



Inference in Bayesian Networks

Chapter 14.4-5
in
Russell/Norvig



Outline



- ◇ Exact inference by Enumeration
- ◇ Exact inference by Variable Elimination
- ◇ Approximate inference by stochastic Simulation
- ◇ Approximate inference by Markov Chain Monte Carlo



Inference Tasks



Simple queries: compute posterior marginal $P(X_i | E=e)$

e.g., $P(\text{NoGas} | \text{Gauge}=\text{empty}, \text{Lights}=\text{on}, \text{Starts}=\text{false})$

Conjunctive queries: $P(X_i, X_j | E=e) = P(X_i | E=e)P(X_j | X_i, E=e)$

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?

Sensitive analysis: which probability values are most critical?

Explanation: why do I need a new starter motor?



Repetition: Inference by enumeration (1)



Start with the joint distribution:

	<i>toothache</i>		\neg <i>toothache</i>	
	<i>catch</i>	\neg <i>catch</i>	<i>catch</i>	\neg <i>catch</i>
<i>cavity</i>	.108	.012	.072	.008
\neg <i>cavity</i>	.016	.064	.144	.576

For any proposition Θ , just sum the atomic events where they are true:

$$P(\Theta) = \sum_{\{\omega: \omega \models \Theta\}} P(\omega)$$

$$P(\text{toothache}) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2$$





Repetition: Inference by enumeration (2)

Start with the joint distribution:

	<i>toothache</i>		\neg <i>toothache</i>	
	<i>catch</i>	\neg <i>catch</i>	<i>catch</i>	\neg <i>catch</i>
<i>cavity</i>	.108	.012	.072	.008
\neg <i>cavity</i>	.016	.064	.144	.576

For any proposition Θ , sum the atomic events where they are true:

$$P(\Theta) = \sum_{\{\omega: \omega \models \Theta\}} P(\omega)$$

$$P(\text{cavity} \vee \text{toothache}) = 0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28$$



Repetition: Inference by enumeration (3)



Start with the joint distribution:

	<i>toothache</i>		\neg <i>toothache</i>	
	<i>catch</i>	\neg <i>catch</i>	<i>catch</i>	\neg <i>catch</i>
<i>cavity</i>	.108	.012	.072	.008
\neg <i>cavity</i>	.016	.064	.144	.576

Can also compute conditional probabilities:

$$\begin{aligned} P(\neg \text{cavity} | \text{toothache}) &= \frac{P(\neg \text{cavity} \wedge \text{toothache})}{P(\text{toothache})} \\ &= \frac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064} = 0.4 \end{aligned}$$



Normalization



Start with the joint distribution:

	<i>toothache</i>		\neg <i>toothache</i>	
	<i>catch</i>	\neg <i>catch</i>	<i>catch</i>	\neg <i>catch</i>
<i>cavity</i>	.108	.012	.072	.008
\neg <i>cavity</i>	.016	.064	.144	.576

Dominator can be viewed as a normalization constant:

$$\begin{aligned} \mathbf{P}(\text{Cavity}|\text{toothache}) &= \frac{P(\neg \text{cavity} \wedge \text{toothache})}{P(\text{toothache})} = \alpha \mathbf{P}(\text{Cavity}, \text{toothache}) \\ &= \alpha [\mathbf{P}(\text{Cavity}, \text{toothache}, \text{catch}) + \mathbf{P}(\text{Cavity}, \text{toothache}, \neg \text{catch})] \\ &= \alpha [<0.108, 0.016> + <0.012, 0.064>] = \alpha <0.12, 0.08> = <0.6, 0.4> \end{aligned}$$

General idea: compute distribution on query variable by fixing evidence variables and summing over hidden variables



Inference by Enumeration



We are interested in:

the posterior joint distribution of the query variables Y
given specific values e for the evidence variables E

Let the hidden variables be $H = X - Y - E$

Then the required summation of joint entries is done by summing out the hidden variables:

$$P(Y|E = e) = \alpha \sum_h P(Y, E = e, H = h)$$

The terms in the summation are joint entries because Y , E , and H together exhaust the set of random variables.



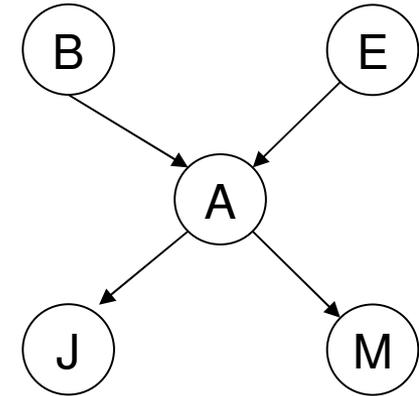
Bayesian Networks: Inference by Enumeration



For Example: Suppose a query $P(\text{Burglary} | \text{JohnCalls}=\text{true}, \text{MaryCalls}=\text{true})$
Hidden variables: *Earthquake, Alarm*

Simple query on the burglary network:

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



Rewrite full joint entries using product of CPT entries:

