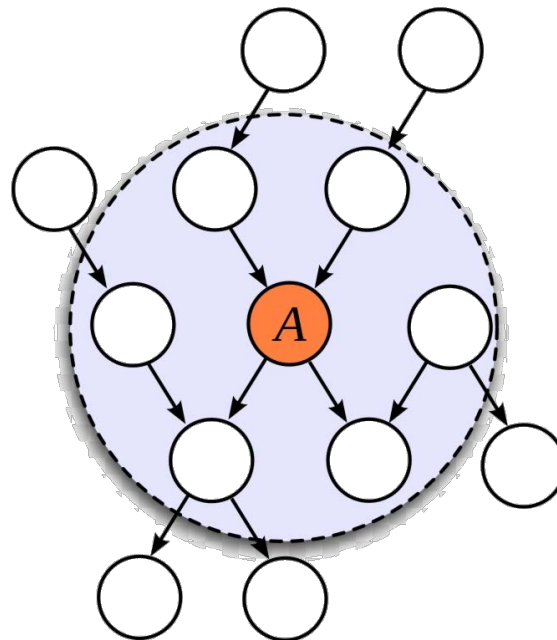


Bayesian Networks

Bang Liu, Jian-Yun Nie

IFT3335: Introduction to Artificial Intelligence



(Adapted from Philipp Koehn, JHU 601.464/664 Artificial Intelligence)

Outline

- Bayesian Networks
- Parameterized distributions
- Exact inference
- Approximate inference

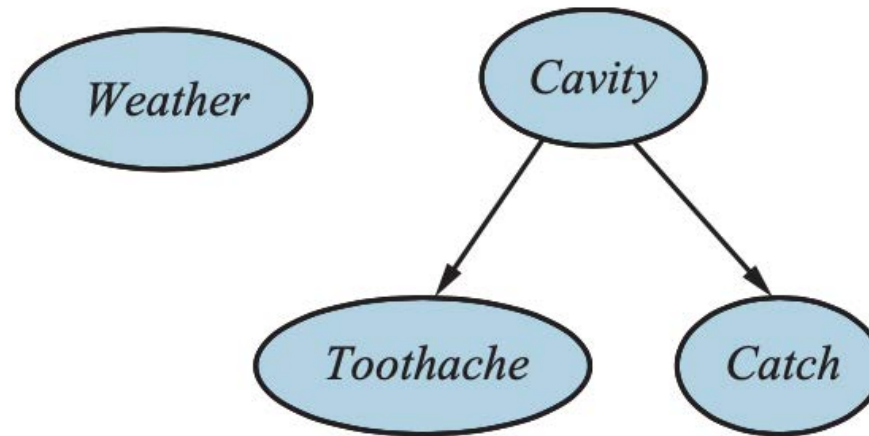
bayesian networks

Bayesian Networks

- A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions
- Syntax
 - a set of nodes, one per variable
 - a directed, acyclic graph (link \approx “directly influences”)
 - a conditional distribution for each node given its parents:
 $\mathbf{P}(X_i | Parents(X_i))$
- In the simplest case, conditional distribution represented as a **conditional probability table** (CPT) giving the distribution over X_i for each combination of parent values

Example

- Topology of network encodes conditional independence assertions:



- *Weather* is independent of the other variables
- *Toothache* and *Catch* are conditionally independent given *Cavity*

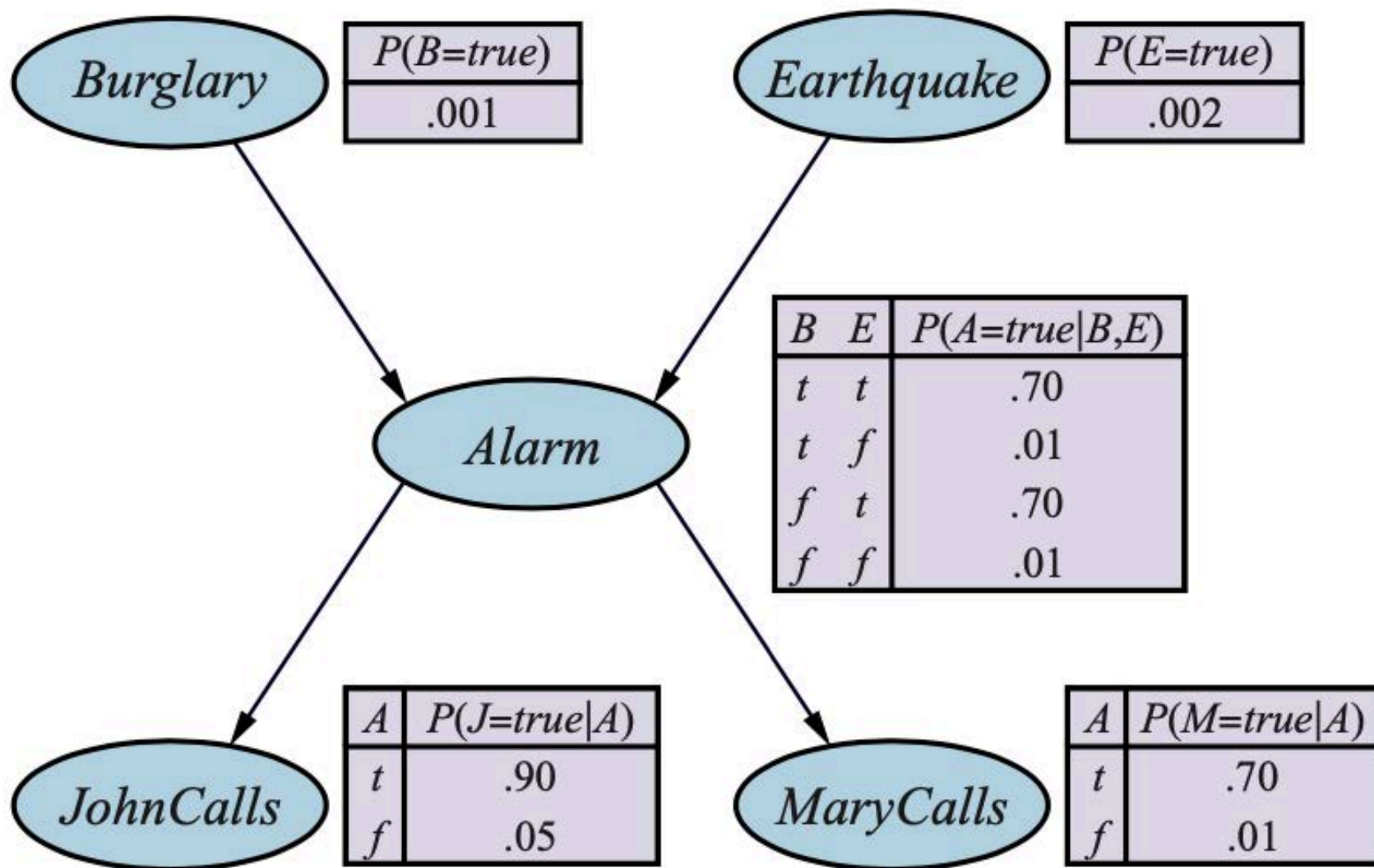
Example

- *I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes.*

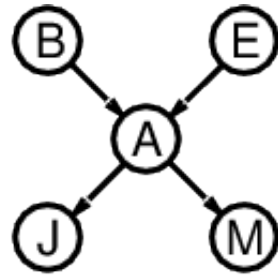
Is there a burglar?■

- Variables: *Burglar, 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

Example

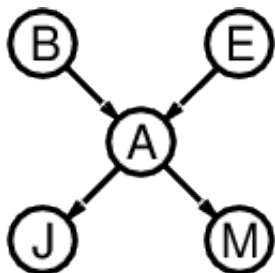


Compactness



- A conditional probability table 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. $2^5 - 1 = 31$)

Global Semantics



- **Global** semantics defines the full joint distribution as the product of the local conditional distributions:

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{parents}(X_i))$$

- E.g., $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$ ■

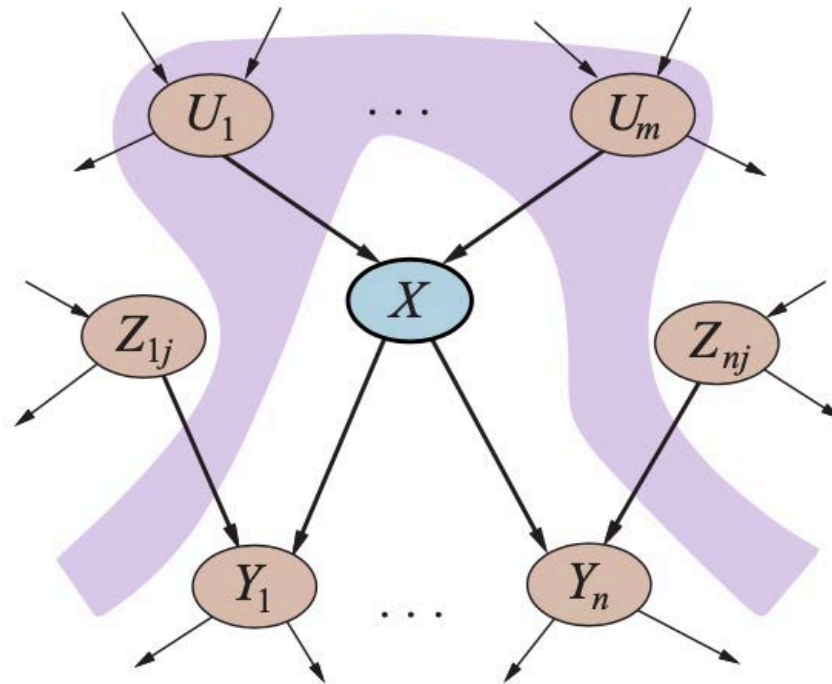
$$= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e)$$

$$= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998$$

$$\approx 0.00063$$

Local Semantics

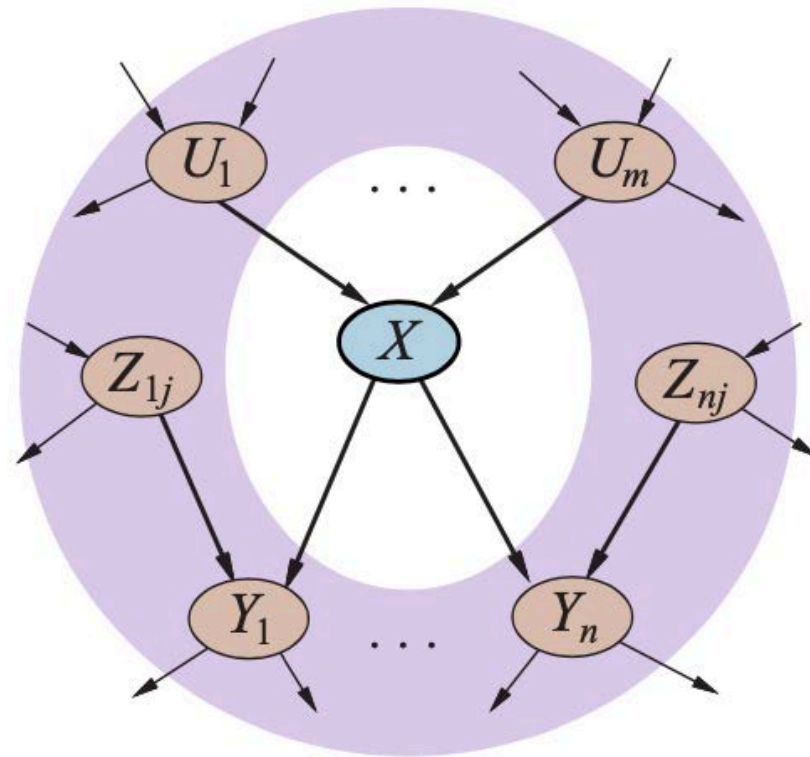
- **Local semantics:** each node is conditionally independent of its nondescendants given its parents



- Theorem: Local semantics \Leftrightarrow global semantics

Markov Blanket

- Each node is conditionally independent of all others given its **Markov blanket**: parents + children + children's parents
(Exercise: prove this.)



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, \dots, X_n
 2. For $i = 1$ to n
 - add X_i to the network
 - select parents from X_1, \dots, X_{i-1} such that
$$\mathbf{P}(X_i | Parents(X_i)) = \mathbf{P}(X_i | X_1, \dots, X_{i-1})$$
- Any order will work, but the resulting network will be more compact if the variables are ordered such that causes precede effects.
- This choice of parents guarantees the global semantics:

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

Example

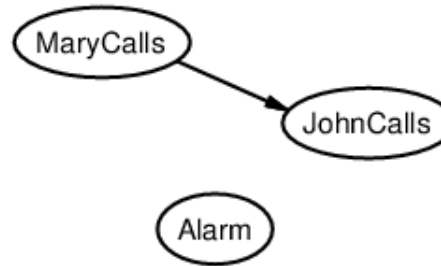
- Suppose we choose the ordering M, J, A, B, E



- $P(J|M) = P(J)$?

Example

- Suppose we choose the ordering M, J, A, B, E

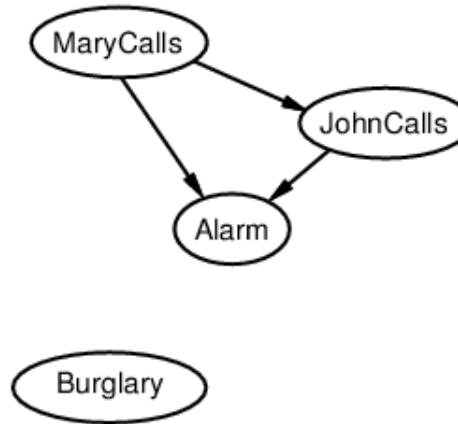


- $P(J|M) = P(J)$? No■
- $P(A|J, M) = P(A|J)$? $P(A|J, M) = P(A)$?

