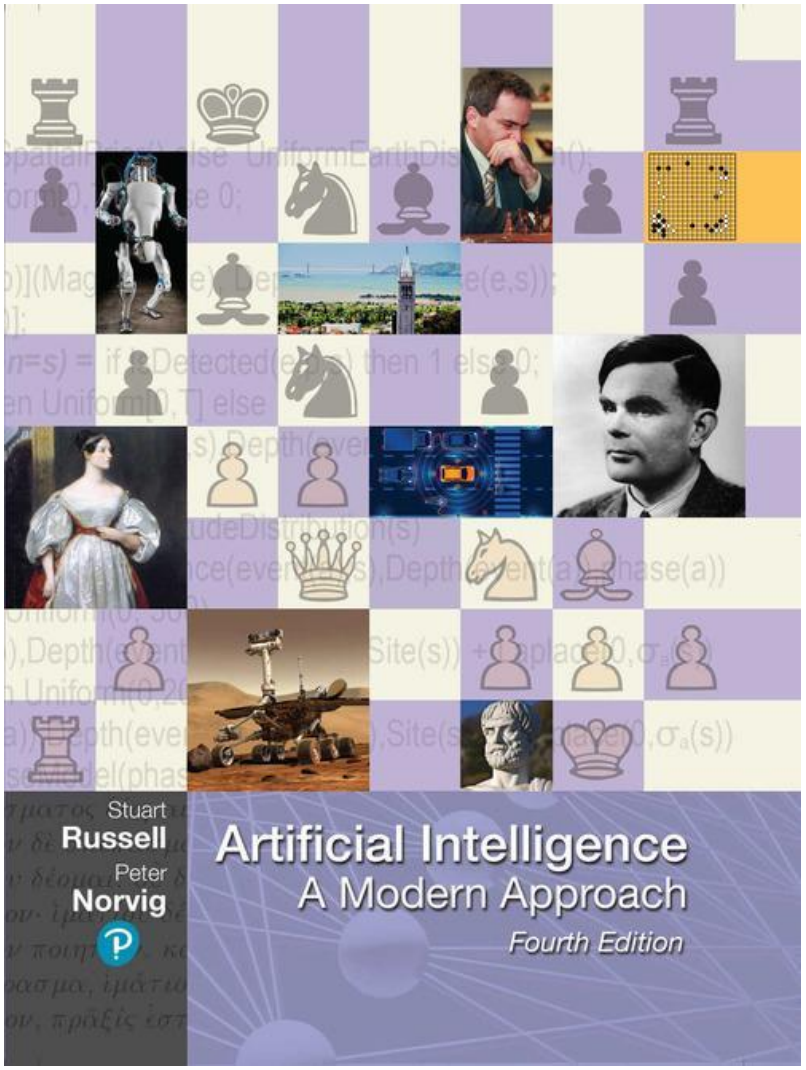
Artificial Intelligence: A Modern Approach

Fourth Edition



Chapter 13

Probabilistic Reasoning

Outline

- Representing Knowledge in an Uncertain Domain
- Semantics of Bayesian Networks
- Exact Inference in Bayesian Networks
- Approximate Inference for Bayesian Networks
- Causal Networks

Representing Knowledge in an Uncertain Domain

Bayesian networks: represents dependencies among variables.

A simple, directed graph in which each node is annotated with quantitative probability information

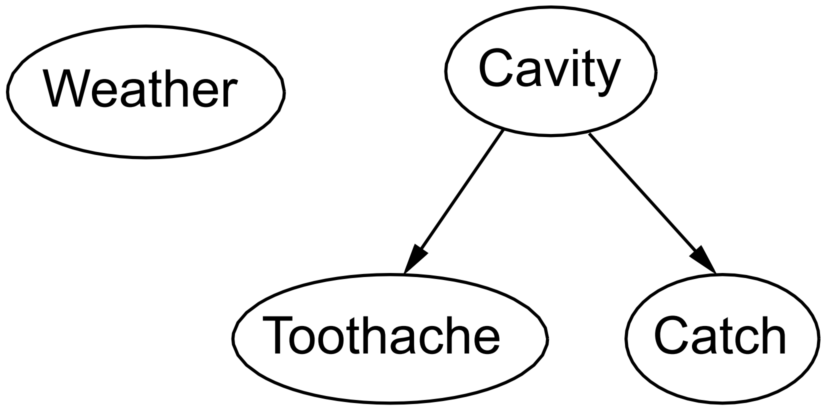
Syntax:

- a set of nodes, one per variable
- a directed, acyclic graph (link $\approx$ "directly influences")
- a conditional distribution for each node given its parents:
  $\mathrm{P}(X_i|Parents(X_i))$

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*

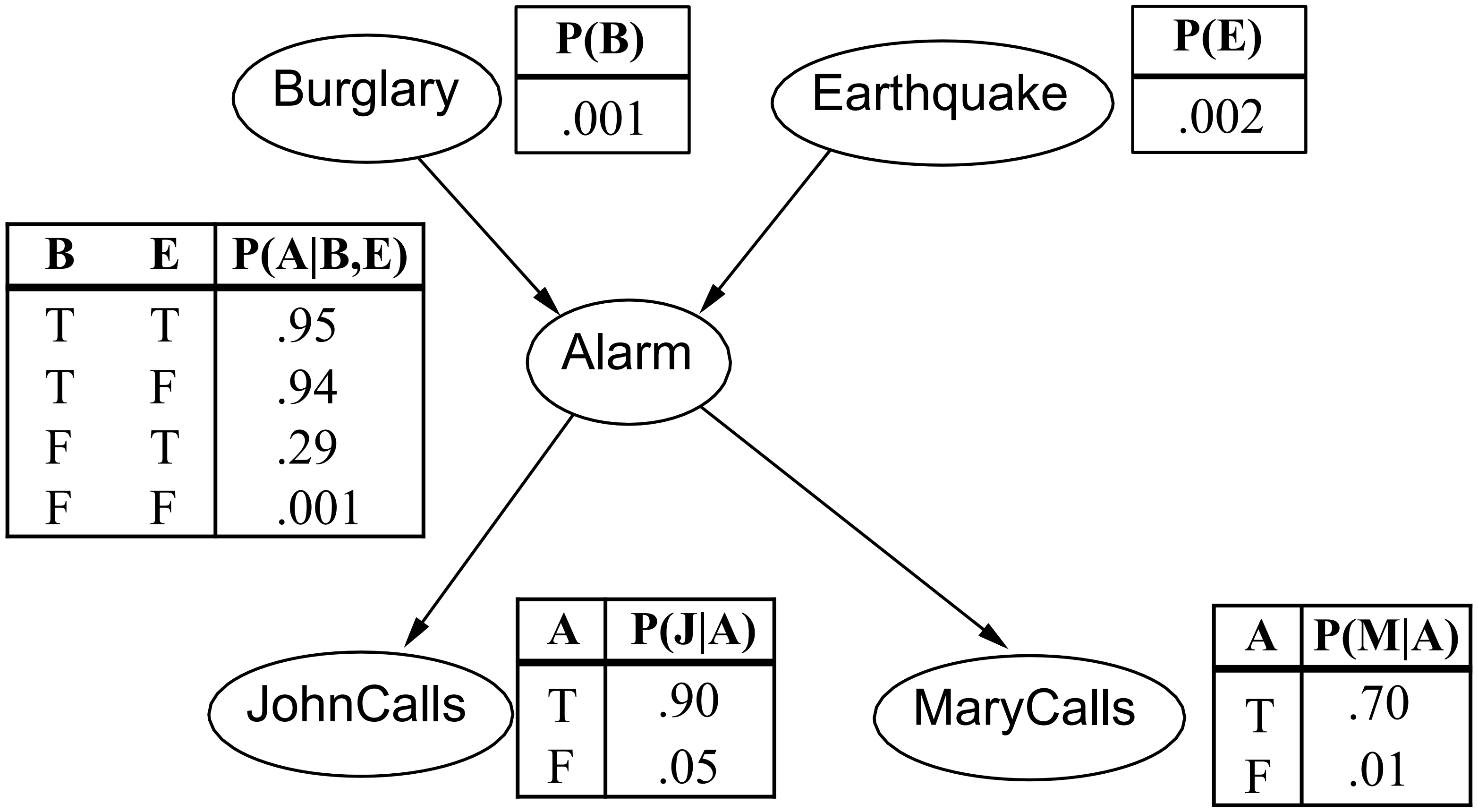
Example

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 contd.



P(B)
.001

P(E)
.002

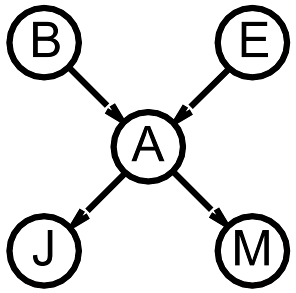
B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01

Compactness

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. $2^5 - 1 = 31$)

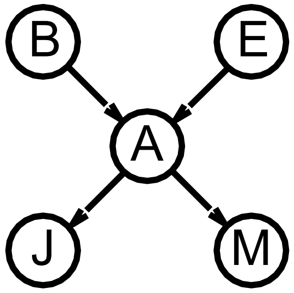
The Semantics of Bayesian Networks

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

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

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

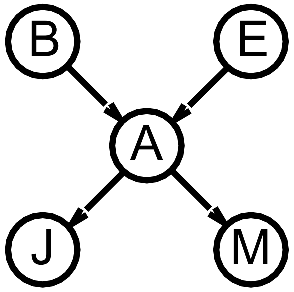
$=$



Global semantics

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

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



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

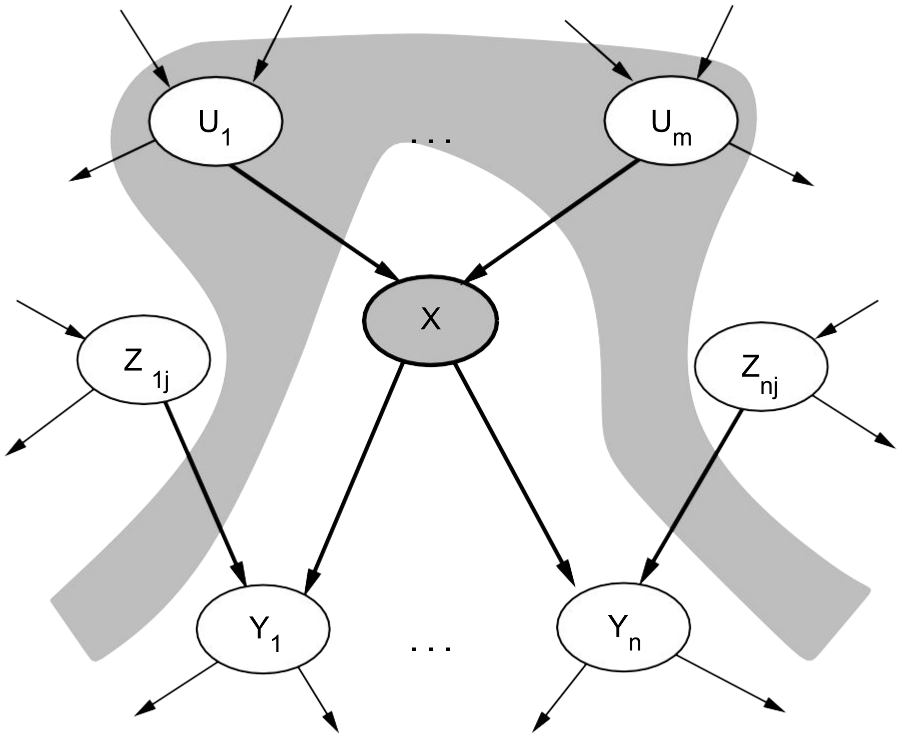
$$= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e)$$

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

$$\approx 0.00063$$

Local semantics

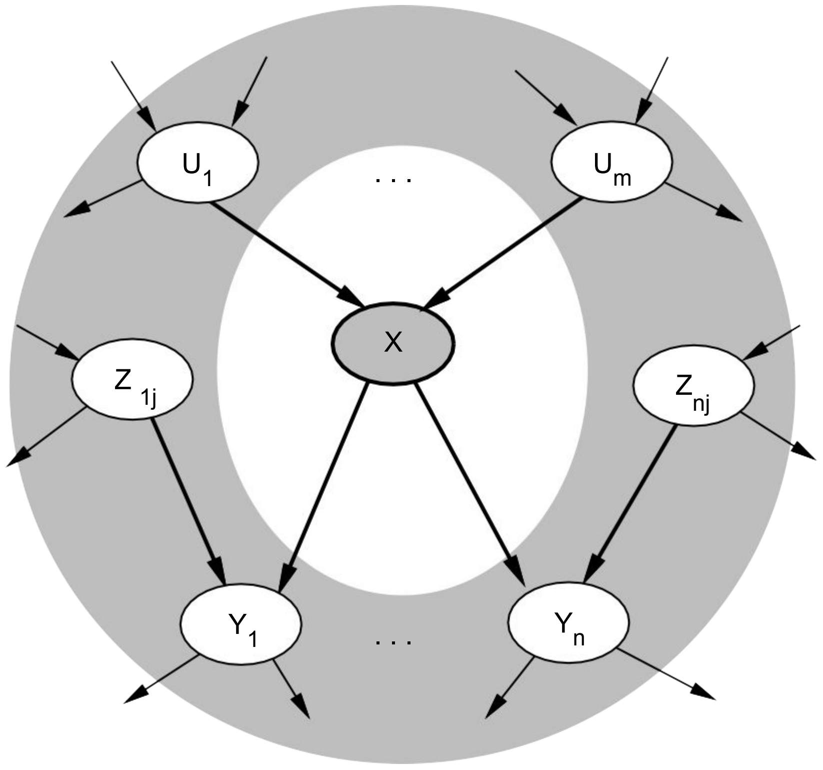
Local semantics: each node is conditionally independent of its nondescendants given its parents