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

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 $\{X\} \cup E \cup Y$

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

for each value x_i of X do

 extend e with value x_i for X

$Q(x_i) \leftarrow$ 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

$Y \leftarrow$ FIRST($vars$)

if Y has value y in e

then return $P(y | Pa(Y)) \times$ ENUMERATE-ALL($REST(vars), e$)

else return $\sum_y P(y | Pa(Y)) \times$ ENUMERATE-ALL($REST(vars), e_y$)

 where e_y is e extended with $Y = y$

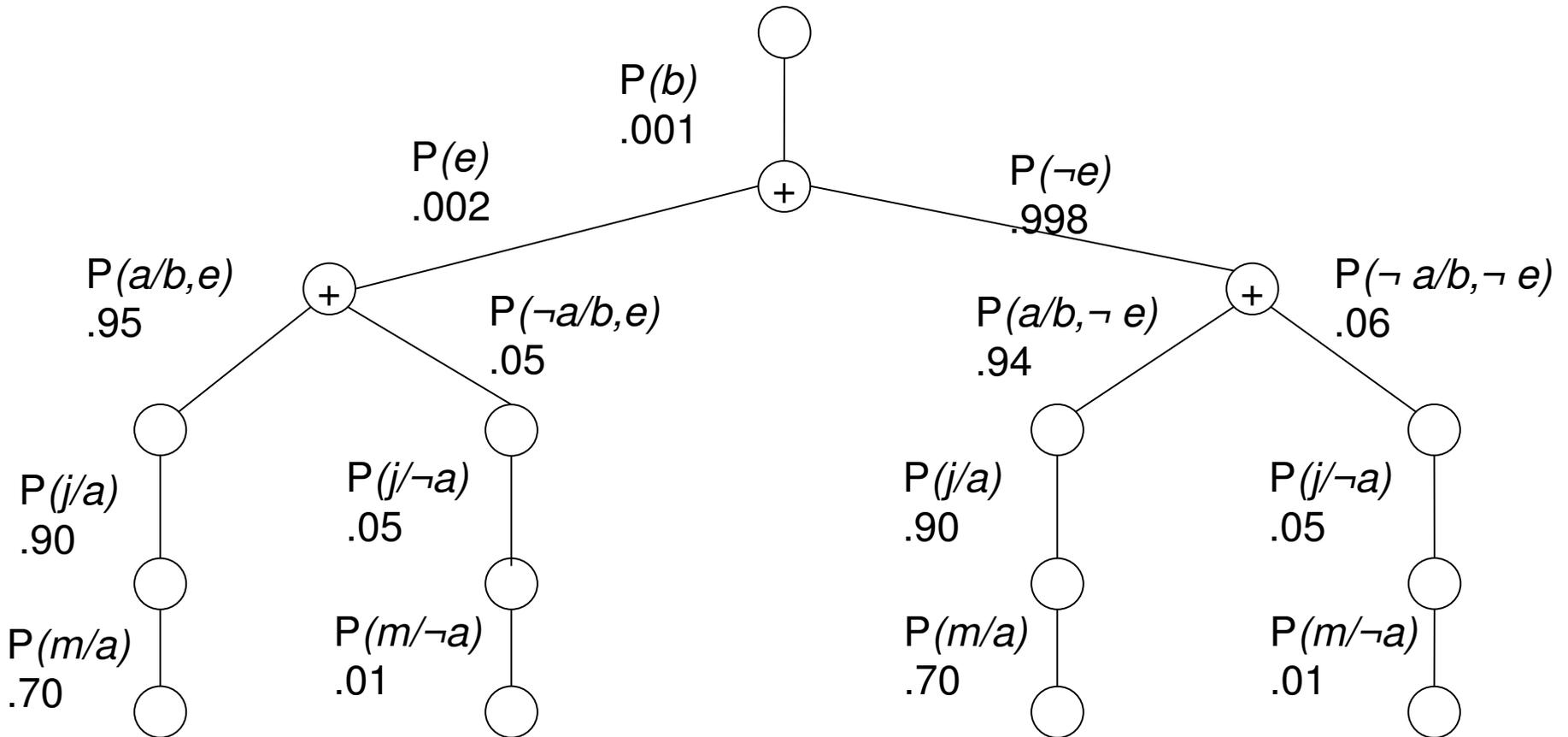




Evaluation Tree of the Enumeration Algorithm

Enumeration is inefficient: repeated computation!

For Example: compute $P(j|a)P(m|a)$ for each value of e



Inference by Variable Elimination



Variable Elimination: carry out summations right-to-left,
storing intermediate results (factors) to avoid re-computation

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

P



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

$$\sum_x f_1 \times \dots \times f_k = f_1 \times \dots \times f_i \sum_x f_{i+1} \times \dots \times f_k = f_1 \times \dots \times f_i \times f_{\bar{X}}$$

assuming f_1, \dots, f_i do not depend on x

Pointwise product of factors f_1 and f_2

$$f_1(x_1, \dots, x_j, y_1, \dots, y_k) \times f_2(y_1, \dots, y_k, z_1, \dots, z_l)$$

$$\dot{=} f(x_1, \dots, x_j, y_1, \dots, y_k, z_1, \dots, z_l)$$

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 $P(X_1, \dots, X_n)$

$factors \leftarrow []$; $var \leftarrow \text{REVERSE}(\text{VARS}[bn])$

for each var in $vars$ do

$factors \leftarrow [\text{MAKE-FACTOR}(var, e) \mid factors]$

if var is a hidden variable the $factors \leftarrow \text{SUM-OUT}(var, factors)$

return $\text{NORMALIZE}(\text{POINTWISE-PRODUCT}(factors))$



Irrelevant variables



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

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

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

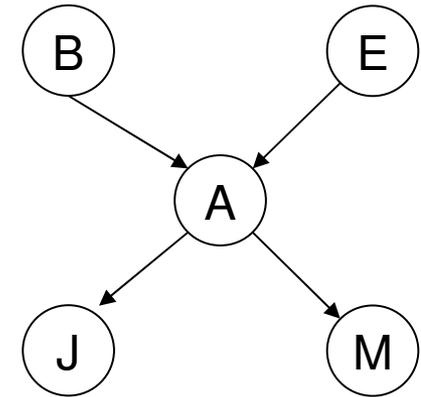
Thm1: Y is irrelevant unless $Y \in \text{Ancestors}(\{X\} \cup E)$