Add JohnCalls: If Mary calls, that probably means the alarm has gone off, which makes it more likely that John calls. Therefore, *JohnCalls* needs *MaryCalls* as a parent.

Example

- Suppose we choose the ordering M, J, A, B, E

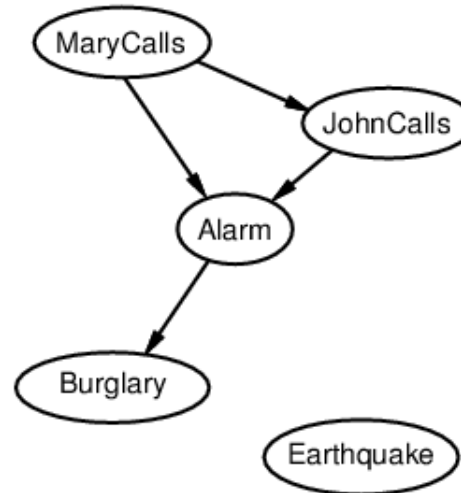


- $P(J|M) = P(J)$? No
- $P(A|J, M) = P(A|J)$? $P(A|J, M) = P(A)$? No
- $P(B|A, J, M) = P(B|A)$?
- $P(B|A, J, M) = P(B)$?

Add Alarm: Clearly, if both call, it is more likely that the alarm has gone off than if just one or neither calls, so we need both *MaryCalls* and *JohnCalls* as parents.

Example

- Suppose we choose the ordering M, J, A, B, E

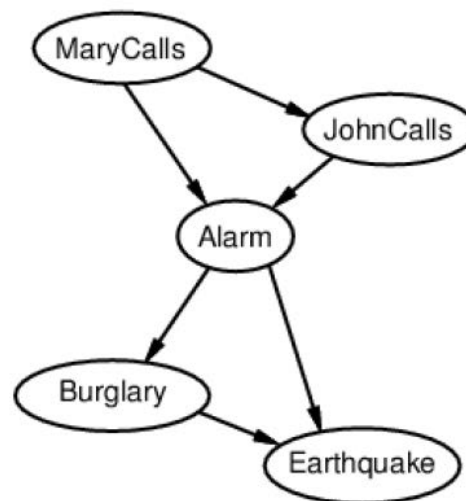


- $P(J|M) = P(J)$? No
- $P(A|J, M) = P(A|J)$? $P(A|J, M) = P(A)$? No
- $P(B|A, J, M) = P(B|A)$? Yes
- $P(B|A, J, M) = P(B)$? No
- $P(E|B, A, J, M) = P(E|A)$?
- $P(E|B, A, J, M) = P(E|A, B)$?

Add *Burglary*: If we know the alarm state, then the call from John or Mary cannot give us more information about burglary. Hence we need just Alarm as parent.

Example

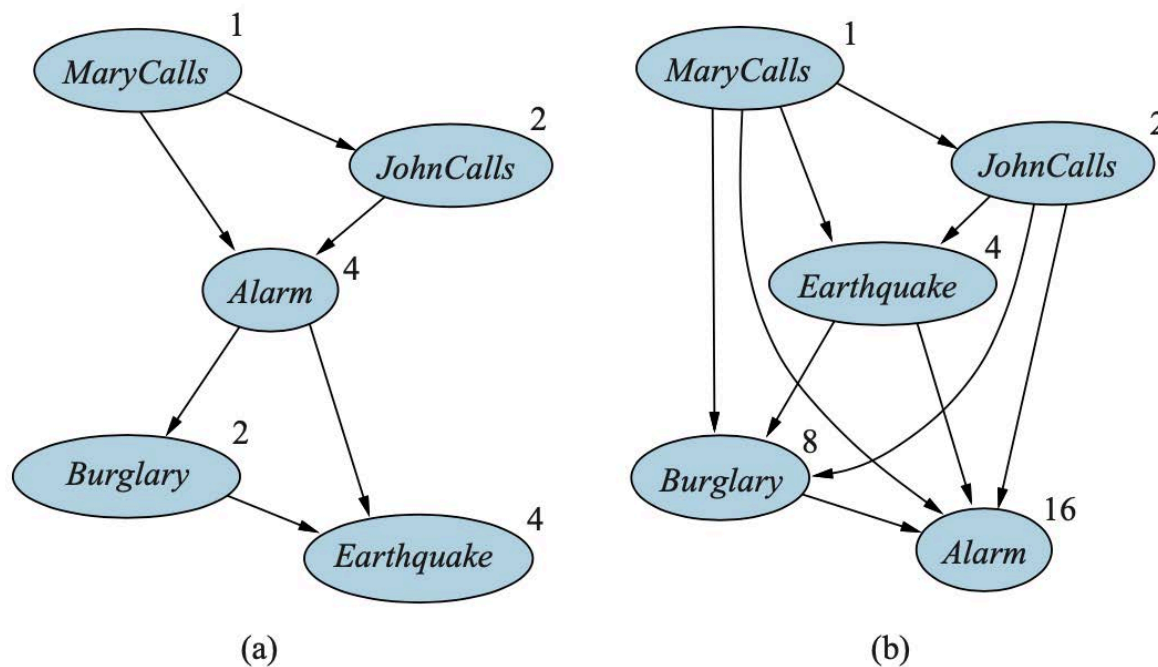
- Suppose we choose the ordering M, J, A, B, E



- $P(J|M) = P(J)$? No
- $P(A|J, M) = P(A|J)$? $P(A|J, M) = P(A)$? No
- $P(B|A, J, M) = P(B|A)$? Yes
- $P(B|A, J, M) = P(B)$? No
- $P(E|B, A, J, M) = P(E|A)$? No
- $P(E|B, A, J, M) = P(E|A, B)$? Yes

Adding *Earthquake*: If the alarm is on, it is more likely that there has been an earthquake. But if we know that there has been a burglary, then that explains the alarm, and the probability of an earthquake would be only slightly above normal. Hence, we need both *Alarm* and *Burglary* as parents.

Example



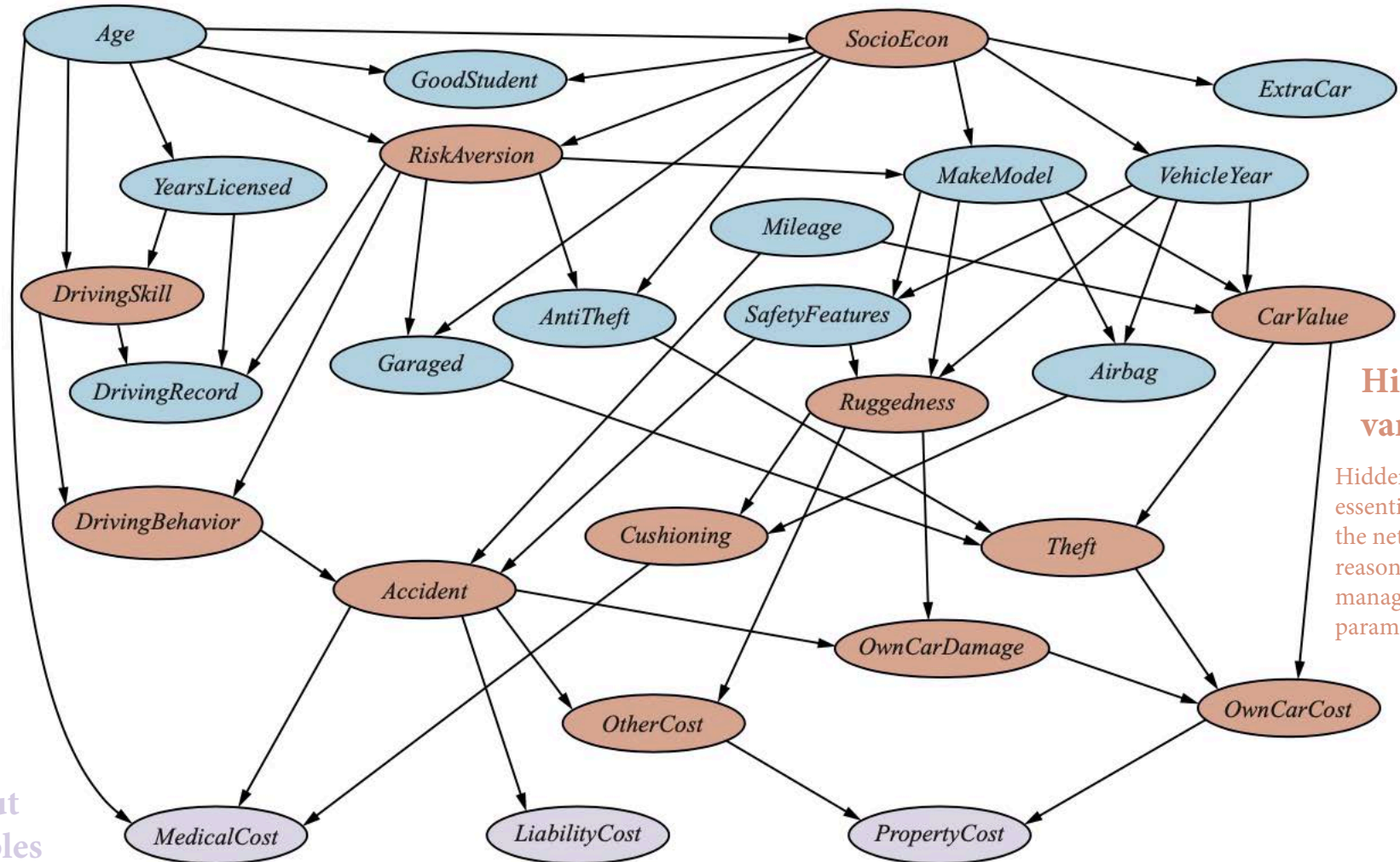
- Deciding conditional independence is hard in noncausal directions
- Assessing conditional probabilities is hard in noncausal directions
- Network is less compact: $1 + 2 + 4 + 2 + 4 = 13$ numbers needed

Figure (b) shows a very bad node ordering: it requires 31 distinct probabilities to be specified—exactly the same number as the full joint distribution.

Example: Car Insurance

A car insurance company receives an application from an individual to insure a specific vehicle and must decide on the appropriate annual premium to charge, based on the anticipated claims it will pay out for this applicant. The task is to build a Bayes net that captures the causal structure of the domain and gives an accurate, well-calibrated distribution over the output variables given the evidence available from the application form (See AIMA 13.2.4)

Input variables



Hidden variables

Hidden variables are essential for structuring the network so that it is reasonably sparse with a manageable number of parameters.

Output variables

Compact Conditional Distributions

- CPT grows exponentially with number of parents
CPT becomes infinite with continuous-valued parent or child
- Solution: **canonical** distributions that are defined compactly
- **Deterministic** nodes are the simplest case:
 $X = f(\text{Parents}(X))$ for some function f
- E.g., Boolean functions
 $\text{NorthAmerican} \Leftrightarrow \text{Canadian} \vee \text{US} \vee \text{Mexican}$
- E.g., numerical relationships among continuous variables

$$\frac{\partial \text{Level}}{\partial t} = \text{inflow} + \text{precipitation} - \text{outflow} - \text{evaporation}$$

Compact Conditional Distributions

- Noisy-OR distributions model multiple noninteracting causes
 - parents $U_1 \dots U_k$ include all causes (can add **leak node**)
 - independent failure probability q_i for each cause alone

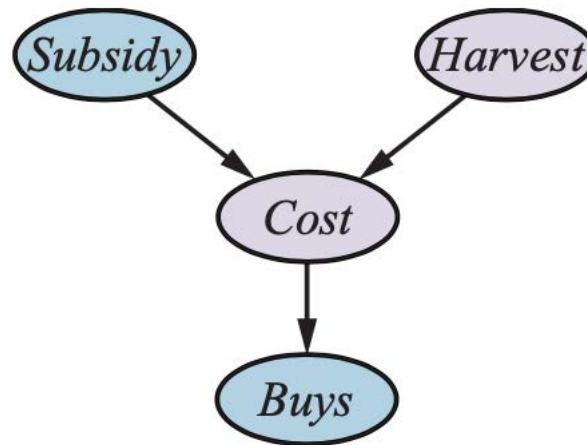
$$\implies P(X|U_1 \dots U_j, \neg U_{j+1} \dots \neg U_k) = 1 - \prod_{i=1}^j q_i$$

<i>Cold</i>	<i>Flu</i>	<i>Malaria</i>	$P(\text{Fever})$	$P(\neg \text{Fever})$
F	F	F	0.0	1.0
F	F	T	0.9	0.1
F	T	F	0.8	0.2
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	0.6
T	F	T	0.94	$0.06 = 0.6 \times 0.1$
T	T	F	0.88	$0.12 = 0.6 \times 0.2$
T	T	T	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

- Number of parameters **linear** in number of parents

Hybrid (Discrete+Continuous) Networks

- Discrete (*Subsidy?* and *Buys?*); continuous (*Harvest* and *Cost*)



- Option 1: discretization—possibly large errors, large CPTs
Option 2: finitely parameterized canonical families (e.g., Gaussian Distribution)
- To specify a hybrid network, we have to specify two new kinds of distributions
 - 1) Continuous variable, discrete+continuous parents (e.g., *Cost*)
 - 2) Discrete variable, continuous parents (e.g., *Buys?*)

