

Last update: November 25, 2008

# INFERENCE IN BAYESIAN NETWORKS

CMSC 421: CHAPTER 14.4–5

# 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  $\mathbf{P}(X_i|\mathbf{E} = \mathbf{e})$

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?

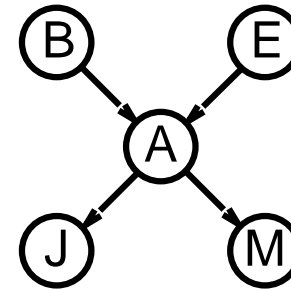
**Explanation:** why do I need a new starter motor?

# Inference by enumeration

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

Simple query on the burglary network:

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



Note: here,  $a$  no longer means “ $A = true$ ”

It's a dummy variable whose range is  $\{A = true, A = false\}$

Rewrite full joint entries using product of CPT entries:

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

Recursive depth-first enumeration

# 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  $\mathbf{e}$

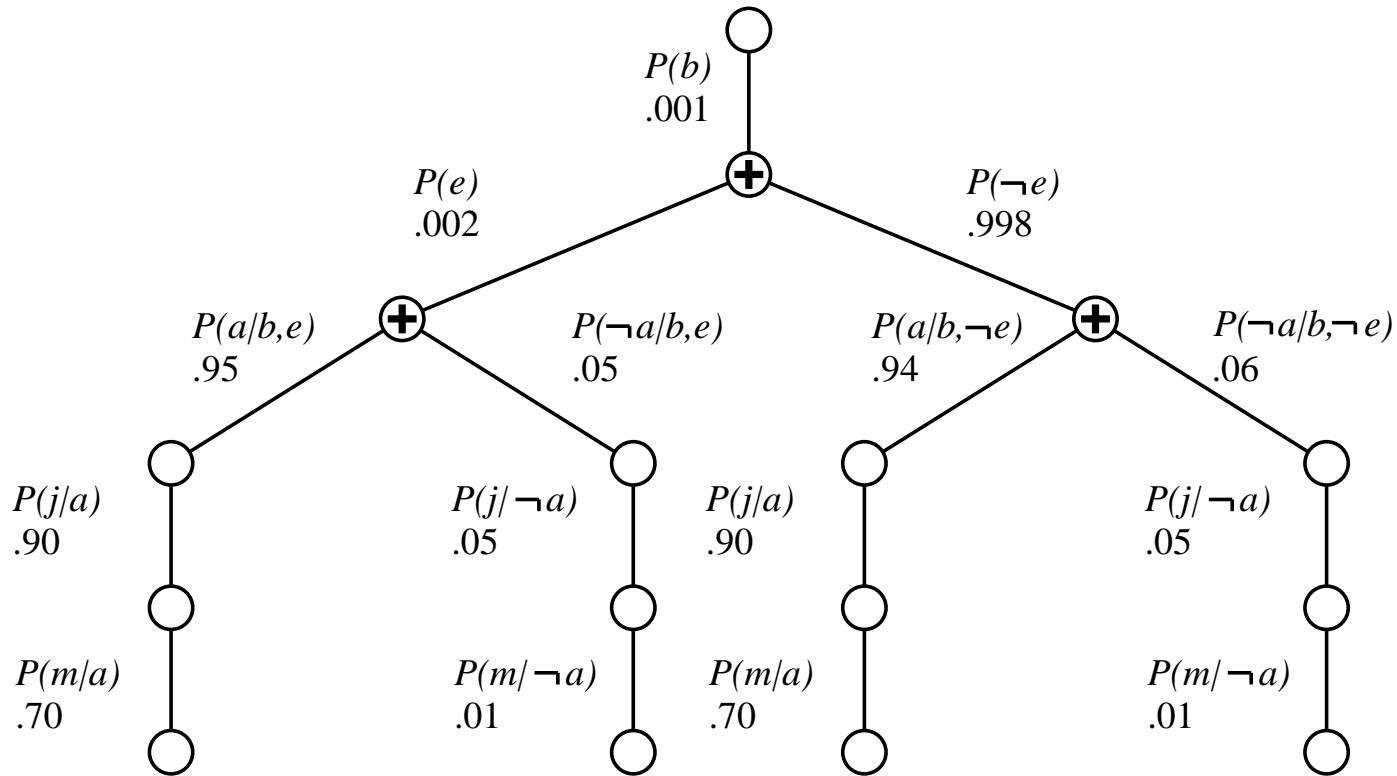
**then return**  $P(y \mid Pa(Y)) \times$  **ENUMERATE-ALL**(**REST**( $vars$ ),  $\mathbf{e}$ )

**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

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



Enumeration is inefficient: repeated computation

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

$O(n)$  space,  $O(d^n)$  time, where  $n$  = variables,  $d$  = values per variable

# Inference by variable elimination

Variable elimination: related to dynamic programming

Carry out summations right-to-left (i.e., bottom-up in the tree),  
storing intermediate results (**factors**) to avoid recomputation

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

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

where  $\mathbf{f}_J(A) = \begin{pmatrix} P(j|a) \\ P(j|\neg a) \end{pmatrix}$ ,  $\mathbf{f}_M(A) = \begin{pmatrix} P(m|a) \\ P(m|\neg a) \end{pmatrix}$ , etc.,

and  $\times$  is the **pointwise** product.

## Pointwise product and summing out variables

Pointwise product: multiply the array elements for the same variable values

$$\mathbf{f}_J(A) \times \mathbf{f}_M(A) = \begin{pmatrix} P(j|a) \\ P(j|\neg a) \end{pmatrix} \times \begin{pmatrix} P(m|a) \\ P(m|\neg a) \end{pmatrix} = \begin{pmatrix} P(j|a)P(m|a) \\ P(j|\neg a)P(m|\neg a) \end{pmatrix}$$

Pointwise product of factors  $f_1$  and  $f_2$ :

$$\begin{aligned} f_1(x_1, \dots, x_j, y_1, \dots, y_k) \times f_2(y_1, \dots, y_k, z_1, \dots, z_l) \\ = f(x_1, \dots, x_j, y_1, \dots, y_k, z_1, \dots, z_l) \end{aligned}$$

E.g.,

$$\begin{aligned} f_A(a, b, e) \times f_J(a) &= f_{AJ}(a, b, e) \\ f_A(a, b, e) \times f_J(a) \times f_M(a) &= f_{AJM}(a, b, e) \end{aligned}$$

