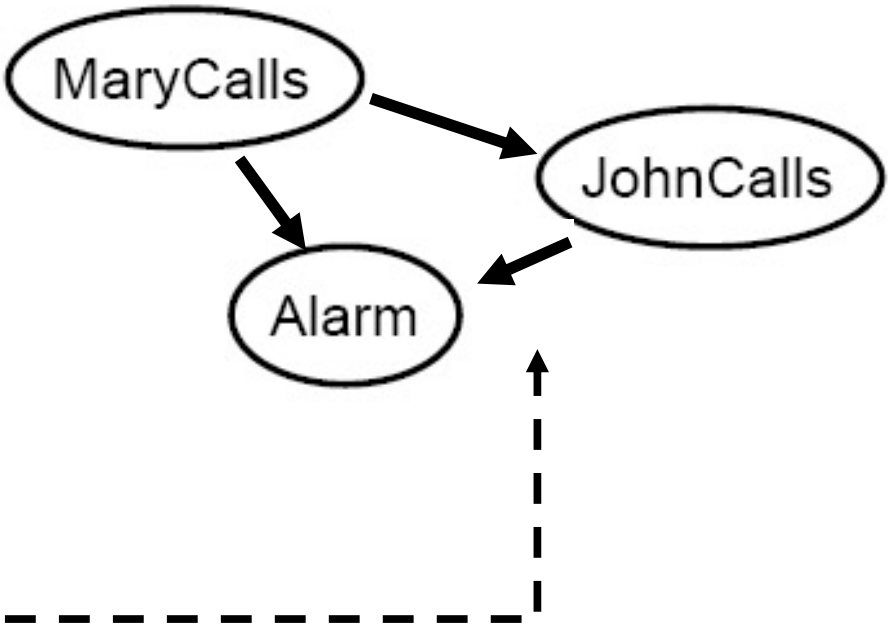
Example

- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

$P(A \mid J, M) = P(A \mid M)$?

If John and Mary call, the probability that the alarm has gone off is higher than if only Mary calls.



Example

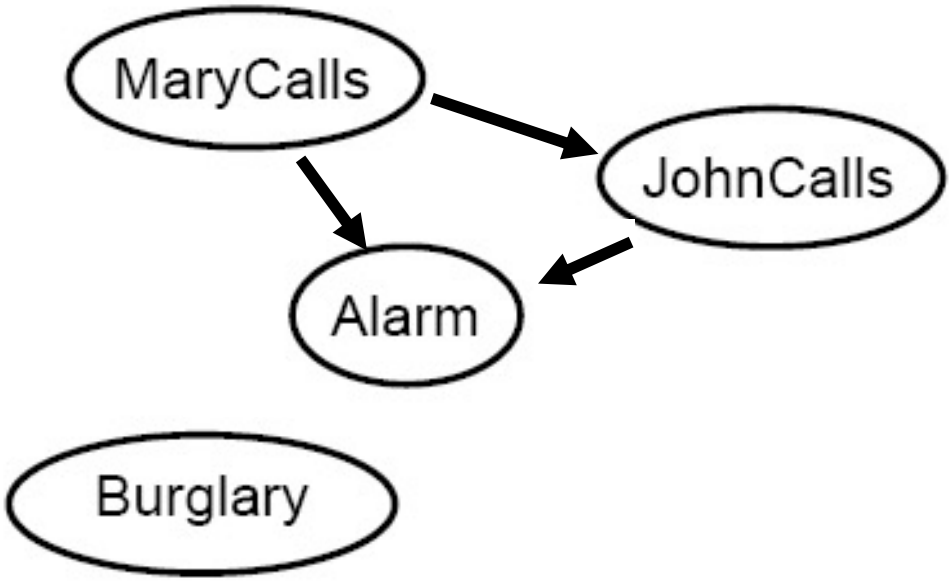
- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

$$P(B \mid A, J, M) = P(B)?$$

Knowing whether Mary or John called and whether the alarm went off influences my knowledge about whether there has been a burglary

Node B needs parents A, J or M



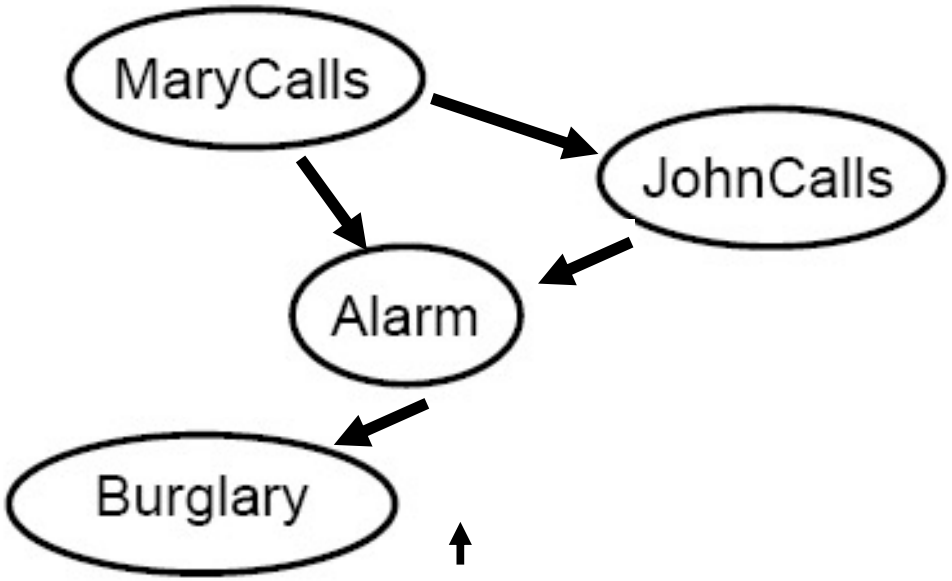
Example

- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

$$P(B \mid A, J, M) = P(B \mid A)?$$ ✓

If I know that the alarm has gone off, knowing that John or Mary have called does not add to my knowledge of whether there has been a burglary or not.



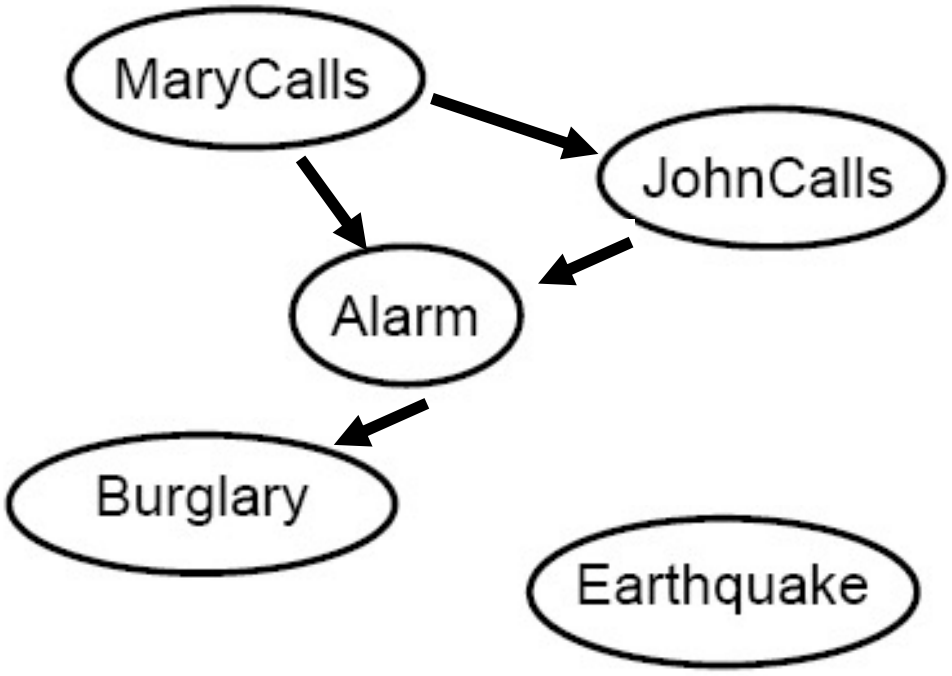
Thus, no edges from M and J, only from A

Example

- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

$$P(E \mid B, A, J, M) = P(E \mid A)?$$ ✗



Knowing whether there has been an Alarm does not suffice to determine the probability of an earthquake, we have to know whether there has been a burglary as well.

Example

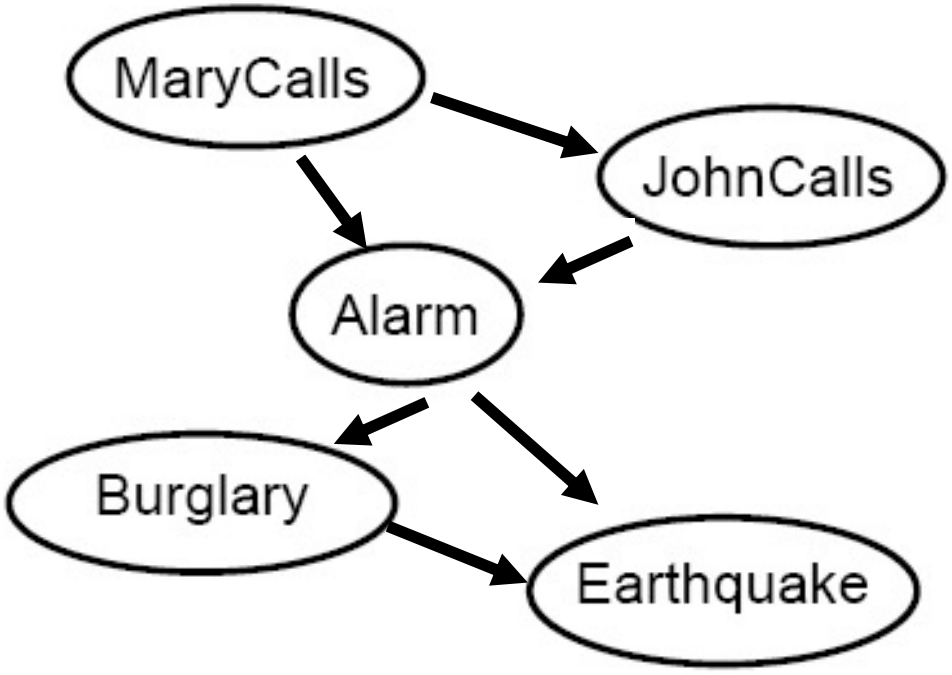
- Suppose we first select the ordering

MaryCalls, JohnCalls, Alarm, Burglary, Earthquake,

$$P(E \mid B, A, J, M) = P(E \mid A)?$$

$$P(E \mid B, A, J, M) = P(E \mid A, B)?$$

Knowing whether there has been an Alarm does not suffice to determine the probability of an earthquake, we have to know whether there has been a burglary as well.



Example - Discussion

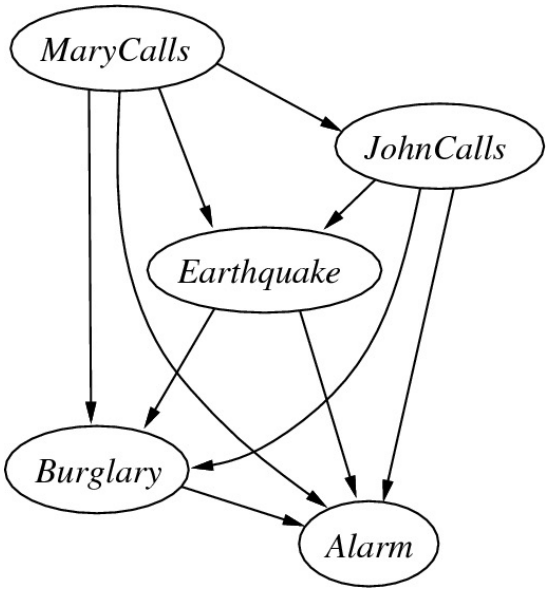
- Deciding conditional independence is hard in non-causal direction
- for assessing whether *X* is conditionally independent of *Z* ask the question:

If I add variable Z in the condition, does it change the probabilities for X?