Continuous Child Variables

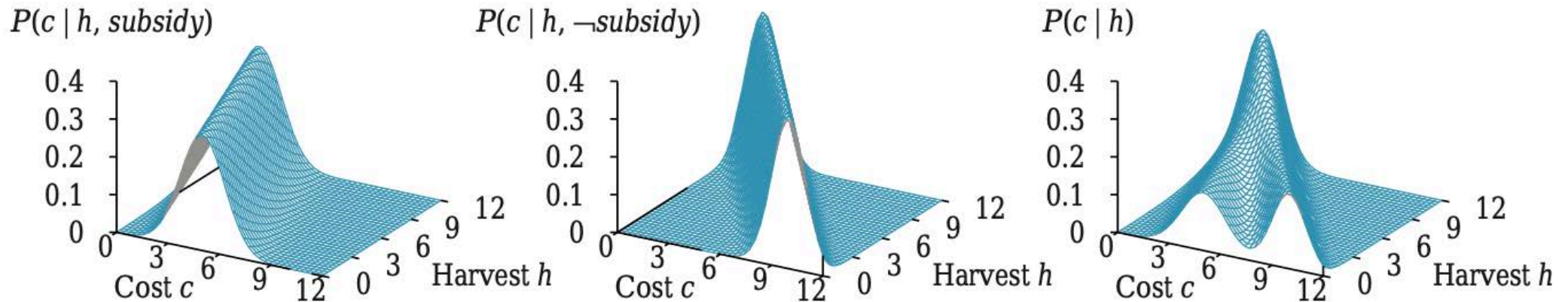
- Need one **conditional density** function for child variable given continuous parents, for each possible assignment to discrete parents
- Most common is the **linear Gaussian** model, in which the child has a Gaussian distribution whose mean μ varies linearly with the value of the parent and whose standard deviation σ is fixed. e.g.,:

$$P(\text{Cost} = c | \text{Harvest} = h, \text{Subsidy?} = \text{true})$$

$$= N(a_t h + b_t, \sigma_t)(c)$$

$$= \frac{1}{\sigma_t \sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{c - (a_t h + b_t)}{\sigma_t}\right)^2\right)$$

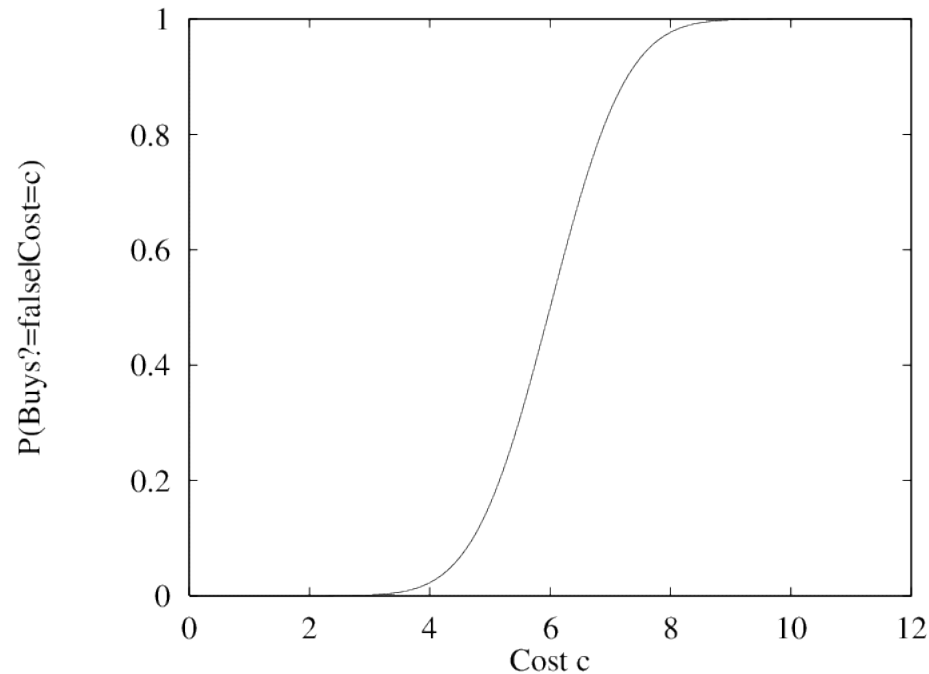
Continuous Child Variables



- All-continuous network with LG distributions
 \implies full joint distribution is a multivariate Gaussian
- Discrete+continuous LG network is a **conditional Gaussian** network i.e., a multivariate Gaussian over all continuous variables for each combination of discrete variable values

Discrete Variable w/ Continuous Parents

- Probability of *Buys?* given *Cost* should be a “soft” threshold:



- **Probit** distribution uses integral of Gaussian:

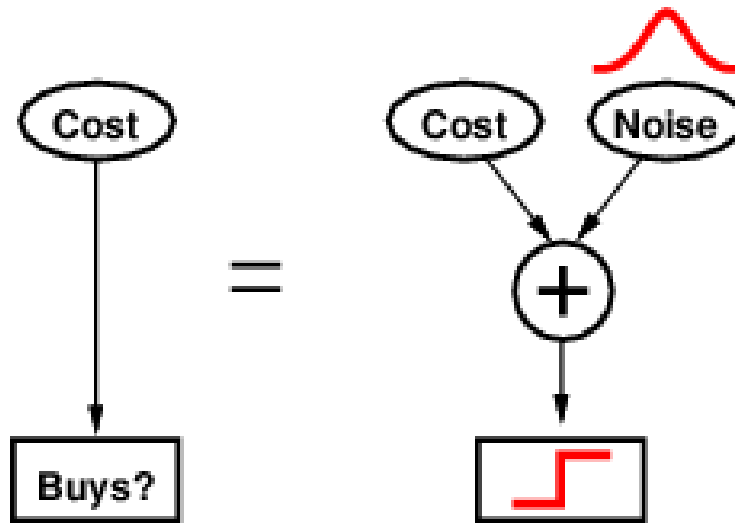
$$\Phi(x) = \int_{-\infty}^x N(0, 1)(x)dx \quad (\text{Cumulative distribution function (CDF) } P(X \leq x))$$

$$P(\text{Buys?} = \text{true} \mid \text{Cost} = c) = \Phi((-c + \mu)/\sigma)$$

which means that the cost threshold occurs around μ , the width of the threshold region is proportional to σ , and the probability of buying decreases as cost increases.

Why the Probit?

- It's sort of the right shape
- Can view as hard threshold whose location is subject to noise

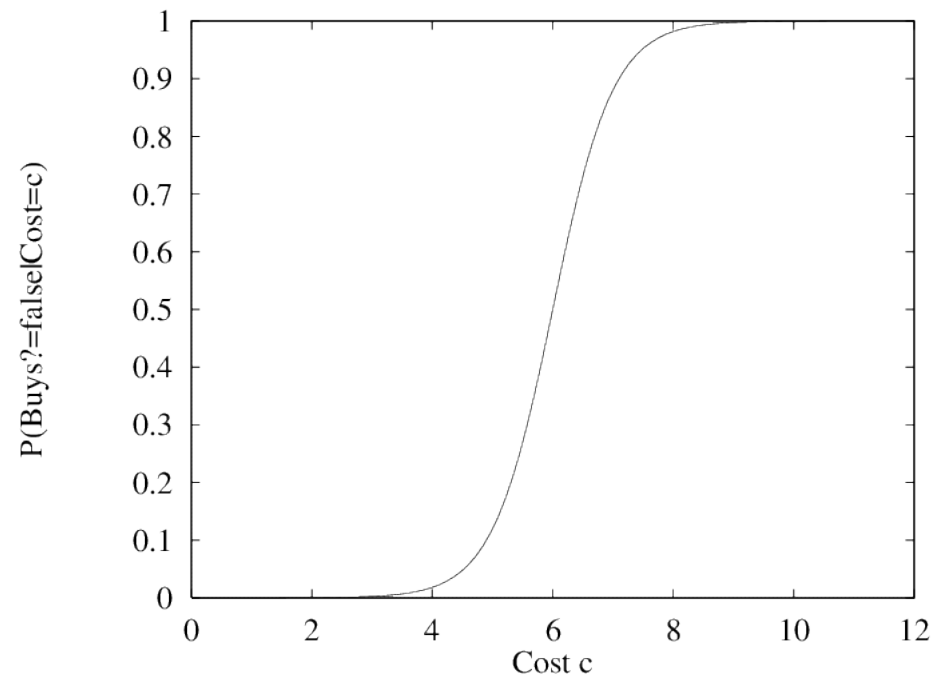


Discrete Variable

- Sigmoid (or logit) distribution also used in neural networks:

$$P(\text{Buys?} = \text{true} \mid \text{Cost} = c) = \frac{1}{1 + \exp\left(-2\frac{-c+\mu}{\sigma}\right)}$$

- Sigmoid has similar shape to probit but much longer tails:



inference

Inference Tasks

- **Simple queries:** compute posterior marginal $\mathbf{P}(X_i|\mathbf{E} = \mathbf{e})$
e.g., $P(\text{NoGas}|\text{Gauge} = \text{empty}, \text{Lights} = \text{on}, \text{Starts} = \text{false})$ ■
- **Conjunctive queries:** $\mathbf{P}(X_i, X_j|\mathbf{E} = \mathbf{e}) = \mathbf{P}(X_i|\mathbf{E} = \mathbf{e})\mathbf{P}(X_j|X_i, \mathbf{E} = \mathbf{e})$ ■
- **Optimal decisions:** decision networks include utility information;
probabilistic inference required for $P(\text{outcome}|\text{action}, \text{evidence})$ ■
- **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

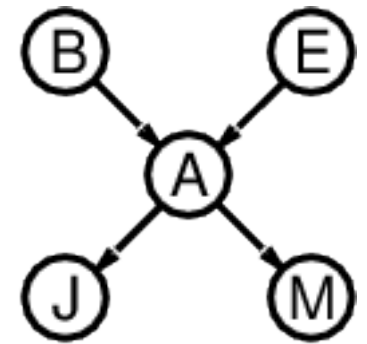
- Simple query on the burglary network

$$\mathbf{P}(B|j, m)$$

$$= \mathbf{P}(B, j, m) / P(j, m) \blacksquare$$

$$= \alpha \mathbf{P}(B, j, m) \blacksquare$$

$$= \alpha \sum_e \sum_a \mathbf{P}(B, e, a, j, m) \blacksquare$$



- Rewrite full joint entries using product of CPT entries:

$$\mathbf{P}(B|j, m)$$

$$= \alpha \sum_e \sum_a \mathbf{P}(B)P(e)\mathbf{P}(a|B, e)P(j|a)P(m|a) \blacksquare$$

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

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

Enumeration Algorithm

function ENUMERATION-ASK(X, \mathbf{e}, bn) **returns** a distribution over X

inputs: X , the query variable

\mathbf{e} , observed values for variables \mathbf{E}

bn , a Bayesian network with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$

$Q(X) \leftarrow$ a distribution over X , initially empty

for each value x_i of X **do**

 extend \mathbf{e} with value x_i for X

$Q(x_i) \leftarrow$ ENUMERATE-ALL(VARS[bn], \mathbf{e})

return NORMALIZE($Q(X)$)

function ENUMERATE-ALL($vars, \mathbf{e}$) **returns** a real number

if EMPTY?($vars$) **then return** 1.0

$Y \leftarrow$ FIRST($vars$)

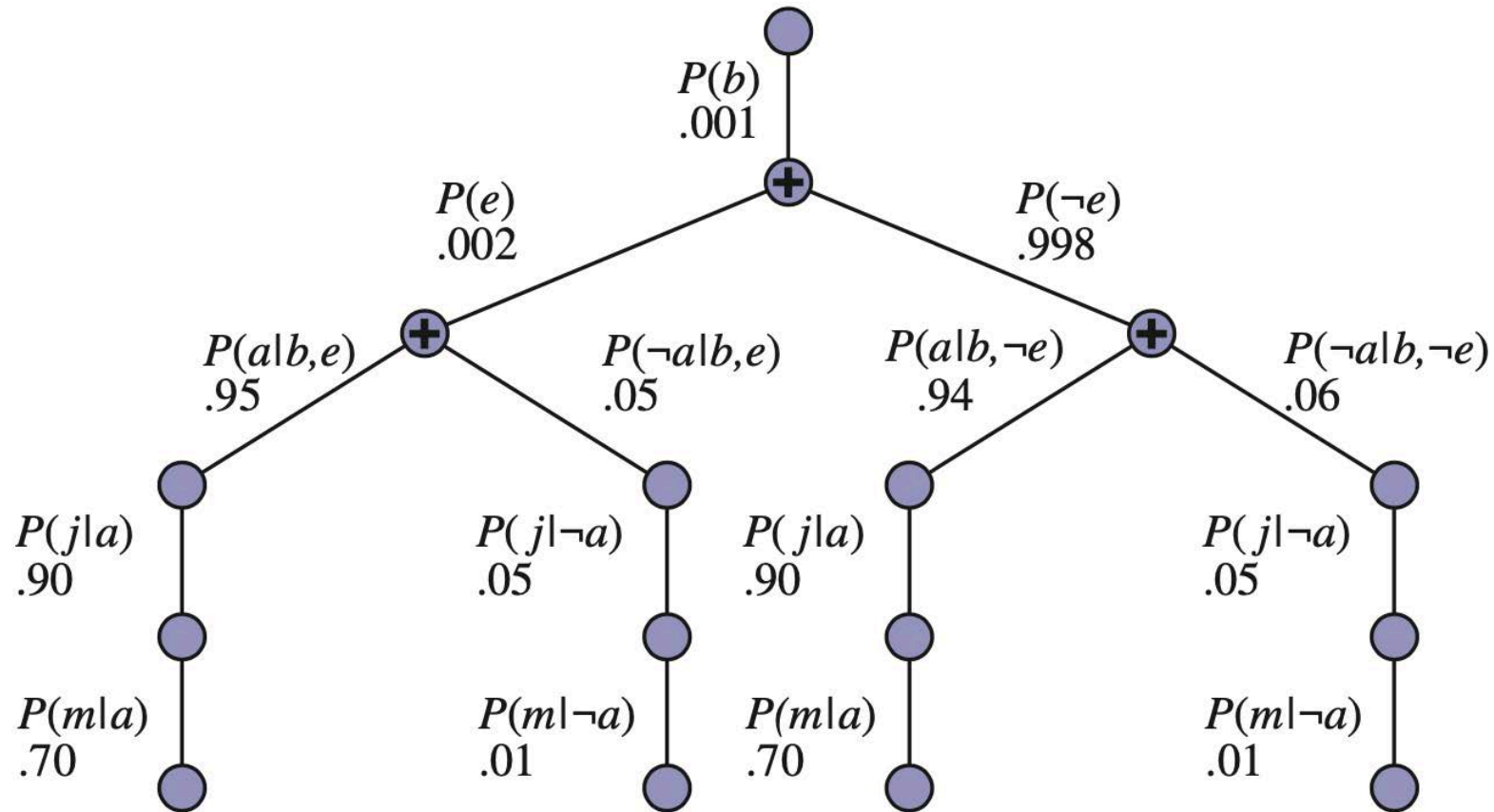
if Y has value y in \mathbf{e}

then return $P(y \mid Pa(Y)) \times$ ENUMERATE-ALL(REST($vars$), \mathbf{e})

else return $\sum_y P(y \mid Pa(Y)) \times$ ENUMERATE-ALL(REST($vars$), \mathbf{e}_y)

 where \mathbf{e}_y is \mathbf{e} extended with $Y = y$

Evaluation Tree



- Enumeration is inefficient: repeated computation
e.g., computes $P(j|a)P(m|a)$ for each value of e