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

If  $f_1, \dots, f_i$  do not depend on  $X$ , then

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

# Variable elimination algorithm

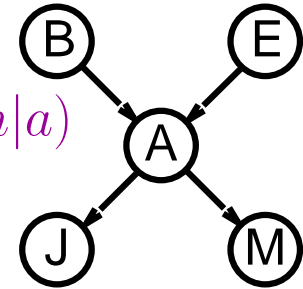
```
function ELIMINATION-ASK( $X, e, bn$ ) returns a distribution over  $X$   
inputs:  $X$ , the query variable  
           $e$ , evidence specified as an event  
           $bn$ , a belief network specifying joint distribution  $\mathbf{P}(X_1, \dots, X_n)$   
 $factors \leftarrow []$ ;  $vars \leftarrow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



Thm 1: For query  $X$  and evidence  $\mathbf{E}$ ,

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

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

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

## Irrelevant variables, continued

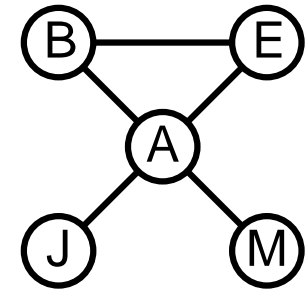
Defn: moral graph of Bayes net:

marry (draw edges between) 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  $P(\text{JohnCalls} | \text{Alarm} = \text{true})$ , both *Burglary* and *Earthquake* are irrelevant



# Complexity of exact inference

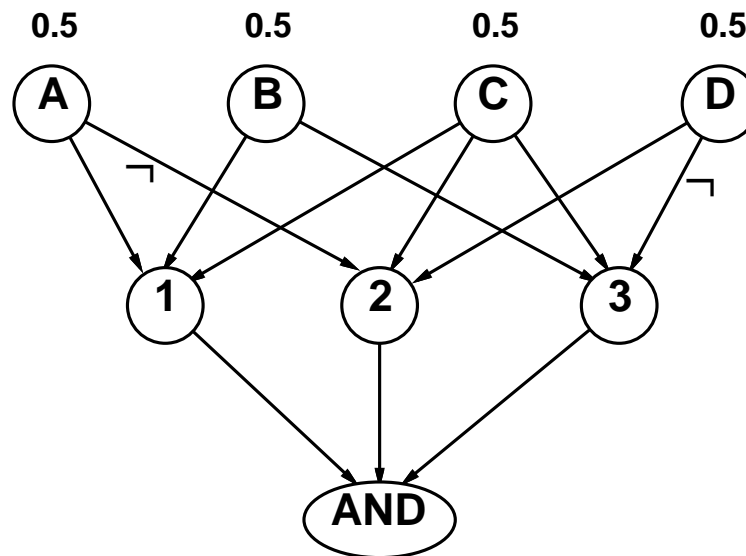
Singly connected networks (or **polytrees**):

- 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  $\Rightarrow$  NP-hard

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



# Inference by stochastic simulation

Outline:

Sampling from an empty network:

- 1) Generate  $N$  random samples of events in the network
- 2) Average the results
- 3) For each event  $x$ , this gives us a posterior probability  $\hat{P}(x)$
- 4) As  $N \rightarrow \infty$  this converges to  $x$ 's true probability  $P(x)$

Rejection sampling, given evidence  $e$ :

- 1) Generate  $N$  random samples of events in the network