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



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
   $$\mathbf{P}(X_i \mid Parents(X_i)) = \mathbf{P}(X_i \mid X_1, \ldots, X_{i-1})$$

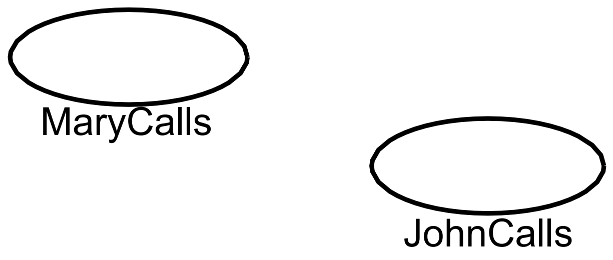
This choice of parents guarantees the global semantics:

$$\mathbf{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbf{P}(X_i \mid X_1, \ldots, X_{i-1})$$ (chain rule)

$$= \prod_{i=1}^{n} \mathbf{P}(X_i \mid Parents(X_i))$$ (by construction)

Example

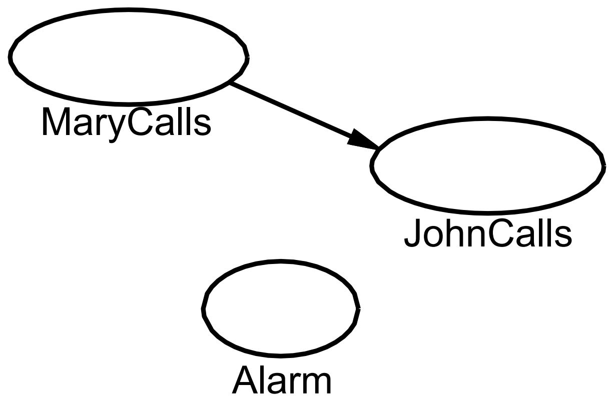
Suppose we choose the ordering M, J, A, B, E



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

Example

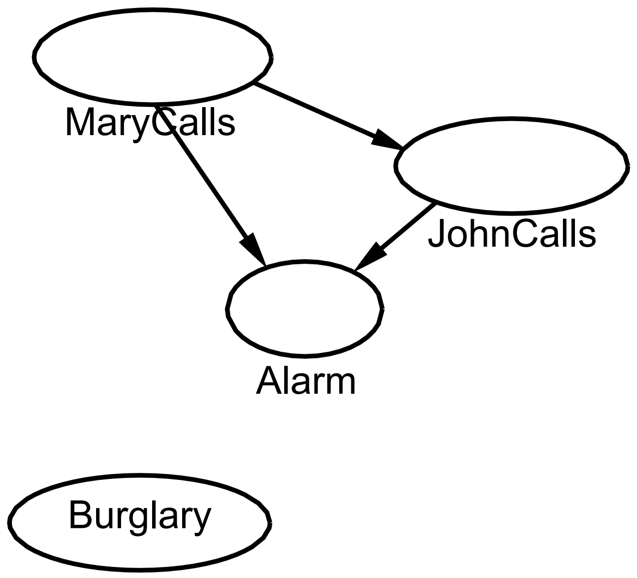
Suppose we choose the ordering M, J, A, B, E



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

Example

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



$P(J|M) = P(J)$? No

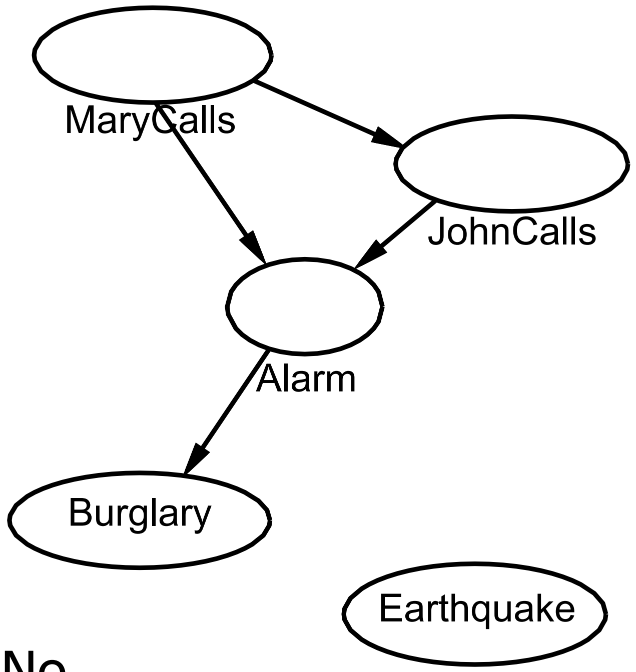
$P(A|J, M) = P(A|J)$? $P(A|J, M) = P(A)$? No

$P(B|A, J, M) = P(B|A)$?

$P(B|A, J, M) = P(B)$?

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

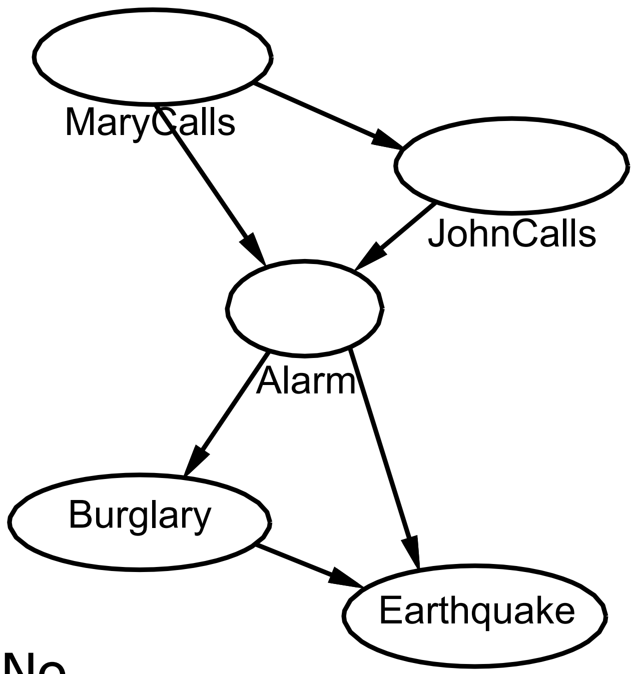
$P(B|A, J, M) = P(B)$? No

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

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

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

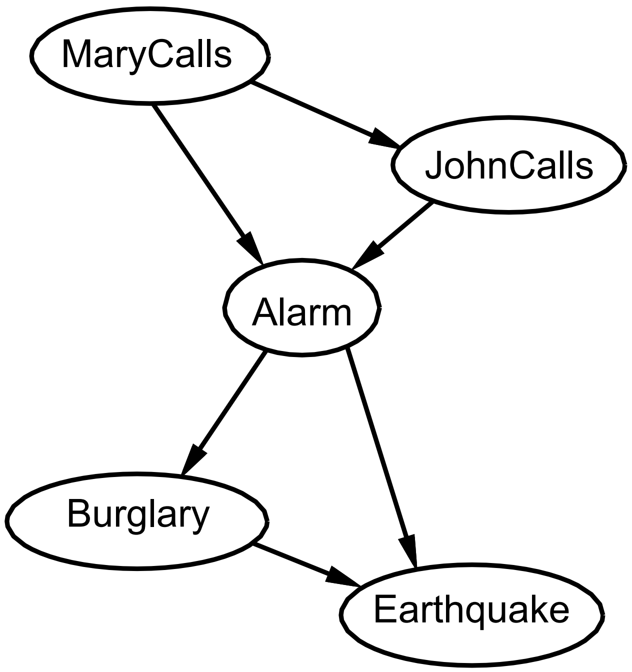
$P(B|A,J,M) = P(B|A)$? Yes

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

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

$P(E|B,A,J,M) = P(E|A,B)$? Yes

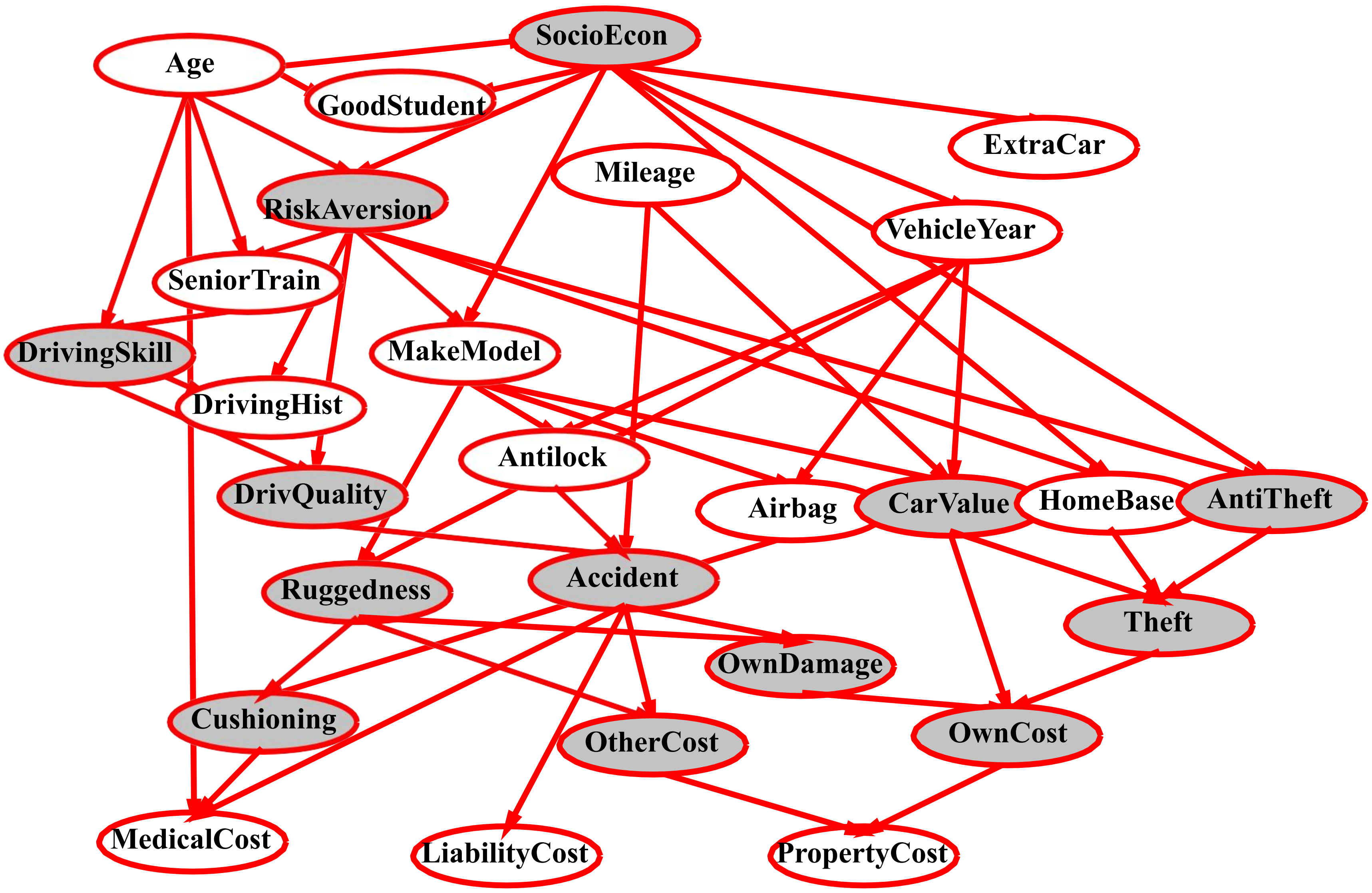
Example contd.