Here, $X = \text{JohnCalls}$, $E = \{\text{Burglary}\}$, and

$$\text{Ancestors}(\{X\} \cup E) = \{\text{Alarm}, \text{Earthquake}\}$$

so M is irrelevant

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



Complexity of exact inference



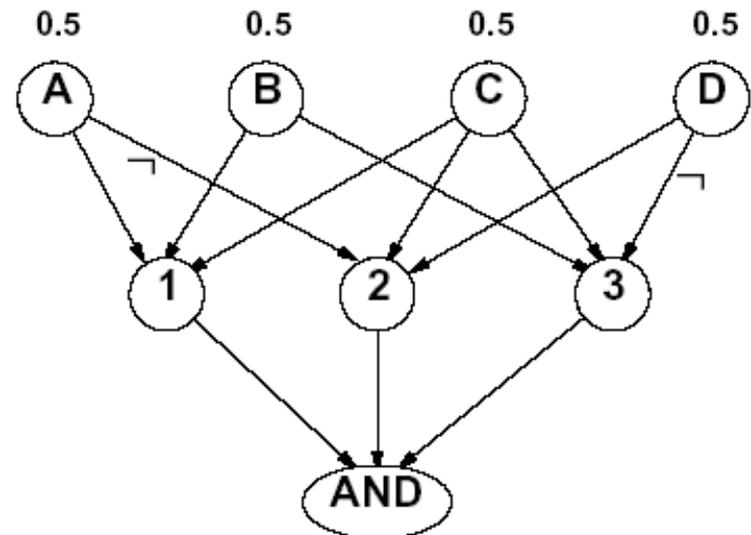
Singly connected networks (or polytrees):

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



Inference by stochastic simulation



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

0.5



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 $P(X_1, \dots, X_n)$

$x \leftarrow$ an event with n elements
for $i = 1$ to n do
 $x_i \leftarrow$ a random sample from $P(X_i \mid \text{Parents}(X_i))$

return x

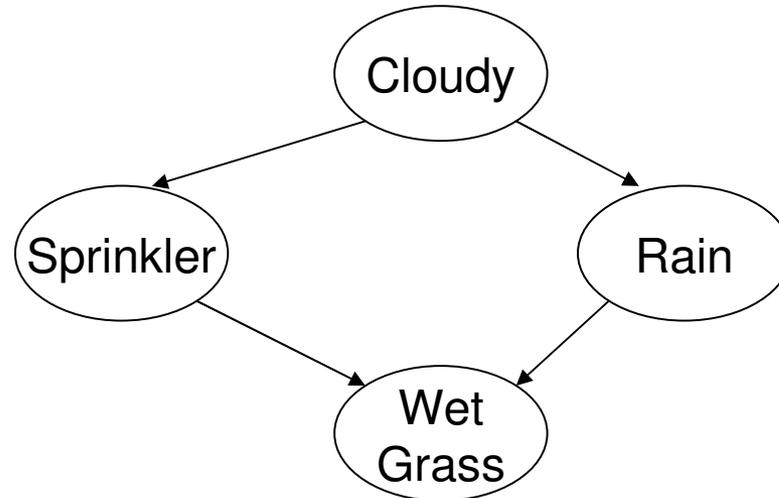


Example



$P(C)$
.50

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



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

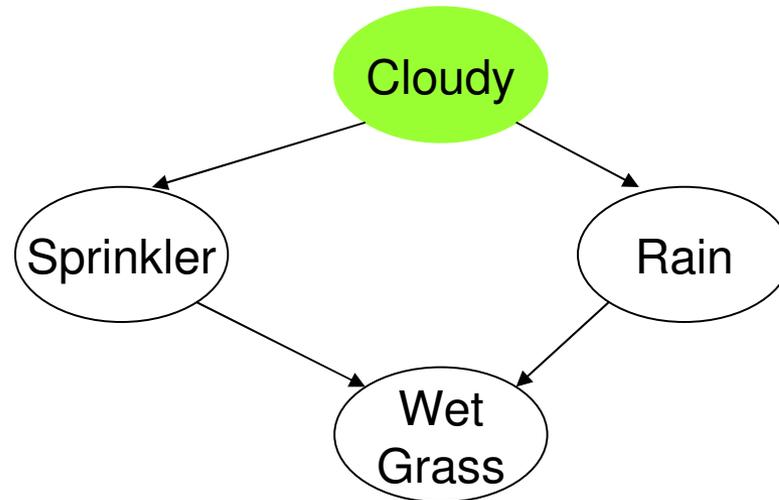
S	R	$P(W S,R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01