Inference by Variable Elimination

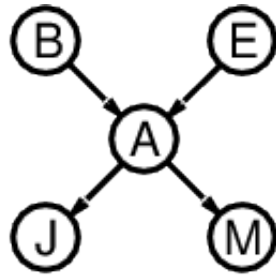
- Variable elimination: carry out summations right-to-left, storing intermediate results (**factors**) to avoid recomputation

$$\begin{aligned} \mathbf{P}(B|j, m) &= \alpha \underbrace{\mathbf{P}(B)}_B \underbrace{\sum_e P(e)}_E \underbrace{\sum_a \mathbf{P}(a|B, e)}_A \underbrace{P(j|a)}_J \underbrace{P(m|a)}_M \blacksquare \\ &= \alpha \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B, e) P(j|a) f_M(a) \blacksquare \\ &= \alpha \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B, e) f_J(a) f_M(a) \blacksquare \\ &= \alpha \mathbf{P}(B) \sum_e P(e) \sum_a f_A(a, b, e) f_J(a) f_M(a) \blacksquare \\ &= \alpha \mathbf{P}(B) \sum_e P(e) f_{\bar{A}JM}(b, e) \text{ (sum out } A) \blacksquare \\ &= \alpha \mathbf{P}(B) f_{\bar{E}\bar{A}JM}(b) \text{ (sum out } E) \blacksquare \\ &= \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b) \end{aligned}$$

Variable Elimination Algorithm

```
function ELIMINATION-ASK( $X, \mathbf{e}, bn$ ) returns a distribution over  $X$   
inputs:  $X$ , the query variable  
           $\mathbf{e}$ , evidence specified as an event  
           $bn$ , a belief network specifying joint distribution  $\mathbf{P}(X_1, \dots, X_n)$   
 $factors \leftarrow [ ]$ ;  $vars \leftarrow \text{REVERSE}(\text{VARS}[bn])$   
for each  $var$  in  $vars$  do  
     $factors \leftarrow [\text{MAKE-FACTOR}(var, \mathbf{e}) | factors]$   
    if  $var$  is a hidden variable then  $factors \leftarrow \text{SUM-OUT}(var, factors)$   
return  $\text{NORMALIZE}(\text{POINTWISE-PRODUCT}(factors))$ 
```

Irrelevant Variables



- Consider the query $P(\text{JohnCalls} | \text{Burglary} = \text{true})$

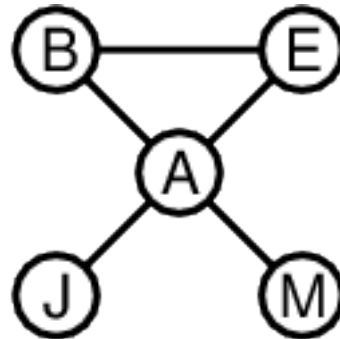
$$P(J|b) = \alpha P(b) \sum_e P(e) \sum_a P(a|b, e) P(J|a) \sum_m P(m|a)$$

Sum over m is identically 1; M is **irrelevant** to the query

- Theorem 1: Y is irrelevant unless $Y \in \text{Ancestors}(\{X\} \cup \mathbf{E})$
- Here
 - $X = \text{JohnCalls}$, $\mathbf{E} = \{\text{Burglary}\}$
 - $\text{Ancestors}(\{X\} \cup \mathbf{E}) = \{\text{Alarm}, \text{Earthquake}\}$ \Rightarrow MaryCalls is irrelevant
- Compare this to backward chaining from the query in Horn clause KBs

Irrelevant Variables

- Definition: moral graph of Bayes net: marry all parents and drop arrows
- Definition: **A** is m-separated from **B** by **C** iff separated by **C** in the moral graph
- Theorem 2: **Y** is irrelevant if m-separated from **X** by **E**

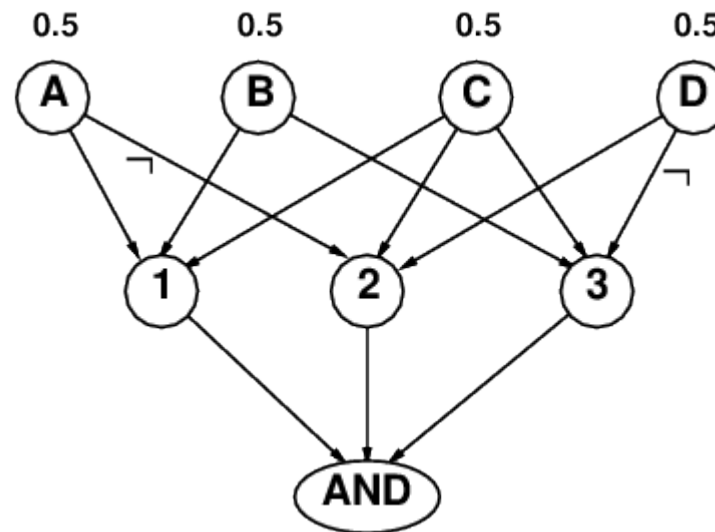


- For $P(\text{JohnCalls} | \text{Alarm} = \text{true})$, both *Burglary* and *Earthquake* are irrelevant

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 \implies NP-hard
 - equivalent to **counting** 3SAT models \implies #P-complete

1. $A \vee B \vee C$
2. $C \vee D \vee \neg A$
3. $B \vee C \vee \neg D$



approximate inference

Inference by Stochastic Simulation

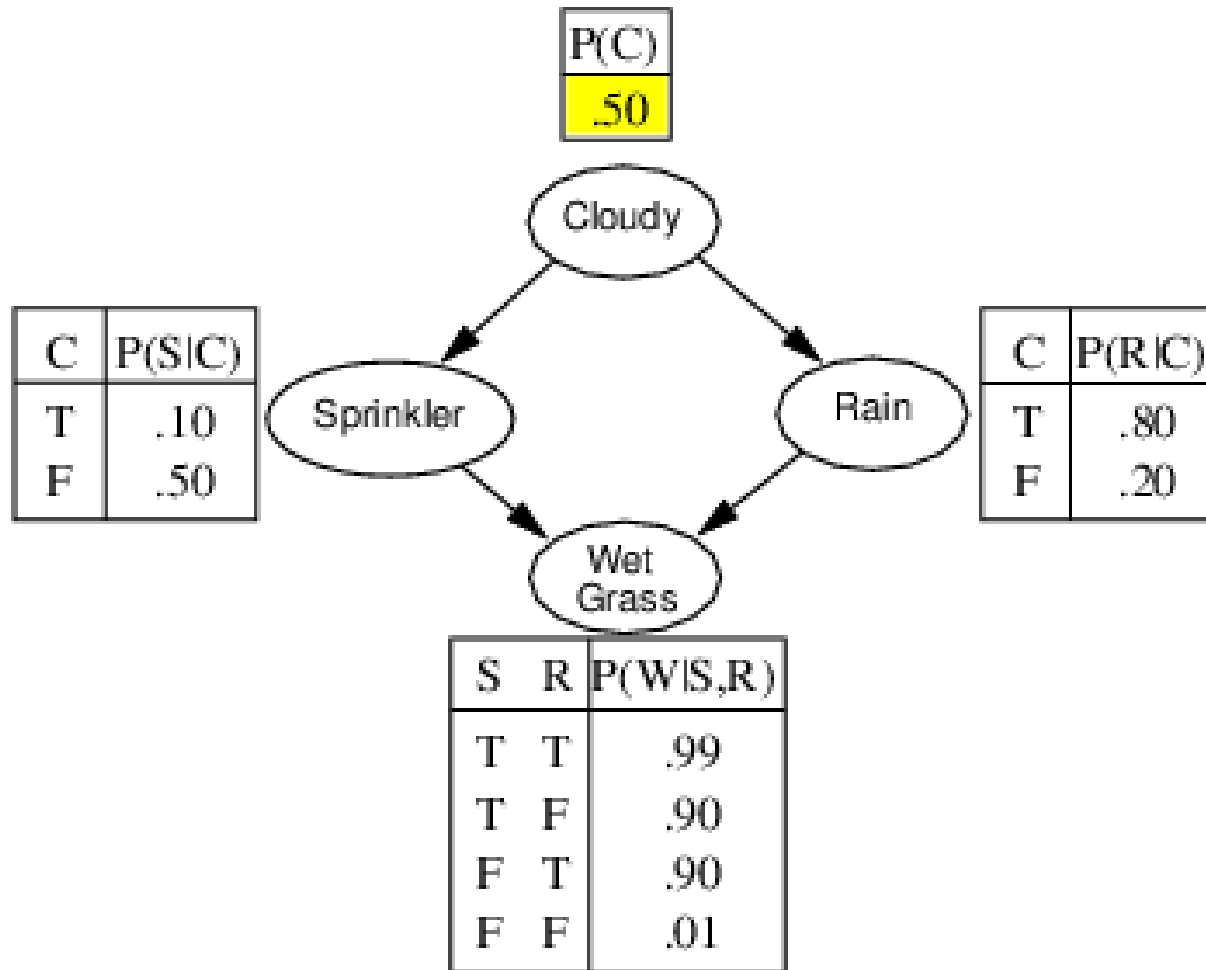
- Basic idea
 - Draw N samples from a sampling distribution S
 - Compute an approximate posterior probability \hat{P}
 - Show this converges to the true probability P
- Outline
 - Sampling from an empty network
 - Rejection sampling: reject samples disagreeing with evidence
 - Likelihood weighting: use evidence to weight samples
 - Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior



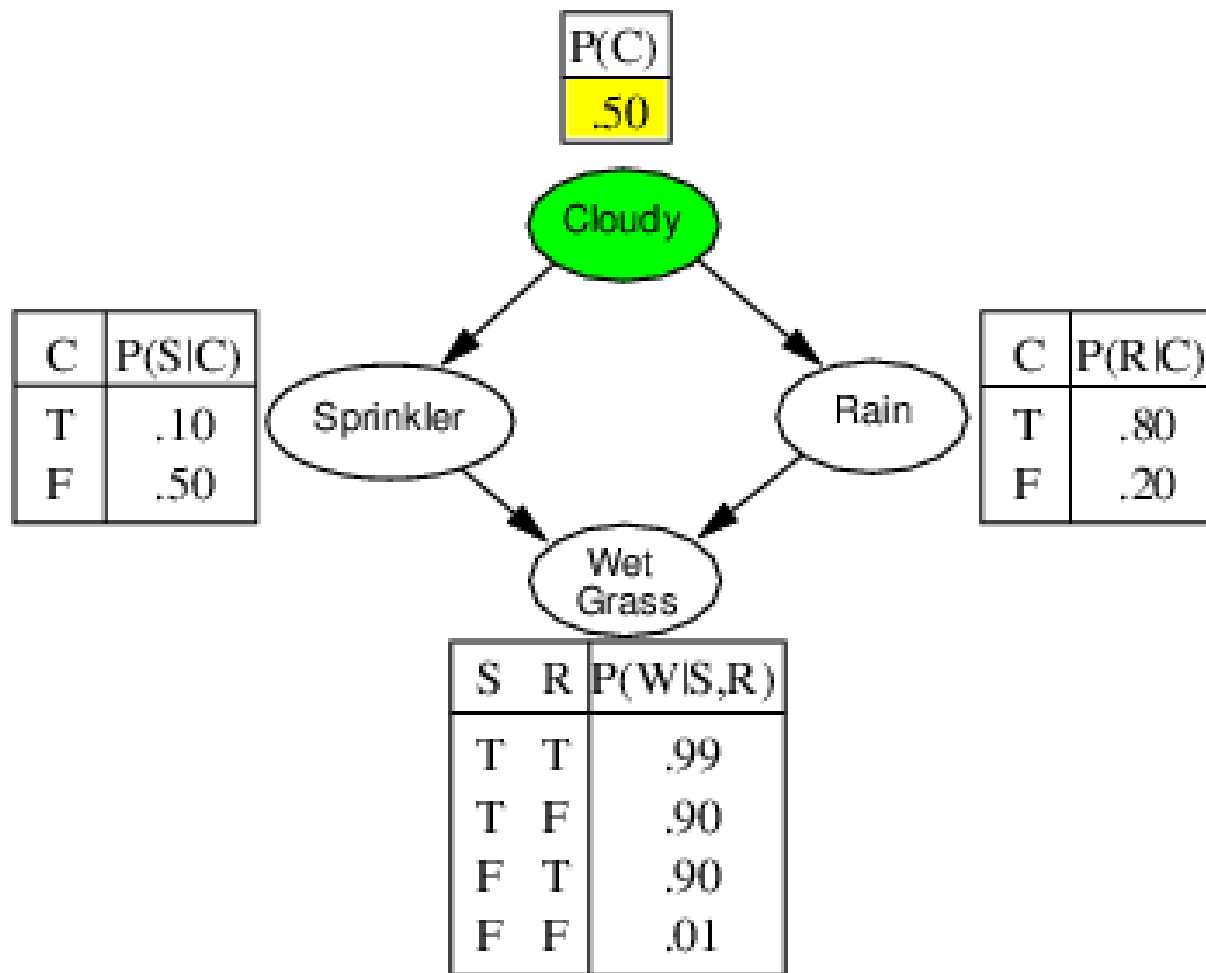
Sampling from an Empty Network

```
function PRIOR-SAMPLE(bn) returns an event sampled from bn  
inputs: bn, a belief network specifying joint distribution  $\mathbf{P}(X_1, \dots, X_n)$   
x  $\leftarrow$  an event with n elements  
for i = 1 to n do  
    xi  $\leftarrow$  a random sample from  $\mathbf{P}(X_i \mid \text{parents}(X_i))$   
    given the values of Parents(Xi) in x  
return x
```

