Inference in Bayesian networks

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., $P(NoGas|Gauge=empty, Lights=on, Starts=false)$

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 $P(outcome|action, evidence)$

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

Conditional probability (CP) query is when we ask for the probability of a variable or set of variables given set of events.

Chapter 13 shows that CP can be computed by summing terms from full joint distribution. i.e. A query can be answered using a Bayesian network by computing sums of products of conditional probabilities from network.

$$\mathbf{P}(X|\mathbf{e}) = \alpha\mathbf{P}(\mathbf{X},\mathbf{e}) = \alpha\Sigma_{\mathbf{y}}\mathbf{P}(\mathbf{X},\mathbf{e},\mathbf{y})$$

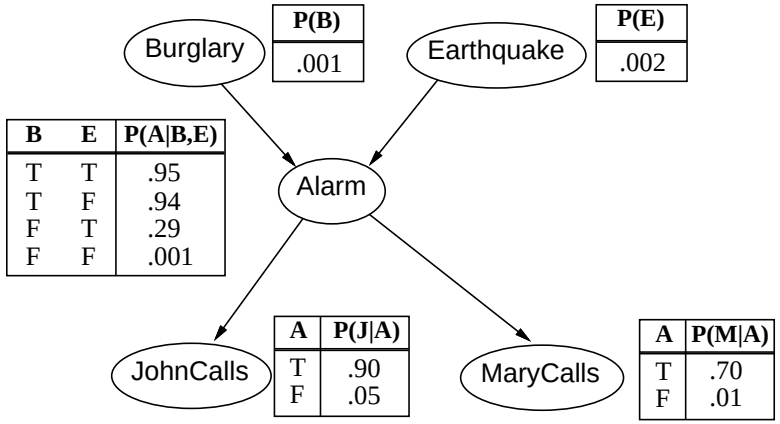
The complexity of this approach is $O(n2^n)$
Chapter 13 shows that CP can be computed by summing terms from full joint distribution.

Inference by enumeration is slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Inference by enumeration

Simple query on the burglary network:

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



Rewrite full joint entries using product of CPT entries:

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

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

Enumeration algorithm

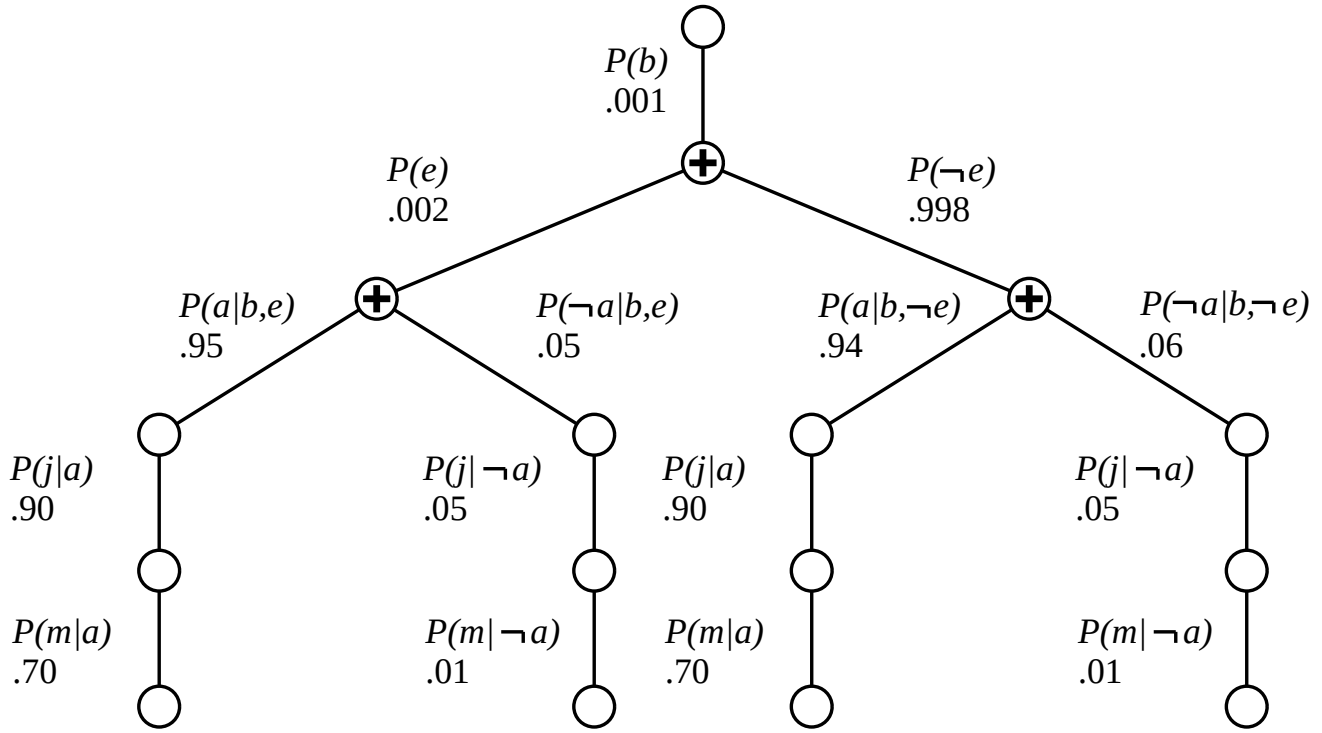
function ENUMERATION-ASK($X, \mathbf{e}, bn$) **returns** a distribution over X
  inputs: X, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayesian network with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$

  $\mathbf{Q}(X) \leftarrow$ a distribution over X, initially empty
  for each value x_i of X **do**
    extend $\mathbf{e}$ with value x_i for X
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL(VARS[bn], $\mathbf{e}$)
  return NORMALIZE($\mathbf{Q}(X)$)