Example



$P(C)$
.50



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

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

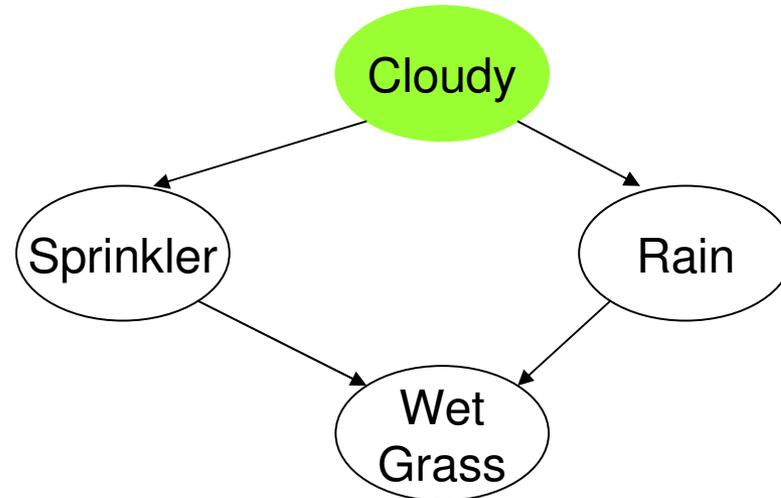
S	R	$P(W S,R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01



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

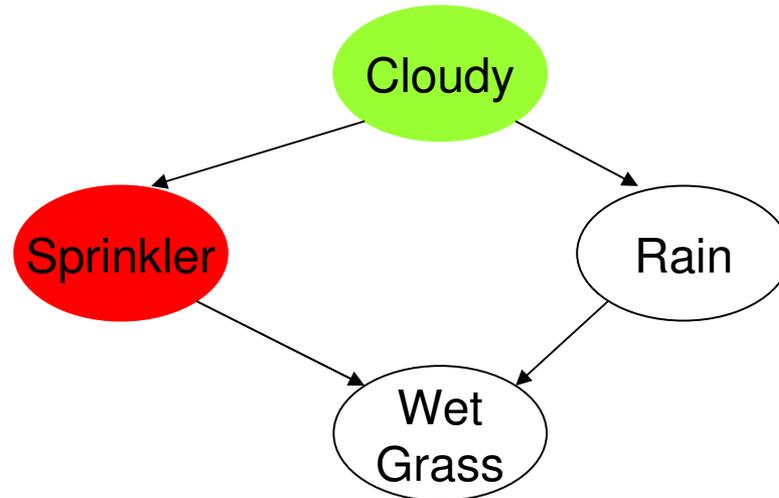


Example



$P(C)$
.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

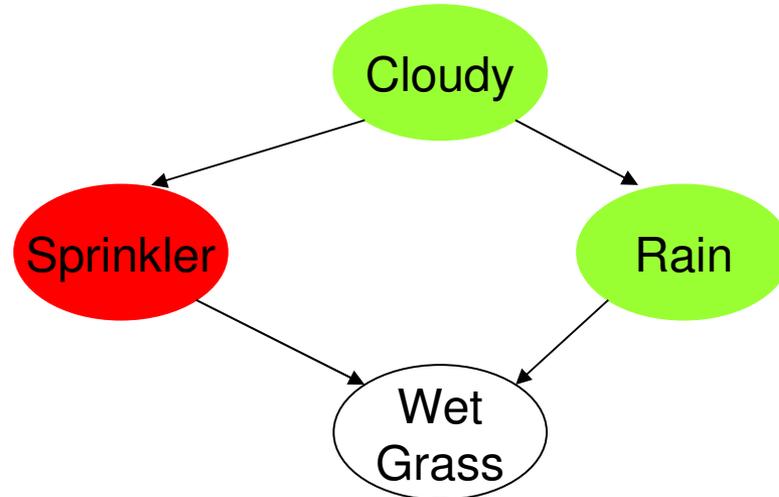


Example



$P(C)$
.50

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



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

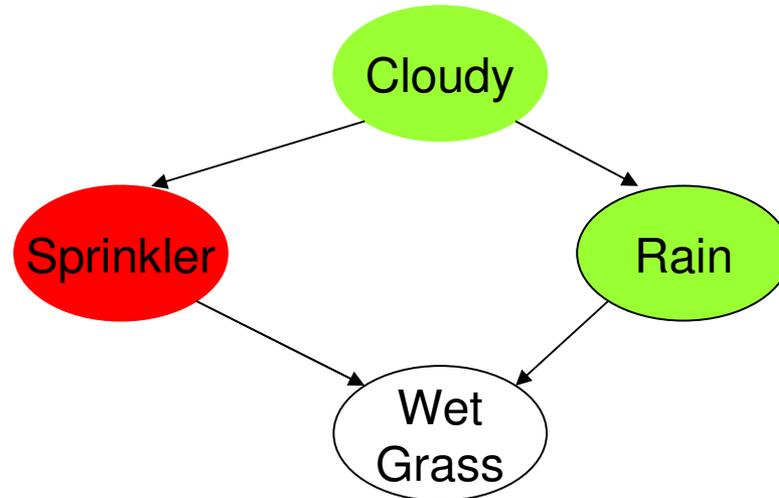
S	R	$P(W S,R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01