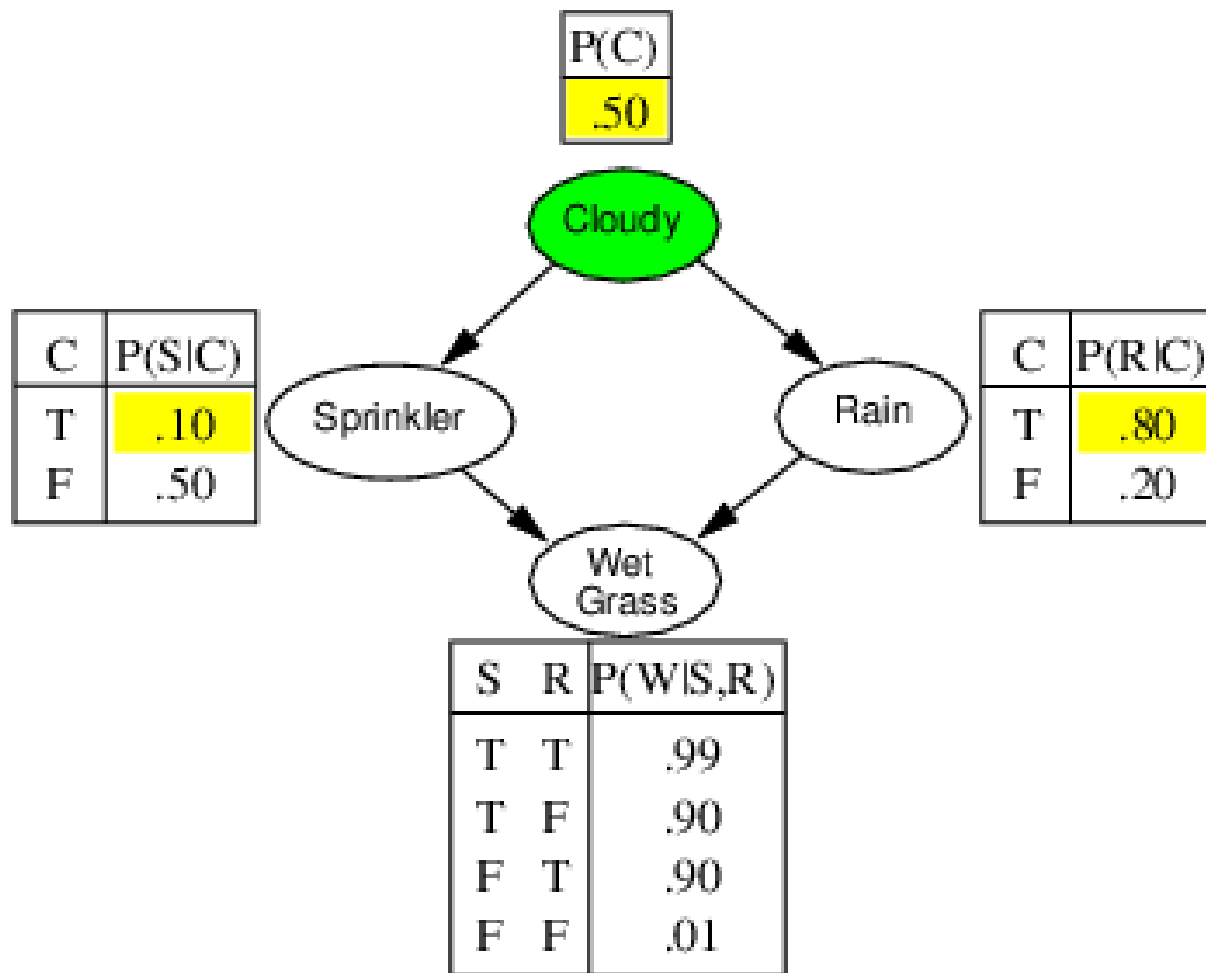

Example



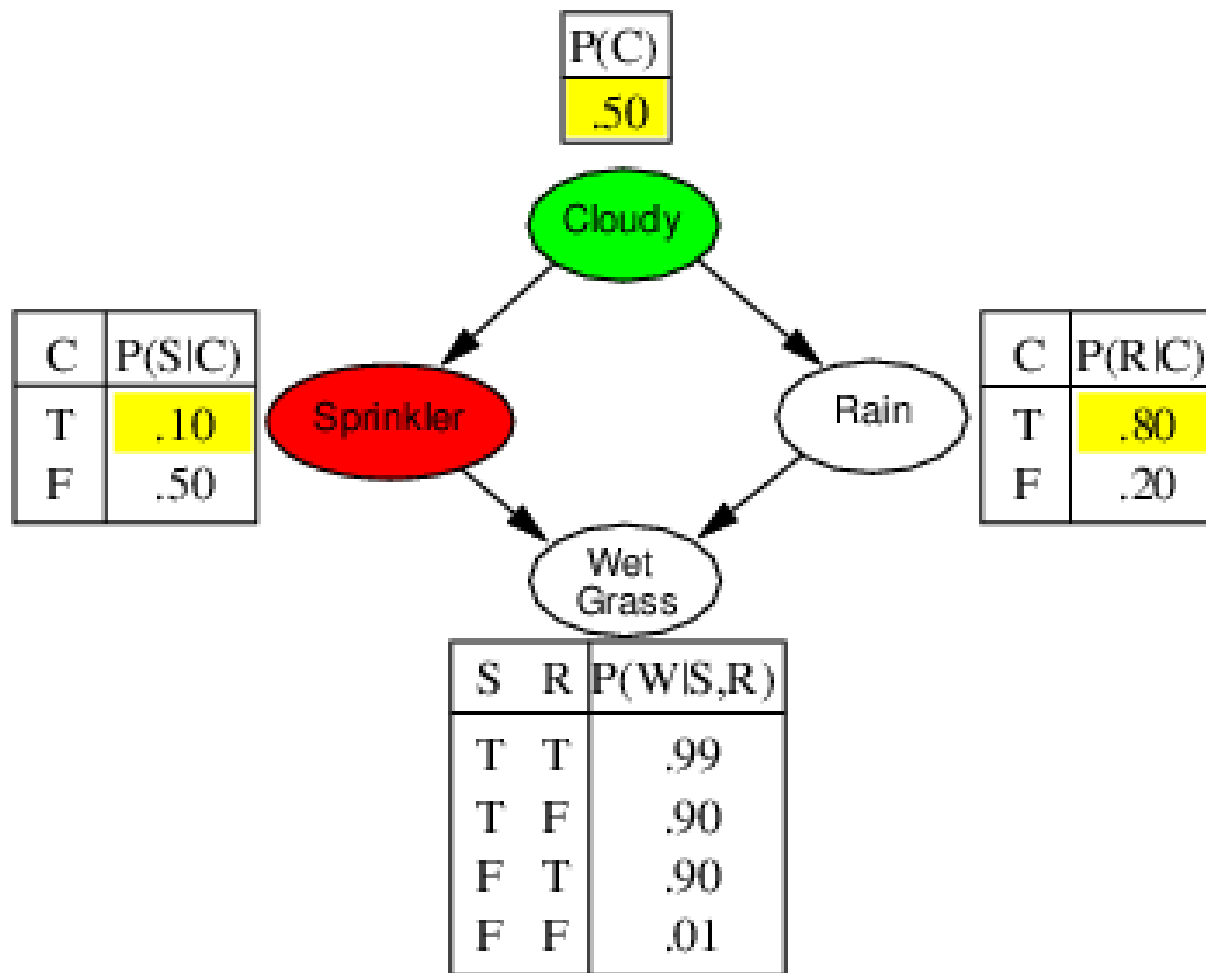
Example



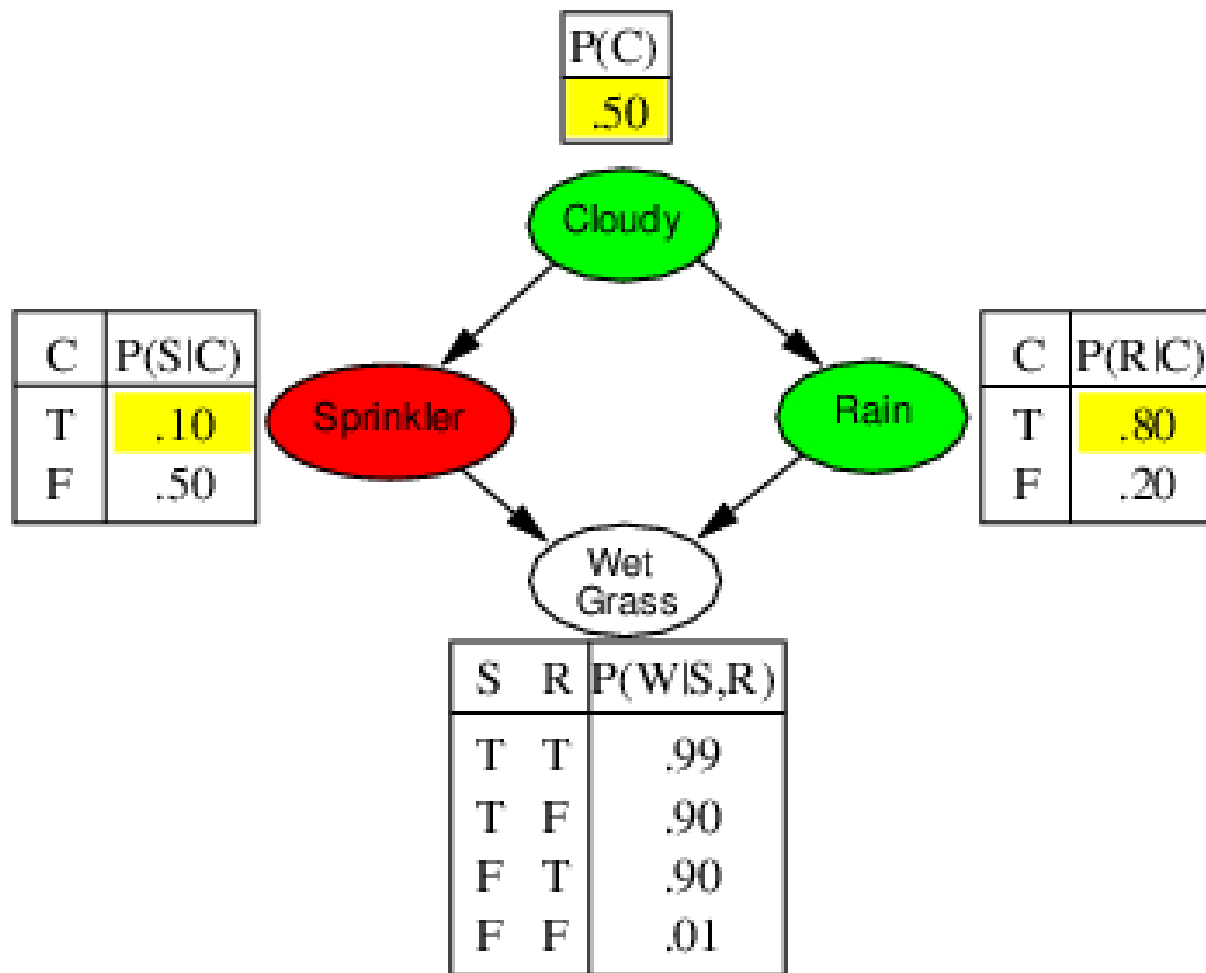
Example



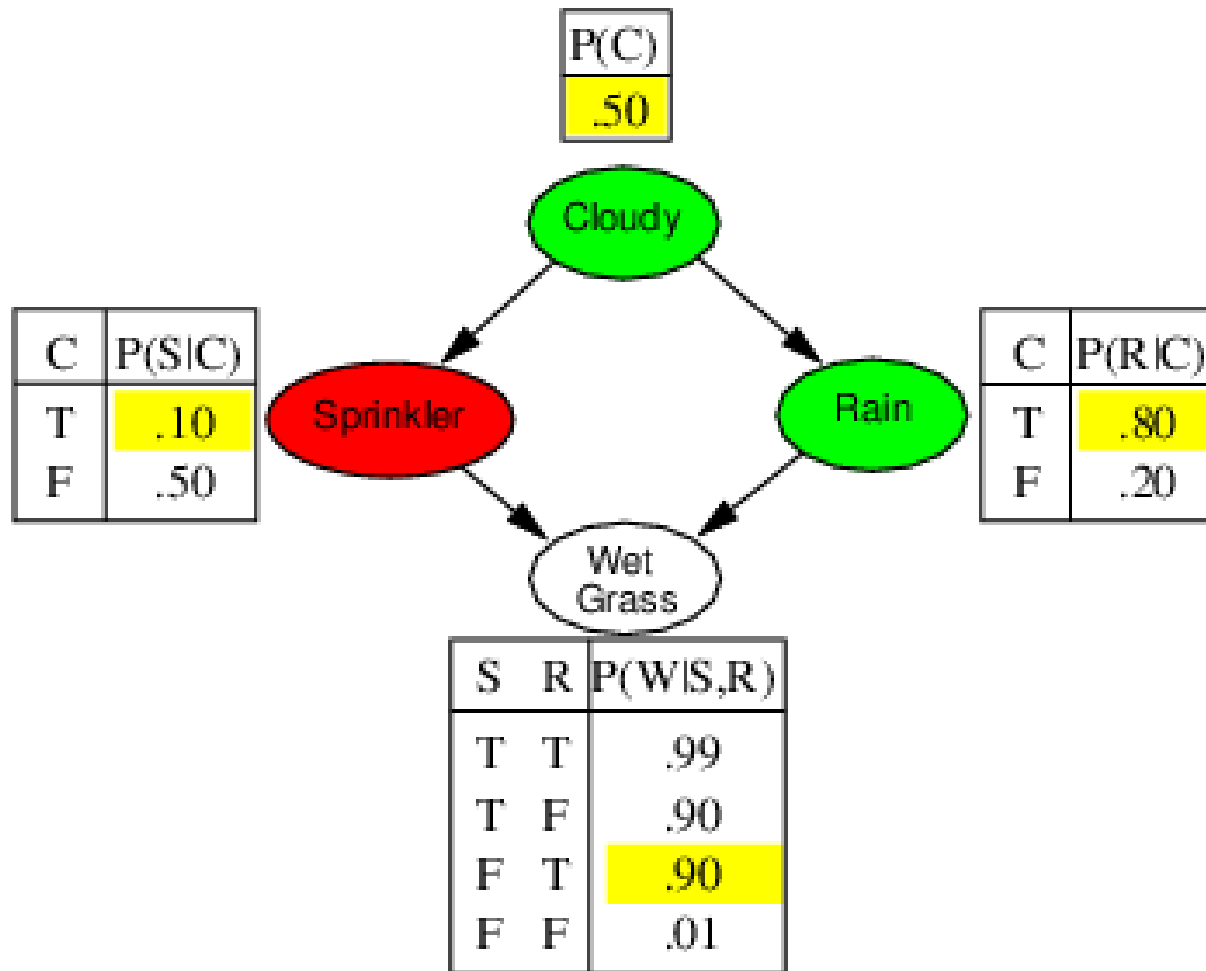
Example



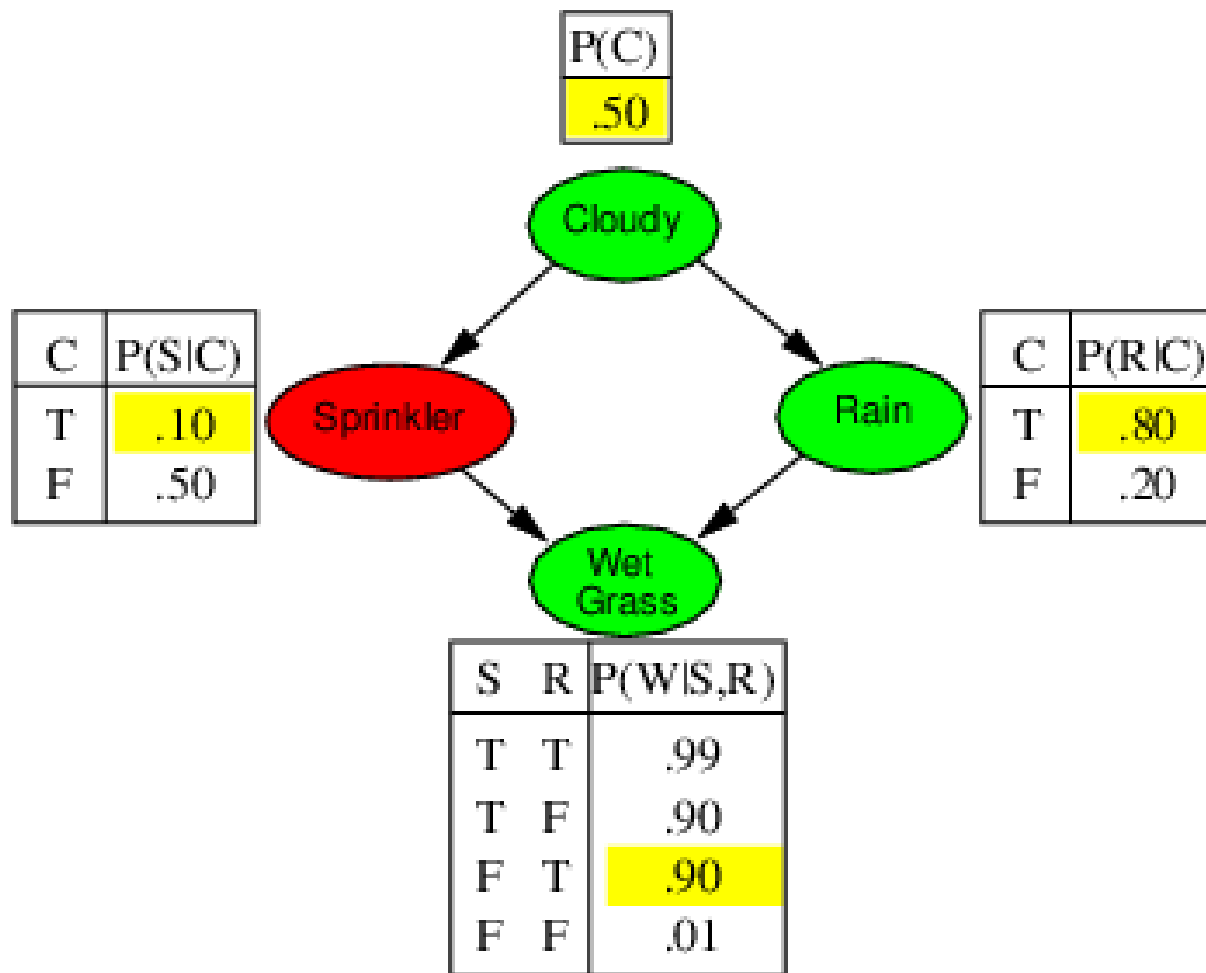
Example



Example



Example



Sampling from an Empty Network

- Probability that PRIORSAMPLE generates a particular event

$$S_{PS}(x_1 \dots x_n) = \prod_{i=1}^n P(x_i | \text{parents}(X_i)) = P(x_1 \dots x_n)$$

i.e., the true prior probability

- E.g., $S_{PS}(t, f, t, t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324 = P(t, f, t, t)$

- Let $N_{PS}(x_1 \dots x_n)$ be the number of samples generated for event x_1, \dots, x_n

- Then we have
$$\begin{aligned} \lim_{N \rightarrow \infty} \hat{P}(x_1, \dots, x_n) &= \lim_{N \rightarrow \infty} N_{PS}(x_1, \dots, x_n) / N \\ &= S_{PS}(x_1, \dots, x_n) \\ &= P(x_1 \dots x_n) \end{aligned}$$

- That is, estimates derived from PRIORSAMPLE are **consistent**

- Shorthand: $\hat{P}(x_1, \dots, x_n) \approx P(x_1 \dots x_n)$

Rejection Sampling

- $\hat{\mathbf{P}}(X|\mathbf{e})$ estimated from samples agreeing with \mathbf{e}

```
function REJECTION-SAMPLING( $X, \mathbf{e}, bn, N$ ) returns an estimate of  $P(X|\mathbf{e})$   
  local variables:  $\mathbf{N}$ , a vector of counts over  $X$ , initially zero  
  for  $j = 1$  to  $N$  do  
     $\mathbf{x} \leftarrow$  PRIOR-SAMPLE( $bn$ )  
    if  $\mathbf{x}$  is consistent with  $\mathbf{e}$  then  
       $\mathbf{N}[\mathbf{x}] \leftarrow \mathbf{N}[\mathbf{x}] + 1$  where  $\mathbf{x}$  is the value of  $X$  in  $\mathbf{x}$   