---

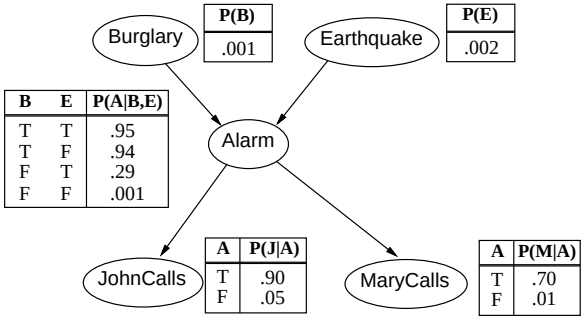
function ENUMERATE-ALL($vars, \mathbf{e}$) **returns** a real number
  if EMPTY?($vars$) **then return** 1.0
  $Y \leftarrow$ FIRST($vars$)
  if Y has value y in $\mathbf{e}$
    then return $P(y \mid Pa(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}$)
    else return $\Sigma_y \; P(y \mid Pa(Y)) \times$ 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

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

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

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

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: Basic operations

A	B	$\mathbf{f}_1(A,B)$
T	T	.3
T	F	.7
F	T	.9
F	F	.1

B	C	$\mathbf{f}_2(B,C)$
T	T	.2
T	F	.8
F	T	.6
F	F	.4

A	B	C	$\mathbf{f}_3(A,B,C)$
T	T	T	$.3 \times .2 = .06$
T	T	F	$.3 \times .8 = .24$
T	F	T	$.7 \times .6 = .42$
T	F	F	$.7 \times .4 = .28$
F	T	T	$.9 \times .2 = .18$
F	T	F	$.9 \times .8 = .72$
F	F	T	$.1 \times .6 = .06$
F	F	F	$.1 \times .4 = .04$

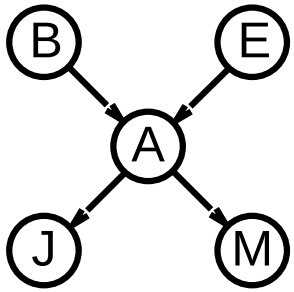
Figure 14.10 Illustrating pointwise multiplication: $\mathbf{f}_1(A,B) \times \mathbf{f}_2(B,C) = \mathbf{f}_3(A,B,C)$.

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 $\mathbf{P}(X_1, \ldots, X_n)$
  $factors \leftarrow [\,]$
  for each var **in** ORDER($bn, vars$) **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 NORMALIZE(POINTWISE-PRODUCT($factors$))

Irrelevant variables

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



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

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

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

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

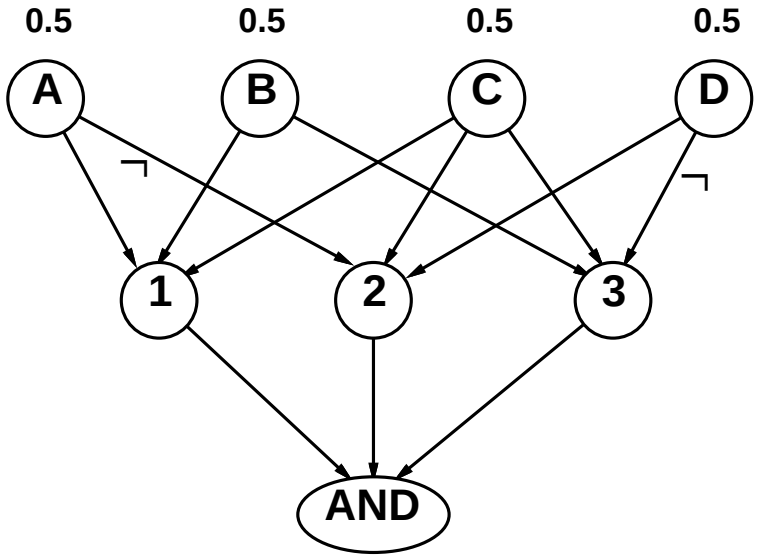
– can reduce 3SAT to exact inference $\Rightarrow$ NP-hard

– equivalent to **counting** 3SAT models $\Rightarrow$ #P-complete

1. A v B v C

2. C v D v ¬A

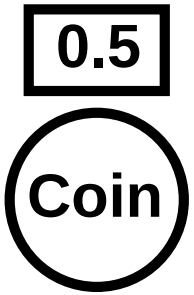
3. B v C v ¬D



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



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 $\mathbf{P}(X_1, \ldots, X_n)$