- causal models and conditional independence seem hardwired for humans!
- Assessing conditional probabilities is also hard in non-causal direction

- Network is less compact
- more edges and more parameters to estimate

- Worst possible ordering
- *MaryCalls, JohnCalls*
Earthquake, Burglary, Alarm
- → fully connected network



Reasoning with Bayesian Networks

- Belief Functions (margin probabilities)
  - given the probability distribution, we can compute a margin probability at each node, which represents the belief into the truth of the proposition
  - → the margin probability is also called the belief function

Reasoning with Bayesian Networks

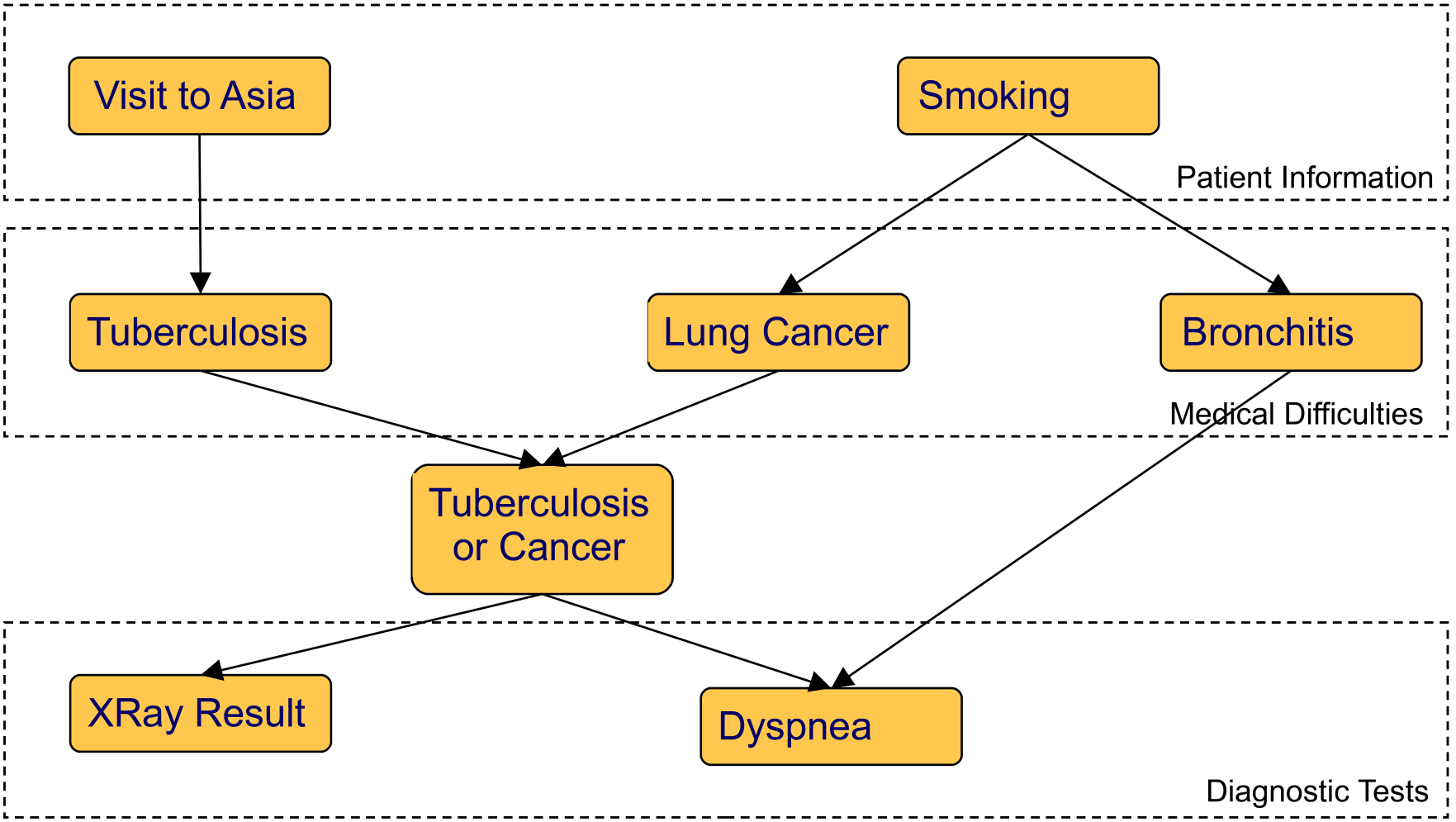
- Belief Functions (margin probabilities)
  - given the probability distribution, we can compute a margin probability at each node, which represents the belief into the truth of the proposition
  - → the margin probability is also called the belief function
- New evidence can be incorporated into the network by changing the appropriate belief functions
  - this may not only happen in unconditional nodes!

Reasoning with Bayesian Networks

- Belief Functions (margin probabilities)
  - given the probability distribution, we can compute a margin probability at each node, which represents the belief into the truth of the proposition
  - → the margin probability is also called the belief function
- New evidence can be incorporated into the network by changing the appropriate belief functions
  - this may not only happen in unconditional nodes!
- changes in the belief functions are then propagated through the network
  - propagation happens in forward (along the causal links) and backward direction (against them)
  - e.g., determining a symptom of a disease does not cause the disease, but changes the probability with which we believe that the patient has the disease