- 2) Reject samples that disagree with the evidence  $e$ , average the others
- 3) For each event  $x$ , this gives us a posterior probability  $\hat{P}(x|e)$
- 4) As  $N \rightarrow \infty$  this converges to  $x$ 's true conditional probability  $P(x|e)$

Likelihood weighting: use evidence to weight samples

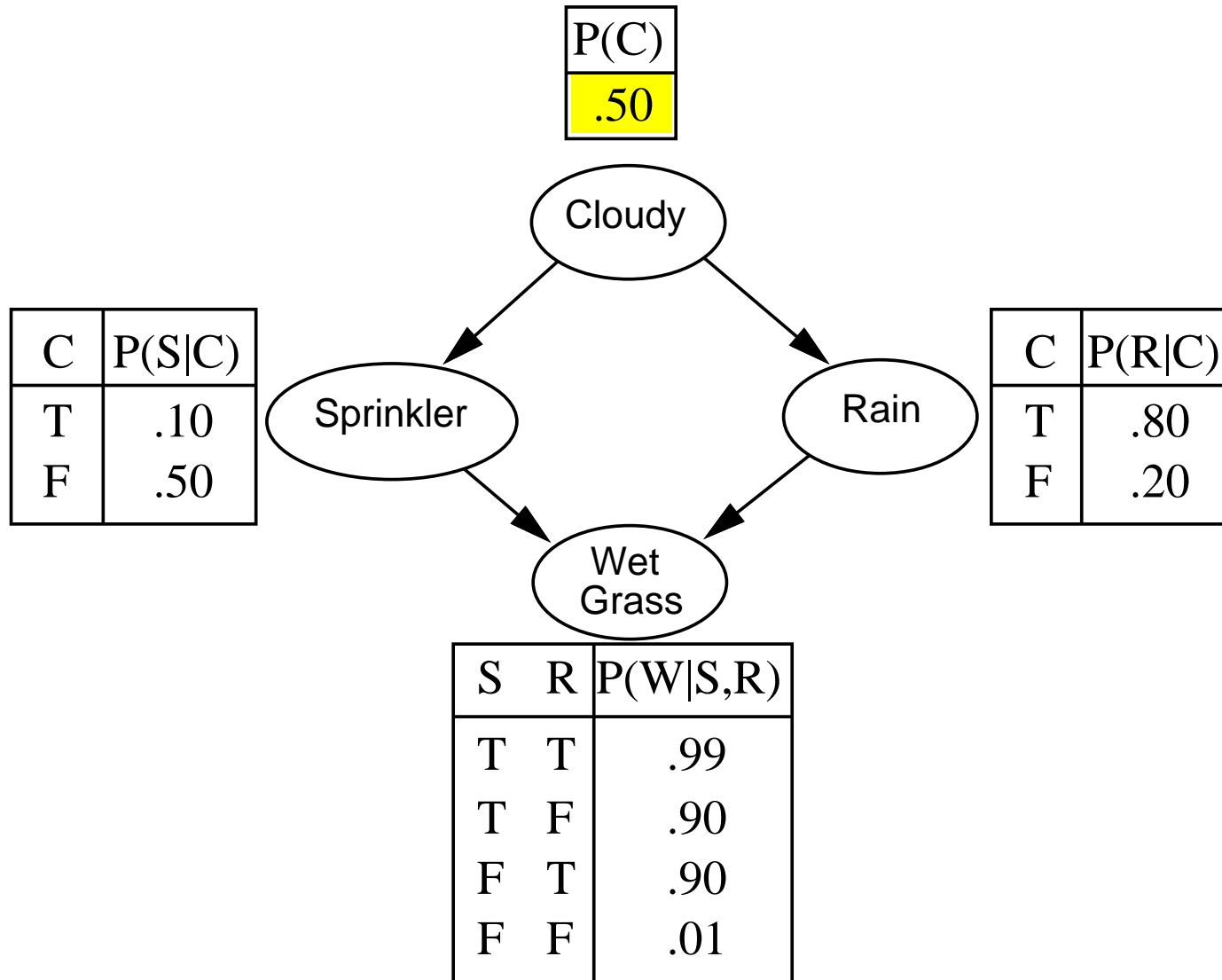
# Sampling from an empty network

Sampling from an empty network:

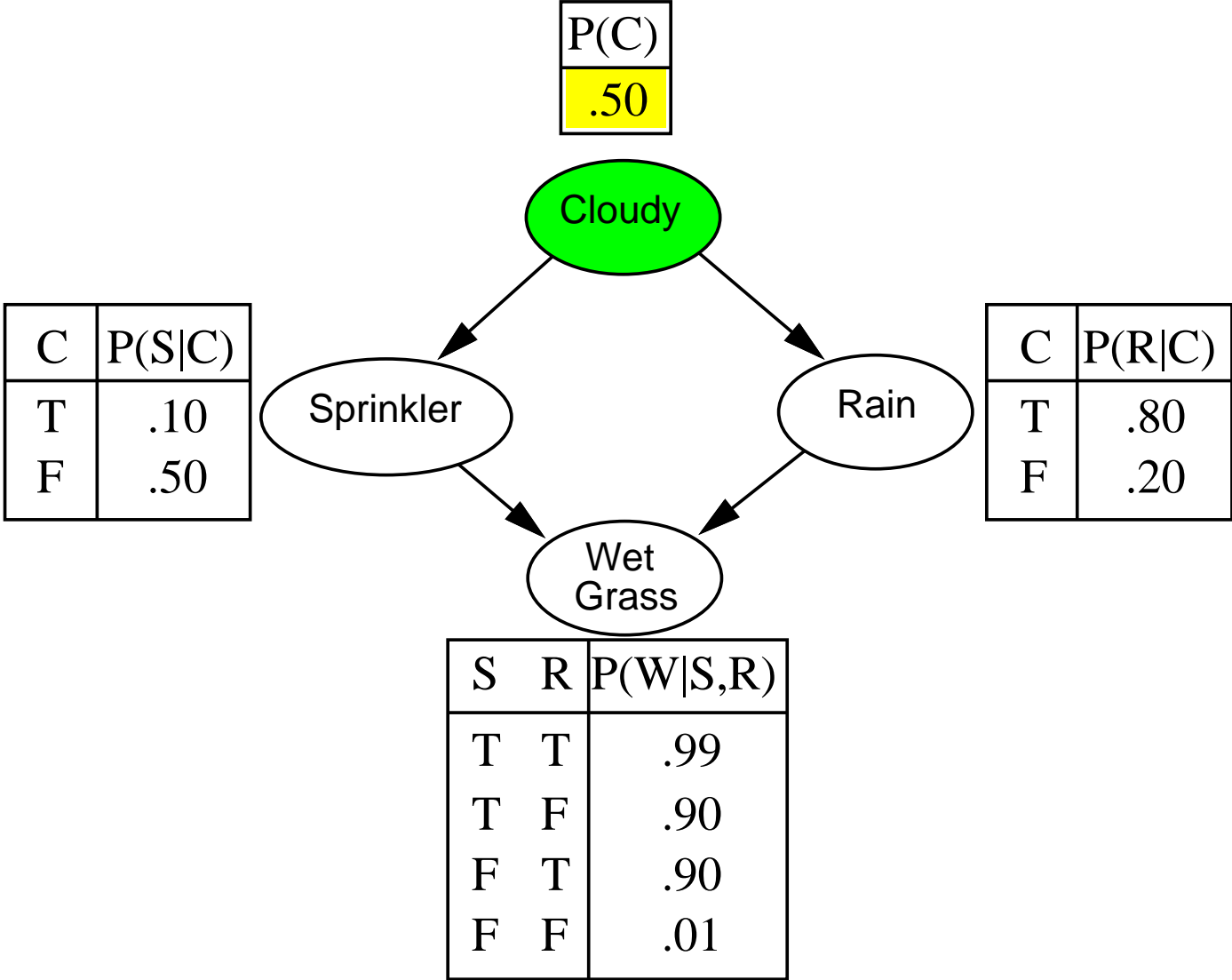
- 1) Generate  $N$  random samples of events in the network
- 2) Average the results
- 3) For each event  $x$ , this gives us a posterior probability  $\hat{P}(x)$
- 4) As  $N \rightarrow \infty$  this converges to  $x$ 's true probability  $P(x)$

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

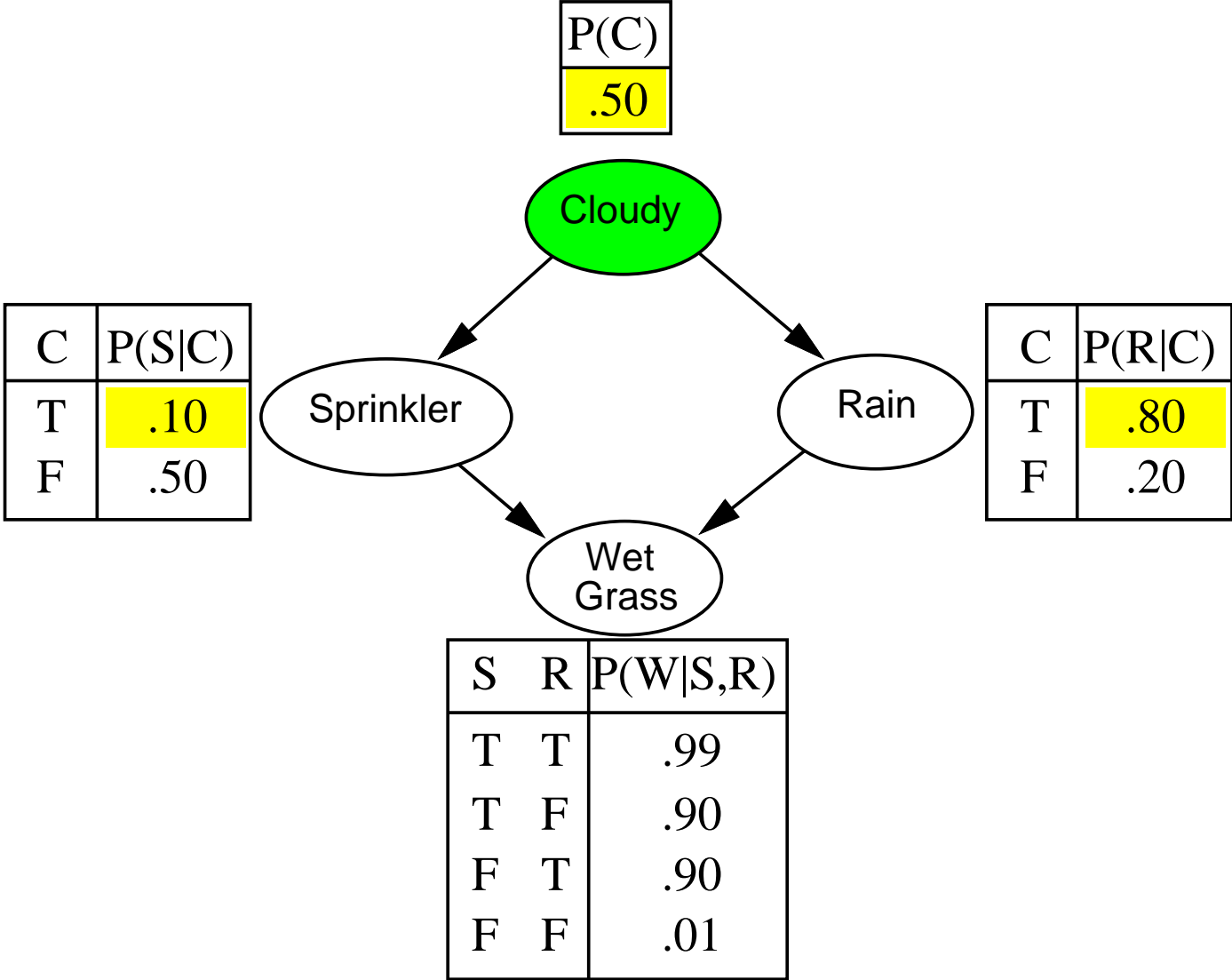
# Example



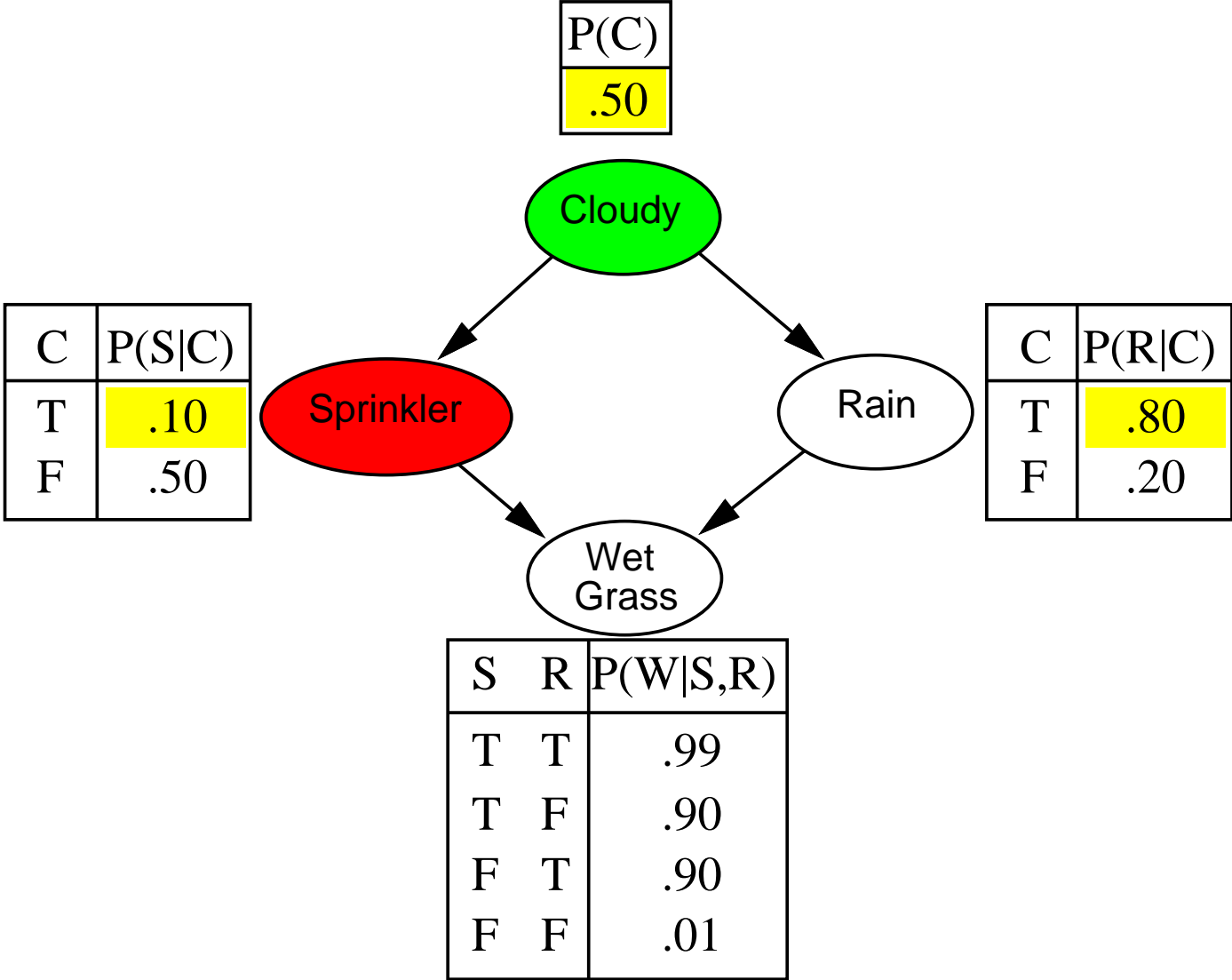