Deciding conditional independence is hard in noncausal directions (Causal models and conditional independence seem hardwired for humans!) Assessing conditional probabilities is hard in noncausal directions

Network is less compact: $1 + 2 + 4 + 2 + 4 = 13$ numbers needed

Example: Car insurance



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:

$X = f(Parents(X))$ for some function f

E.g., Boolean functions

$NorthAmerican \Leftrightarrow Canadian \lor US \lor Mexican$

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

E.g., numerical relationships among continuous variables

Compact conditional distributions contd.

Noisy-OR distributions model multiple noninteracting causes

1) Parents $U_1 \ldots U_k$ include all causes (can add leak node)
2) Independent failure probability q_i for each cause alone

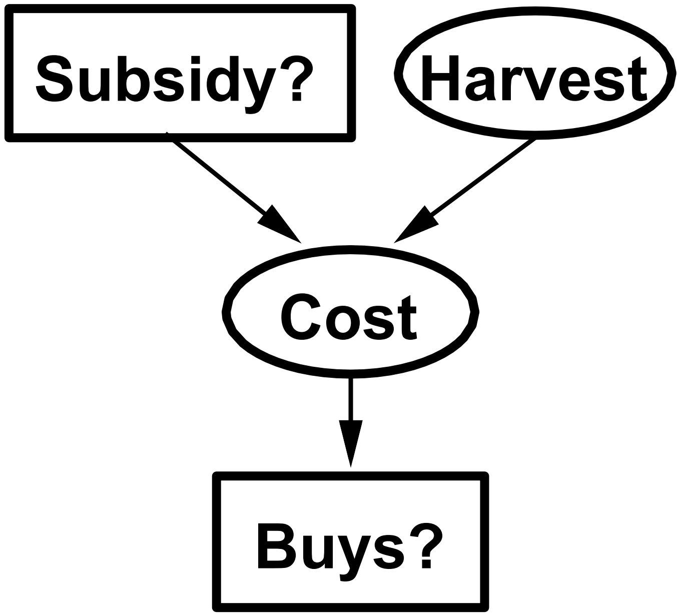
$$\Rightarrow P(X|U_1 \ldots U_j, \neg U_{j+1} \ldots \neg U_k) = 1 - \prod_{i=1}^{j} q_i$$

$Cold$	Flu	$Malaria$	$P(Fever)$	$P(\neg Fever)$
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

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

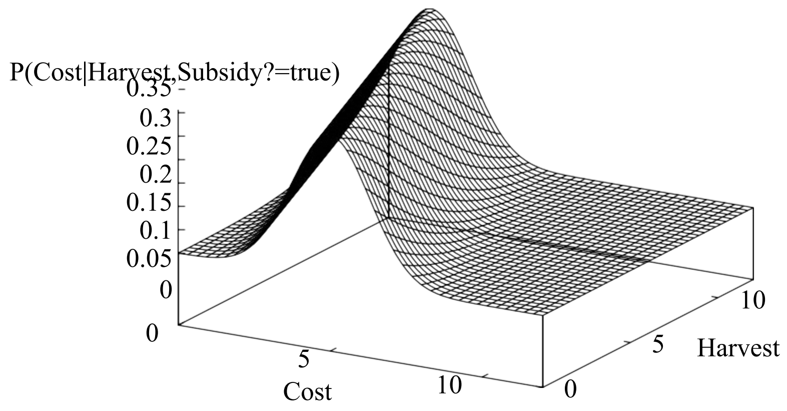
Most common is the linear Gaussian model, e.g.,:

$$P(Cost = c \mid Harvest = h, Subsidy? = true)$$
$$= N(a_t h + b_t, \sigma_t)(c) = \frac{1}{\sigma_t \sqrt{2\pi}} exp\left(-\frac{1}{2}\left(\frac{c - (a_t h + b_t)}{\sigma_t}\right)^2\right)$$

Mean *Cost* varies linearly with *Harvest*, variance is fixed

Linear variation is unreasonable over the full range
but works OK if the likely range of *Harvest* is narrow

Continuous child variables



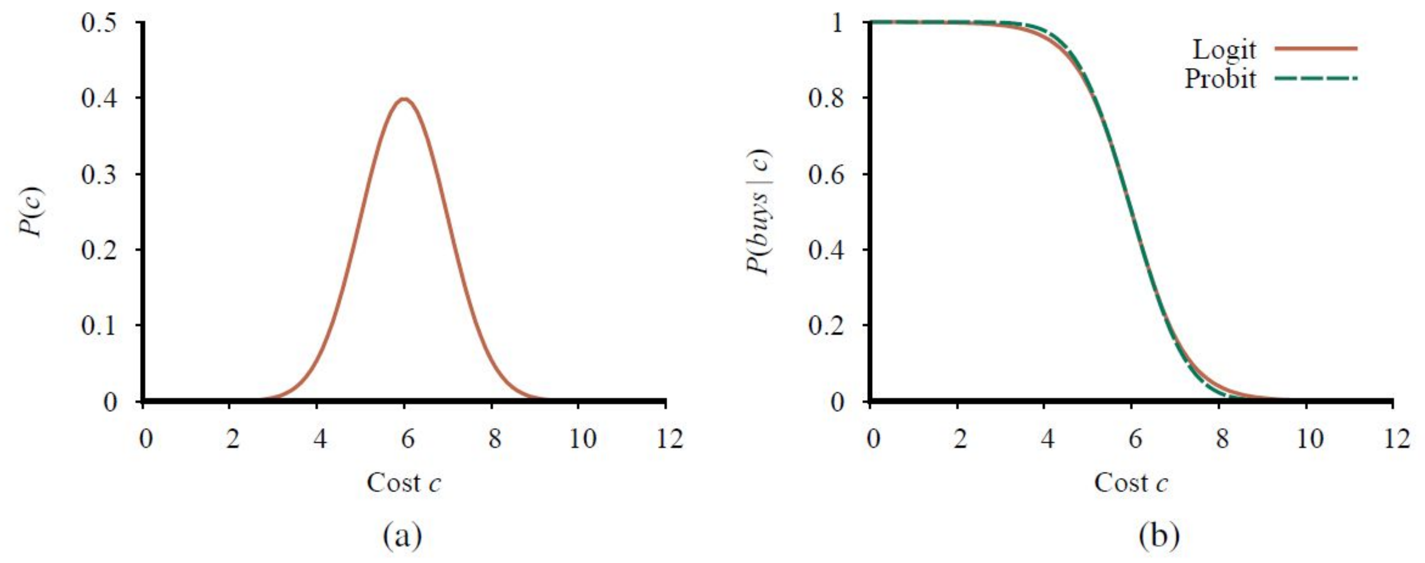
All-continuous network with LG distributions

$\Rightarrow$ 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”



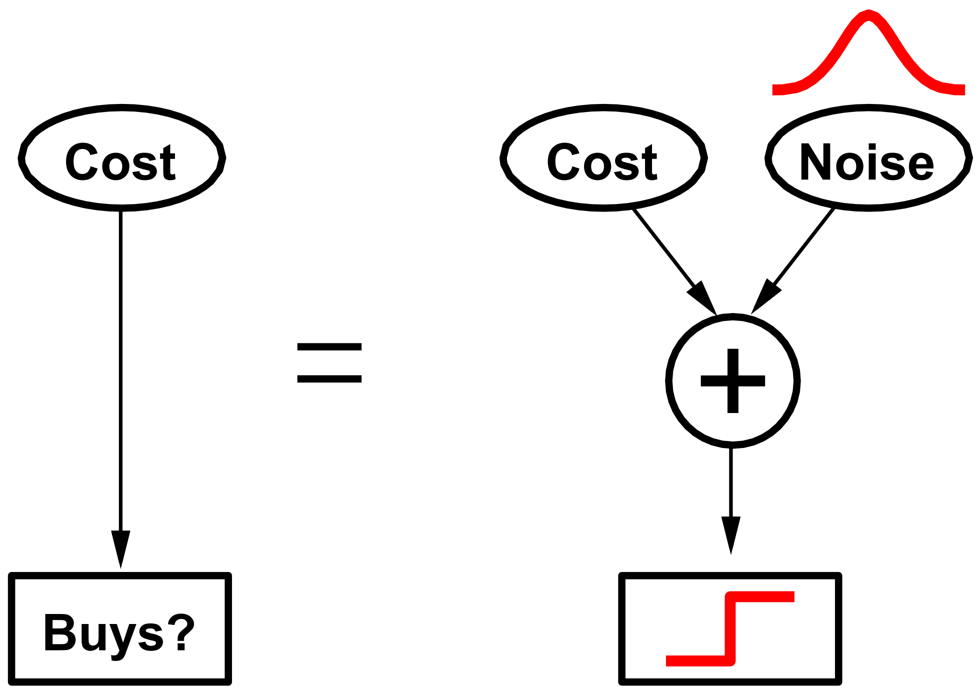
(a) A normal (Gaussian) distribution for the cost threshold, centered on μ = 6.0 with standard deviation σ = 1.0. (b) Expit and probit models for the probability of *buys* given *cost*, for the parameters μ = 6.0 and σ = 1.0.

Probit distribution uses integral of Gaussian: $\Phi(x) = \int_{-\infty}^{x} N(0,1)(x)dx$

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

Why the probit?

1. It’s sort of the right shape
2. Can view as hard threshold whose location is subject to noise

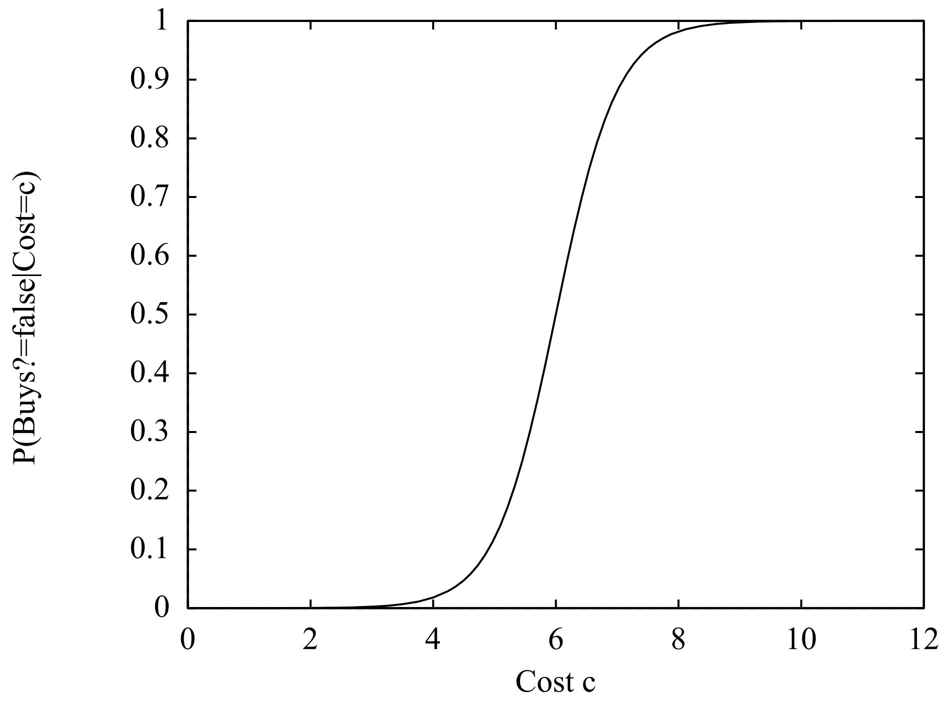


Discrete variable contd.