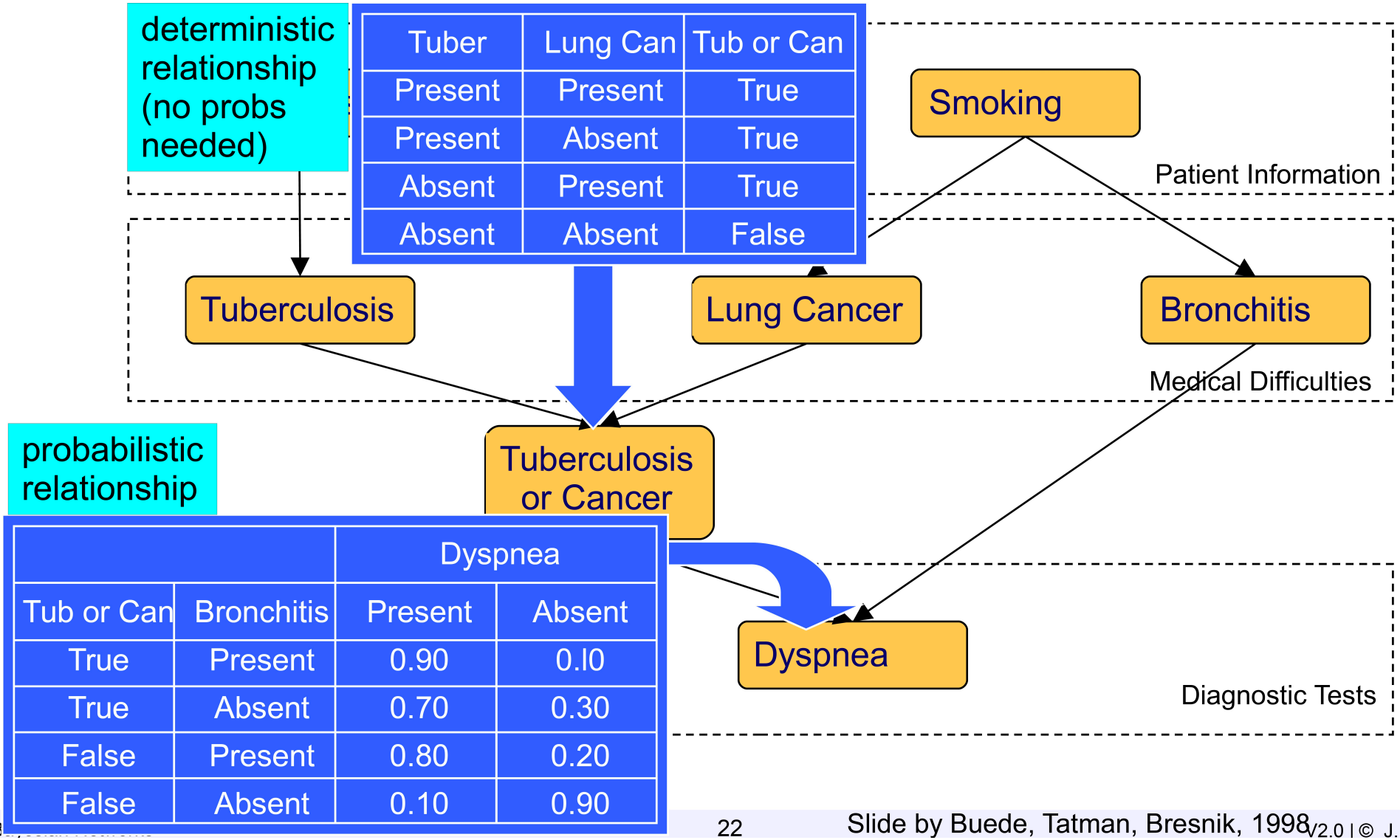
Example: Medical Diagnosis

- Structure of the network



Example: Medical Diagnosis

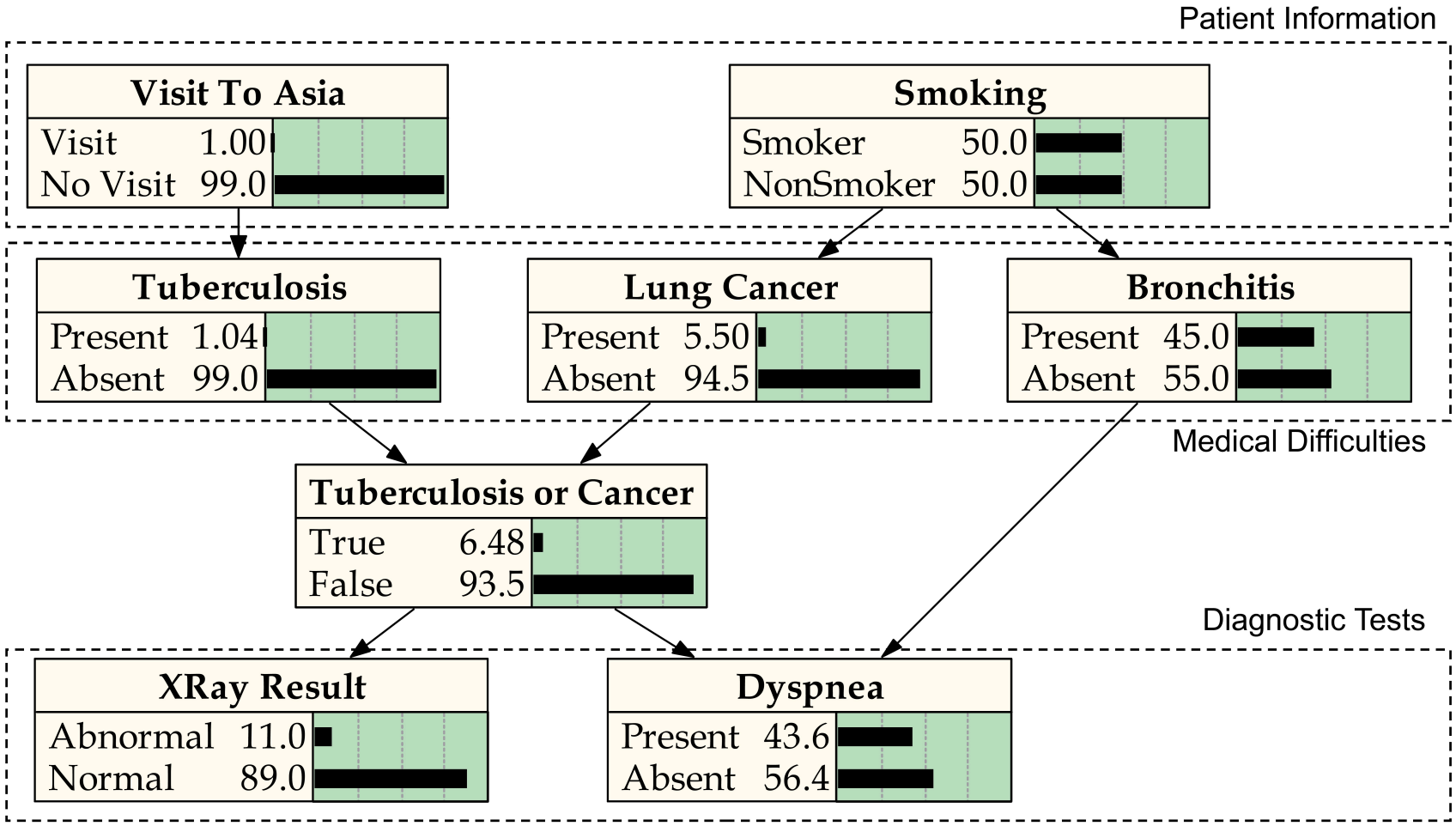
- Adding Probability Distributions



Slide by Buede, Tatman, Bresnik, 1998

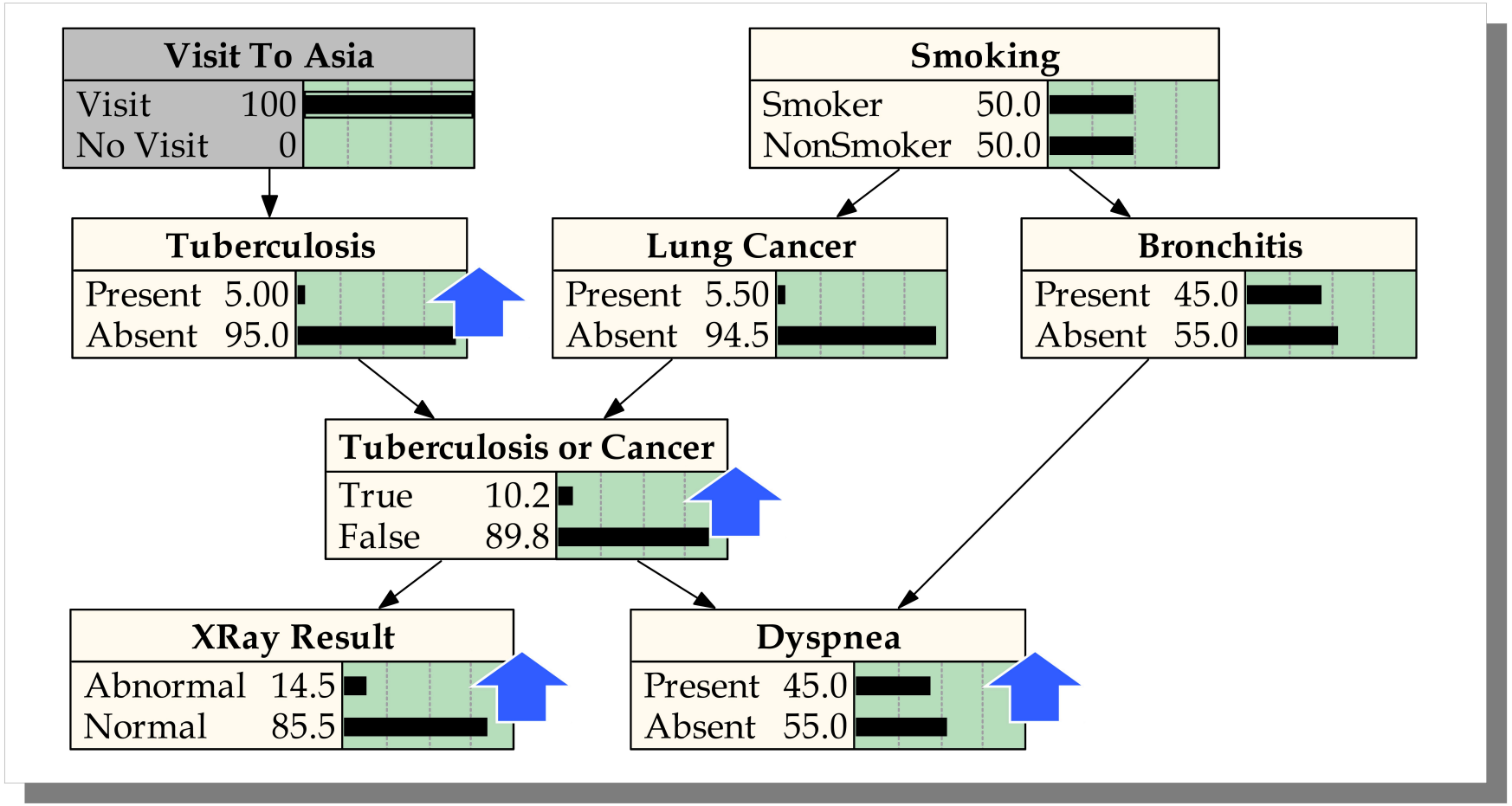
Example: Medical Diagnosis

- Belief functions



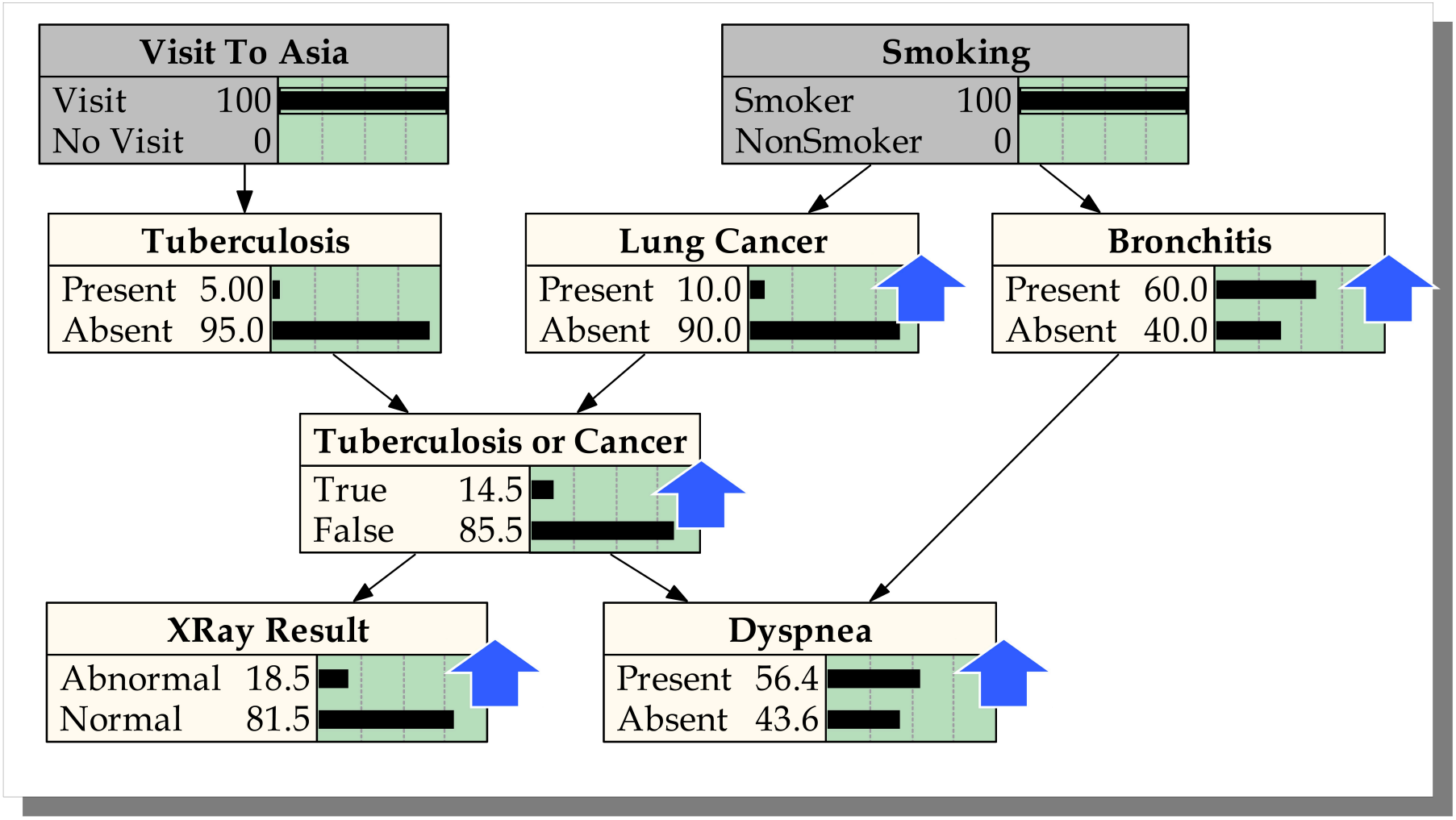
Example: Medical Diagnosis

- Interviewing the patient results in change of probability for variable for “visit to Asia”



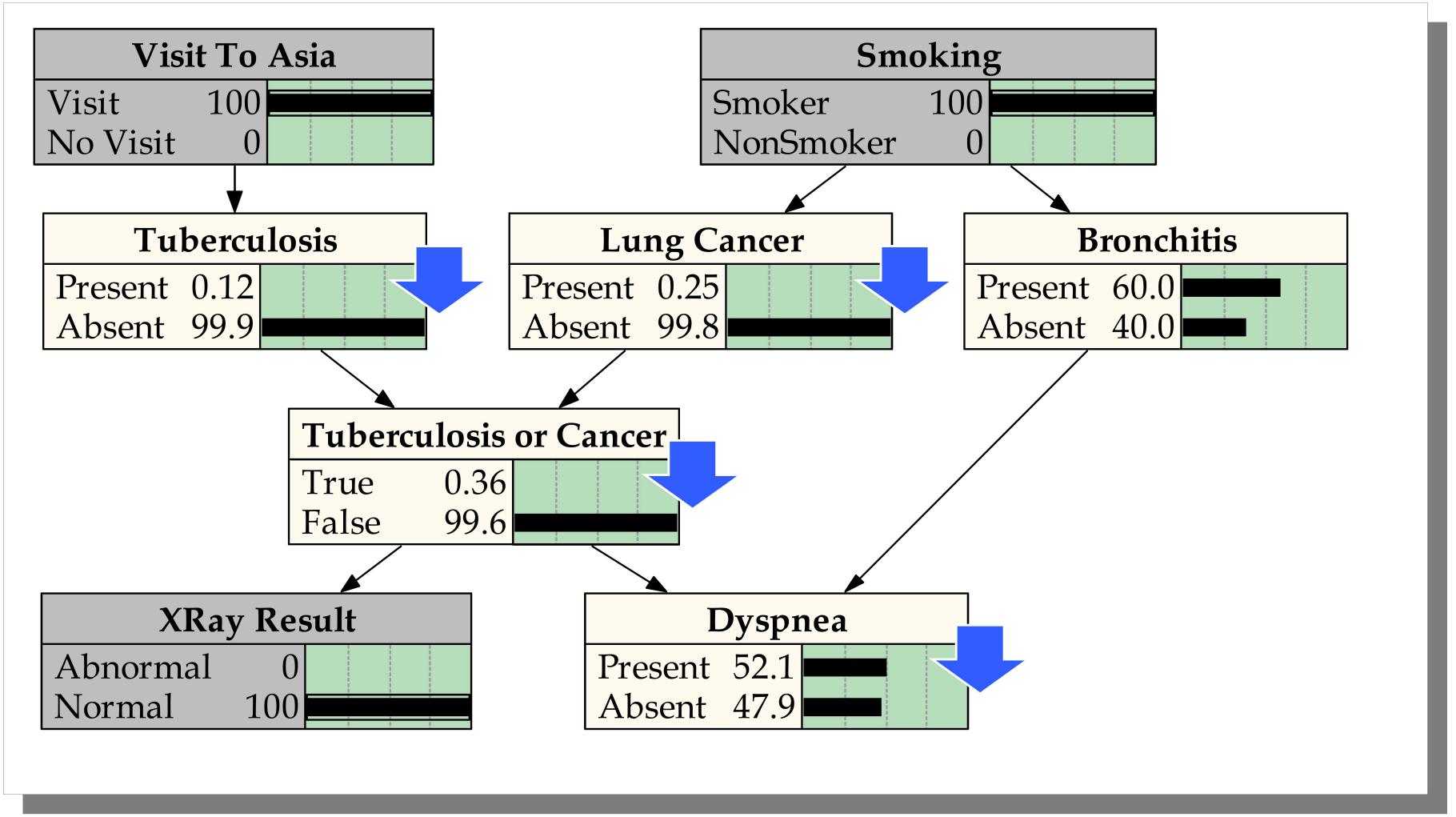
Example: Medical Diagnosis

- Patient is also a smoker...