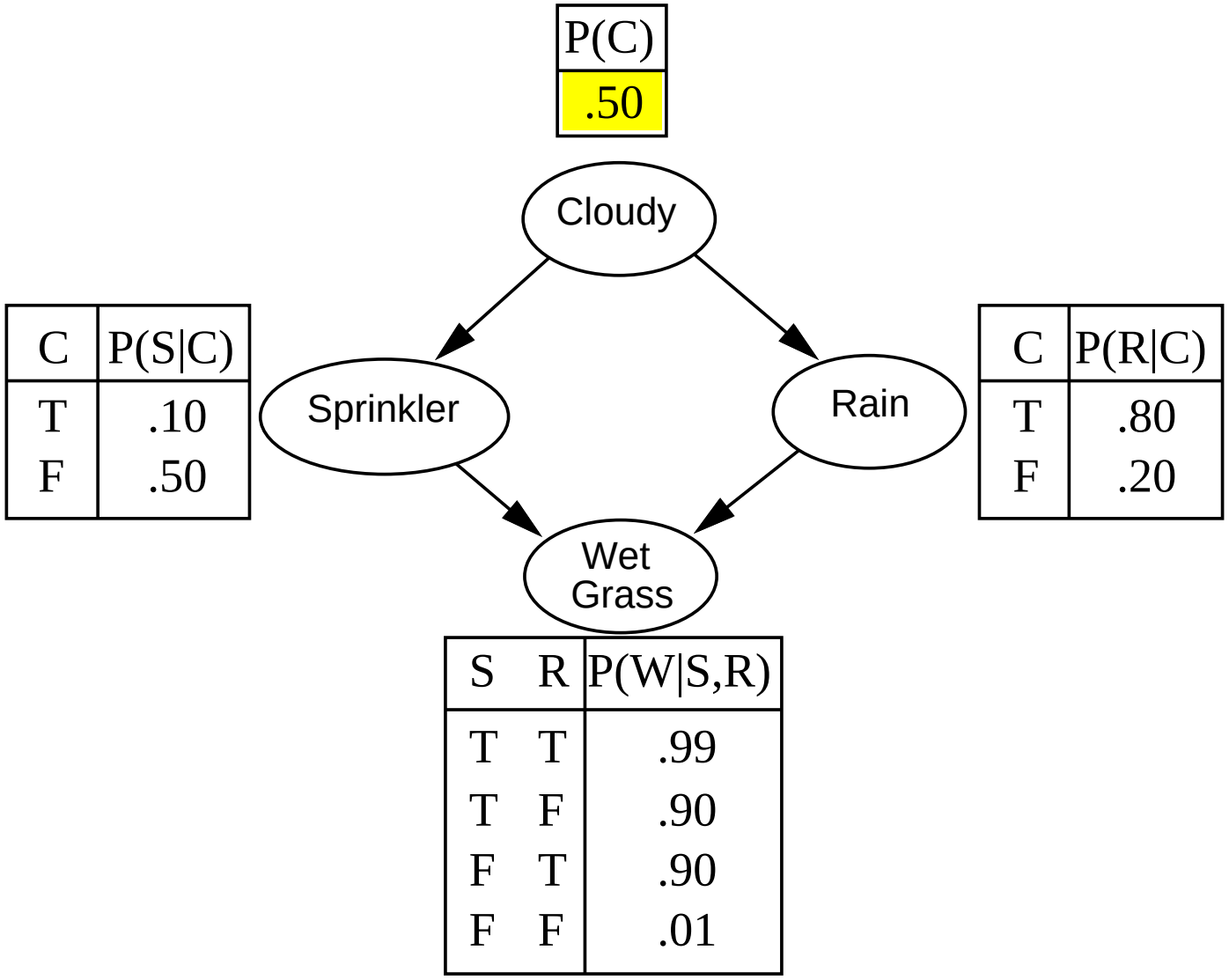
$\mathbf{x} \leftarrow$ an event with n elements

for $i = 1$ **to** n **do**

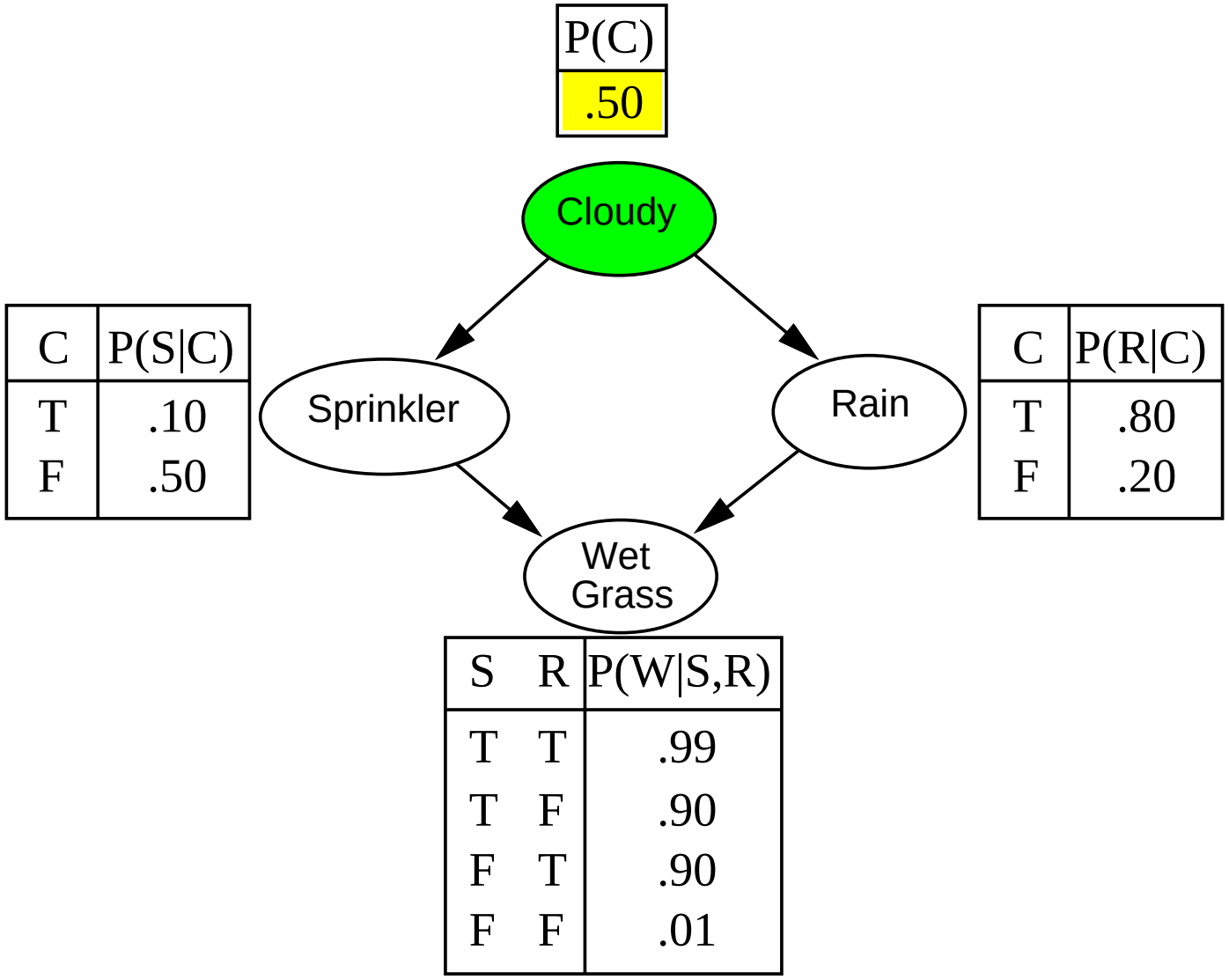
$x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
given the values of $Parents(X_i)$ in $\mathbf{x}$