# Example



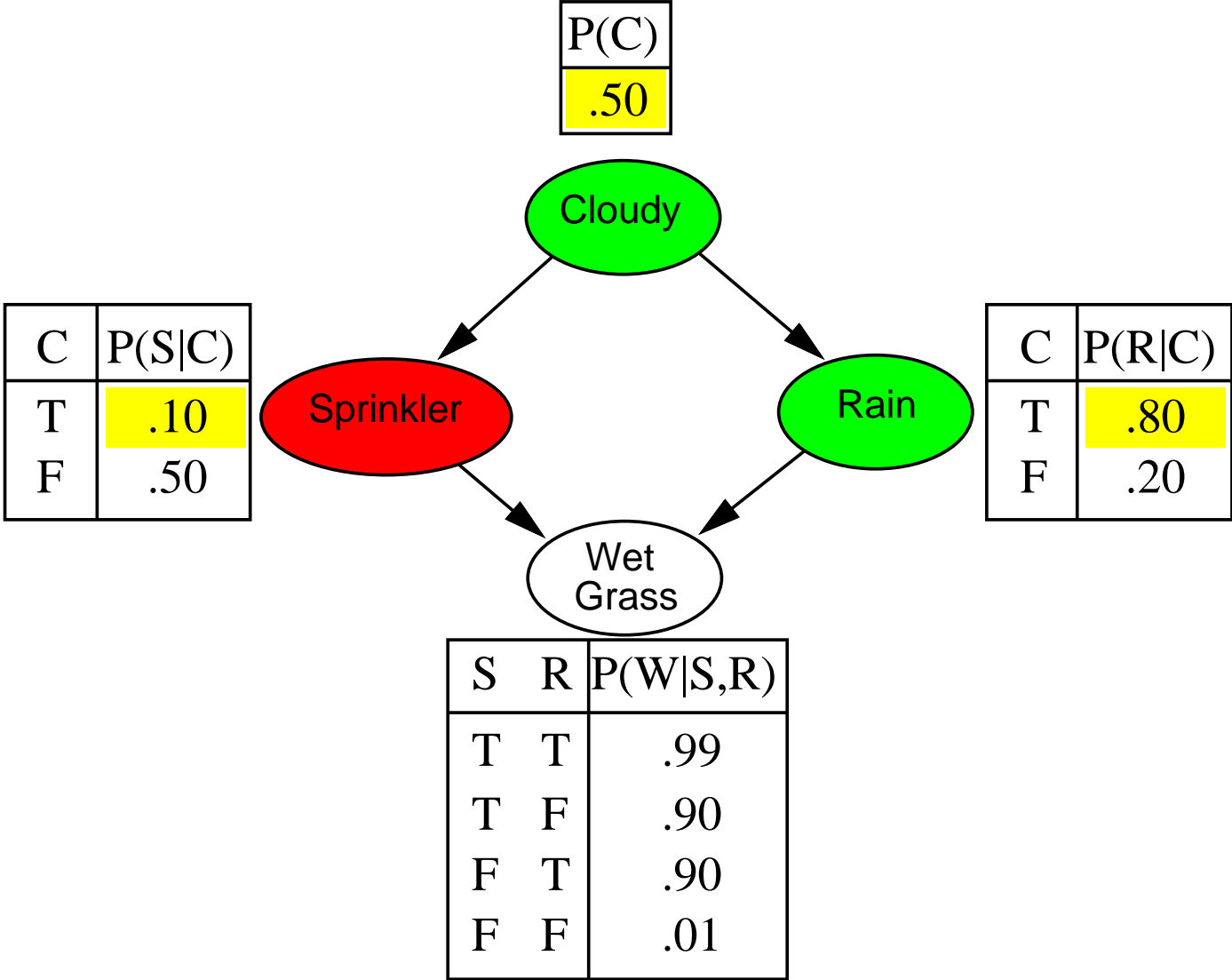
# Example



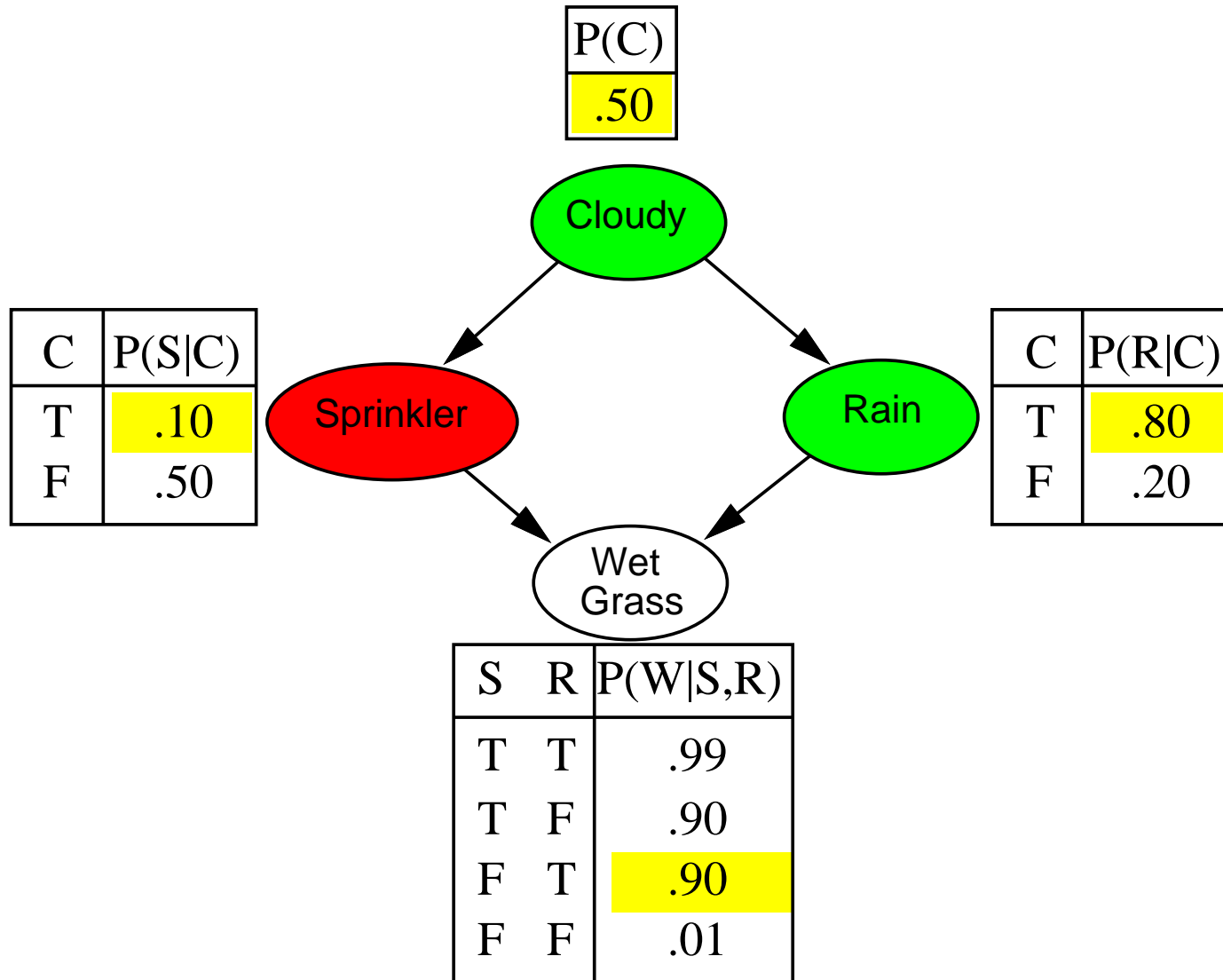
# Example



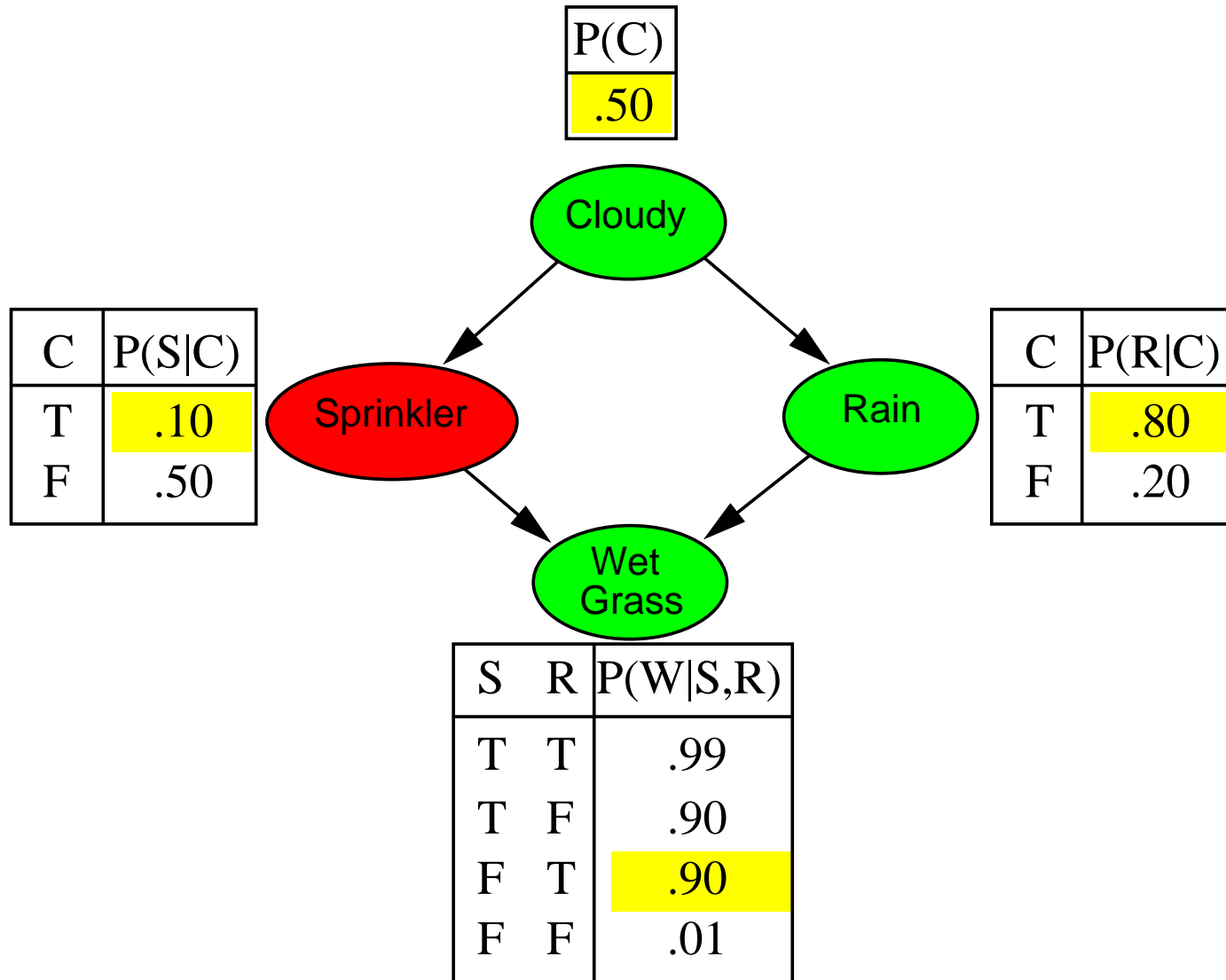
# Example



# Example



# Example



## Sampling from an empty network, continued

Probability that PRIORSAMPLE generates a particular set of events

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

i.e., the true prior probability of  $x_1, \dots, x_n$

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

Suppose we collect  $N$  samples. Let  $N_{PS}(x_1 \dots x_n)$  be the number of samples in which  $x_1, \dots, x_n$  occurred

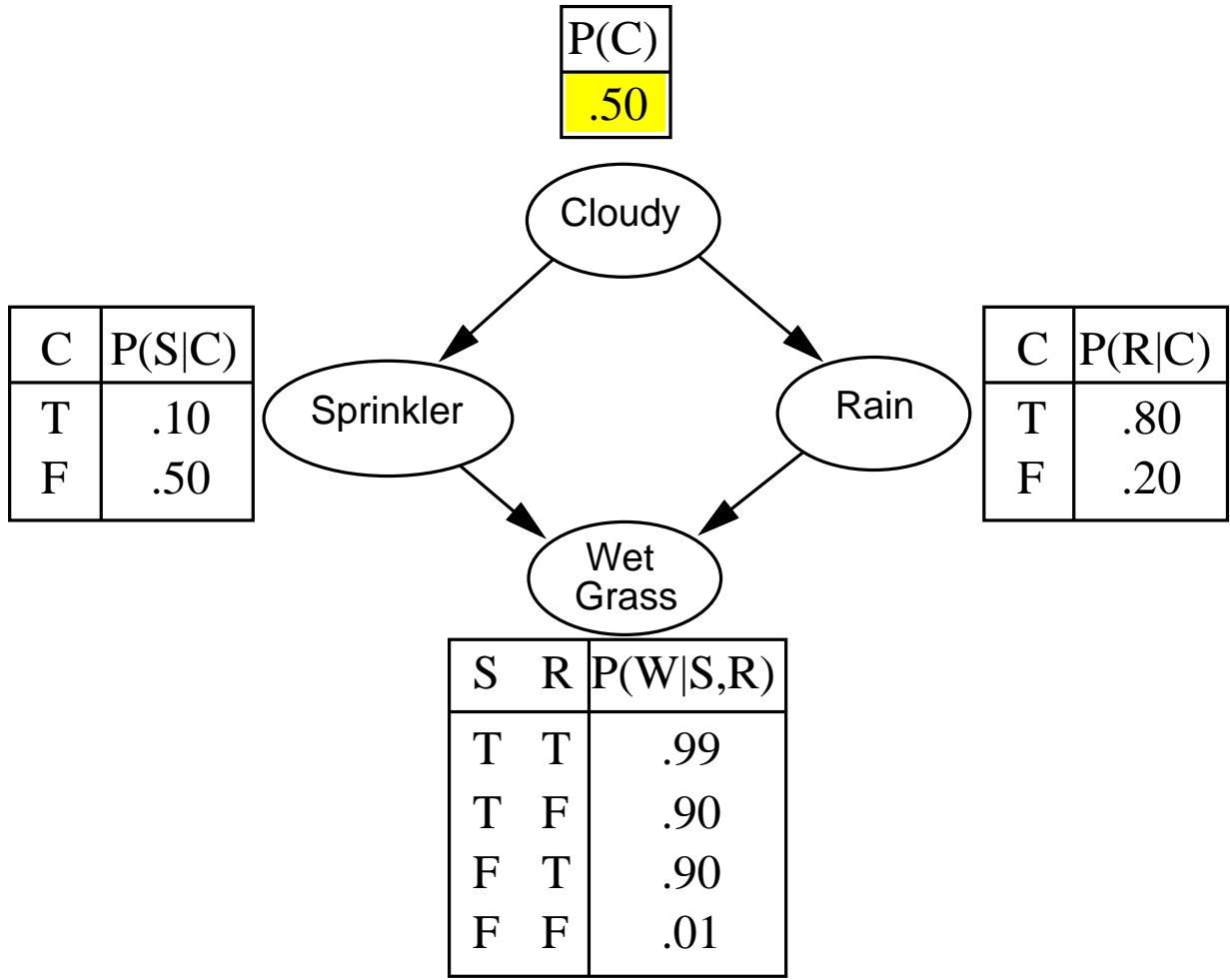
Then we have

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

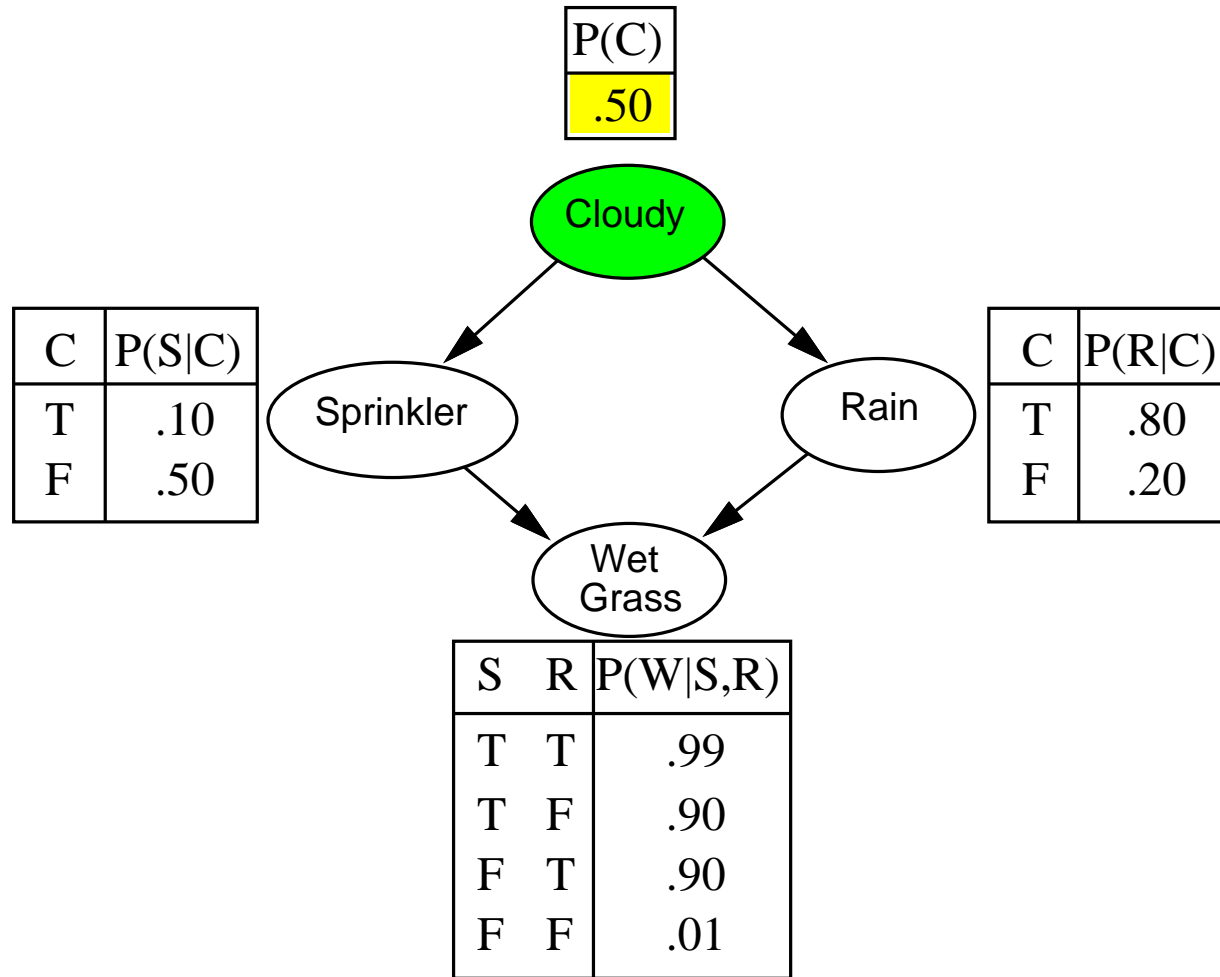
That is, estimates derived from PRIORSAMPLE are **consistent**

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

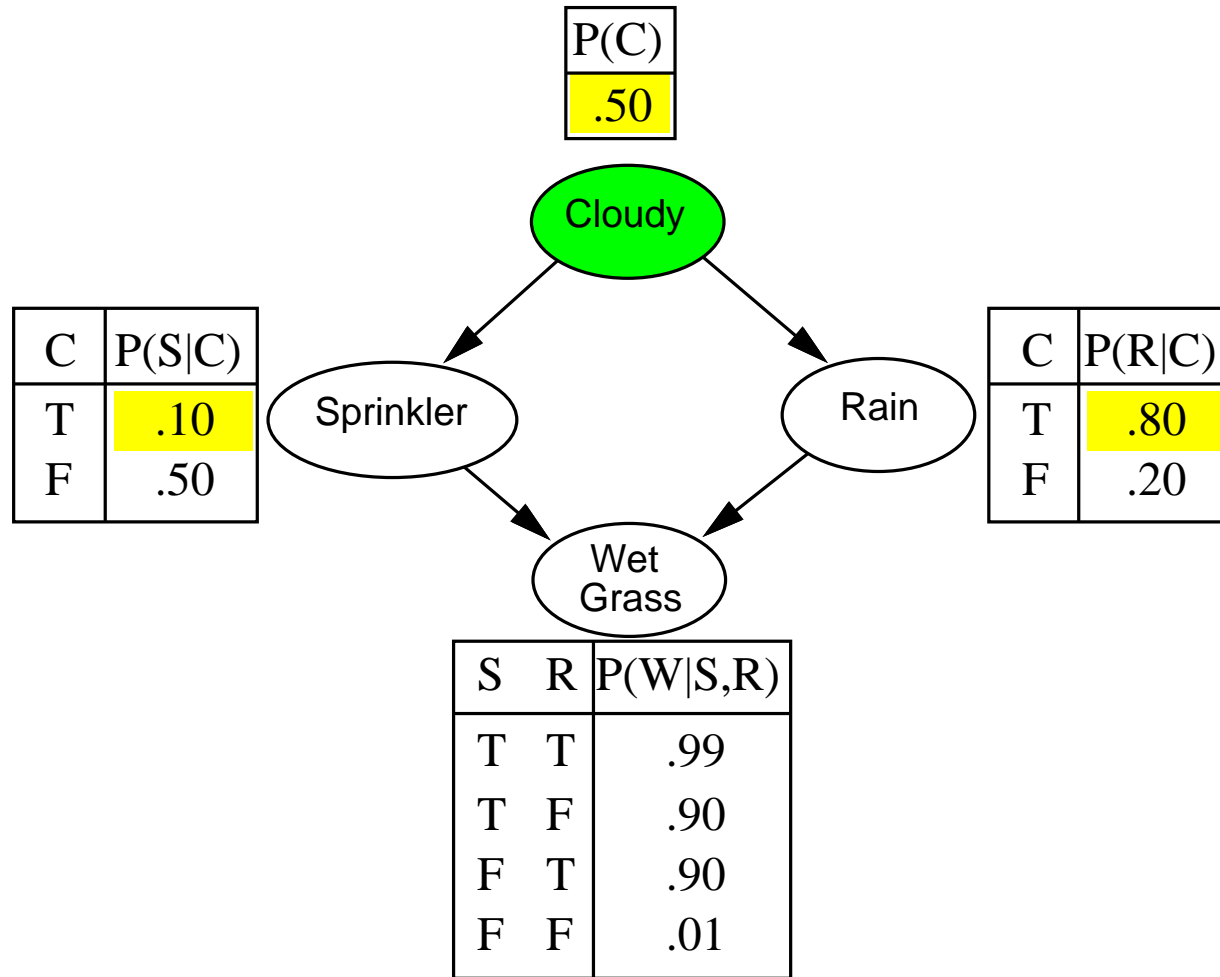