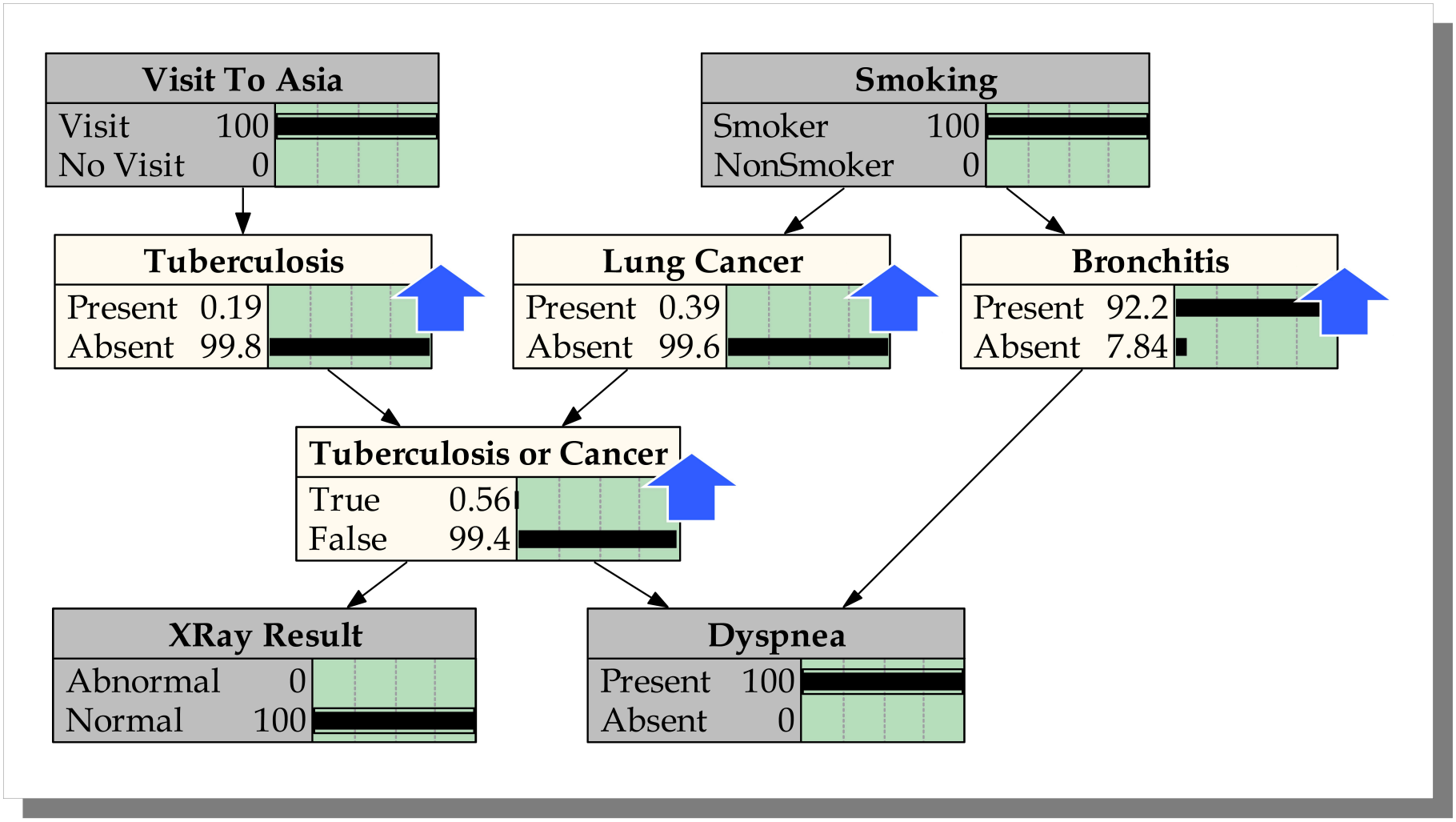
Example: Medical Diagnosis

- but fortunately the X-ray is normal...



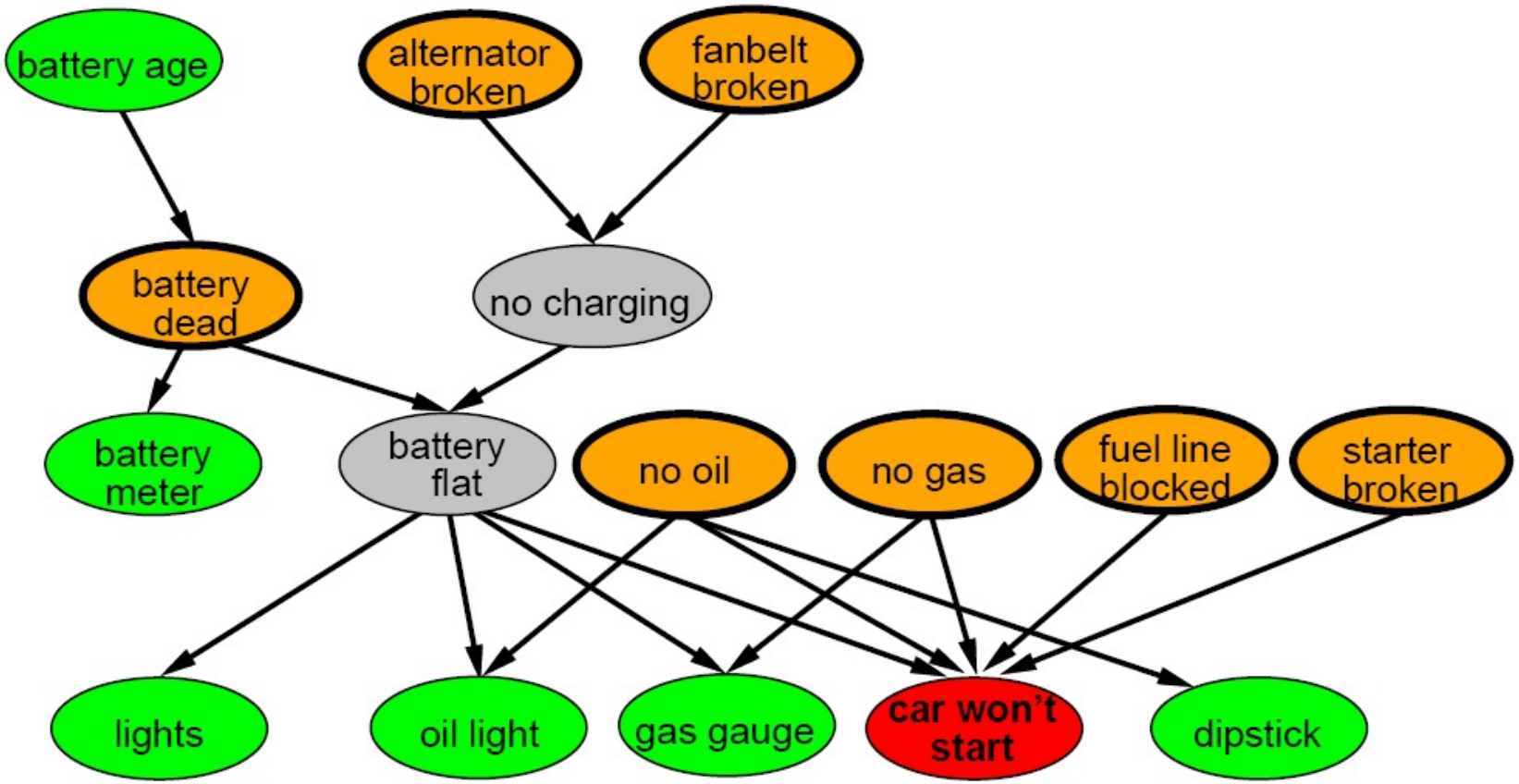
Example: Medical Diagnosis

- but then again patient has difficulty in breathing.



More Complex Example: Car Diagnosis

- Initial evidence: Car does not start
- Test variables
- Variables for possible failures
- Hidden variables: ensure spare structure, reduce parameters

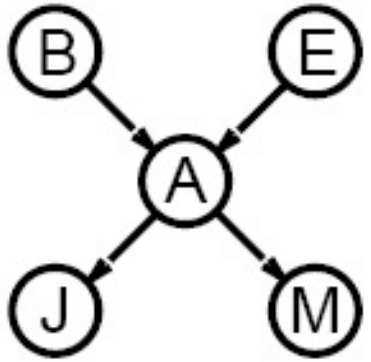


Compactness of a BN

A CPT for Boolean X_i with k Boolean parents has 2^k rows for the combinations of parent values

Each row requires one number p for $X_i = true$ (the number for $X_i = false$ is just $1 - p$)

If each variable has no more than k parents, the complete network requires $O(n \cdot 2^k)$ numbers



I.e., grows linearly with n, vs. $O(2^n)$ for the full joint distribution

For burglary net, $1 + 1 + 4 + 2 + 2 = 10$ numbers (vs. $2^5 - 1 = 31$)

Complexity of Exact Inference

Singly connected networks (or polytrees):

- any two nodes are connected by at most one (undirected) path
- time and space cost of variable elimination are $O(d^k n)$

Complexity of Exact Inference

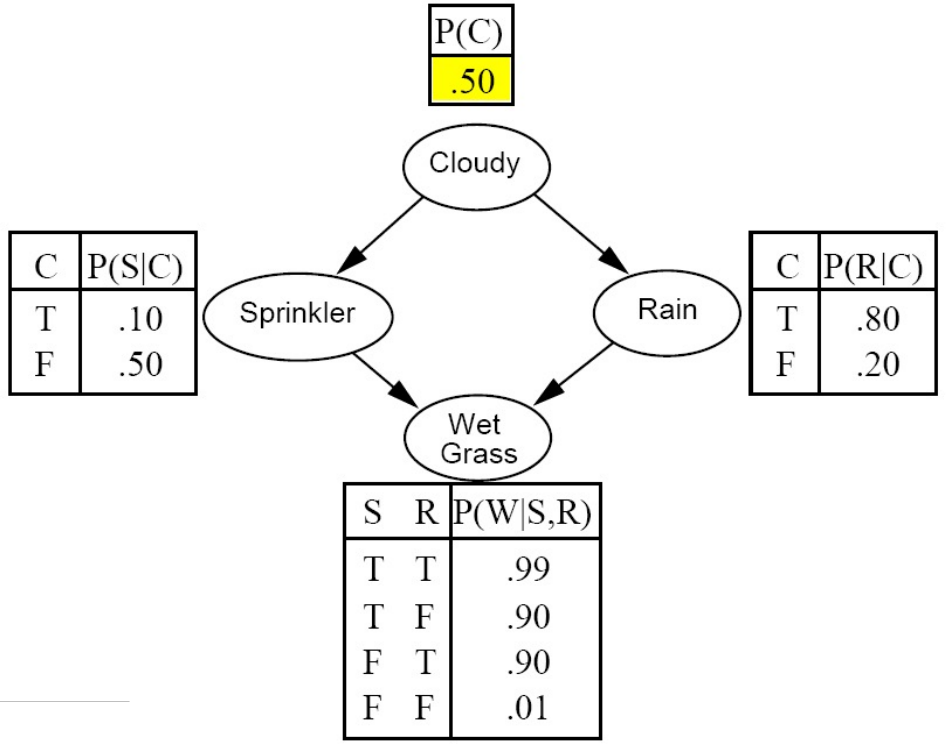
Singly connected networks (or polytrees):

- any two nodes are connected by at most one (undirected) path
- time and space cost of variable elimination are $O(d^k n)$

Multiply connected networks:

- can reduce 3SAT to exact inference $\Rightarrow$ NP-hard

Example:
Two paths from
Cloudy to Wet Grass



Complexity of Exact Inference

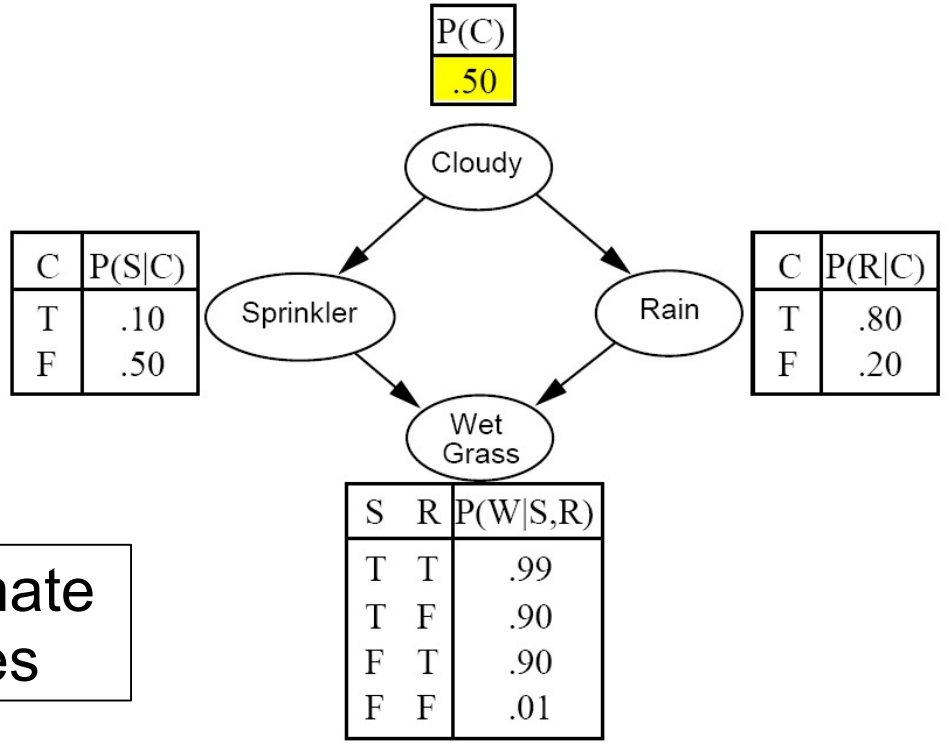
Singly connected networks (or polytrees):

- any two nodes are connected by at most one (undirected) path
- time and space cost of variable elimination are $O(d^k n)$

Multiply connected networks:

- can reduce 3SAT to exact inference $\Rightarrow$ NP-hard

Example:
Two paths from
Cloudy to Wet Grass



→ we "need" approximate inference techniques

Complexity of Inference

Theorem:

Inference in Bayesian networks (even approximate, without proof) is NP-hard

Inference Tasks

- **Simple queries**
  - compute the posterior marginal distribution for a variable
- **Conjunctive queries**
  - compute the posterior for a conjunction of variables

$$\mathbf{P}(X_i, X_j \mid \mathbf{E}=\mathrm{e}) = \mathbf{P}(X_i \mid \mathbf{E}=\mathrm{e}) \cdot \mathbf{P}(X_j \mid X_i, \mathbf{E}=\mathrm{e})$$

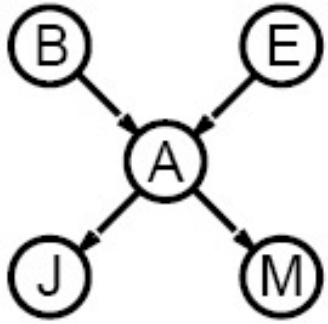
- **Optimal decisions**
  - decision networks include utility information
  - probabilistic inference required for $P(outcome|action,evidence)$
- **Value of Information**
  - Which evidence to seek next?
- **Sensitivity Analysis**
  - Which probability values are most critical?
- **Explanation**
  - Why do I need a new starter motor?

Inference by Enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:

$$\begin{aligned}&\mathbf{P}(B|j,m)\\&=\mathbf{P}(B,j,m)/P(j,m)\\&=\alpha\mathbf{P}(B,j,m)\\&=\alpha\sum_e\sum_a\mathbf{P}(B,e,a,j,m)\end{aligned}$$



Rewrite full joint entries using product of CPT entries:

$$\begin{aligned}&\mathbf{P}(B|j,m)\\&=\alpha\sum_e\sum_a\mathbf{P}(B)P(e)\mathbf{P}(a|B,e)P(j|a)P(m|a)\\&=\alpha\mathbf{P}(B)\sum_e P(e)\sum_a\mathbf{P}(a|B,e)P(j|a)P(m|a)\end{aligned}$$

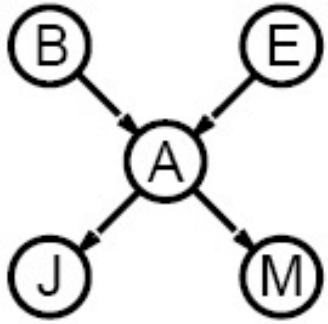
Inference by Enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:

$$\begin{aligned}&\mathbf{P}(B|j,m)\\&=\mathbf{P}(B,j,m)/P(j,m)\\&=\alpha\mathbf{P}(B,j,m)\\&=\alpha\sum_e\sum_a\mathbf{P}(B,e,a,j,m)\end{aligned}$$

Worst case: $O(n\,d^{\,n})$ time
$O(d^{\,n})$ terms, each consisting of a product of $O(n)$ probabilities



Rewrite full joint entries using product of CPT entries:

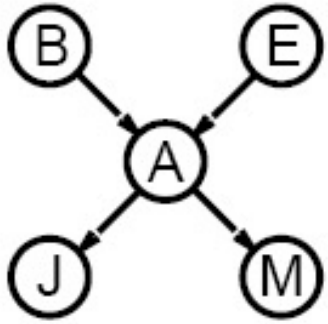
$$\begin{aligned}&\mathbf{P}(B|j,m)\\&=\alpha\sum_e\sum_a\mathbf{P}(B)P(e)\mathbf{P}(a|B,e)P(j|a)P(m|a)\\&=\alpha\mathbf{P}(B)\sum_e P(e)\sum_a\mathbf{P}(a|B,e)P(j|a)P(m|a)\end{aligned}$$

Inference by Enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:

$$\begin{aligned}&\mathbf{P}(B|j,m)\\&=\mathbf{P}(B,j,m)/P(j,m)\\&=\alpha\mathbf{P}(B,j,m)\\&=\alpha\,\Sigma_e\,\Sigma_a\,\mathbf{P}(B,e,a,j,m)\end{aligned}$$



Worst case: $O(n\,d^{\,n})$ time
$O(d^{\,n})$ terms, each consisting of a product of $O(n)$ probabilities

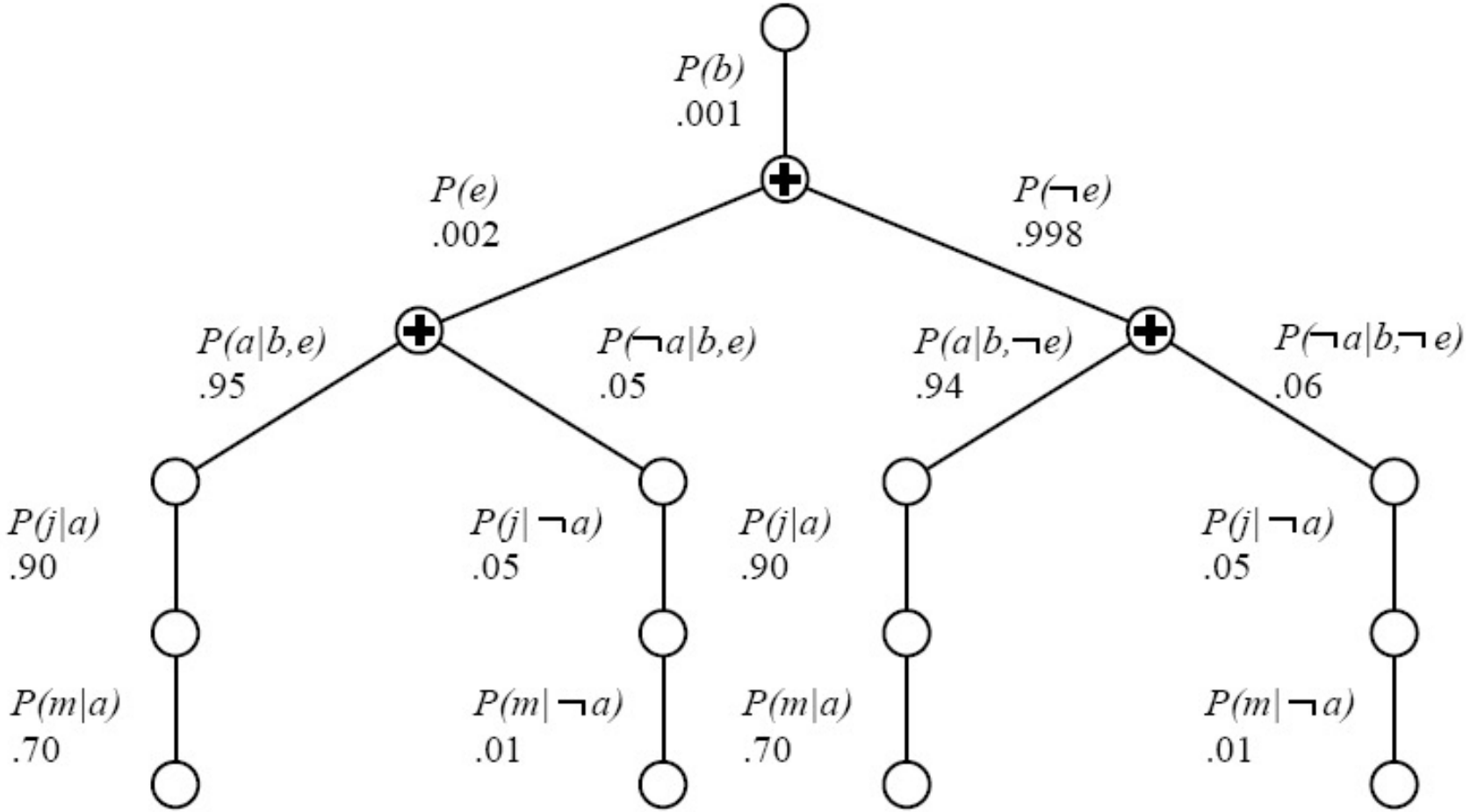
Rewrite full joint entries using product of CPT entries:

$$\begin{aligned}&\mathbf{P}(B|j,m)\\&=\alpha\,\Sigma_e\,\Sigma_a\,\mathbf{P}(B)P(e)\mathbf{P}(a|B,e)P(j|a)P(m|a)\\&=\alpha\mathbf{P}(B)\,\Sigma_e\,P(e)\,\Sigma_a\,\mathbf{P}(a|B,e)P(j|a)P(m|a)\end{aligned}$$