return $\mathbf{x}$

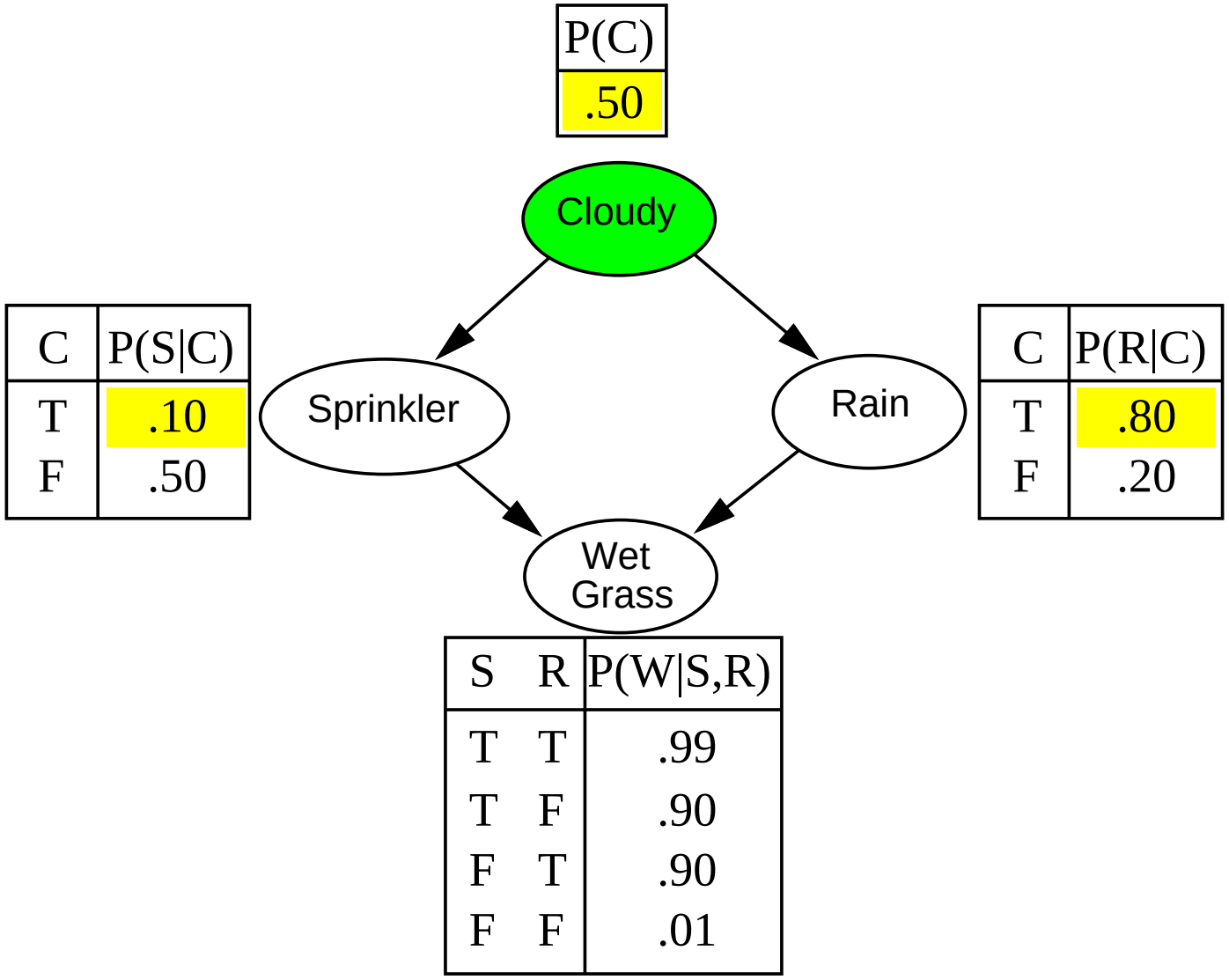
Example


P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01

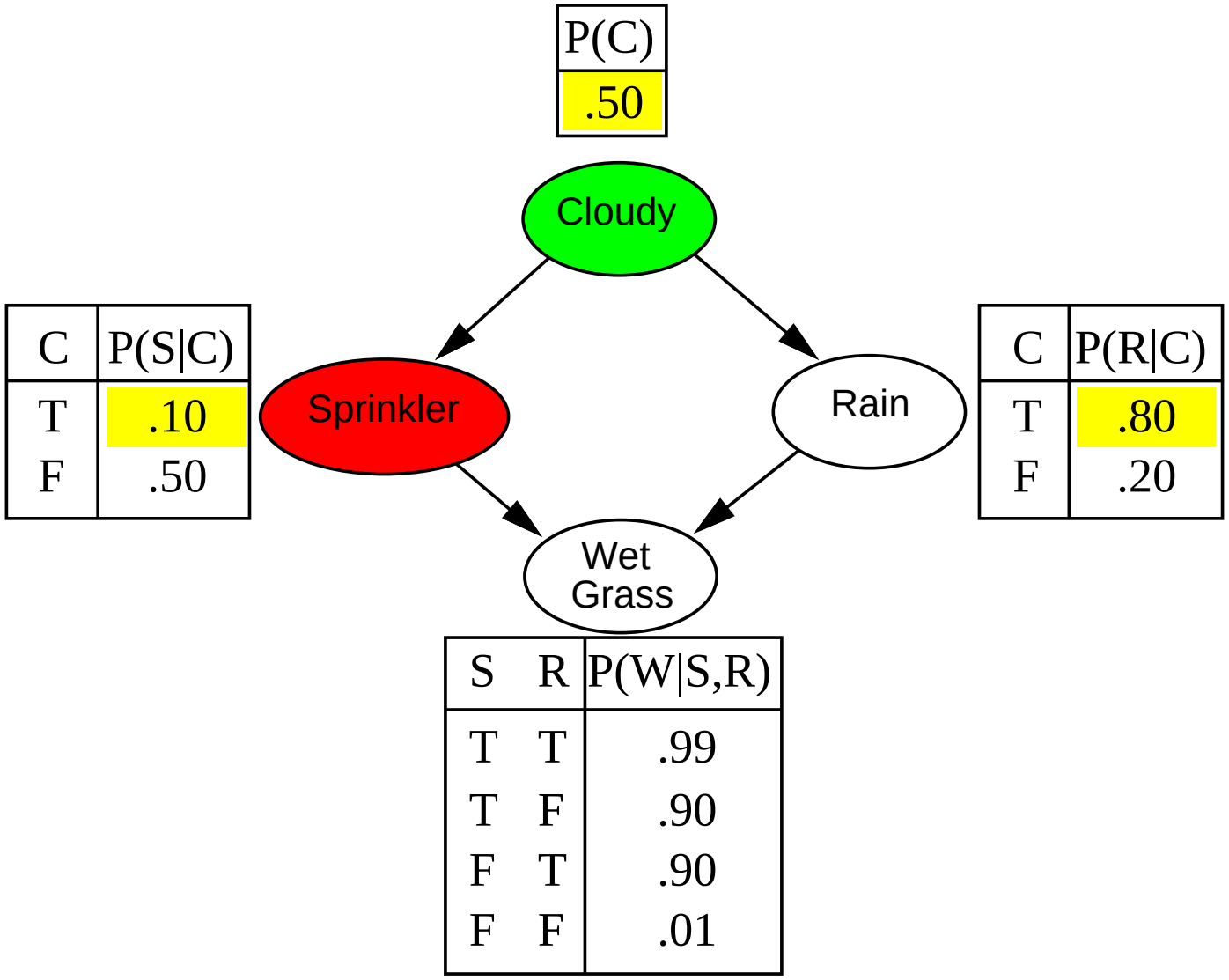
Example