  return NORMALIZE( $\mathbf{N}[X]$ )
```

- E.g., estimate $\mathbf{P}(Rain|Sprinkler = true)$ using 100 samples
 27 samples have *Sprinkler = true*
 Of these, 8 have *Rain = true* and 19 have *Rain = false*
- $\hat{\mathbf{P}}(Rain|Sprinkler = true) = \text{NORMALIZE}(\langle 8, 19 \rangle) = \langle 0.296, 0.704 \rangle$
- Similar to a basic real-world empirical estimation procedure

Analysis of Rejection Sampling

- $\hat{\mathbf{P}}(X|\mathbf{e}) = \alpha \mathbf{N}_{PS}(X, \mathbf{e})$ (algorithm defn.)
= $\mathbf{N}_{PS}(X, \mathbf{e}) / N_{PS}(\mathbf{e})$ (normalized by $N_{PS}(\mathbf{e})$)
 $\approx \mathbf{P}(X, \mathbf{e}) / P(\mathbf{e})$ (property of PRIORSAMPLE)
= $\mathbf{P}(X|\mathbf{e})$ (defn. of conditional probability)
- Hence rejection sampling returns consistent posterior estimates
- Problem: hopelessly expensive if $P(\mathbf{e})$ is small
- $P(\mathbf{e})$ drops off exponentially with number of evidence variables!

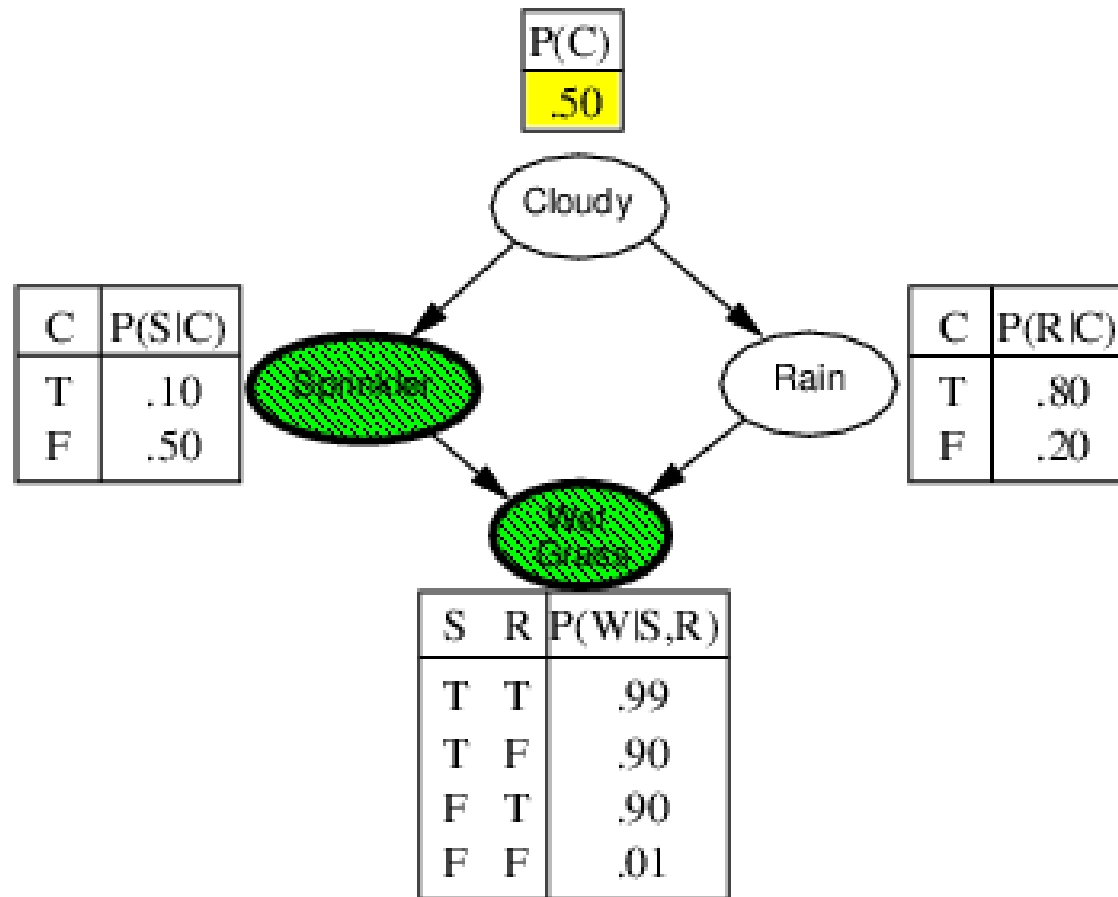
Likelihood Weighting

- Idea: fix evidence variables, sample only nonevidence variables, and weight each sample by the likelihood it accords the evidence

```
function LIKELIHOOD-WEIGHTING( $X, \mathbf{e}, bn, N$ ) returns an estimate of  $P(X|\mathbf{e})$   
local variables:  $\mathbf{W}$ , a vector of weighted counts over  $X$ , initially zero  
for  $j = 1$  to  $N$  do  
     $\mathbf{x}, w \leftarrow$  WEIGHTED-SAMPLE( $bn$ )  
     $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$  where  $x$  is the value of  $X$  in  $\mathbf{x}$   
return NORMALIZE( $\mathbf{W}[X]$ )
```