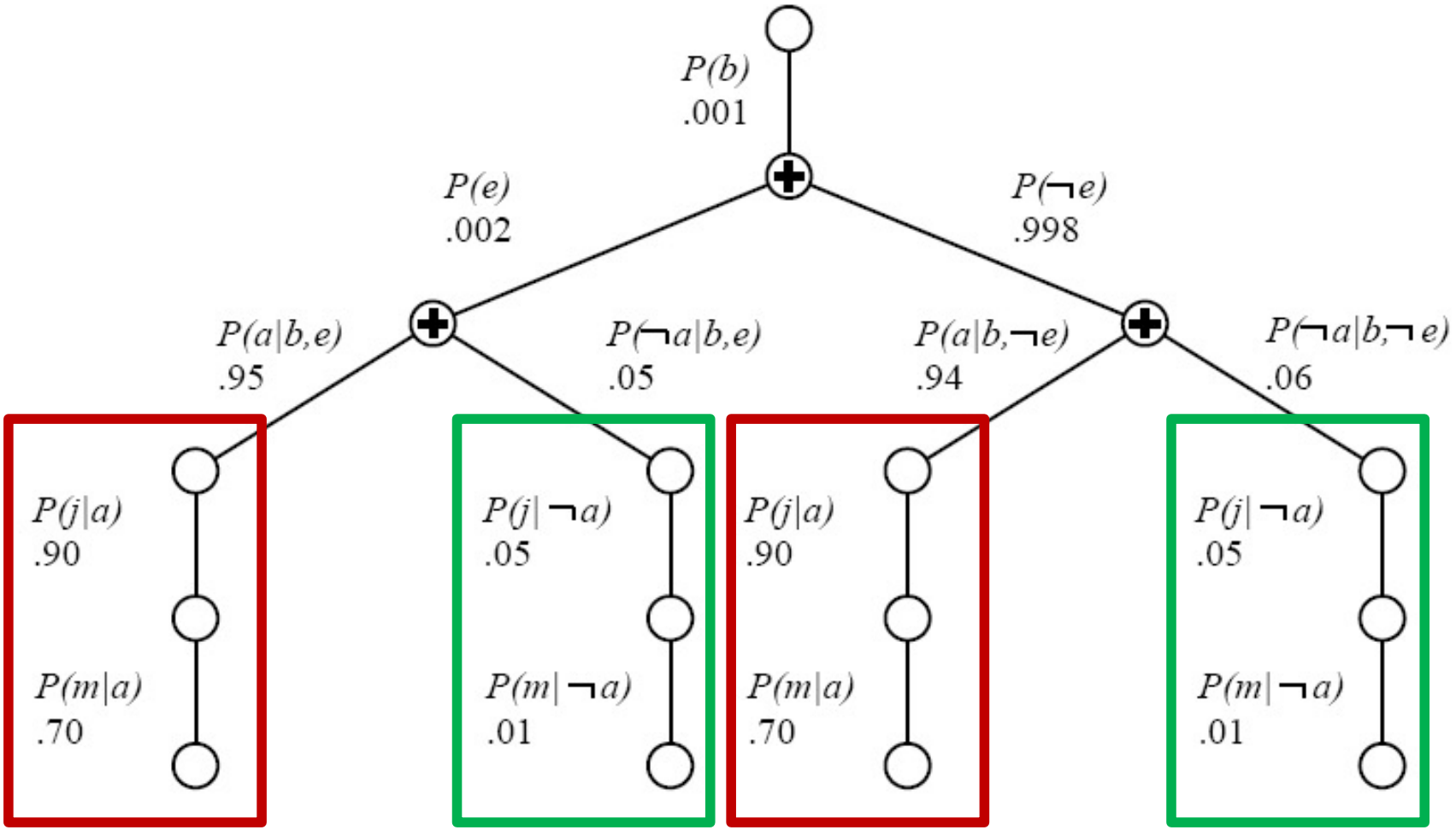
Recursive depth-first enumeration: $O(n)$ space, $O(d^n)$ time

where n is the number of variables and d is the number of values per variable

Evaluation Tree



Evaluation Tree



Enumeration is inefficient: repeated computation

e.g., computes $P(j|a)P(m|a)$ for each value of e

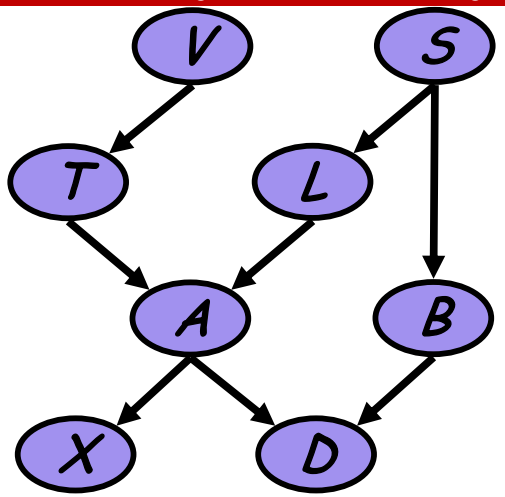
Variable Elimination

Move the sums into the products

- Key idea:
  - Do not multiply left-to-right but right-to-left.
  - Thus, terms that appear inside sums are evaluated first
    - intermediate results are stored as so-called **factors**
    - factors can be re-used several times in the same computation
  - is a form of **dynamic programming**
- Example: $\mathbf{P}(B|j,m)$

$$
\begin{aligned}
&= \alpha \underbrace{\mathbf{P}(B)}_{B} \Sigma_e \underbrace{P(e)}_{E} \Sigma_a \underbrace{\mathbf{P}(a|B,e)}_{A} \underbrace{P(j|a)}_{J} \underbrace{P(m|a)}_{M} \\
&= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) P(j|a) f_M(a) \\
&= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) f_J(a) f_M(a) \\
&= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a f_A(a,b,e) f_J(a) f_M(a) \\
&= \alpha \mathbf{P}(B) \Sigma_e P(e) f_{\bar{A}JM}(b,e) \text{ (sum out } A\text{)} \\
&= \alpha \mathbf{P}(B) f_{\bar{E}\bar{A}JM}(b) \text{ (sum out } E\text{)} \\
&= \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)
\end{aligned}
$$

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



Initial factors

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

Eliminate: *v*

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

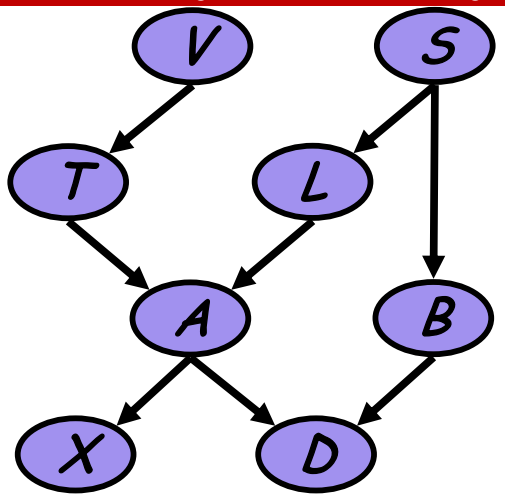
$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

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 eliminate: *v,s,x,t,l,a,b*



Initial factors

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

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

Eliminate: *s*

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

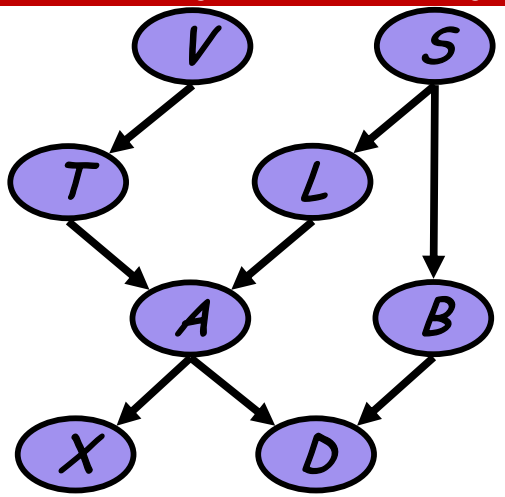
$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

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: *v,s,x,t,l,a,b*



Initial factors

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

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

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

Eliminate: *x*

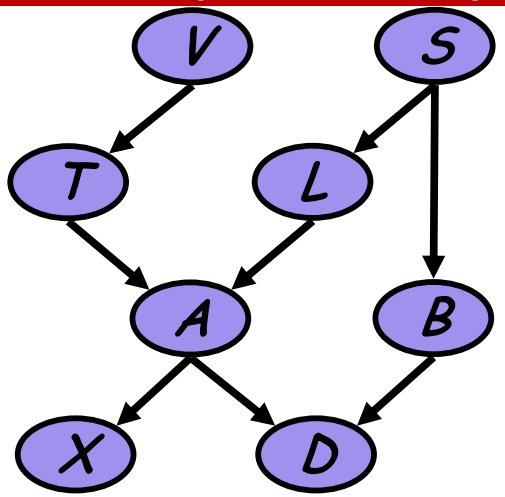
Compute: $f_x(a)=\sum_x P(x\mid a)$

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

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

We want to compute *P(d)*

Need to eliminate: *v,s,x,t,l,a,b*



Initial factors

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

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

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

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

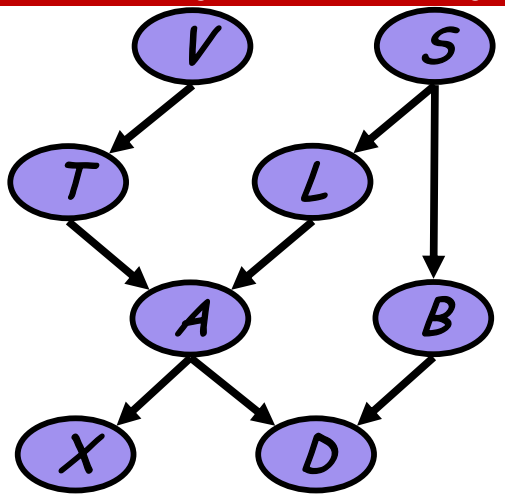
Eliminate: *t*

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

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

We want to compute *P(d)*

Need to eliminate: *v,s,x,t,l,a,b*



Initial factors

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

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

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

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

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

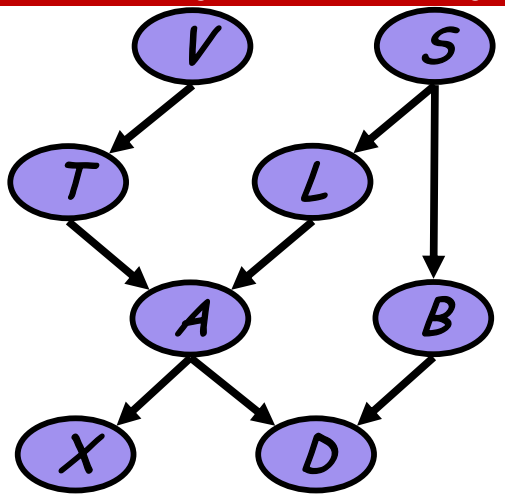
Eliminate: *l*

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

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

We want to compute *P(d)*

Need to eliminate: *v,s,x,t,l,a,b*



Initial factors

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

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

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

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

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

$$\Rightarrow \underline{f_l(a,b)}\,\underline{f_x(a)}\,\underline{P(d\mid a,b)} \Rightarrow \underline{f_a(b,d)} \Rightarrow \underline{f_b(d)}$$

Eliminate: *a,b*

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

Inference by Stochastic Simulation
(Sampling from a Bayesian Network)

Basic idea:

1) Draw N samples from a sampling distribution S
2) Compute an approximate posterior probability $\hat{P}$
3) Show this converges to the true probability P

Outline:

- Sampling from an empty network
- Rejection sampling: reject samples disagreeing with evidence
- Likelihood weighting: use evidence to weight samples
- Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior

How to draw a sample ?

- Given random variable X, D(X)={0, 1}
- Given P(X) = {0.3, 0.7}

How to draw a sample ?

- Given random variable X, D(X)={0, 1}
- Given P(X) = {0.3, 0.7}

- **Sample X ← P (X)**
  - **draw random number r ∈ [0, 1]**
  - **If (r < 0.3) then set X=0**
  - **Else set X=1**

- Can generalize for any domain size

Markov Chain Monte Carlo (MCMC) Sampling

"State" of network = current assignment to all variables.

Generate next state by sampling one variable given Markov blanket
Sample each variable in turn, keeping evidence fixed

function MCMC-Ask$(X, \mathbf{e}, bn, N)$ **returns** an estimate of $P(X|\mathbf{e})$
  local variables: $\mathbf{N}[X]$, a vector of counts over X, initially zero
    $\mathbf{Z}$, the nonevidence variables in bn
    $\mathbf{x}$, the current state of the network, initially copied from $\mathbf{e}$

  initialize $\mathbf{x}$ with random values for the variables in $\mathbf{Y}$
  for $j = 1$ to N **do**
    for each Z_i in $\mathbf{Z}$ **do**
      sample the value of Z_i in $\mathbf{x}$ from $\mathbf{P}(Z_i|mb(Z_i))$
        given the values of $MB(Z_i)$ in $\mathbf{x}$
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in $\mathbf{x}$
  return Normalize$(\mathbf{N}[X])$

Gibbs Sampling

Can also choose a variable to sample at random each time

Ordered Gibbs Sampler

- Generate sample x^{t+1} from x^t :

Process all variables in some order

$$X_1 = x_1^{t+1} \leftarrow P(x_1 \mid x_2^t, x_3^t, \ldots, x_N^t, e)$$

$$X_2 = x_2^{t+1} \leftarrow P(x_2 \mid x_1^{t+1}, x_3^t, \ldots, x_N^t, e)$$

...