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

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

Sigmoid has similar shape to probit but much longer tails:



Exact Inference in Bayesian Networks

Simple queries: compute posterior marginal $\mathrm{P}(X_i|\mathrm{E} = \mathrm{e})$
e.g., $P(NoGas|Gauge = empty, Lights = on, Starts = false)$

Conjunctive queries: $\mathrm{P}(X_i, X_j|\mathrm{E} = \mathrm{e}) = \mathrm{P}(X_i|\mathrm{E} = \mathrm{e})\mathrm{P}(X_j|X_i, \mathrm{E} = \mathrm{e})$

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

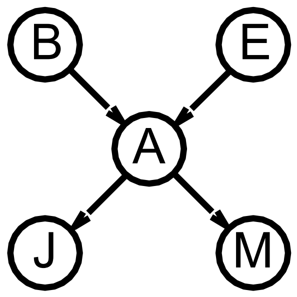
Simple query on the burglary network:

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

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

$= a\mathrm{P}(B, j, m)$

$= a\,\Sigma_e\,\Sigma_a\,\mathrm{P}(B, e, a, j, m)$



Rewrite full joint entries using product of CPT entries:

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

$= a\,\Sigma_e\,\Sigma_a\,\mathrm{P}(B)P(e)\mathrm{P}(a|B, e)P(j|a)P(m|a)$

$= a\mathrm{P}(B)\,\Sigma_e\,P(e)\,\Sigma_a\,\mathrm{P}(a|B, e)P(j|a)P(m|a)$

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

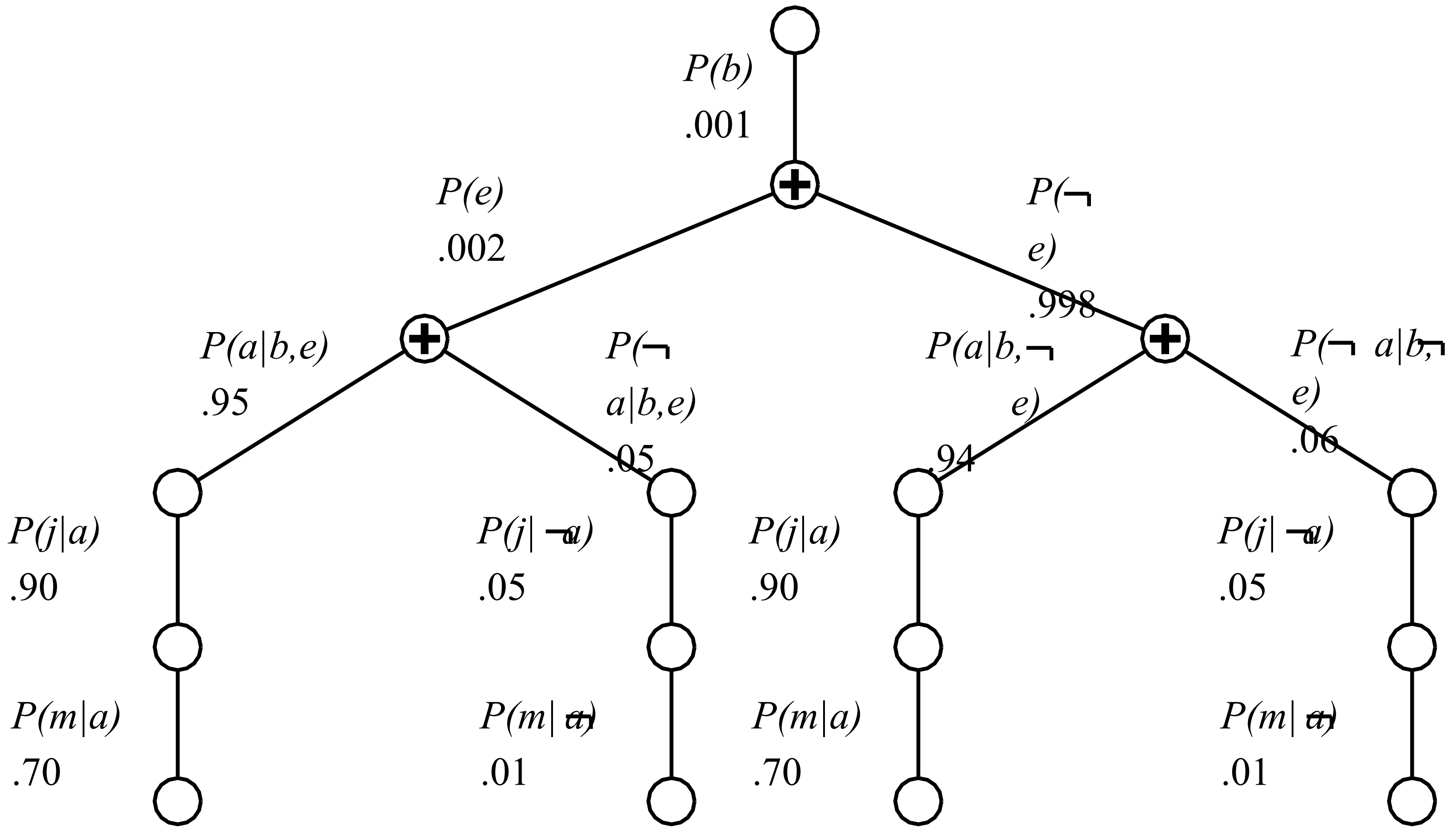
Enumeration algorithm

```
function Enumeration-Ask($X$, e, $bn$) returns a distribution over $X$
    inputs: $X$, the query variable
            e, observed values for variables E
            $bn$, a Bayesian network with variables $\{X\} \cup E \cup Y$

    Q($X$) ← a distribution over $X$, initially empty
    for each value $x_i$ of $X$ do
        extend e with value $x_i$ for $X$
        Q($x_i$) ← Enumerate-All(Vars[$bn$], e)
    return Normalize(Q($X$))
```

```
function Enumerate-All($vars$, e) returns a real number
    if Empty?($vars$) then return 1.0
    $Y$ ← First($vars$)
    if $Y$ has value $y$ in e
        then return $P(y \mid Pa(Y)) \times$ Enumerate-All(Rest($vars$), e)   else
        return $_y$ $P(y \mid Pa(Y)) \times$ Enumerate-All(Rest($vars$), $e_y$)
            where $e_y$ is 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

$$\mathbf{P}(B|j,m) = \alpha \underbrace{\mathbf{P}(B)}_{B} \sum_e \underbrace{P(e)}_{E} \sum_a \underbrace{\mathbf{P}(a|B,e)}_{A} \underbrace{P(j|a)}_{J} \underbrace{P(m|a)}_{M}$$

$$= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) P(j|a) f_M(a)$$

$$= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) f_J(a) f_M(a)$$

$$= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a f_A(a,b,e) f_J(a) f_M(a)$$

$= \alpha \mathbf{P}(B) \Sigma_e P(e) f_{\bar{A}JM}(b,e)$ (sum out A)

$= \alpha \mathbf{P}(B) f_{\bar{E}\bar{A}JM}(b)$ (sum out E)

$$= \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)$$

Variable elimination: Basic operations

Summing out a variable from a product of
factors: move any constant factors outside
the summation
add up submatrices in pointwise product of remaining factors

$$\Sigma_x f_1 \times \cdot\cdot\cdot \times f_k = f_1 \times \cdot\cdot\cdot \times f_i \Sigma_x f_{i+1} \times \cdot\cdot\cdot \times f_k = f_1 \times \cdot\cdot\cdot \times f_i \times f_{\bar{X}}$$

assuming $f_1, \ldots, f_i$ do not depend on X

Pointwise product of factors f_1 and f_2:

$$f_1(x_1, \ldots, x_j, y_1, \ldots, y_k) \times f_2(y_1, \ldots, y_k, z_1, \ldots, z_l)$$
$$= f(x_1, \ldots, x_j, y_1, \ldots, y_k, z_1, \ldots, z_l)$$

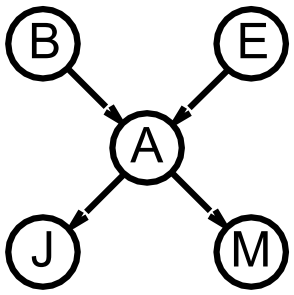
E.g., $f_1(a, b) \times f_2(b, c) = f(a, b, c)$

Variable elimination algorithm

```
function Elimination-Ask($X$, e, $bn$) returns a distribution over $X$
    inputs: $X$, the query variable
            e, evidence specified as an event
            $bn$, a belief network specifying joint distribution $P(X_1, \ldots, X_n)$
    $factors \leftarrow$ [ ]; $vars \leftarrow$ Reverse(Vars[$bn$])
    for each $var$ in $vars$ do
        $factors \leftarrow$ [Make-Factor($var$ , e)|$factors$]
        if $var$ is a hidden variable then $factors \leftarrow$ Sum-Out($var$, $factors$)
    return Normalize(Pointwise-Product($factors$))
```

Irrelevant variables

Consider the query $P(JohnCalls|Burglary = true)$



$$P(J|b) = \alpha P(b) {}_e P(e) {}_a P(a|b, e) P(J|a) {}_m P(m|a)$$

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

Thm 1: Y is irrelevant unless $Y \in Ancestors(\{X\} \cup \mathrm{E})$

Here, $X = JohnCalls$, $\mathrm{E} = \{Burglary\}$, and $Ancestors(\{X\} \cup \mathrm{E}) = \{Alarm, Earthquake\}$ so $MaryCalls$ is irrelevant

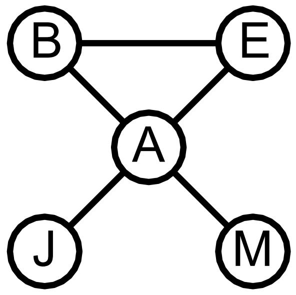
(Compare this to backward chaining from the query in Horn clause KBs)

Pearson

Irrelevant variables contd.

Defn: moral graph of Bayes net: marry all parents and drop arrows
Defn: A is m-separated from B by C iff separated by C in the moral graph

Thm 2: Y is irrelevant if m-separated from X by E



For $P(JohnCalls|Alarm = true)$, both $Burglary$ and $Earthquake$ are irrelevant

Complexity of exact inference

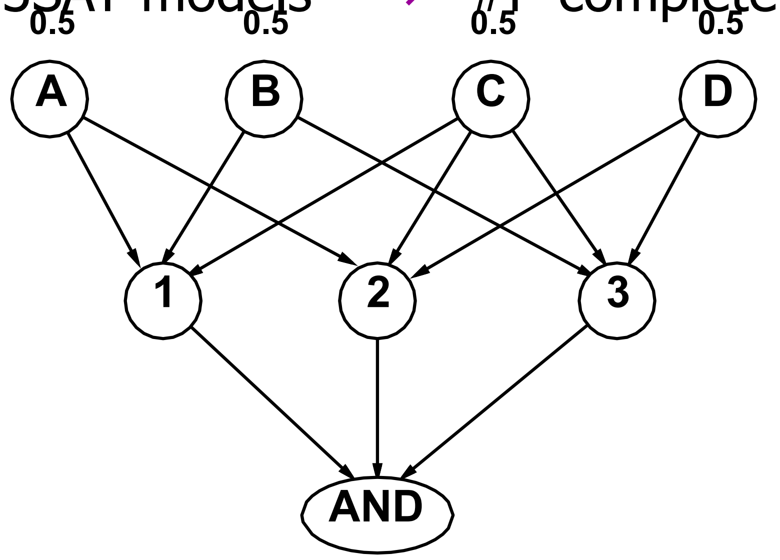
Singly connected networks (or polytrees):
- any two nodes are connected by at most one (undirected) path
- time and space cost of variable elimination are $O(d^k n)$

Multiply connected networks:
- can reduce 3SAT to exact inference $\Rightarrow$ NP-hard
- equivalent to counting 3SAT models $\Rightarrow$ #P-complete

1. A v B v C

2. C v D v A

3. B v C v D



Approximate Inference for Bayesian Networks

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

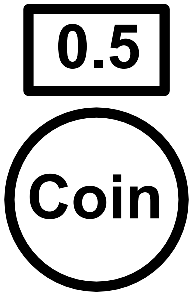
Outline:

– 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

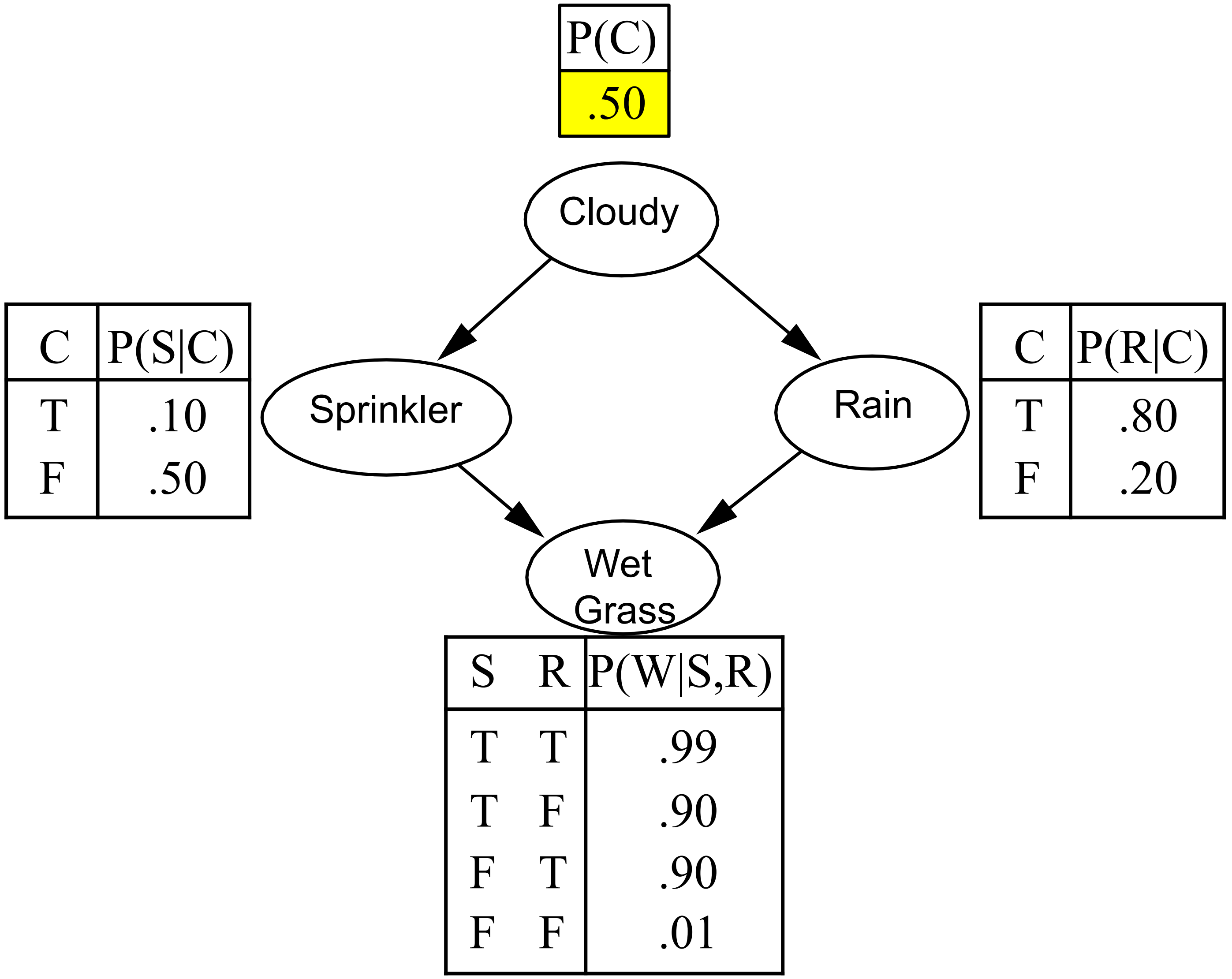


Sampling from an empty network