P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01

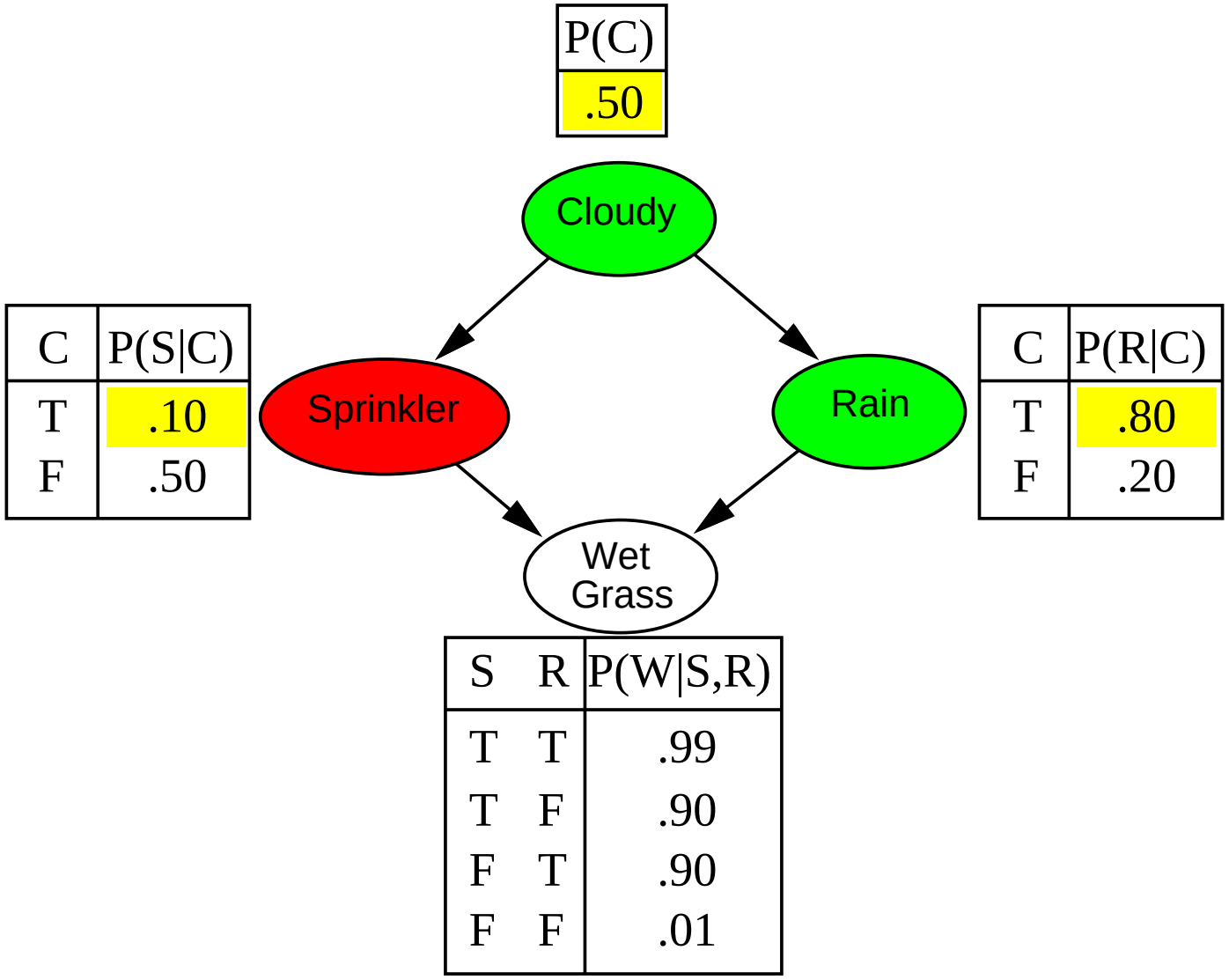
Example


P(C)
.50
Cloudy
C
P(S|C)
T
.10
F
.50
Sprinkler
Rain
C
P(R|C)
T
.80
F
.20
Wet
Grass
S
R
P(W|S,R)
T
T
.99
T
F