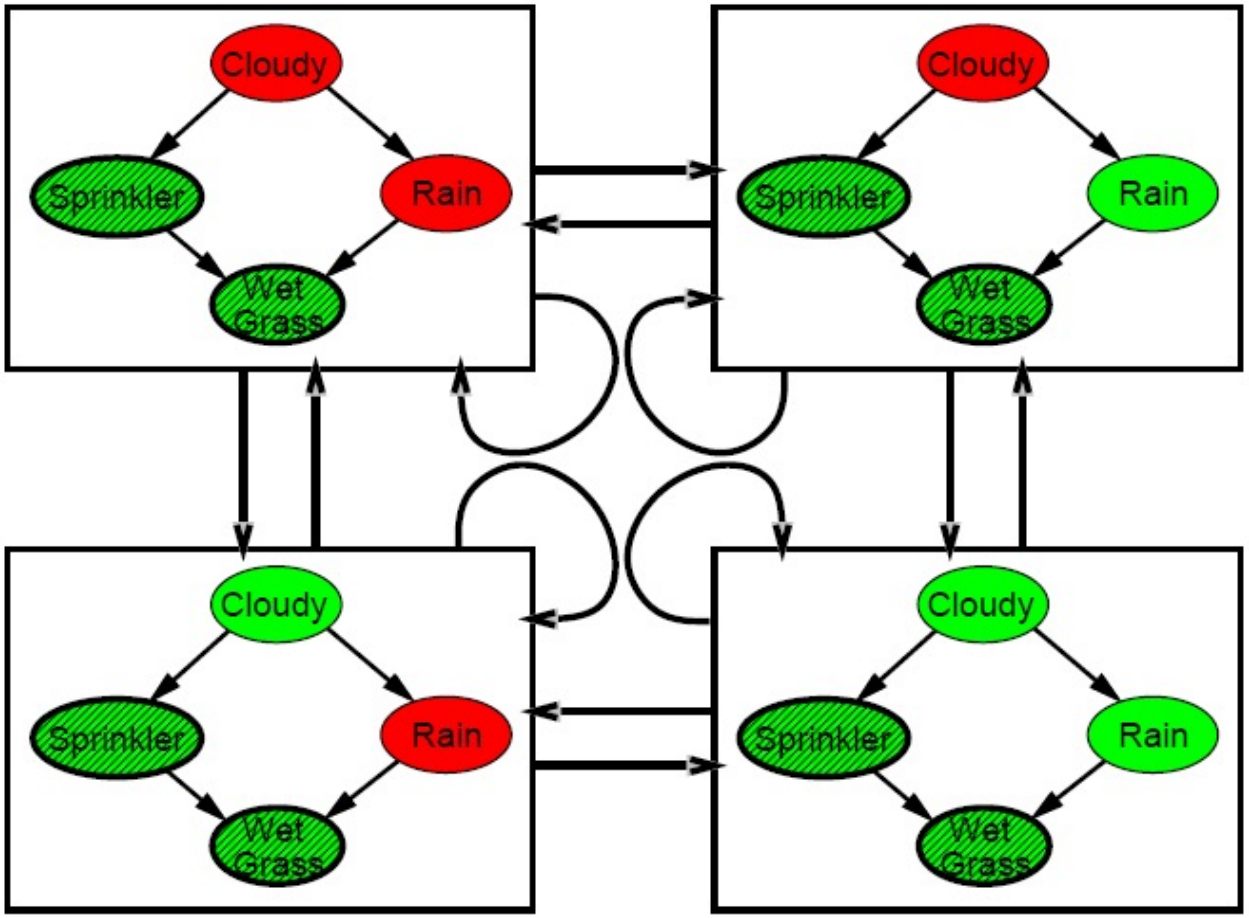
$$X_N = x_N^{t+1} \leftarrow P(x_N \mid x_1^{t+1}, x_2^{t+1}, \ldots, x_{N-1}^{t+1}, e)$$

- In short, for i=1 to N:

$$X_i = x_i^{t+1} \leftarrow \textbf{sampled from } P(x_i \mid x^t \setminus x_i, e)$$

The Markov Chain

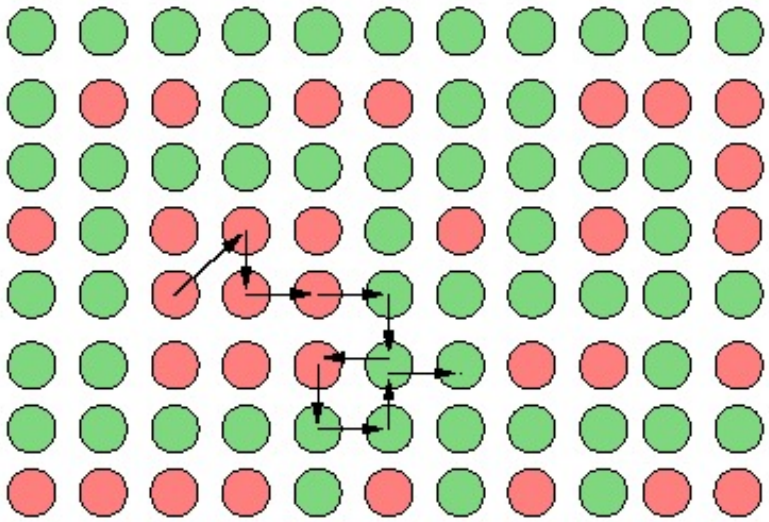
With $Sprinkler = true, WetGrass = true$, there are four states:



Wander about for a while, average what you see

Gibbs Sampling: Illustration

The process of Gibbs sampling can be understood as a *random walk* in the space of all instantiations with $\boldsymbol{Y} = \boldsymbol{u}$:



Reachable in one step: instantiations that differ from current one by value assignment to at most one variable (assume randomized choice of variable X_k).

Guaranteed to converge iff chain is :

- irreducible (every state reachable from every other state)
- aperiodic (returns to state i can occur at irregular times)
- ergodic (returns to every state with probability 1)

Example

Estimate $\mathbf{P}(Rain|Sprinkler = true, WetGrass = true)$

Sample $Cloudy$ or $Rain$ given its Markov blanket, repeat.
Count number of times $Rain$ is true and false in the samples.

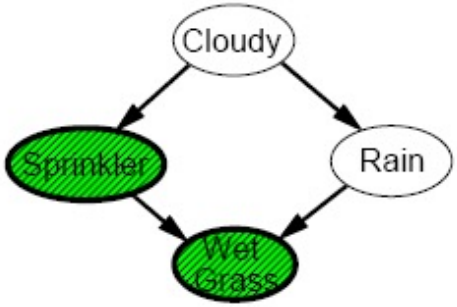
E.g., visit 100 states
31 have $Rain = true$, 69 have $Rain = false$

$\hat{\mathbf{P}}(Rain|Sprinkler = true, WetGrass = true)$
$= \textsc{Normalize}(\langle 31, 69 \rangle) = \langle 0.31, 0.69 \rangle$

Theorem: chain approaches stationary distribution:
long-run fraction of time spent in each state is exactly proportional to its posterior probability

Markov Blanket Sampling

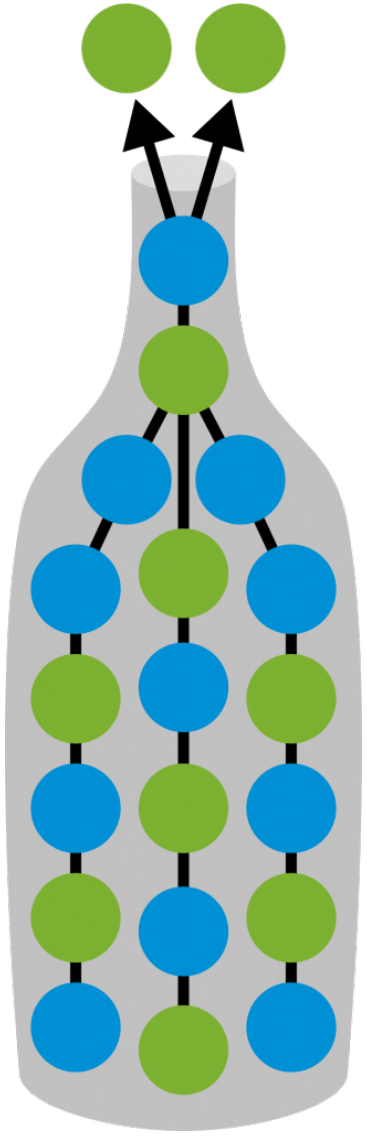
Markov blanket of $Cloudy$ is
$Sprinkler$ and $Rain$
Markov blanket of $Rain$ is
$Cloudy$, $Sprinkler$, and $WetGrass$



Probability given the Markov blanket is calculated as follows:

$$P(x_i'|mb(X_i)) = P(x_i'|parents(X_i))\Pi_{Z_j \in Children(X_i)} P(z_j|parents(Z_j))$$

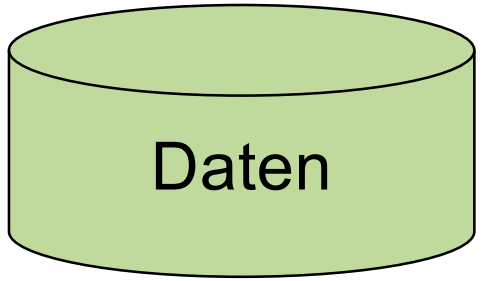
Where do the numbers come from?



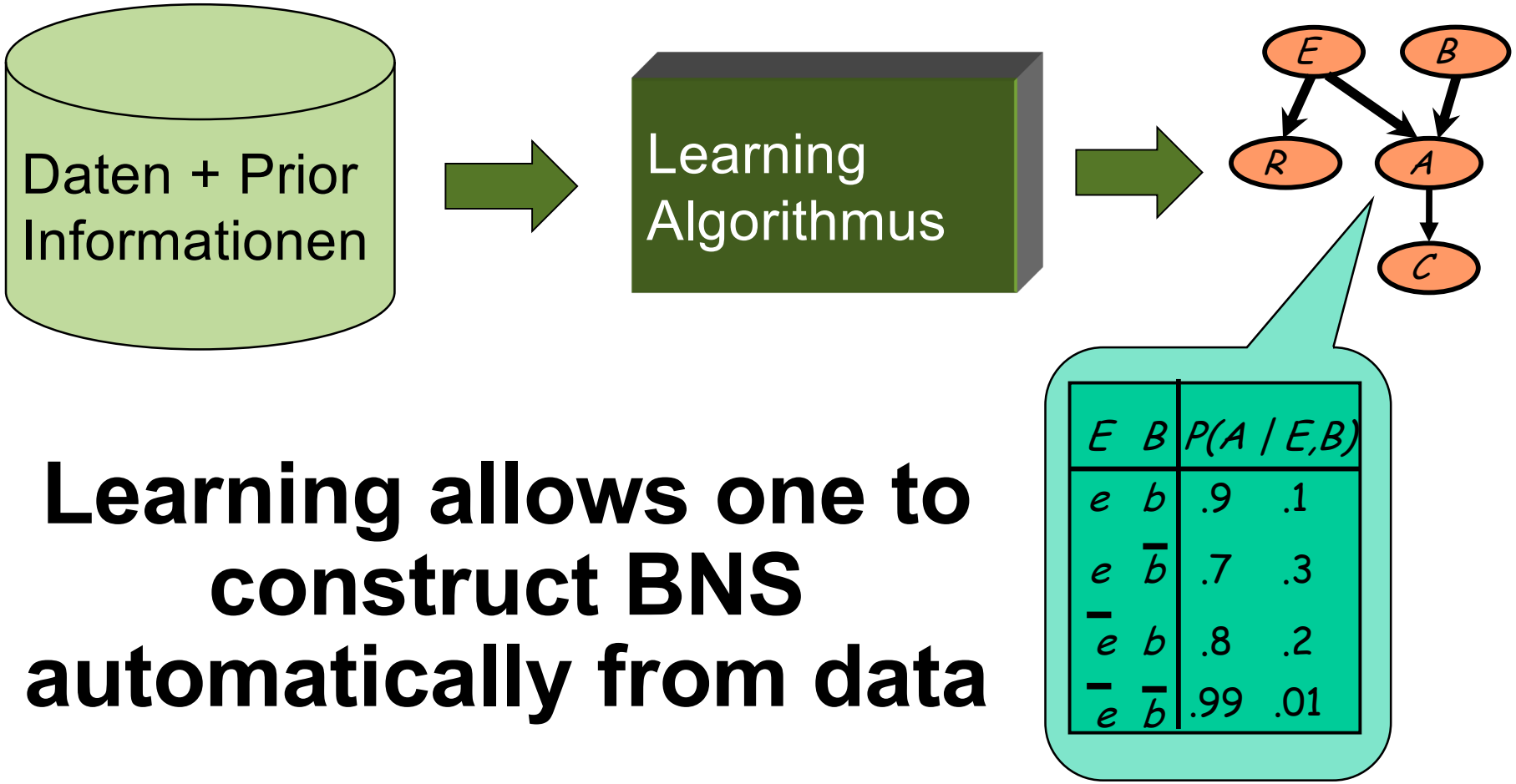
Knowledge acquisition bottleneck

- Experts are expensive
- It is difficilut to get hands on experts

Data is cheap



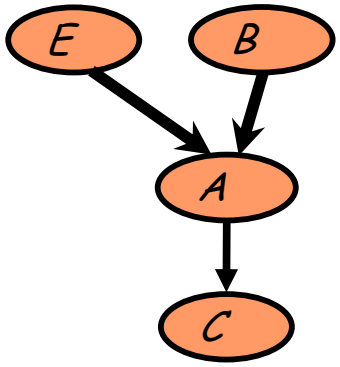
Independencies and structure provide insights into the domain at hand