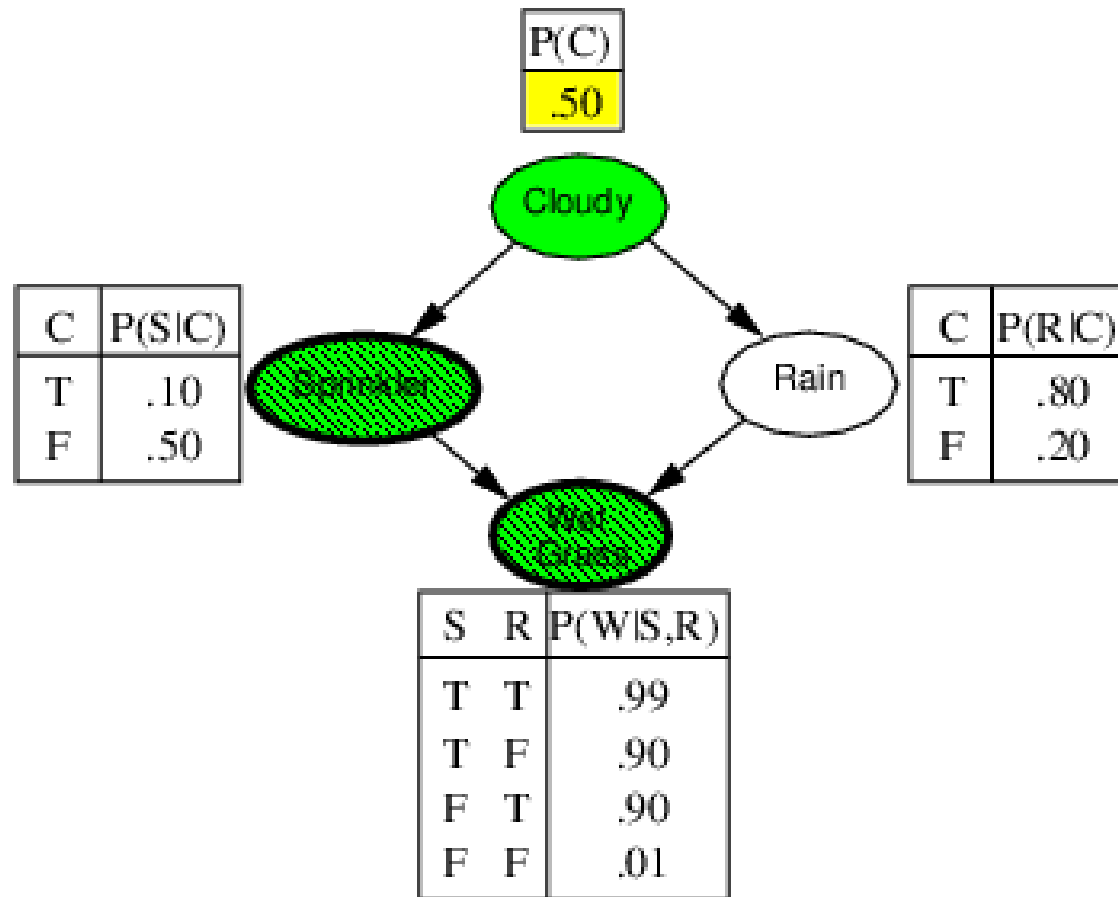
```
function WEIGHTED-SAMPLE( $bn, \mathbf{e}$ ) returns an event and a weight  
 $\mathbf{x} \leftarrow$  an event with  $n$  elements;  $w \leftarrow 1$   
for  $i = 1$  to  $n$  do  
    if  $X_i$  has a value  $x_i$  in  $\mathbf{e}$   
        then  $w \leftarrow w \times P(X_i = x_i \mid \text{parents}(X_i))$   
        else  $x_i \leftarrow$  a random sample from  $\mathbf{P}(X_i \mid \text{parents}(X_i))$   
return  $\mathbf{x}, w$ 
```