Example



$P(C)$
.50



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

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

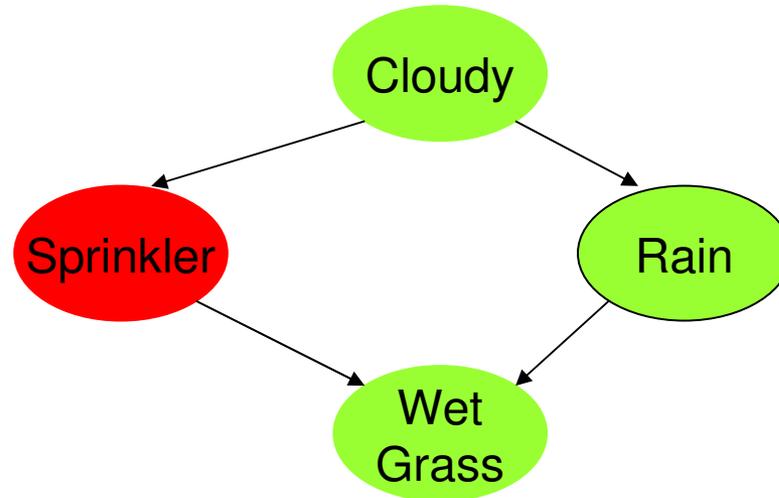
S	R	$P(W S,R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01



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



Probability that PRIORSAMPLE generates a particular event

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

i.e., the true prior probability

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

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

Then we have

$$\lim_{N \rightarrow \infty} \hat{P}(x_1, \dots, x_n) = \lim_{N \rightarrow \infty} N_{PS}(x_1, \dots, x_n) / N$$

$$\dot{=} S_{PS}(x_1, \dots, x_n)$$

$$\dot{=} P(x_1 \dots x_n)$$

That is, estimates derived from PRIORSAMPLE are consistent

Shorthand: $P(x_1, \dots, x_n) \approx P(x_1 \dots x_n)$



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

$x \leftarrow$ PRIOR-SAMPLE(bn)

 if x is consistent with e then

$N[x] \leftarrow N[x]+1$ where x is the value of X in x

return NORMALIZE($N[X]$)

E.g., estimate $P(\text{Rain Sprinkler} = \text{true})$ using 100 samples

27 samples have *Sprinkler = true*

Of these, 8 have *Rain = true* and 19 have *Rain = false*.

$\hat{P}(\text{Rain Sprinkler} = \text{true}) = \text{NORMALIZE}(\langle 8, 9 \rangle) = \langle 0.296, 0.704 \rangle$

Similar to a basic real-world empirical estimation procedure



Analysis of rejection sampling



$$\begin{aligned}\hat{P}(X|e) &= \alpha N_{PS}(X, e) \quad (\text{algorithm defn.}) \\ &= N_{PS}(X, e) / N_{PS}(e) \quad (\text{normalized by } N_{PS}(e)) \\ &\stackrel{!}{=} P(X, e) / P(e) \quad (\text{property of PRIORSAMPLE}) \\ &= P(X|e) \quad (\text{defn. of conditional probability})\end{aligned}$$