```
function Prior-Sample($bn$) returns an event sampled from $bn$
    inputs: $bn$, a belief network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

    $\mathbf{x} \leftarrow$ an event with $n$ elements
    for $i$ = 1 to $n$ do
        $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
            given the values of $Parents(X_i)$ in $\mathbf{x}$
    return $\mathbf{x}$
```

Example



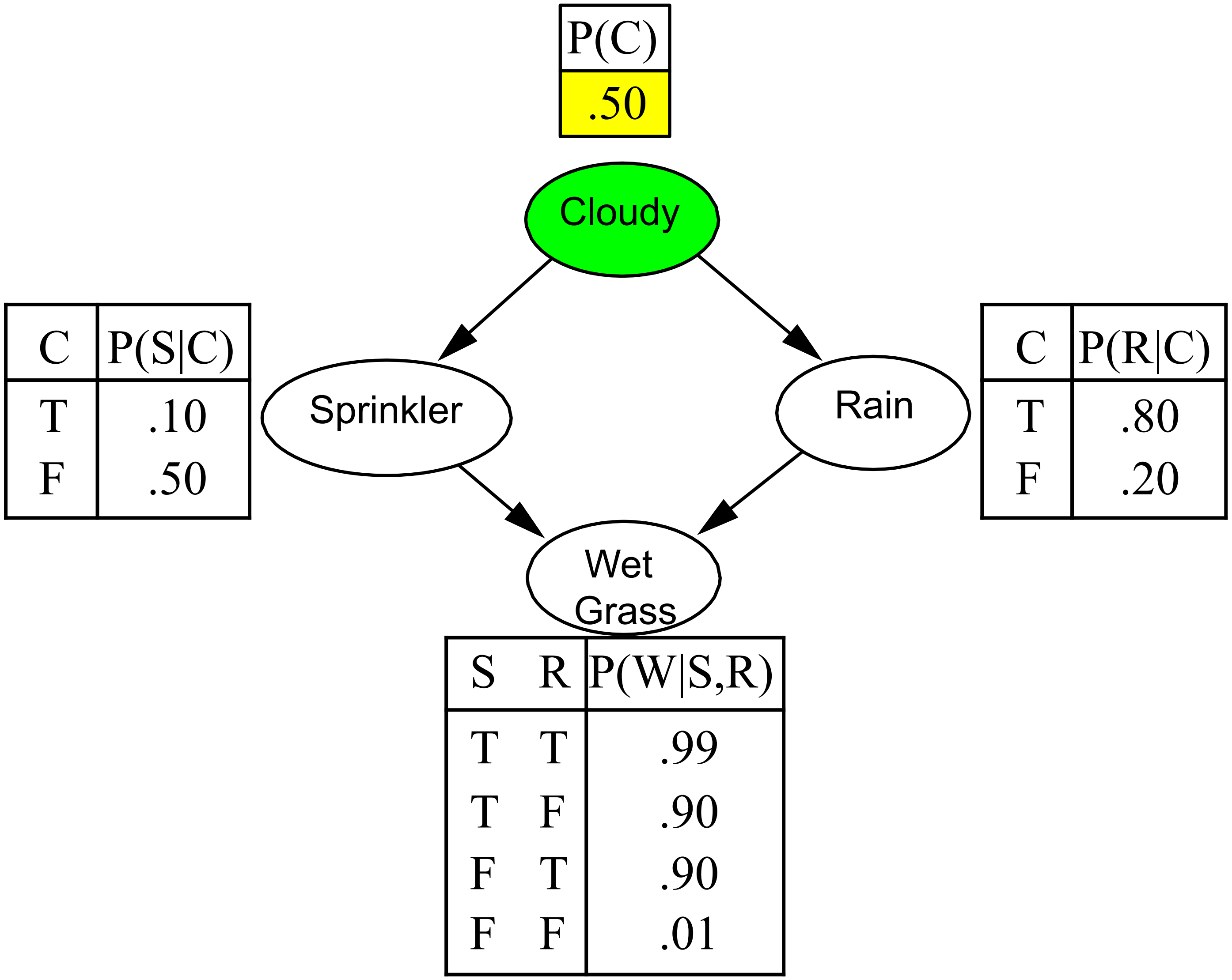
P(C)
.50

C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example



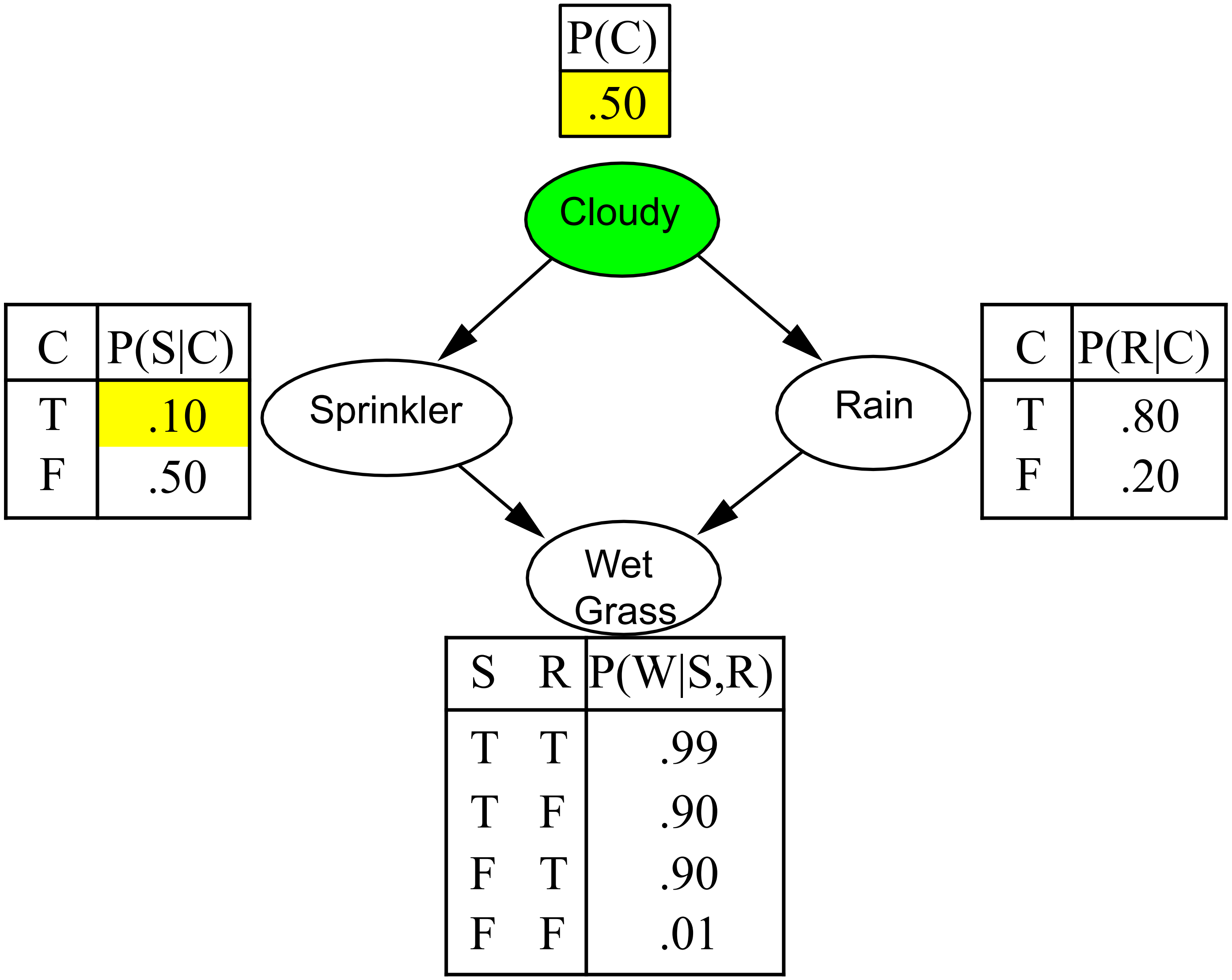
P(C)
.50

C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example



P(C)
.50

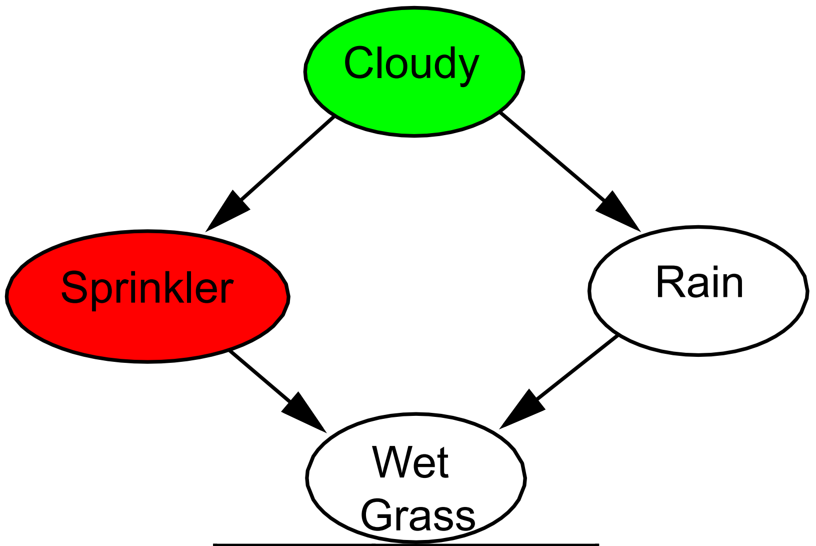
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example

P(C)
.50

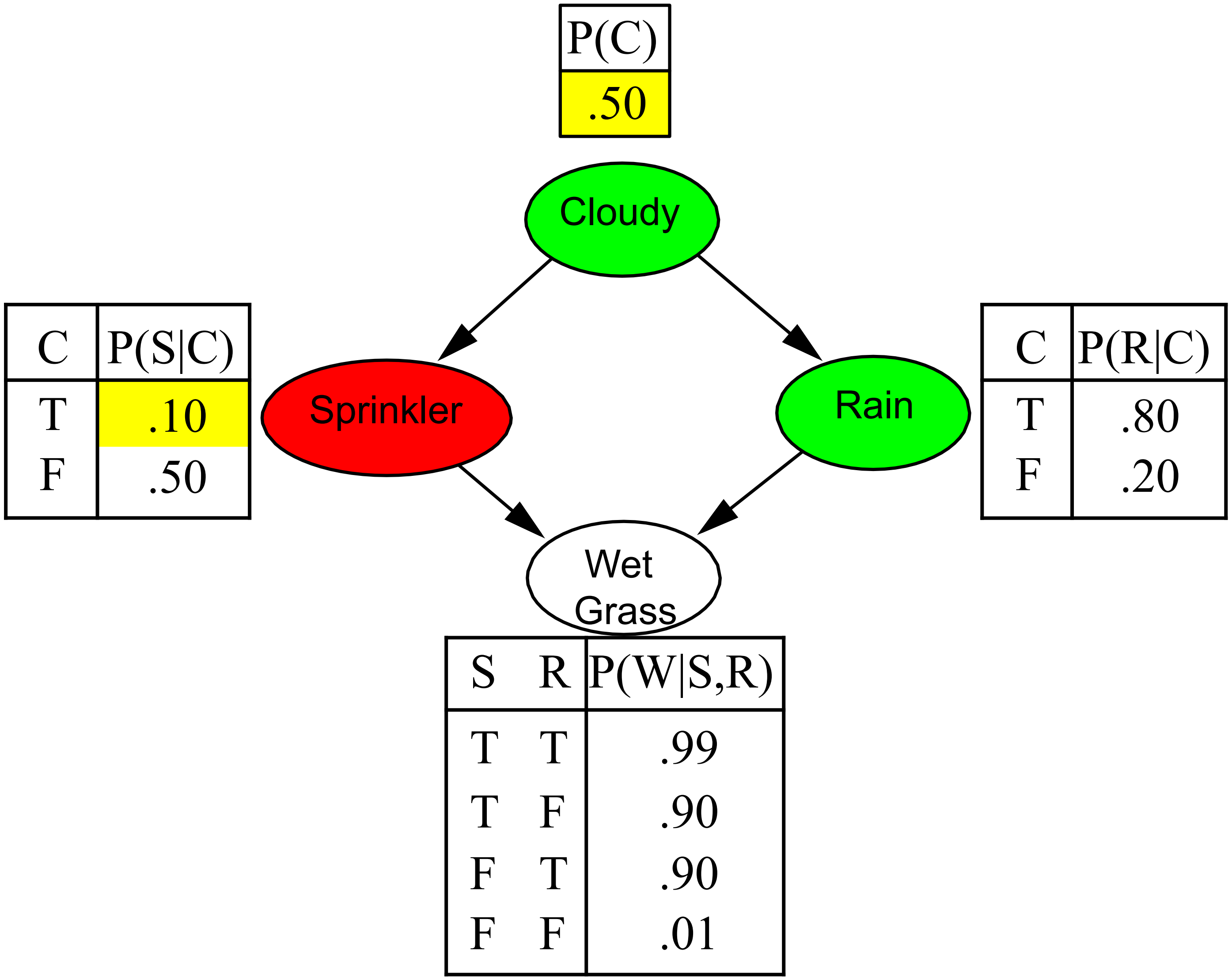


C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example



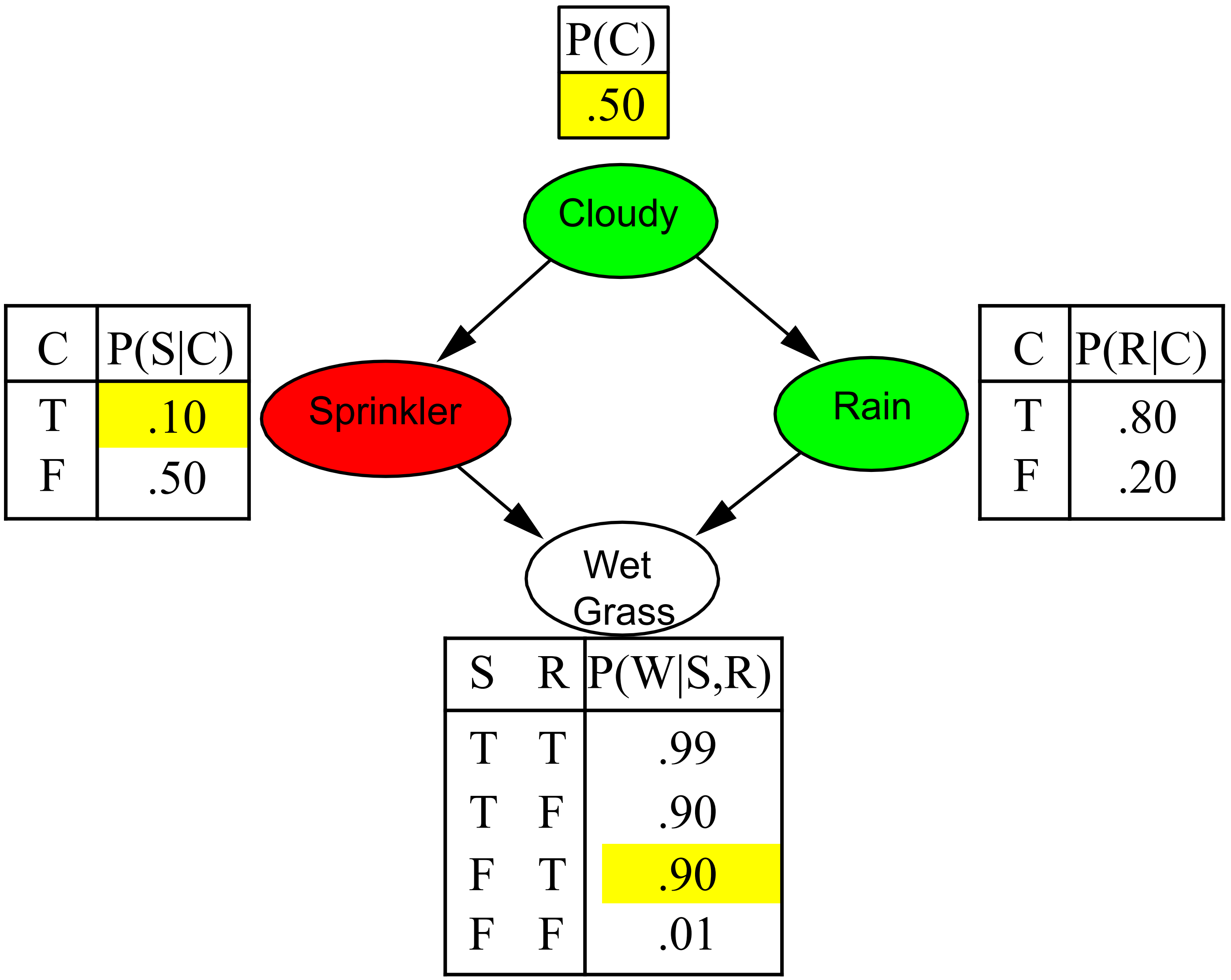
P(C)
.50

C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example



P(C)
.50

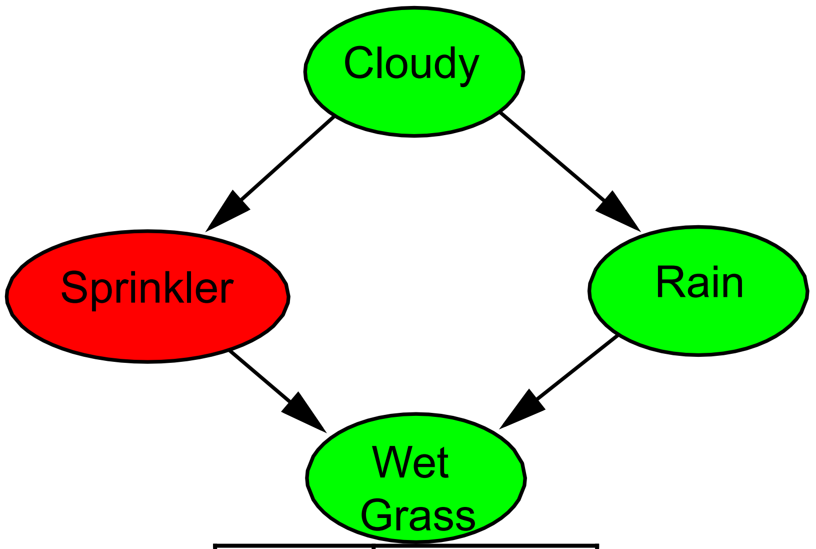
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example

P(C)
.50



C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Sampling from an empty network contd.

Probability that PriorSample generates a particular event

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

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

Then we have

$$\lim_{N\to\infty} \hat{P}(x_1, \ldots, x_n) = \lim_{N\to\infty} N_{PS}(x_1, \ldots, x_n)/N = S_{PS}(x_1, \ldots, x_n) = P(x_1 \ldots x_n)$$

That is, estimates derived from PriorSample are consistent

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

Rejection sampling

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