.90
F
T
.90
F
F
.01

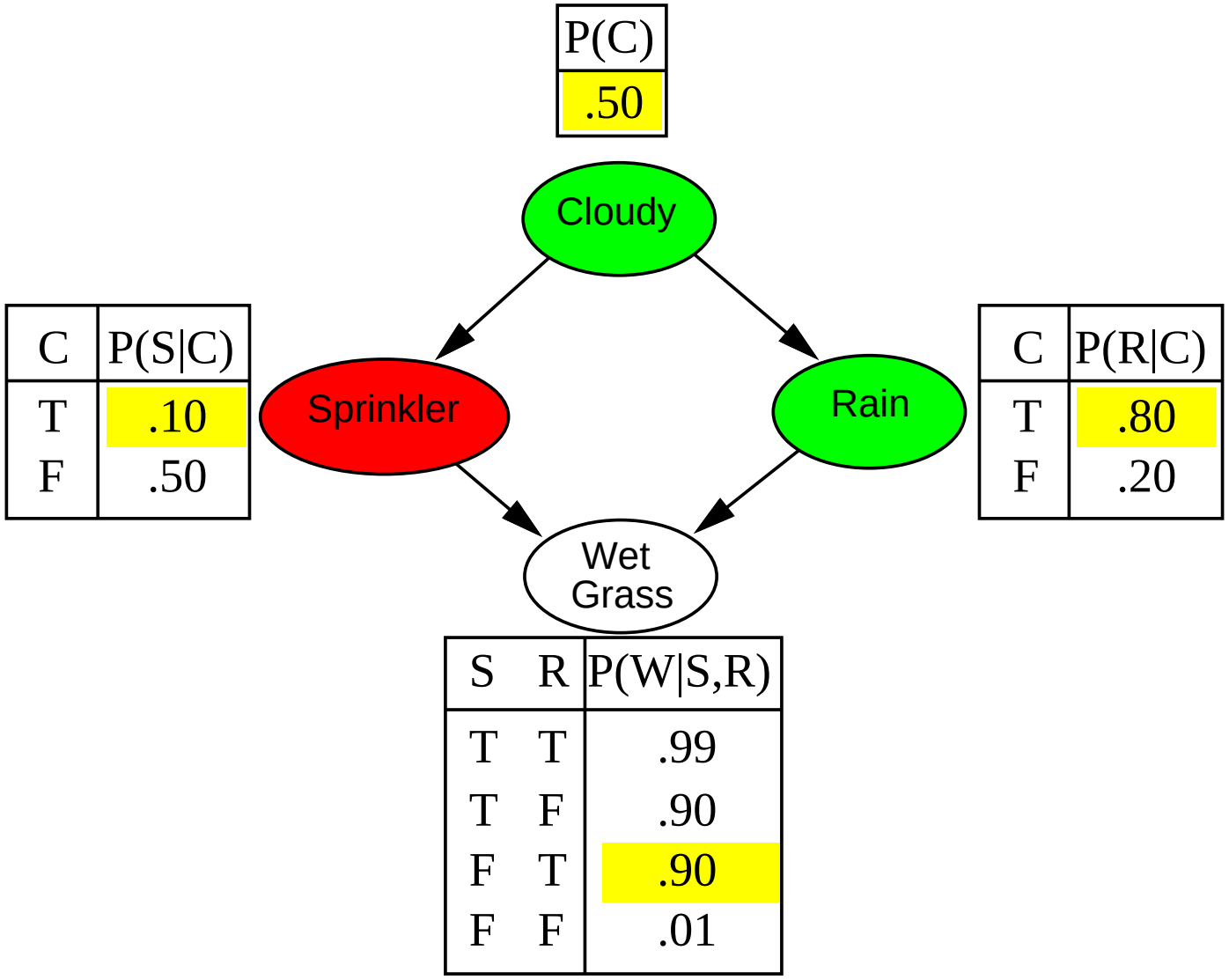
Example


P(C)
.50
Cloudy
C
P(S|C)
T
.10
F
.50
Sprinkler
Rain
C
P(R|C)
T
.80
F
.20
Wet
Grass
S
R
P(W|S,R)
T
T
.99
T
F
.90
F
T
.90
F
F
.01