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

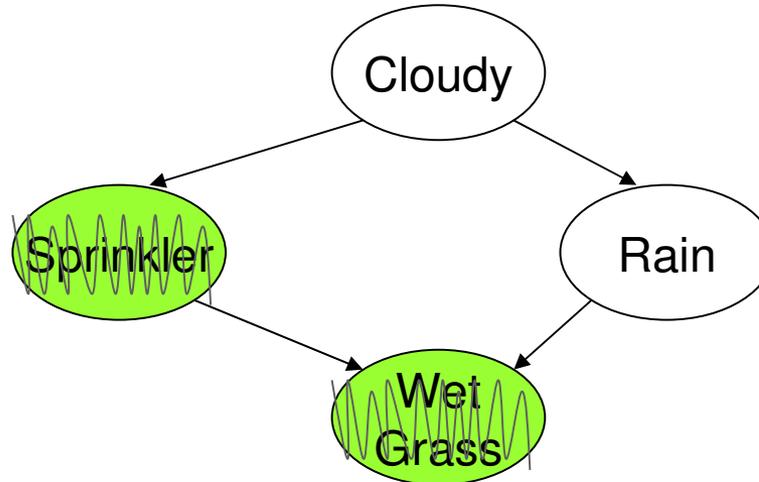
```
function WEIGHTED-SAMPLE( $bn, e$ ) returns an event and a weight
   $x \leftarrow$  an event with  $n$  elements;  $w \leftarrow 1$ 
  for  $i = 1$  to  $n$  do
    if  $X_i$  has a value  $x_i$  in  $e$ 
      then  $w \leftarrow w \times P(X_i = x_i \mid \text{Parent}(X_i))$ 
    else  $x_i \leftarrow$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



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

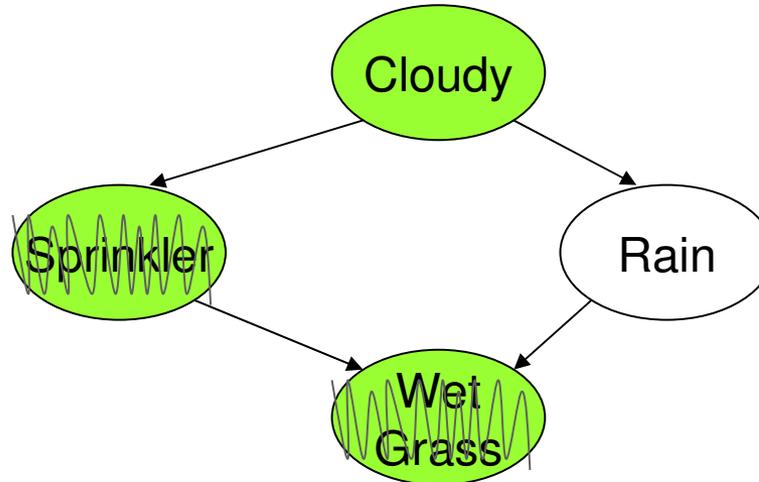
$$\omega = 1.0$$



Example



P(C)
.50



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

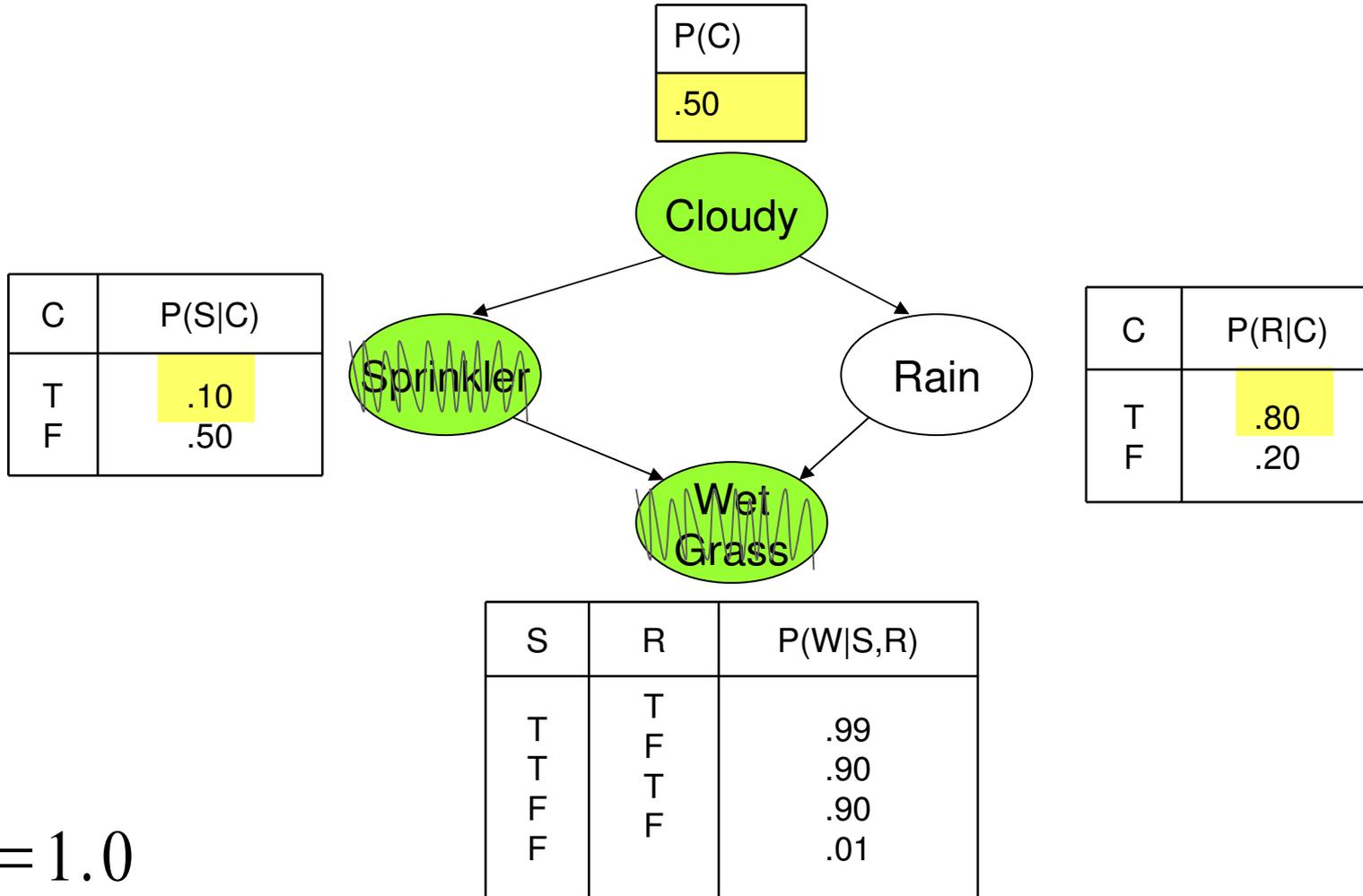
C	P(R C)
T	.80
F	.20

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

$$\omega = 1.0$$



Example



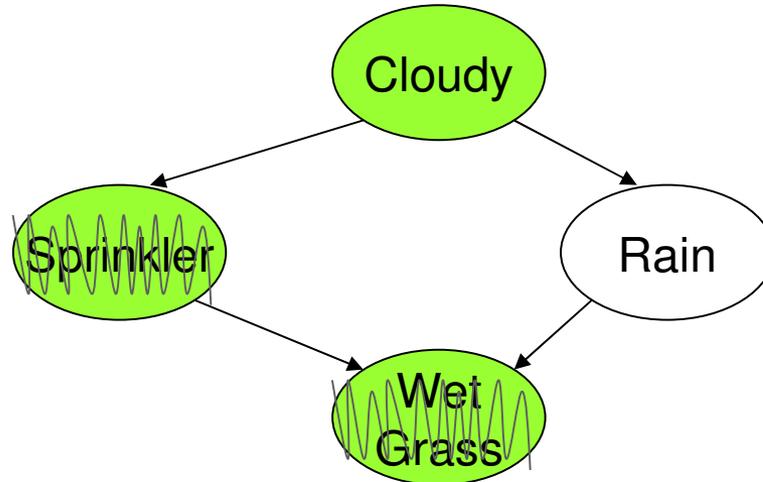
$$\omega = 1.0$$



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

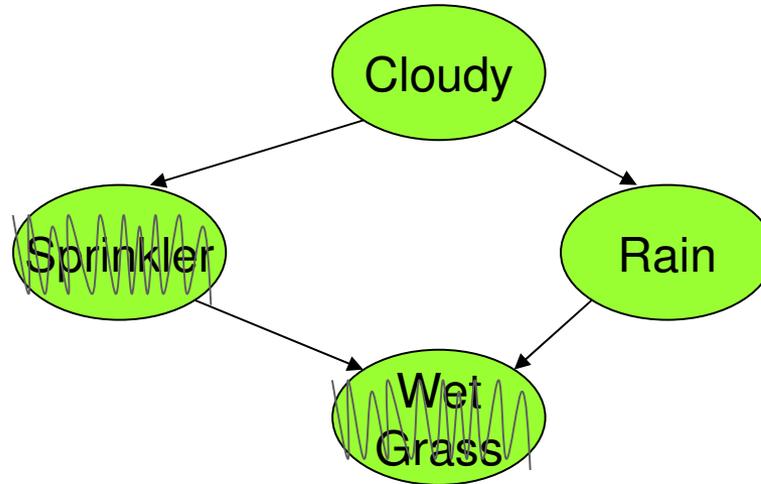
$$\omega = 1.0 \times 0.1$$