# Example



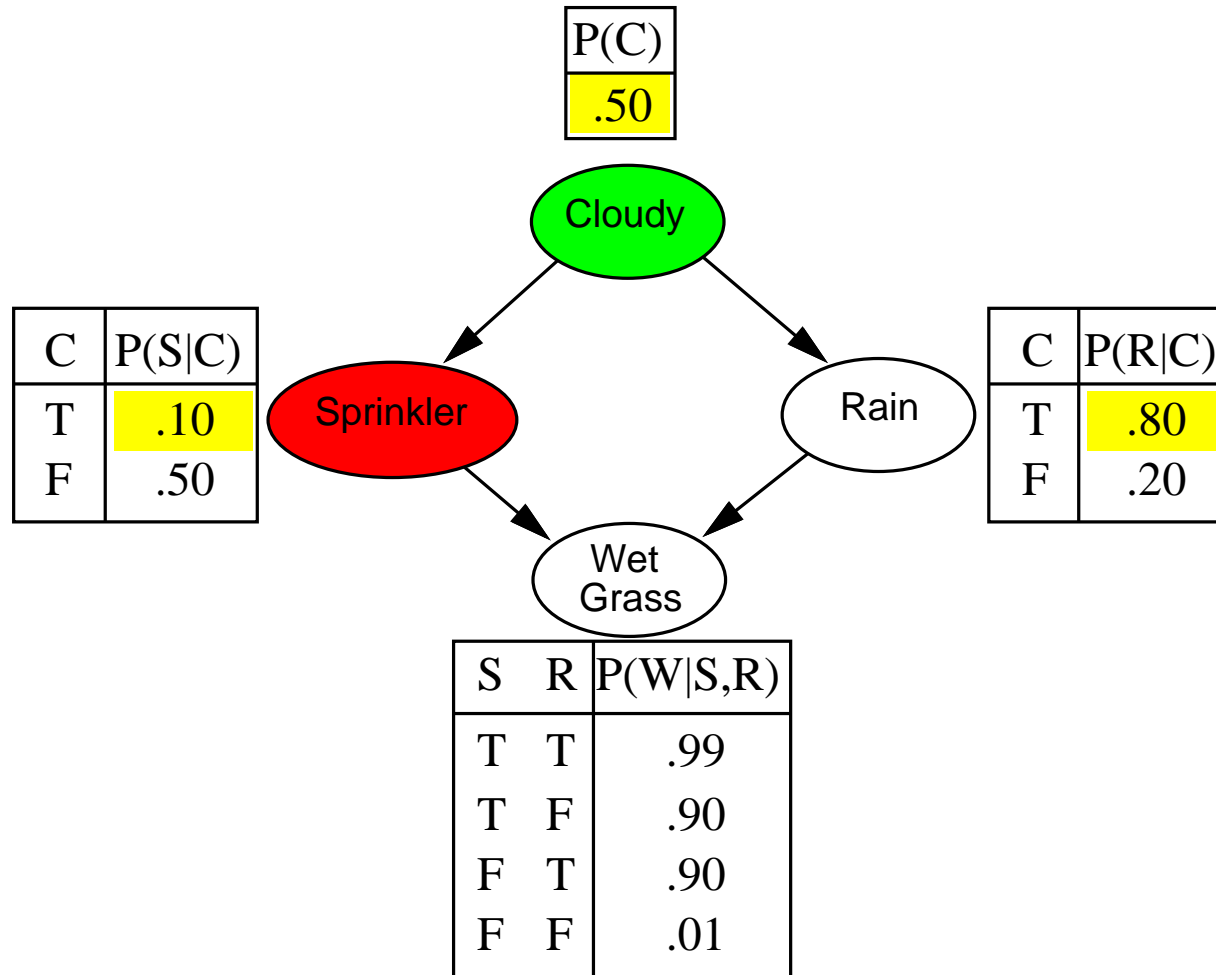
# Example



# Example



# Example



Reject this sample, start running the next one

# 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
       $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$  where  $x$  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

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

Hence rejection sampling returns consistent posterior estimates

Problem: if  $P(\mathbf{e})$  is small, this is hopelessly expensive to compute:  
must reject most of the samples because they disagree with  $\mathbf{e}$

$P(\mathbf{e})$  drops off exponentially with number of evidence variables!

# Likelihood weighting

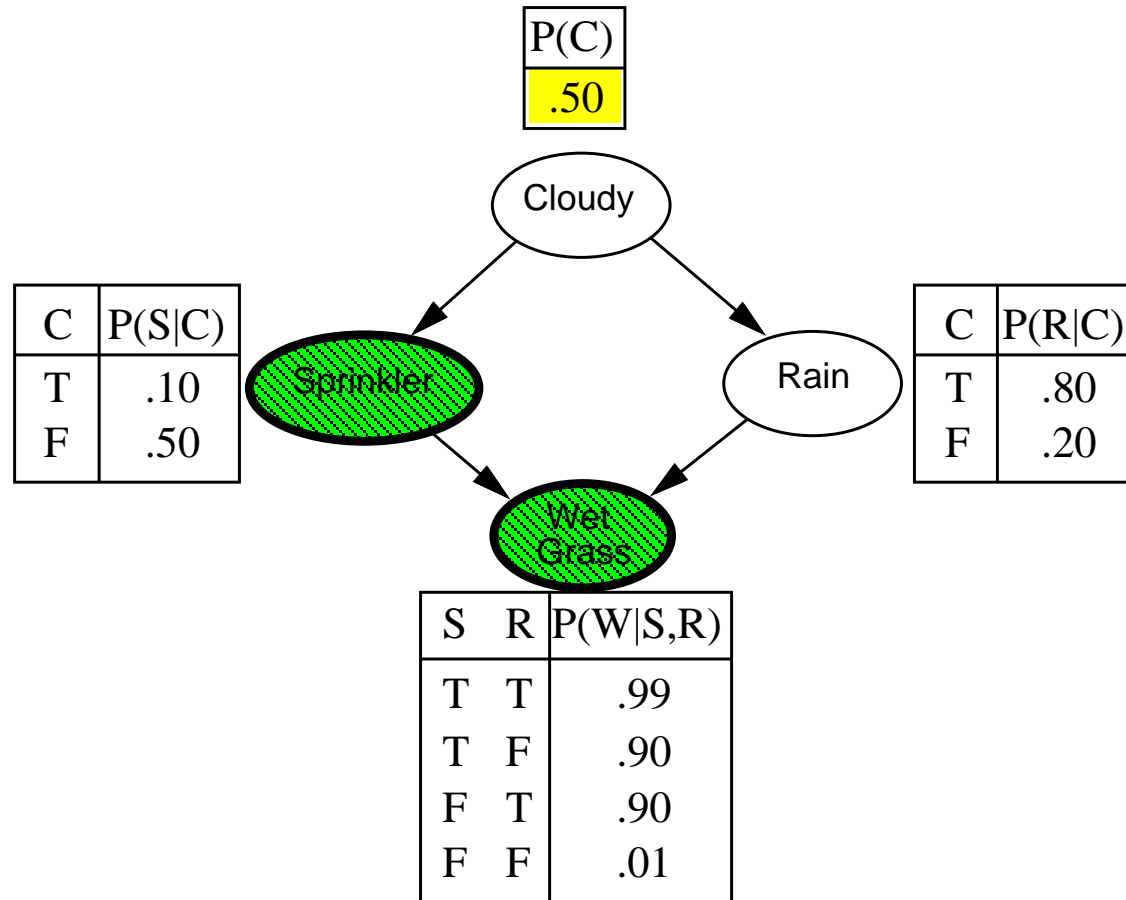
Idea: fix evidence variables, sample only nonevidence variables, and weight each sample by the likelihood it accords the evidence