Likelihood Weighting Example



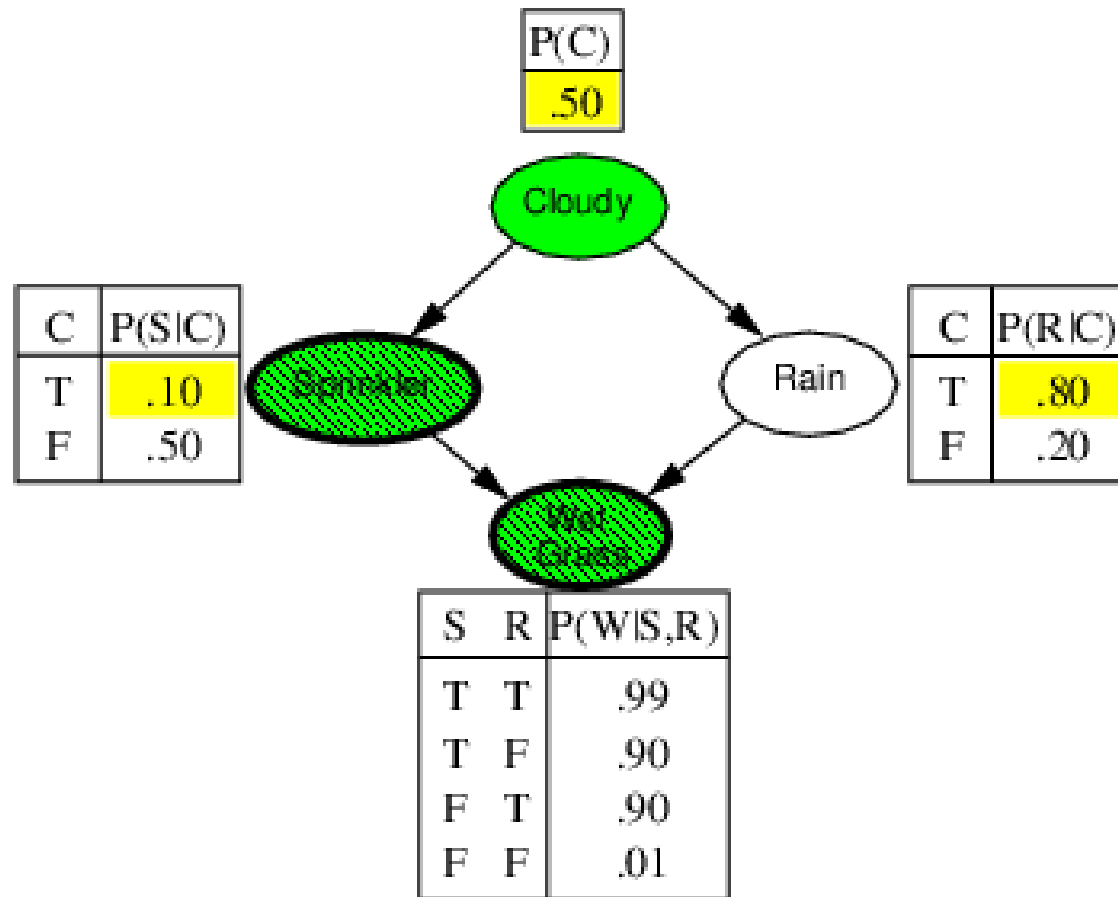
$$w = 1.0$$

Likelihood Weighting Example



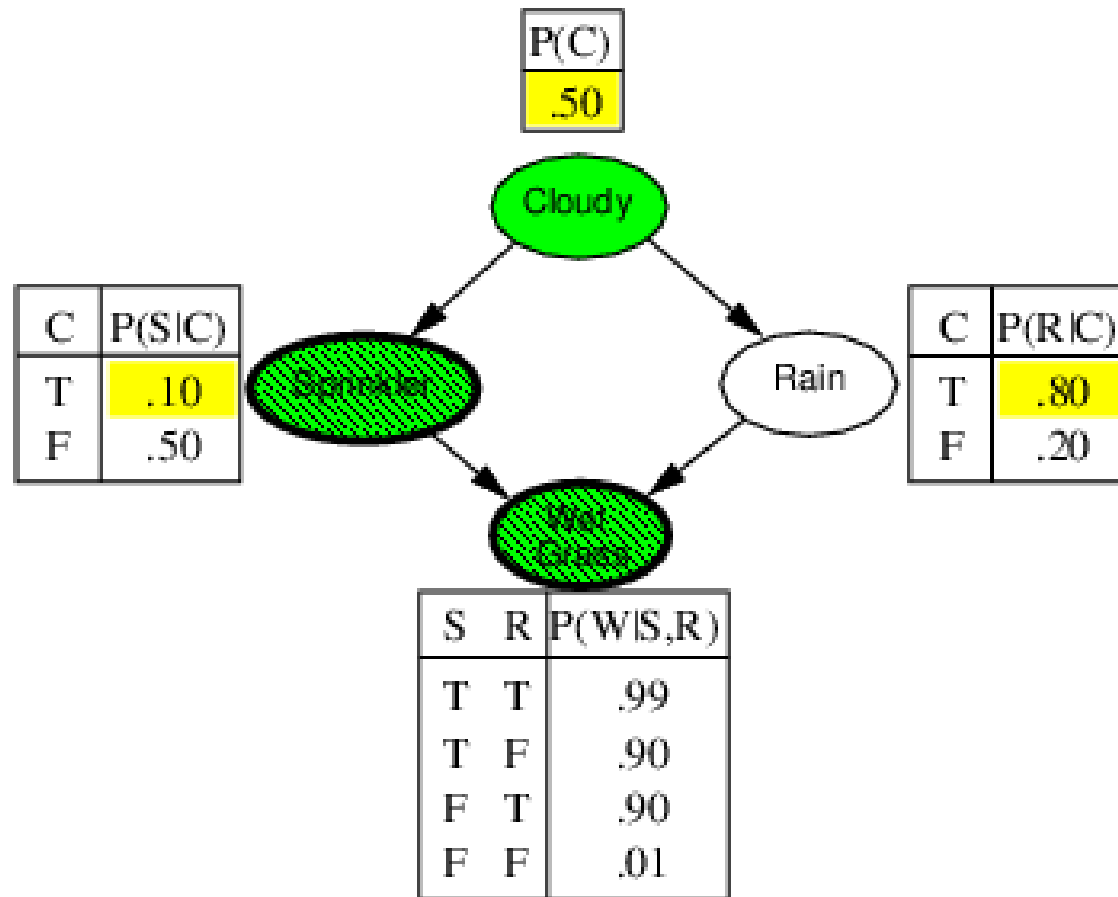
$$w = 1.0$$

Likelihood Weighting Example



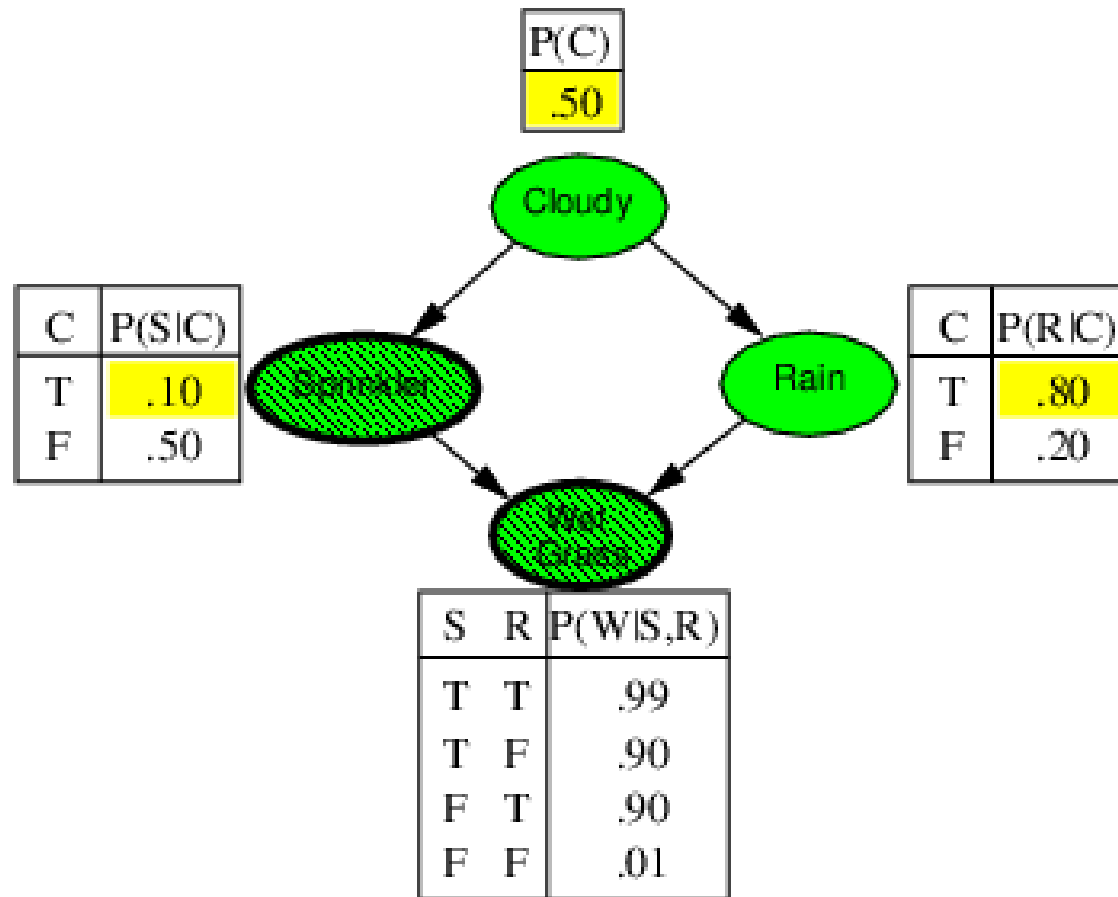
$$w = 1.0$$

Likelihood Weighting Example



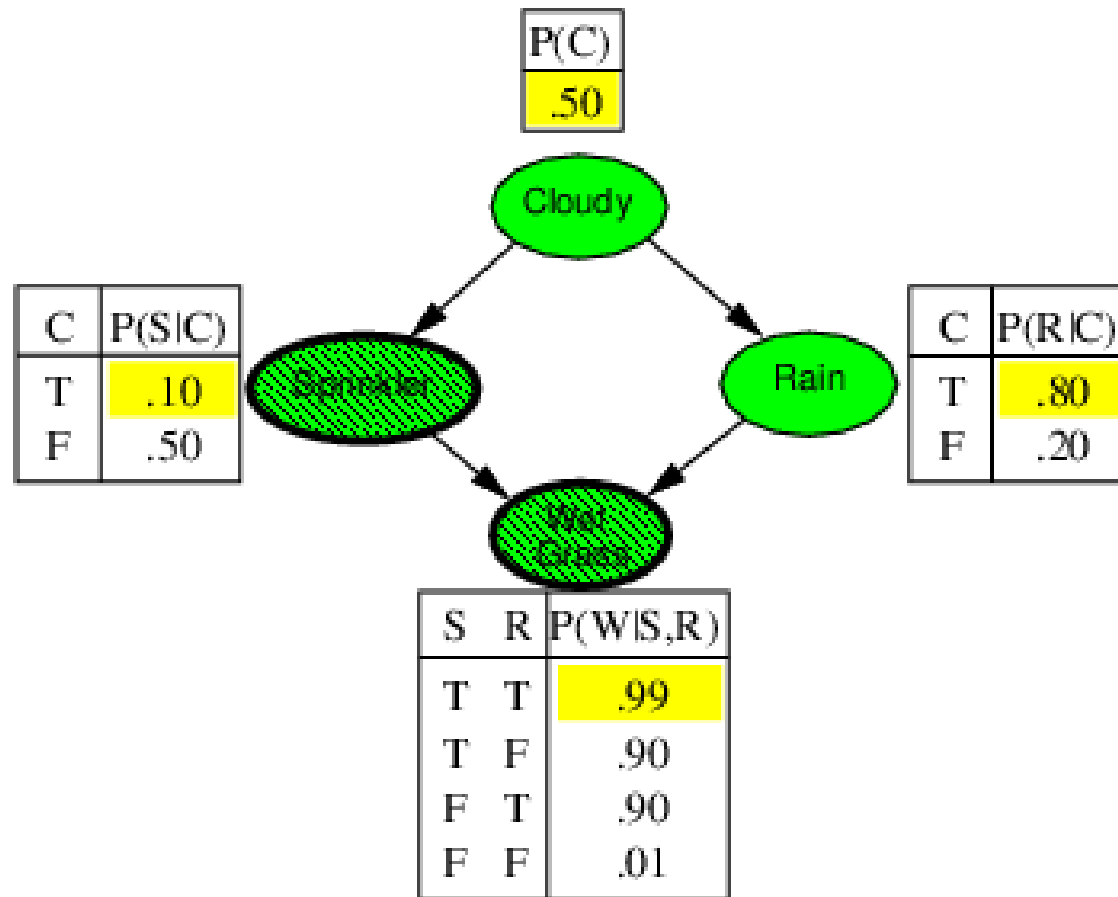
$$w = 1.0 \times 0.1$$

Likelihood Weighting Example



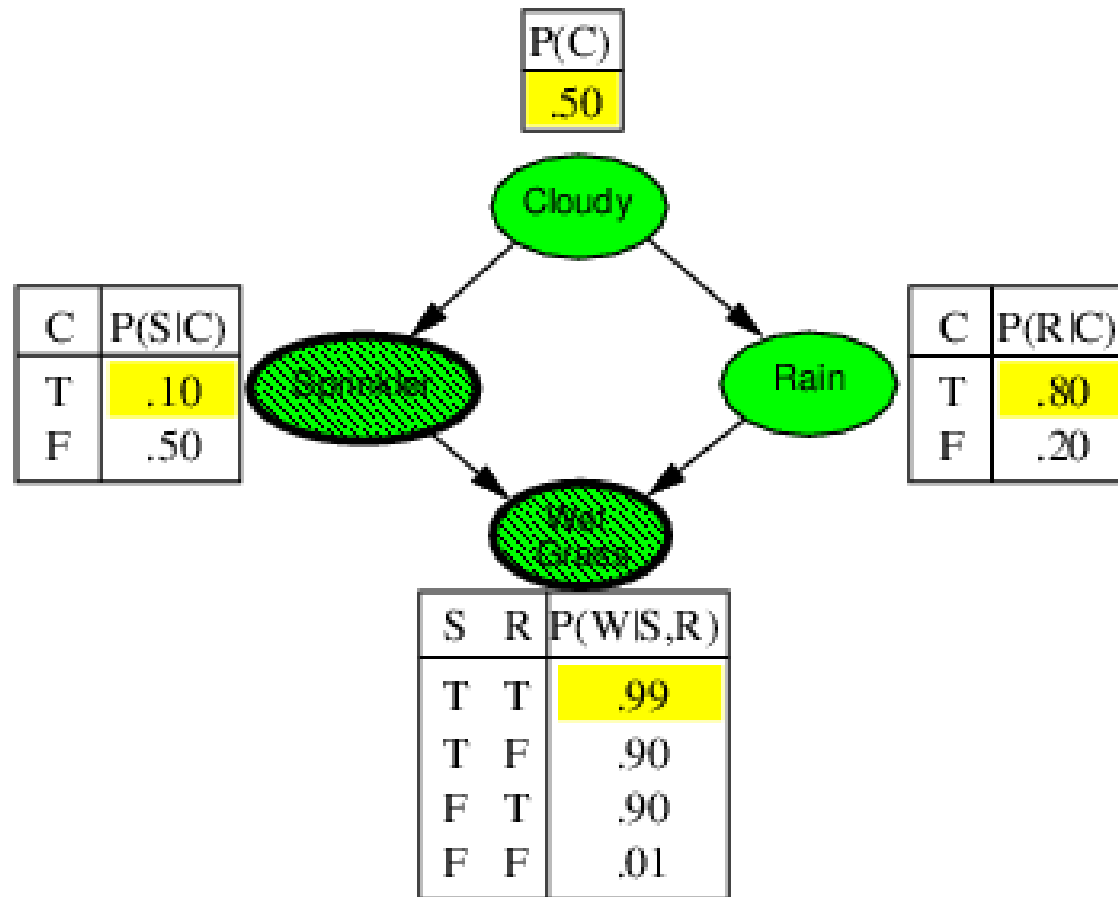
$$w = 1.0 \times 0.1$$

Likelihood Weighting Example



$$w = 1.0 \times 0.1$$

Likelihood Weighting Example



$$w = 1.0 \times 0.1 \times 0.99 = 0.099$$

Likelihood Weighting Analysis

- Sampling probability for WEIGHTEDSAMPLE is

$$S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^l P(z_i | \text{parents}(Z_i))$$

- Note: pays attention to evidence in **ancestors** only
⇒ somewhere “in between” prior and posterior distribution

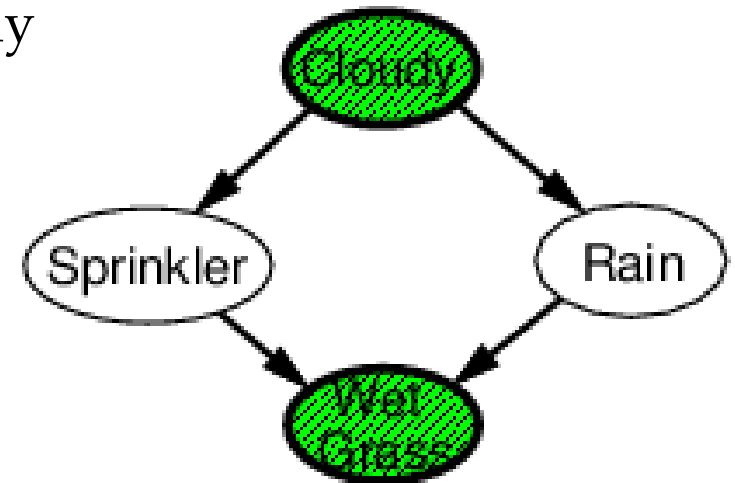
- Weight for a given sample \mathbf{z}, \mathbf{e} is

$$w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^m P(e_i | \text{parents}(E_i))$$

- Weighted sampling probability is

$$\begin{aligned} S_{WS}(\mathbf{z}, \mathbf{e}) w(\mathbf{z}, \mathbf{e}) &= \prod_{i=1}^l P(z_i | \text{parents}(Z_i)) \prod_{i=1}^m P(e_i | \text{parents}(E_i)) \\ &= P(\mathbf{z}, \mathbf{e}) \text{ (by standard global semantics of network)} \end{aligned}$$

- Hence likelihood weighting returns consistent estimates but performance still degrades with many evidence variables because a few samples have nearly all the total weight



Approximate Inference using MCMC

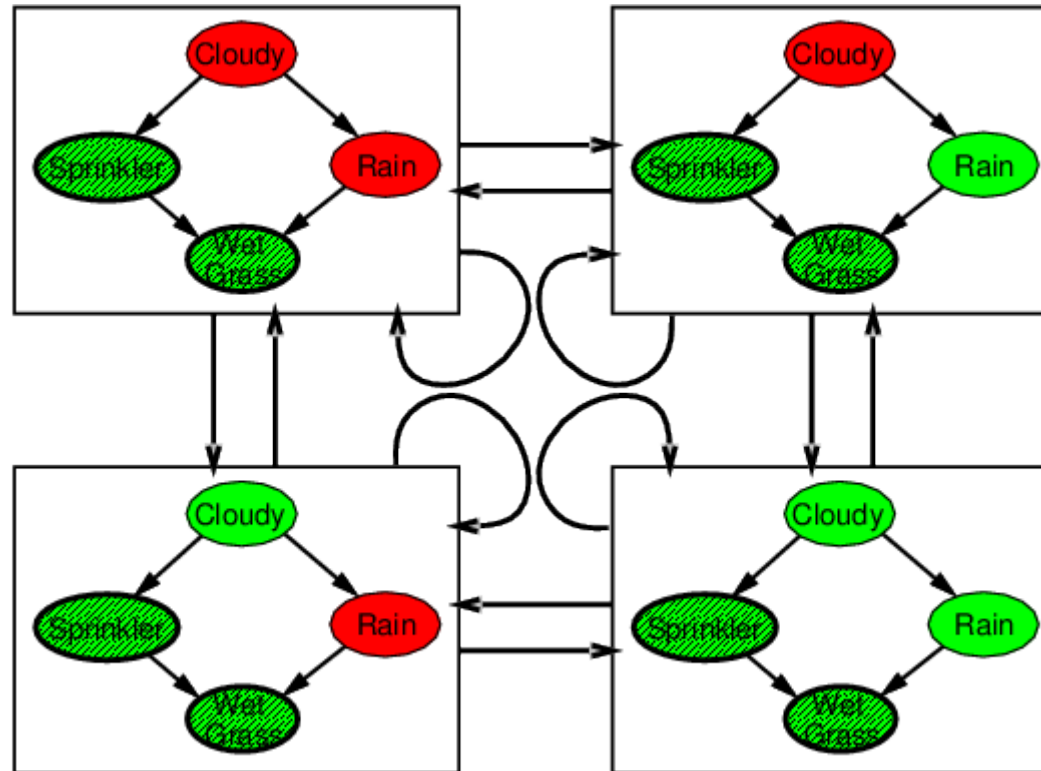
- “State” of network = current assignment to all variables
- Generate next state by sampling one variable given Markov blanket
Sample each variable in turn, keeping evidence fixed

```
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  $\mathbf{Z}$   
  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( $\mathbf{N}[X]$ )
```

- Can also choose a variable to sample at random each time

The Markov Chain

- With *Sprinkler = true*, *WetGrass = true*, there are four states:



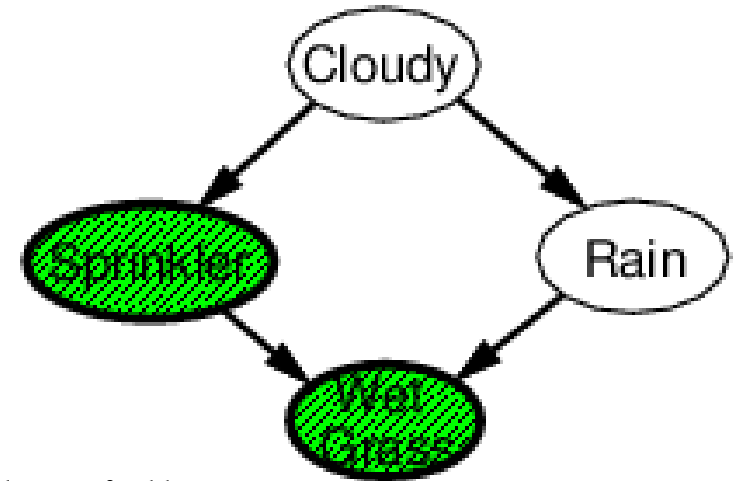
- Wander about for a while, average what you see

MCMC 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*
- $\hat{\mathbf{P}}(Rain|Sprinkler = true, WetGrass = true)$
 $= \text{NORMALIZE}(\langle 31, 69 \rangle) = \langle 0.31, 0.69 \rangle$
- 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:

$$P(x'_i | mb(X_i)) = P(x'_i | parents(X_i)) \prod_{Z_j \in Children(X_i)} P(z_j | parents(Z_j))$$

- Easily implemented in message-passing parallel systems, brains
- Main computational problems
 - difficult to tell if convergence has been achieved
 - can be wasteful if Markov blanket is large:
 $P(X_i | mb(X_i))$ won't change much (law of large numbers)

Summary

- Bayes nets provide a natural representation for (causally induced) conditional independence
- Topology + CPTs = compact representation of joint distribution
- Generally easy for (non)experts to construct
- Canonical distributions (e.g., noisy-OR) = compact representation of CPTs
- Continuous variables \implies parameterized distributions (e.g., linear Gaussian)
- Exact inference by variable elimination
 - polytime on polytrees, NP-hard on general graphs
 - space = time, very sensitive to topology
- Approximate inference by LW, MCMC
 - LW does poorly when there is lots of (downstream) evidence
 - LW, MCMC generally insensitive to topology
 - Convergence can be very slow with probabilities close to 1 or 0
 - Can handle arbitrary combinations of discrete and continuous variables