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

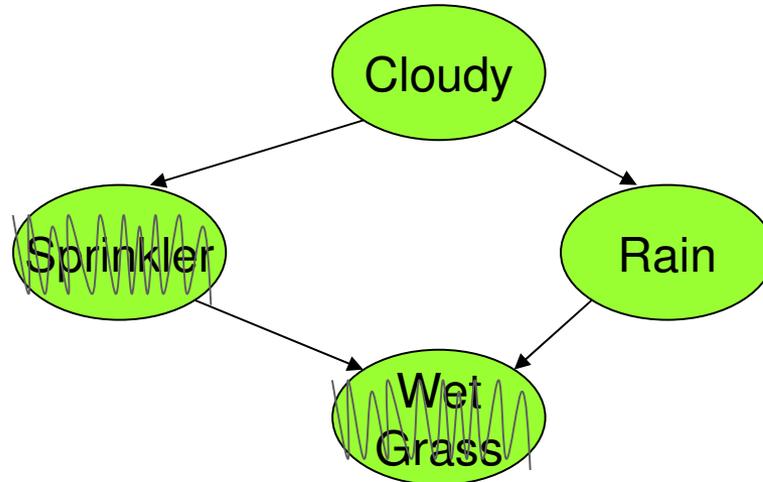
$$\omega = 1.0 \times 0.1$$



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

$$\omega = 1.0 \times 0.1$$

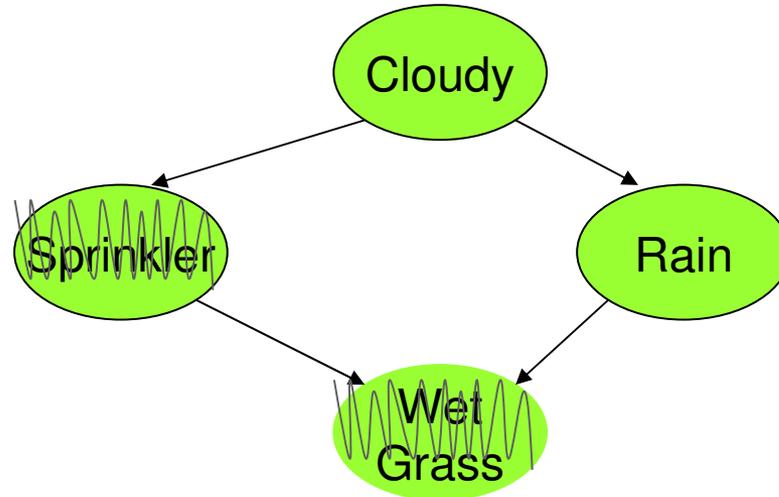


Example



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

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



Likelihood weighting analysis



Sampling probability for WEIGHTEDSAMPLE is

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

Note: pays attention to evidence in ancestors only

⇒ somewhere „in between“ prior and posterior distribution

Weight for a given sample z, e is

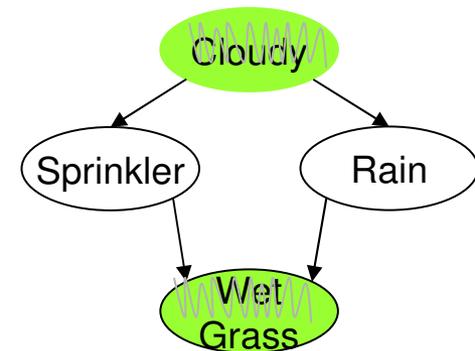
$$w(z, e) = \prod_{i=1}^m P(e_i \text{ Parents}(E_i))$$

Weighted sampling probability is

$$S_{WS}(z, e) w(z, e) = \prod_{i=1}^l P(z_i \text{ Parents}(Z_i)) \prod_{i=1}^m P(e_i \text{ Parents}(E_i))$$

∝ $P(z, e)$ (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 $P(X | e)$