```
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( $\mathbf{W}[X]$ )
```

---

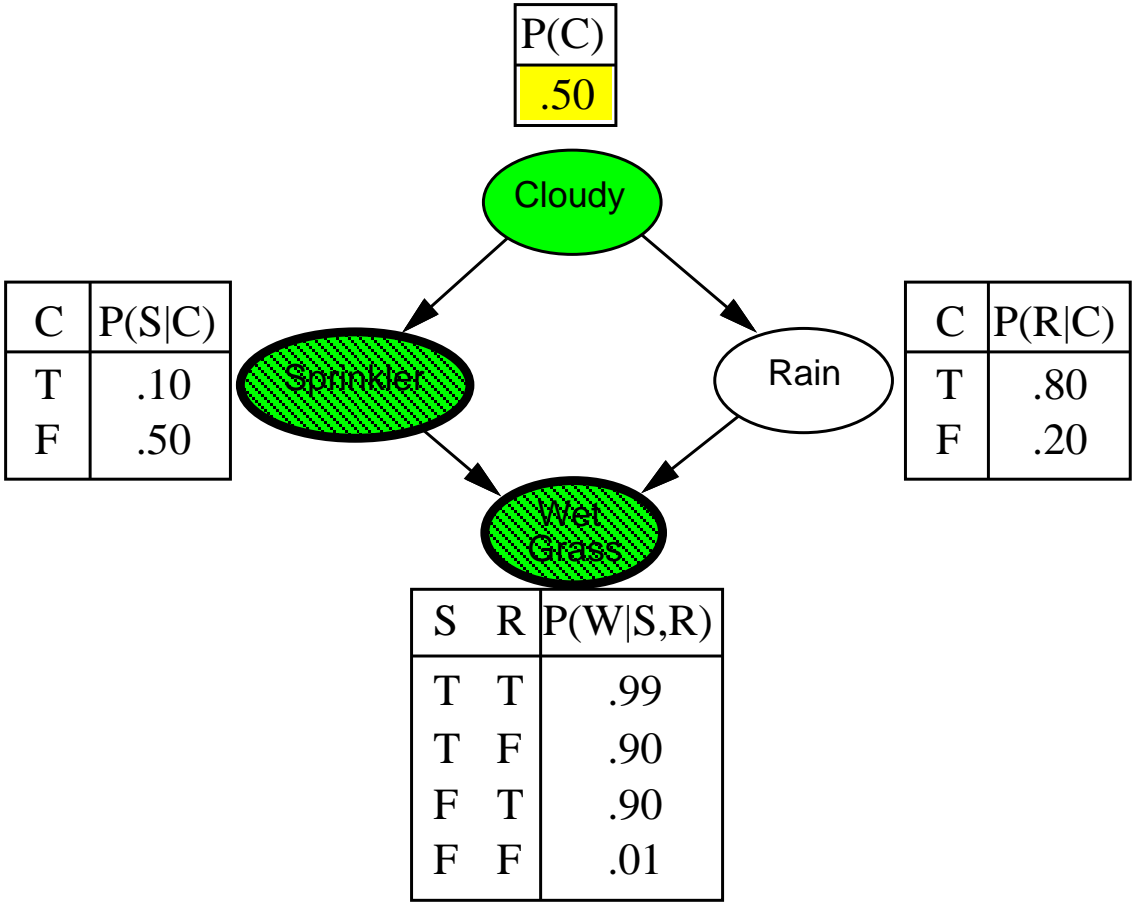
```
function WEIGHTED-SAMPLE( $bn, \mathbf{e}$ ) returns an event and a weight  
   $\mathbf{x} \leftarrow$  an event with  $n$  elements;  $w \leftarrow 1$   
  for  $i = 1$  to  $n$  do  
    if  $X_i$  has a value  $x_i$  in  $\mathbf{e}$   
      then  $w \leftarrow w \times P(X_i = x_i \mid \text{parents}(X_i))$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