```
function Rejection-Sampling(X, e, bn, N) returns an estimate of P(X|e)
    local variables: N, a vector of counts over X, initially zero
    for j = 1 to N do
        x ← Prior-Sample(bn)
        if x is consistent with e then
            N[x] ← N[x]+1 where x is the value of X in x
    return Normalize(N[X])
```

E.g., estimate $P(Rain|Sprinkler = true)$ using 100 samples 27 samples have $Sprinkler = true$
Of these, 8 have $Rain = true$ and 19 have $Rain = false$.

$\hat{P}(Rain|Sprinkler = true) = \text{Normalize}(\langle 8, 19\rangle) = \langle 0.296, 0.704\rangle$

Similar to a basic real-world empirical estimation procedure

Analysis of rejection sampling

$\hat{P}(X|e) = \alpha N_{PS}(X, e)$ (algorithm defn.)

$= N_{PS}(X, e)/N_{PS}(e)$ (normalized by $N_{PS}(e)$)

$\approx P(X, e)/P(e)$ (property of PriorSample)

$= P(X|e)$ (defn. of conditional probability)

Hence rejection sampling returns consistent posterior estimates

Problem: hopelessly expensive if $P(e)$ is small

$P(e)$ 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

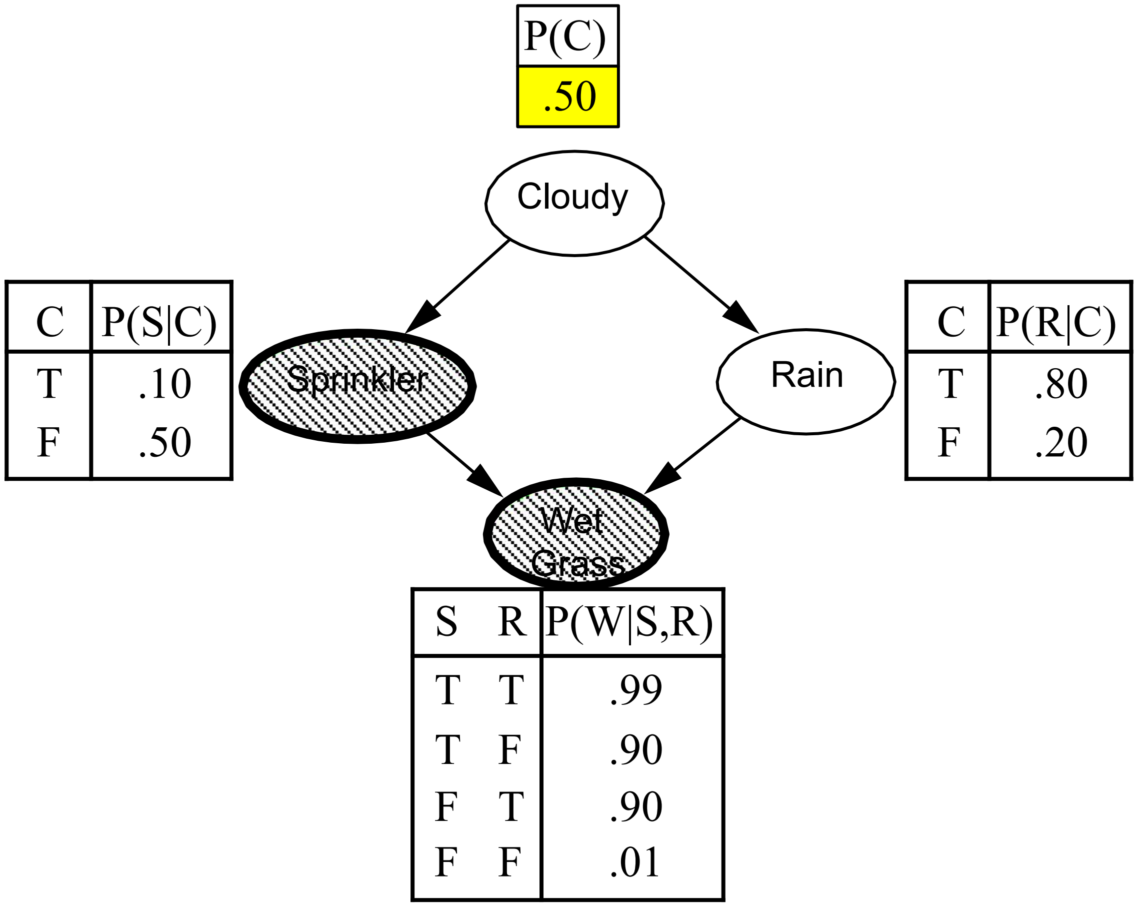
```
function Likelihood-Weighting($X$, e, $bn$, $N$) returns an estimate of $P(X|\mathrm{e})$
    local variables: W, a vector of weighted counts over $X$, initially zero

    for $j$ = 1 to $N$ do
        x, $w$ ← Weighted-Sample($bn$)
        W[$x$] ← W[$x$] + $w$ where $x$ is the value of $X$ in
    x  return Normalize(W[$X$])
```