Learning allows one to construct BNS automatically from data

Easiest setup: Learning the parameters from fully observed data

Structure of the BN is given. The Conditional Distributions associated with the nodes have to be estimated from data



Random Varaibles

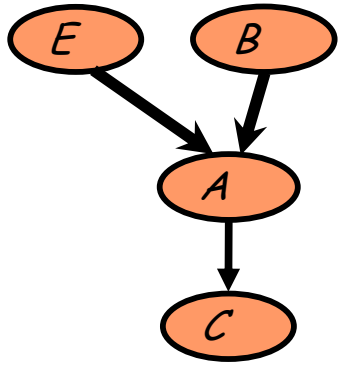
$$D = \begin{bmatrix} E[1] & B[1] & A[1] & C[1] \\ E[2] & B[2] & A[2] & C[2] \\ \vdots & \vdots & \vdots & \vdots \\ E[M] & B[M] & A[M] & C[M] \end{bmatrix}$$

Training examples (joint configurations of all random variables)

Likelihood-Function L

How likely are the data given the current model

We assume that the examples are independent.
Thus we can multiply together their probabilities



$$L(\Theta|D) = \prod_m P(E[m], B[m], A[m], C[m]|\Theta)$$

$$= \prod_m \begin{bmatrix} P(E[m]|\Theta) \\ P(B[m]|\Theta) \\ P(A[m]|B[m], E[m], \Theta) \\ P(C[m]|A[m], \Theta) \end{bmatrix} = \begin{matrix} \prod_m P(E[m]|\Theta) \\ \prod_m P(B[m]|\Theta) \\ \prod_m P(A[m]|B[m], E[m], \Theta) \\ \prod_m P(C[m]|A[m], \Theta) \end{matrix}$$

BN Factorization **Product now over the examples**

The likelihood function decomposes into local computations

Maximum-Likelihood Estimation (MLE)

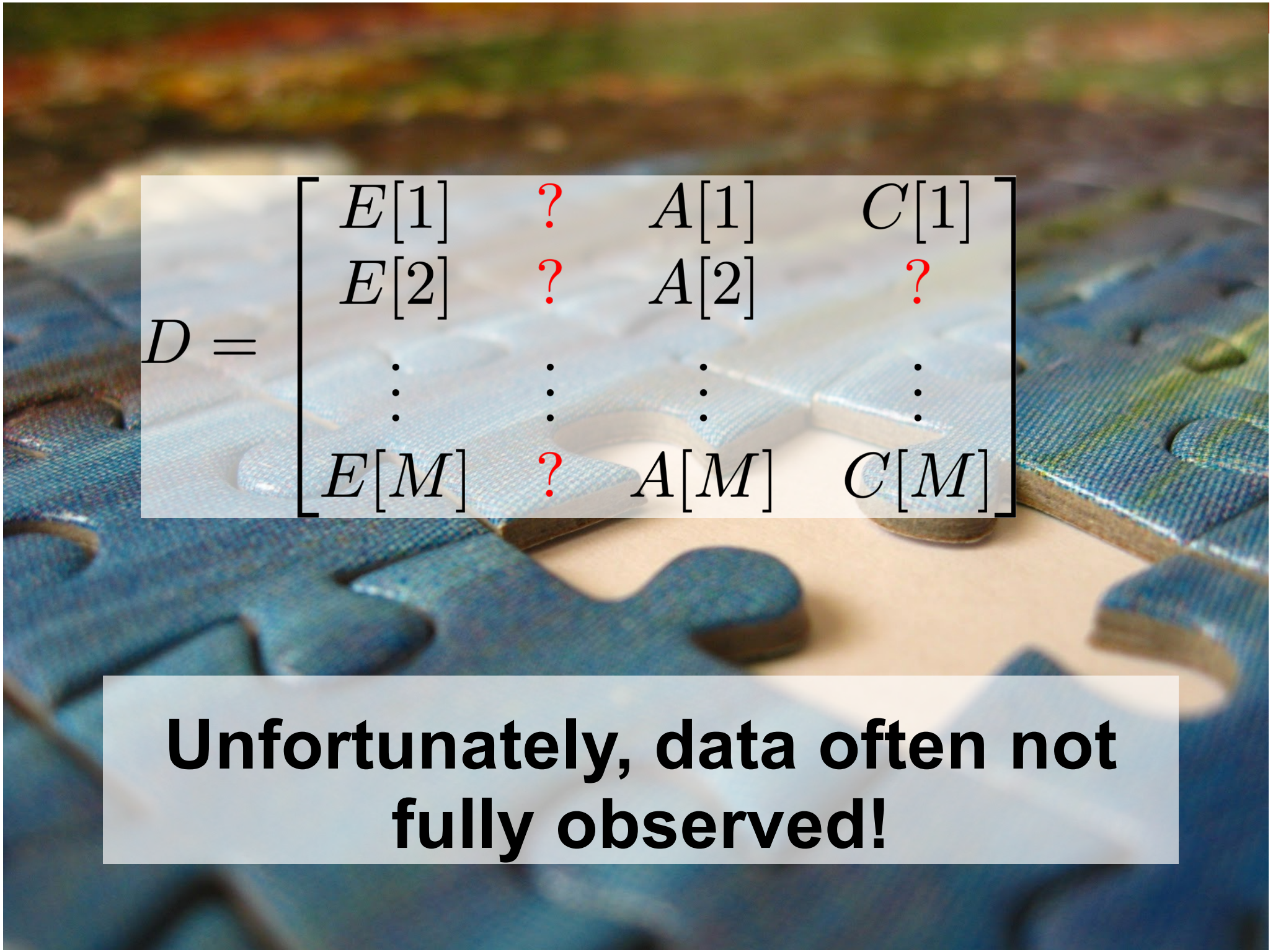
Given compelte data we compute **sufficient statistics**

Multinomial RVs: counts how often RVs are in a particular configuration

$$N(x_i, pa_i)$$

Then the next estimate of the best parameters

$$\hat{\theta}_{x_i|pa_i} = \frac{N(x_i, pa_i)}{N(pa_i)}$$


$D = \begin{bmatrix} E[1] & ? & A[1] & C[1] \\ E[2] & ? & A[2] & ? \\ \vdots & \vdots & \vdots & \vdots \\ E[M] & ? & A[M] & C[M] \end{bmatrix}$
Unfortunately, data often not fully observed!

Expectation Maximization (EM)

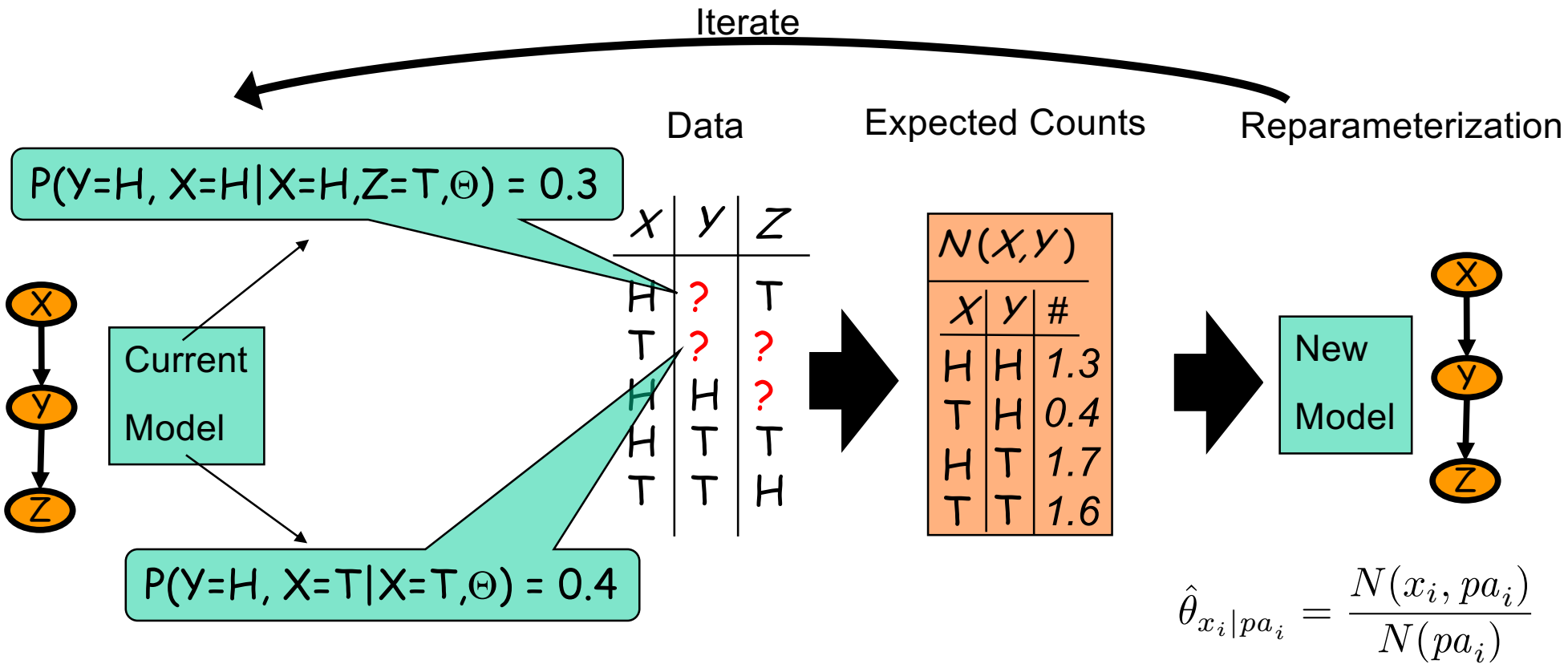
General framework for learning if data is partially observed

Intuition:

- If **data is complete**, ML parameter estimation is easy:
  - **simple counting** (1 iteration)
- But what if there are missing values, i.e., we are facing **incomplete data**?

1. **Complete data** (Imputation)
   - most probable?, average?, ... value
2. **Count**
3. **Iterate**

Expectation Maximization (EM)



$$\hat{\theta}_{x_i|pa_i} = \frac{N(x_i, pa_i)}{N(pa_i)}$$

Since we do something wrong we iterate

Real-World Applications of BN

- Industrial
- Processor Fault Diagnosis - by Intel
- Auxiliary Turbine Diagnosis - GEMS by GE
- Diagnosis of space shuttle propulsion systems - VISTA by NASA/Rockwell
- Situation assessment for nuclear power plant – NRC

- Military
- Automatic Target Recognition - MITRE
- Autonomous control of unmanned underwater vehicle - Lockheed Martin
- Assessment of Intent

Real-World Applications of BN

- Medical Diagnosis (also at TUDA)
- Internal Medicine
- Pathology diagnosis - Intellipath by Chapman & Hall
- Breast Cancer Manager with Intellipath

- NLP, CV, Cognitive Science, Plant Phenotpying (also at TUDA)

- Commercial
- Financial Market Analysis
- Information Retrieval
- Software troubleshooting and advice - Windows 95 & Office 97
- Pregnancy and Child Care – Microsoft
- Software debugging - American Airlines' SABRE online reservation system