# Likelihood weighting example



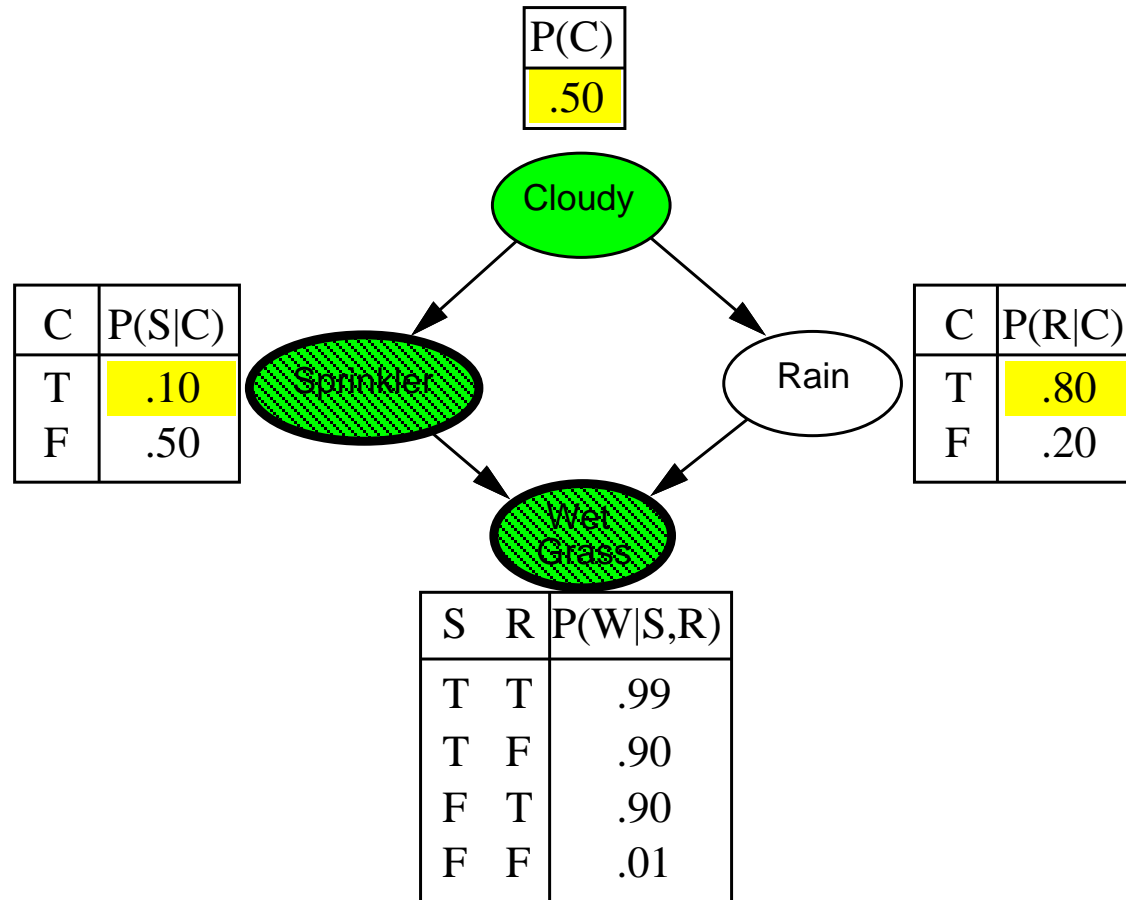
$$w = 1.0$$

# Likelihood weighting example



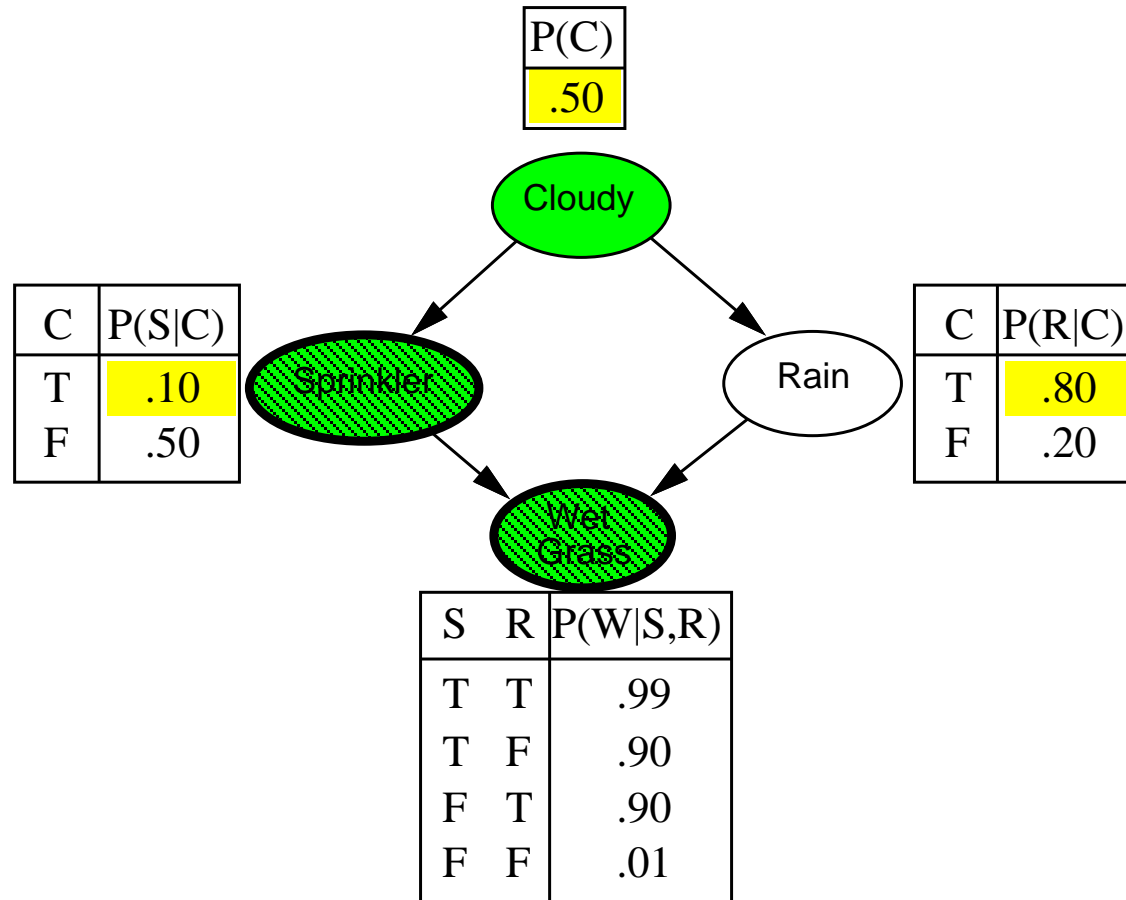
$w = 1.0$

# Likelihood weighting example



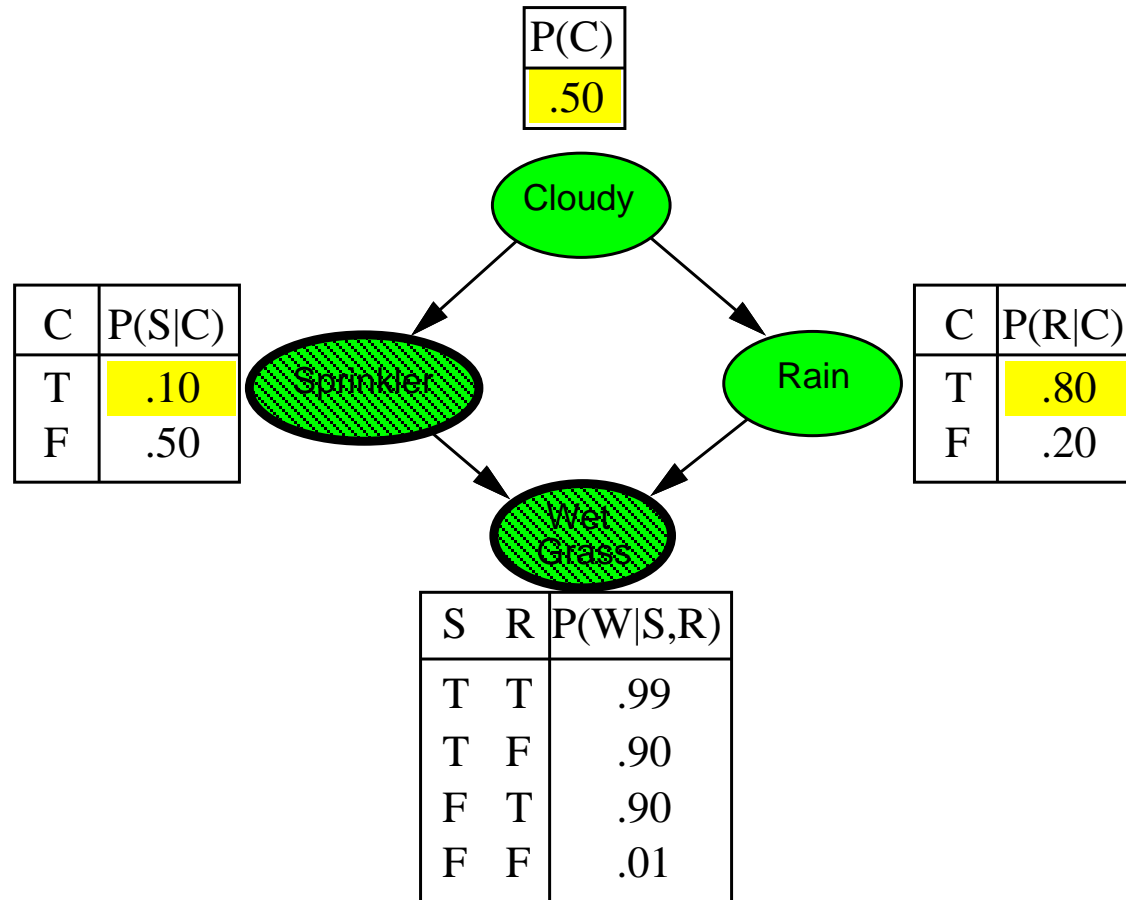
$$w = 1.0$$

# Likelihood weighting example



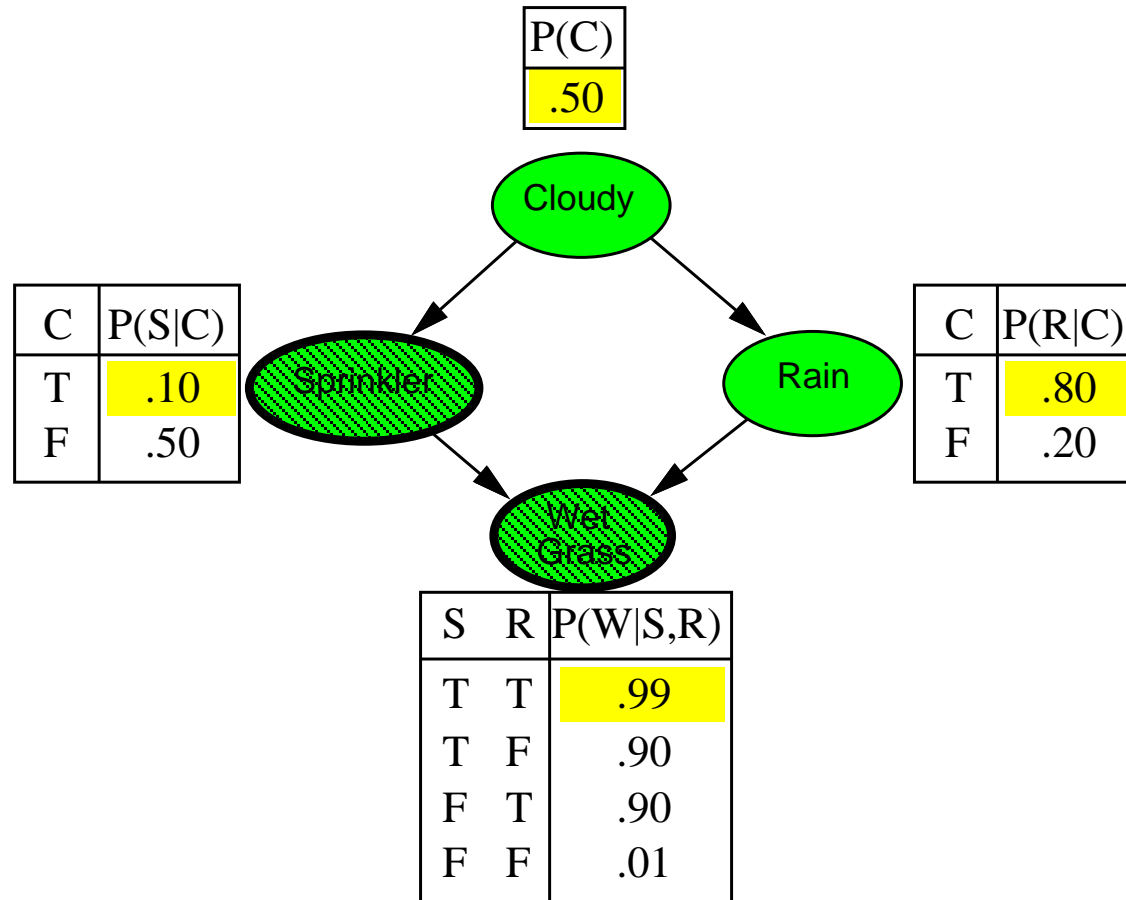
$$w = 1.0 \times 0.1$$

# Likelihood weighting example



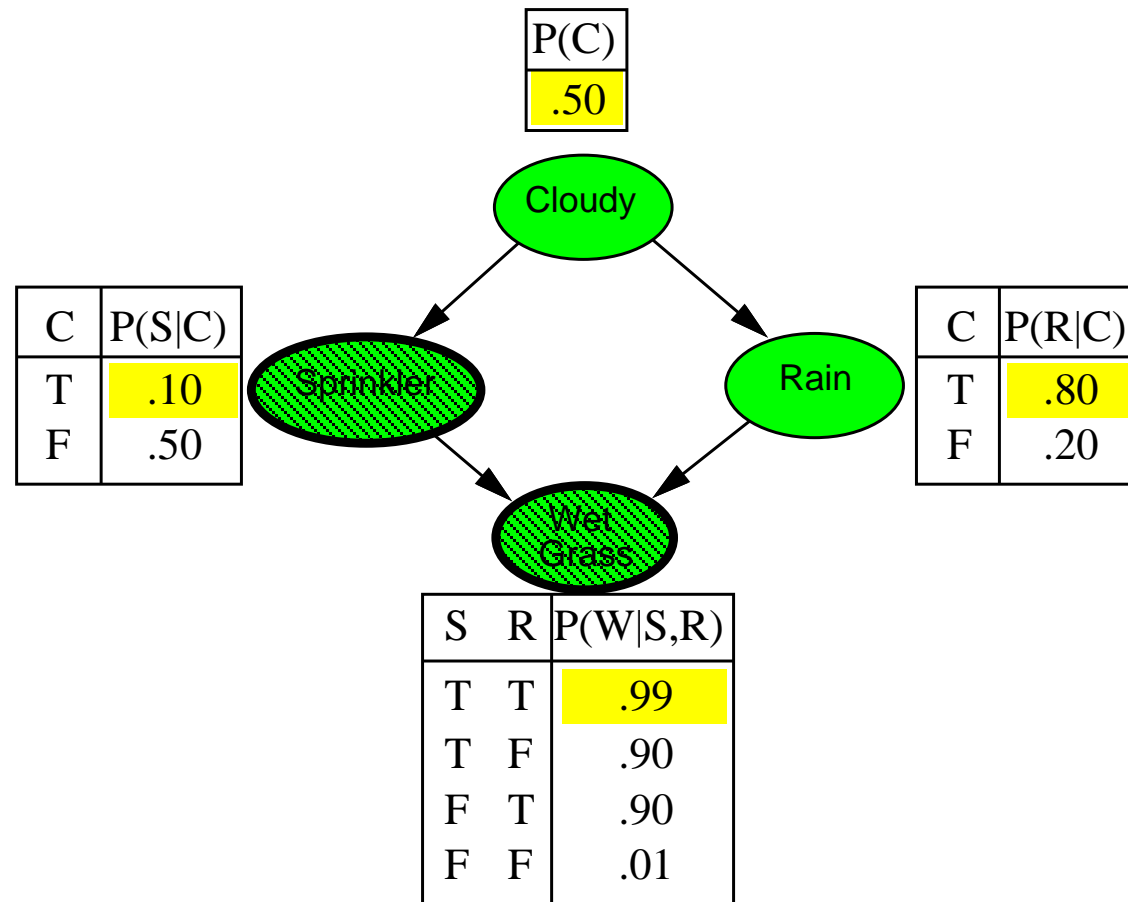
$$w = 1.0 \times 0.1$$

# 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

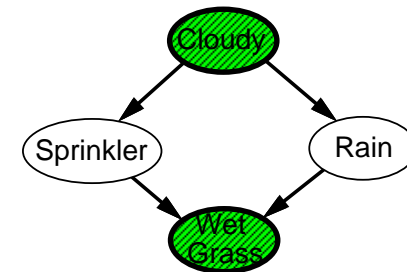
Weight for a given sample  $\mathbf{z}, \mathbf{e}$  is

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

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



# Summary

Exact inference by variable elimination:

- polytime on polytrees, NP-hard on general graphs
- space = time, very sensitive to topology

Approximate inference by stochastic simulation

- Convergence can be very slow with probabilities close to 1 or 0