```
function Weighted-Sample($bn$, e) returns an event and a weight

    x ← an event with $n$ elements; $w$ ← 1
    for $i$ = 1 to $n$ do
        if $X_i$ has a value $x_i$ in e
            then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
            else $x_i$ ← a random sample from $\mathrm{P}(X_i \mid parents(X_i))$
    return x, $w$
```

Likelihood weighting example



P(C)
.50

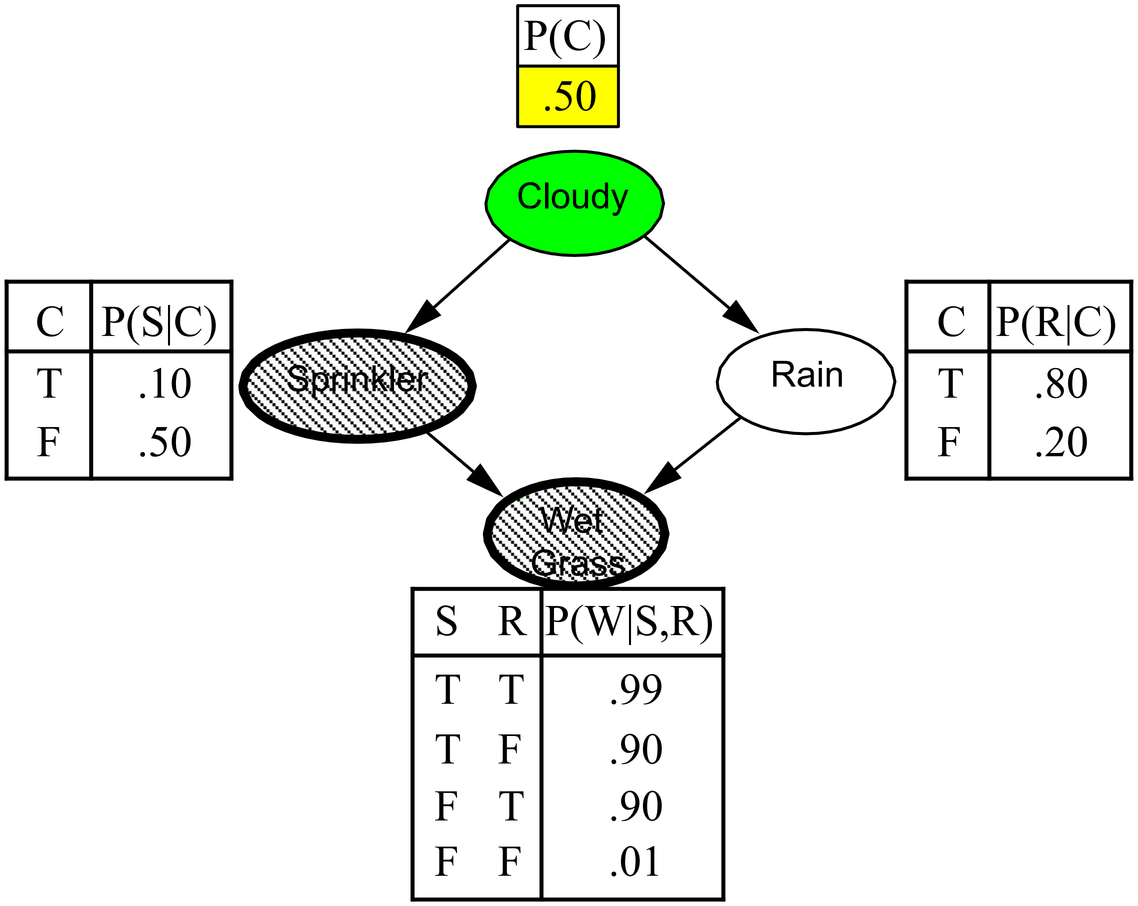
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0$

Likelihood weighting example



P(C)
.50

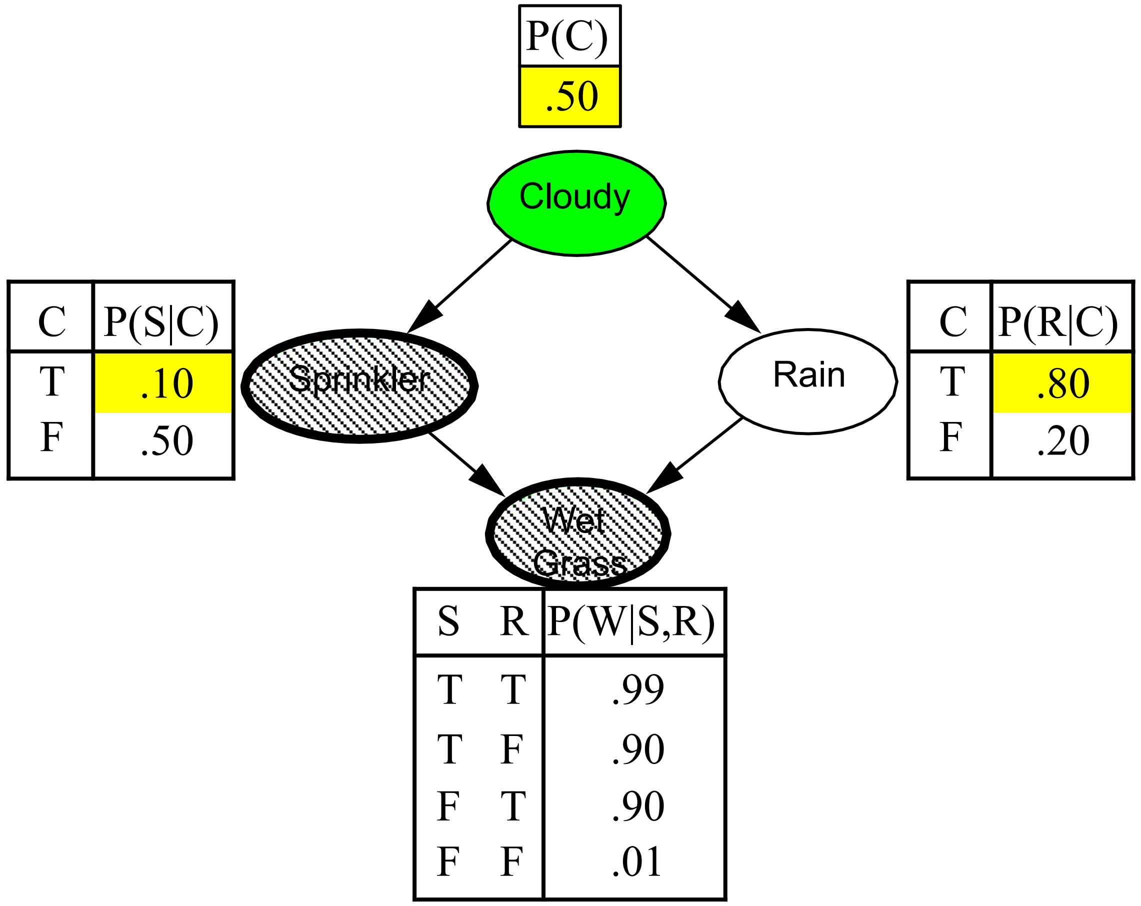
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0$

Likelihood weighting example



P(C)
.50

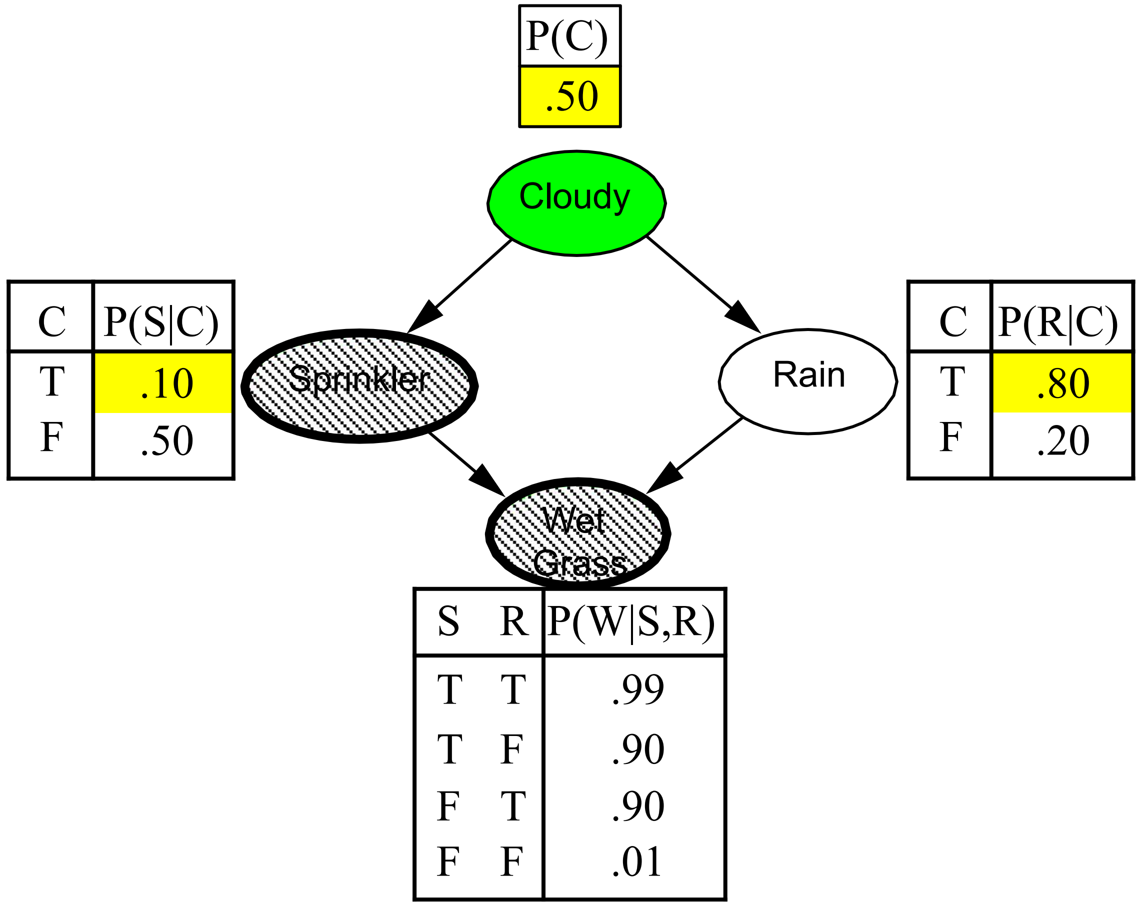
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0$

Likelihood weighting example



P(C)
.50

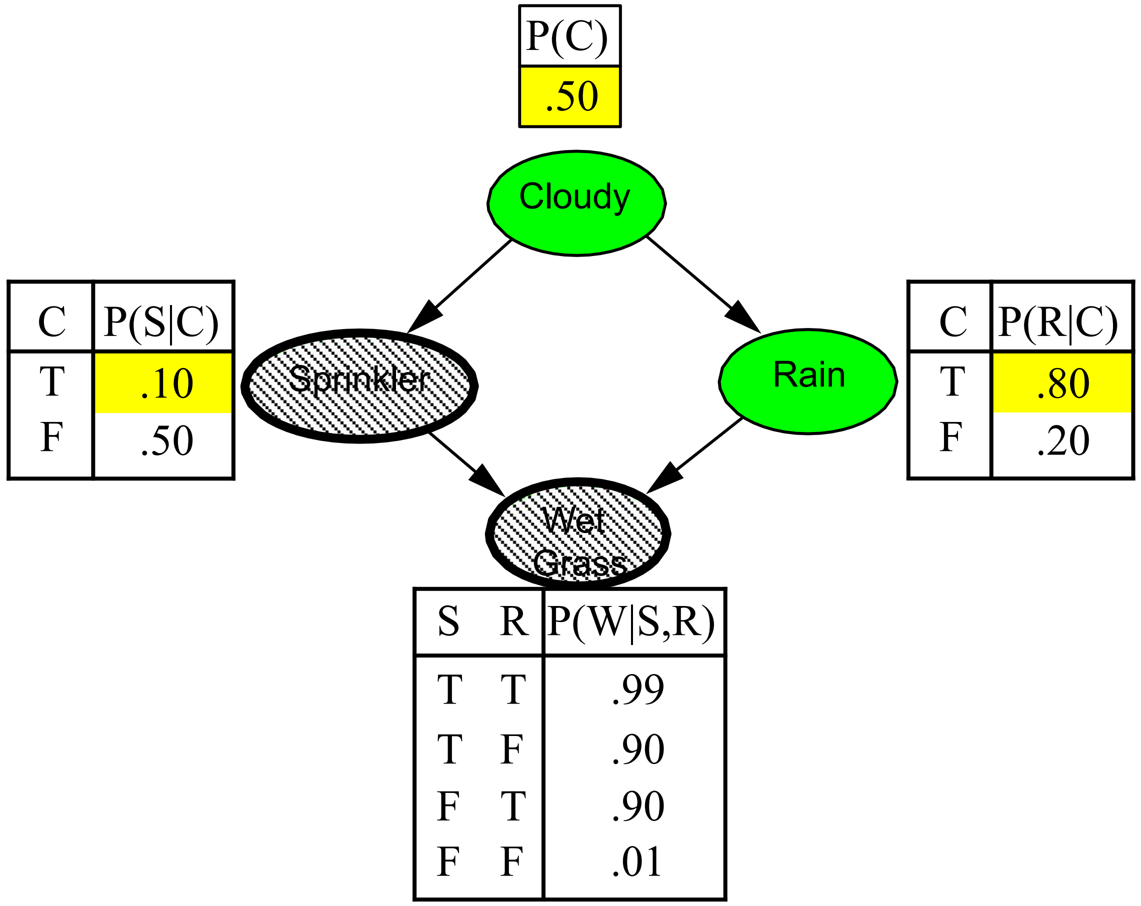
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0 \times 0.1$

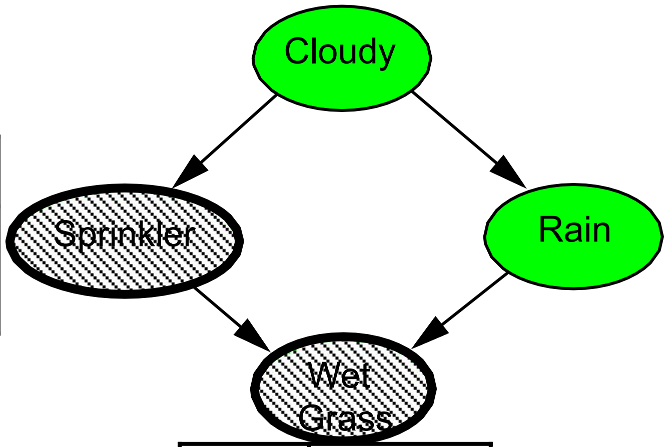
Likelihood weighting example



$w = 1.0 \times 0.1$

Likelihood weighting example

P(C)
.50



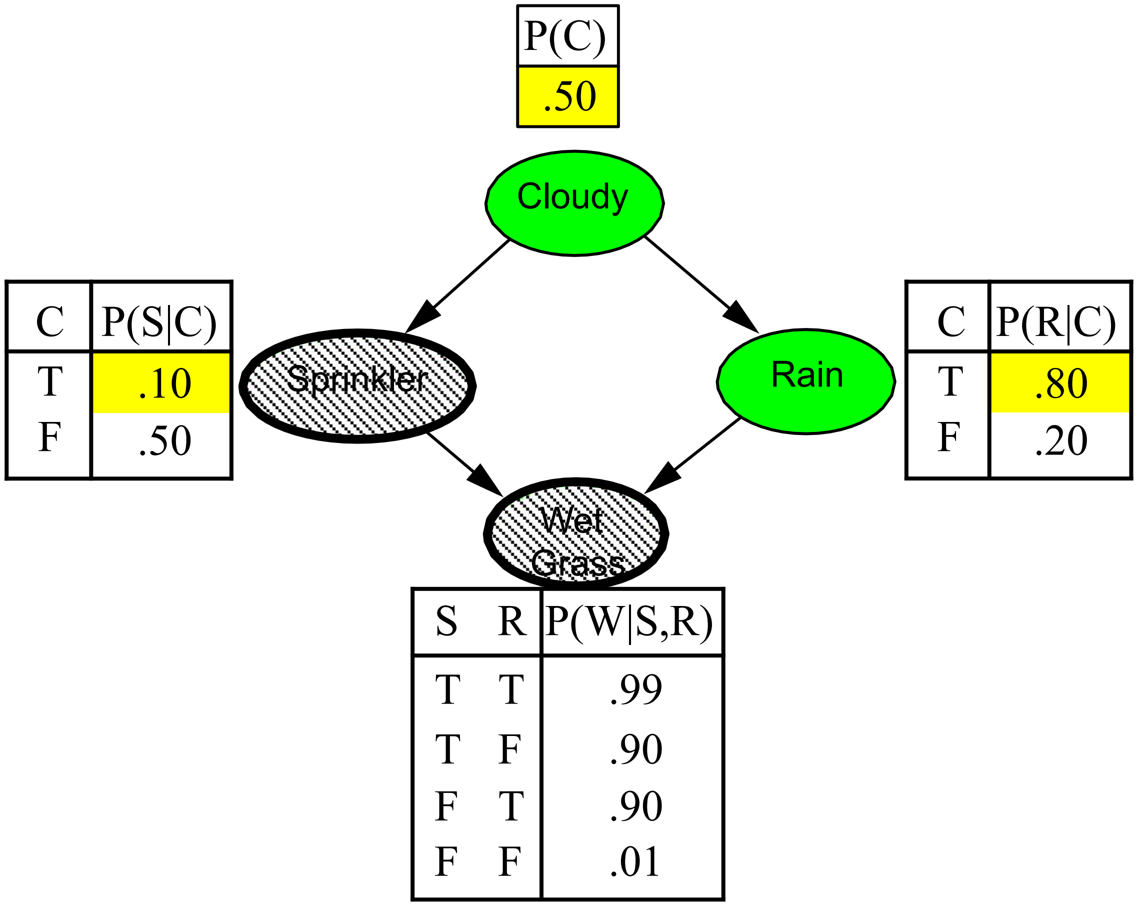
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0 \times$
0.1

Likelihood weighting example



P(C)
.50

C	P(S|C)
T	.10
F	.50

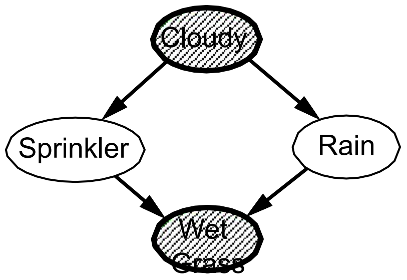
C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

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

Likelihood weighting analysis

Sampling probability for WeightedSample
is $S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i | parents(Z_i))$
Note: pays attention to evidence in ancestors
only $\Rightarrow$ somewhere "in between" prior
and posterior distribution



Weight for a given sample $\mathbf{z}$,
$\mathbf{e}$ is $w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{m} P(e_i | parents(E_i))$

Weighted sampling
probability is
$S_{WS}(\mathbf{z}, \mathbf{e}) w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i | parents(Z_i)) \prod_{i=1}^{m} P(e_i | parents(E_i))$
$= P(\mathbf{z}, \mathbf{e})$ (by standard global semantics of
network)

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.

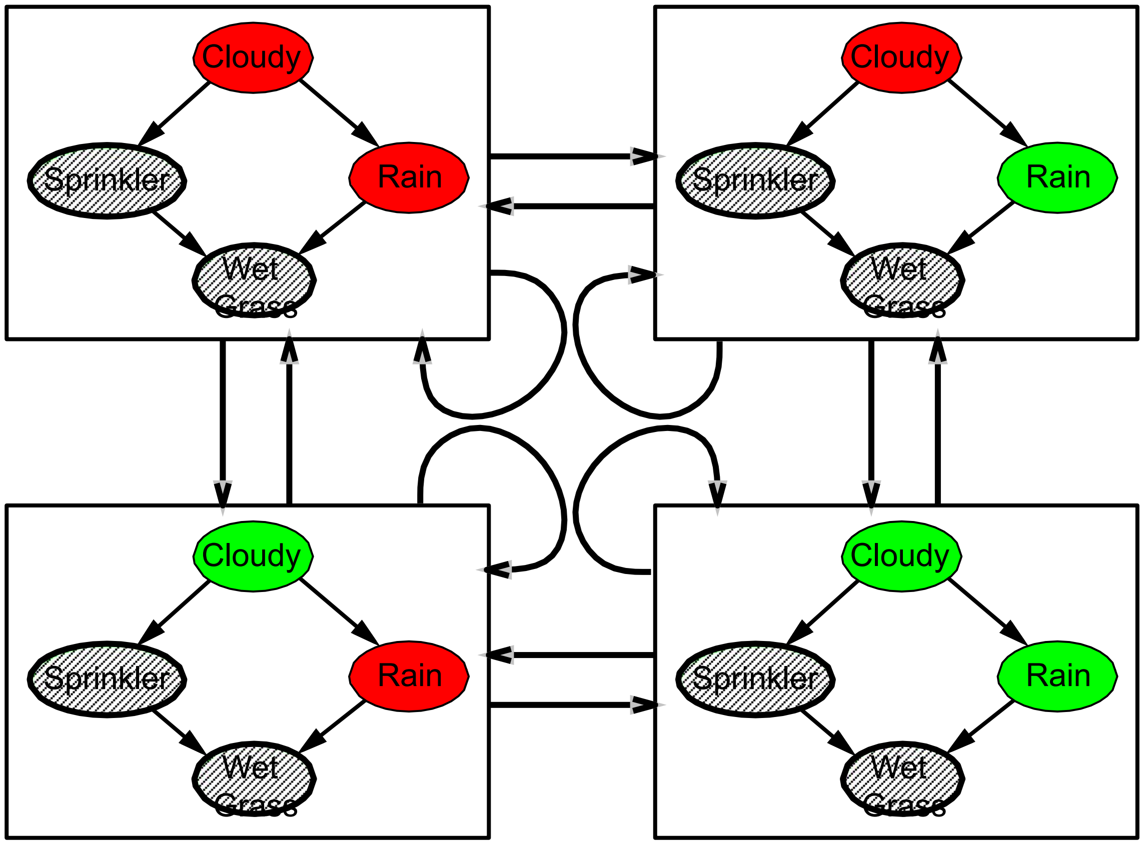
Generate next state by sampling one variable given Markov blanket Sample each variable in turn, keeping evidence fixed