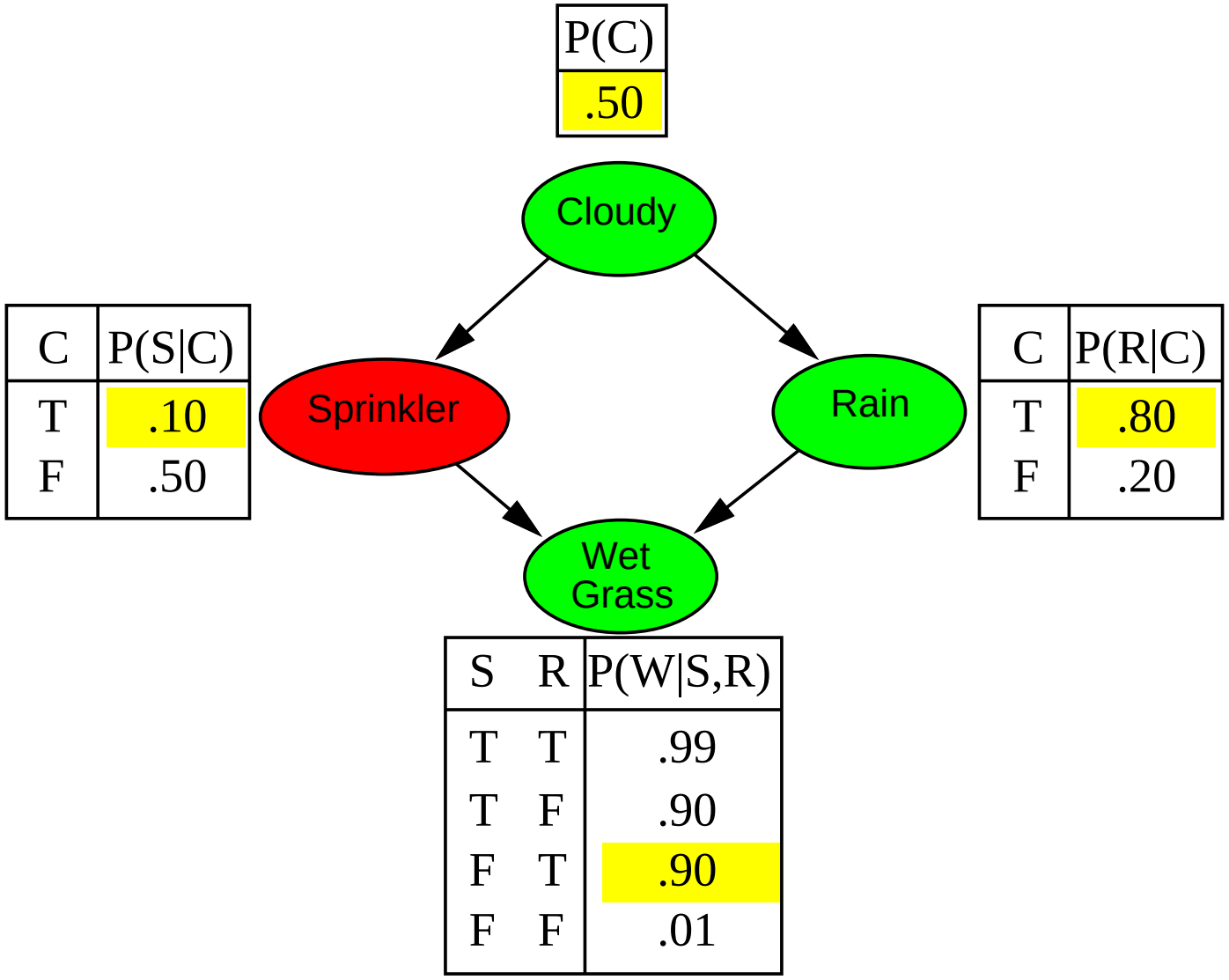
Example


P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01

Example


P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01

Example


P(C)
.50
Cloudy
C
P(S|C)
T
.10
F
.50
Sprinkler
Rain
C
P(R|C)
T
.80
F
.20
Wet Grass
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

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

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

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

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

$\hat{\mathbf{P}}(X|\mathbf{e})$ 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

$\hat{\mathbf{P}}(X|\mathbf{e}) = \alpha \mathbf{N}_{PS}(X,\mathbf{e})$ (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**