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 Z

for $j = 1$ to N do

$N[x] \leftarrow N[x] + 1$ where x is the value of X in x

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

return NORMALIZE($N[X]$)

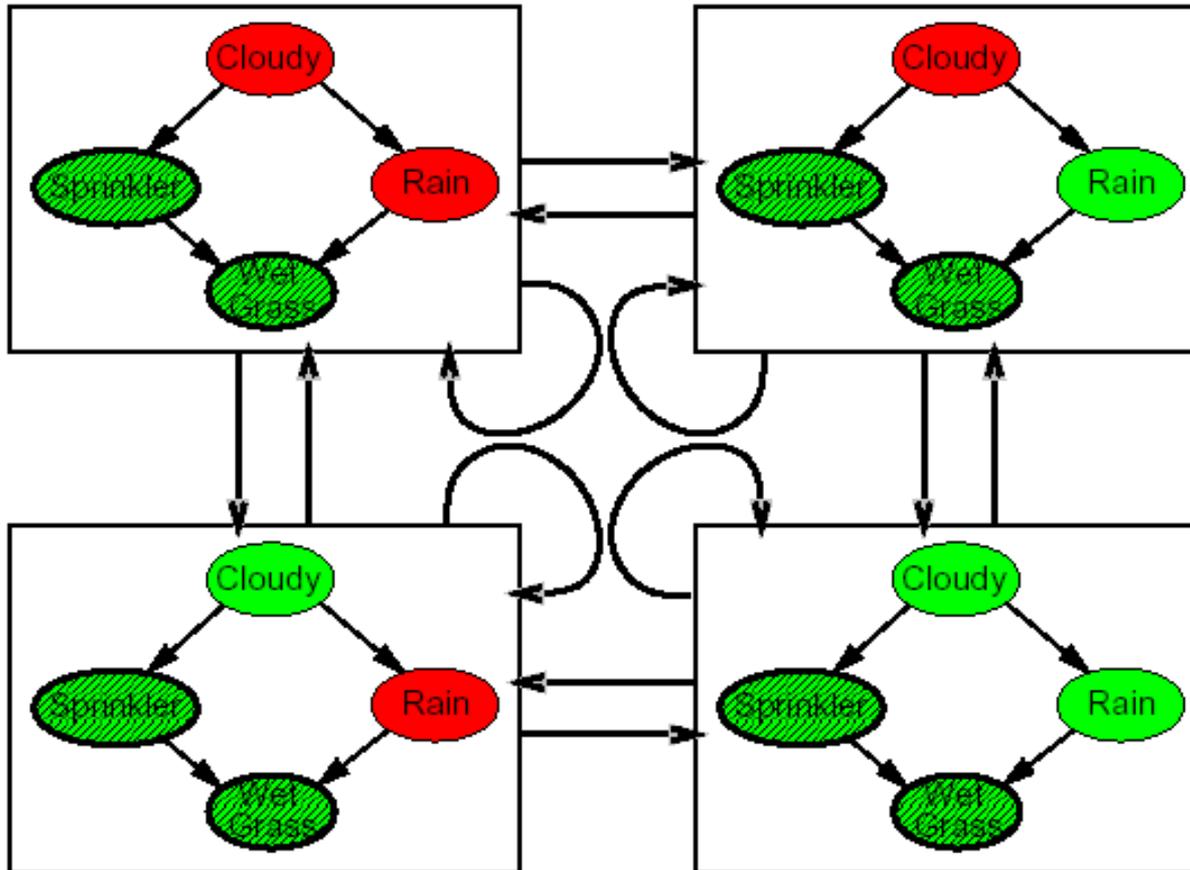
Can also choose a variable to sample at random each time



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 $P(\text{Rain Sprinkler}=\text{true}, \text{WetGrass}=\text{true})$

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

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

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

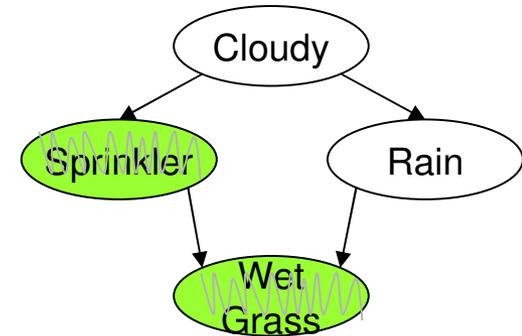


Markov blanket sampling



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:

$$P(x_i \mid MB(X_i)) = P(x_i \mid Parents(X_i)) \prod_{Z_j \in Children(X_i)} P(z_j \mid Parents(Z_j))$$

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:

$P(X_i \mid MB(X_i))$ won't change much (law of large numbers)



Summary



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