```
function MCMC-Ask($X$, e, $bn$, $N$) returns an estimate of $P(X|\mathrm{e})$
    local variables: N[$X$], a vector of counts over $X$, initially zero
                     Z, the nonevidence variables in $bn$
                     x, the current state of the network, initially copied from e
    initialize x with random values for the variables in Y
    for $j$ = 1 to $N$ do
        for each $Z_i$ in Z do
            sample the value of $Z_i$ in x from P($Z_i$|$mb(Z_i)$)
                given the values of $MB(Z_i)$ in x
            N[$x$] ← N[$x$] + 1 where $x$ is the value of $X$ in
    x  return Normalize(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 contd.

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.

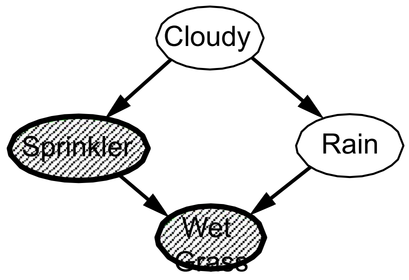
E.g., visit 100 states
31 have $Rain = true$, 69 have $Rain = false$

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

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 *W etGrass*



Probability given the Markov blanket is calculated as follows:

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

Easily implemented in message-passing parallel

systems, brains Main computational problems:

1) Difficult to tell if convergence has been achieved
2) Can be wasteful if Markov blanket is large:
   $P(X_i | mb(X_i))$ won't change much (law of large numbers)

Causal Networks

Causal Networks: a restricted class of Bayesian networks that forbids all but causally compatible orderings.

$P(c, r, s, w, g) = P(c)\, P(r \mid c)\, P(s \mid c)\, P(w \mid r, s)\, P(g \mid w)$

$C = f_C(U_C)$
$R = f_R(C, U_R)$
$S = f_S(C, U_S)$
$W = f_W(R, S, U_W)$
$G = f_G(W, U_G)$

For example, suppose we turn the sprinkler on—*do*(*Sprinkler* = *true*)

$P(c, r, w, g \mid do(S = true)) = P(c)\, P(r \mid c)\, P(w \mid r, s = true)\, P(g \mid w)$

Causal Networks

Example:

Predict the effect of turning on the sprinkler on a downstream variable such as *GreenerGrass*, but the adjustment formula must take into account not only the direct route from Sprinkler, but also the "back door" route via Cloudy and Rain.

$$P(g|do(S = true)) = \sum_r P(g|S = true, r)P(r)$$

we wish to find the effect of $do(X_j = x_{jk})$ on a variable X_i,

Back-door criterion

allows us to write an adjustment formula that conditions on any set of variables **Z** that closes the back door, so to speak

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 ⇒ 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

Random sampling techniques such as likelihood weighting and Markov chain Monte
Carlo can give reasonable estimates of the true posterior probabilities in a network and can cope with much larger networks than can exact algorithms.