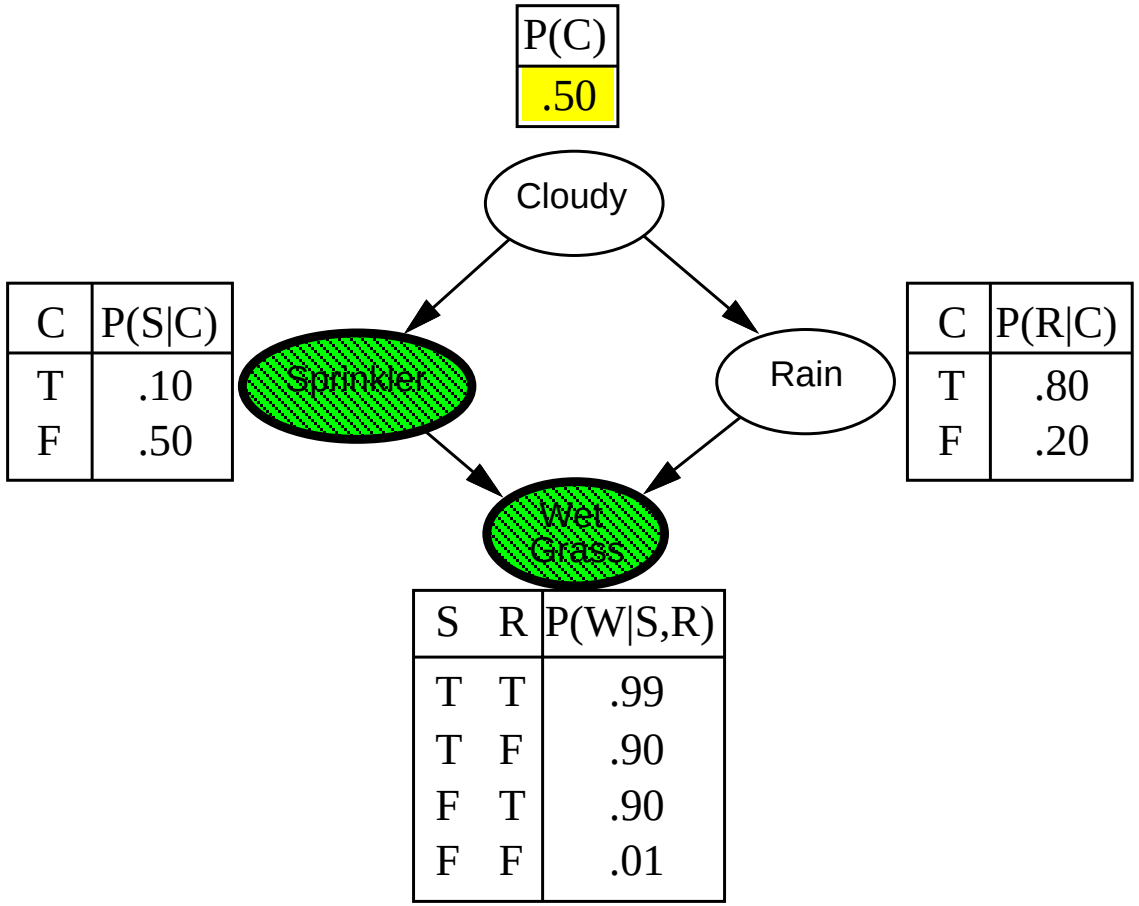
    if X_i has a value x_i in $\mathbf{e}$

      then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$

      else $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$

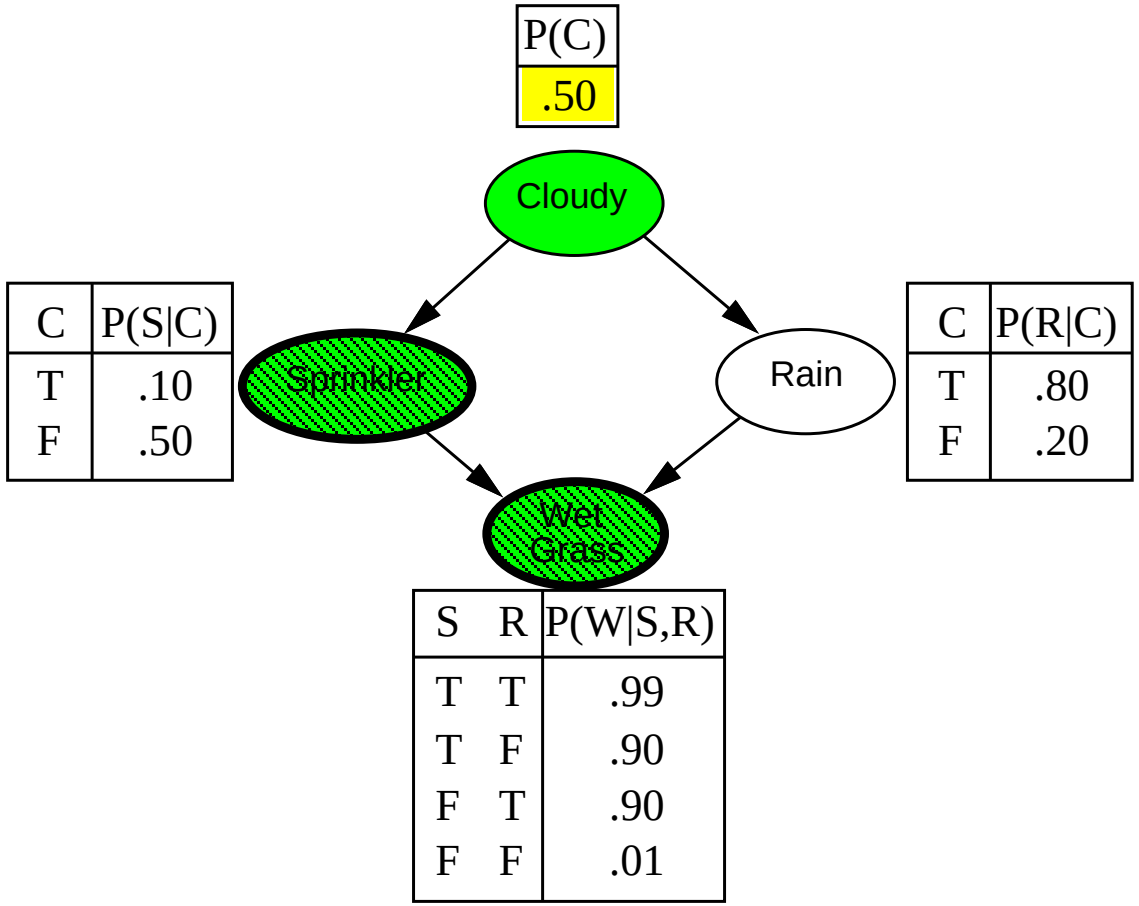
  return $\mathbf{x}$, w

Likelihood weighting example



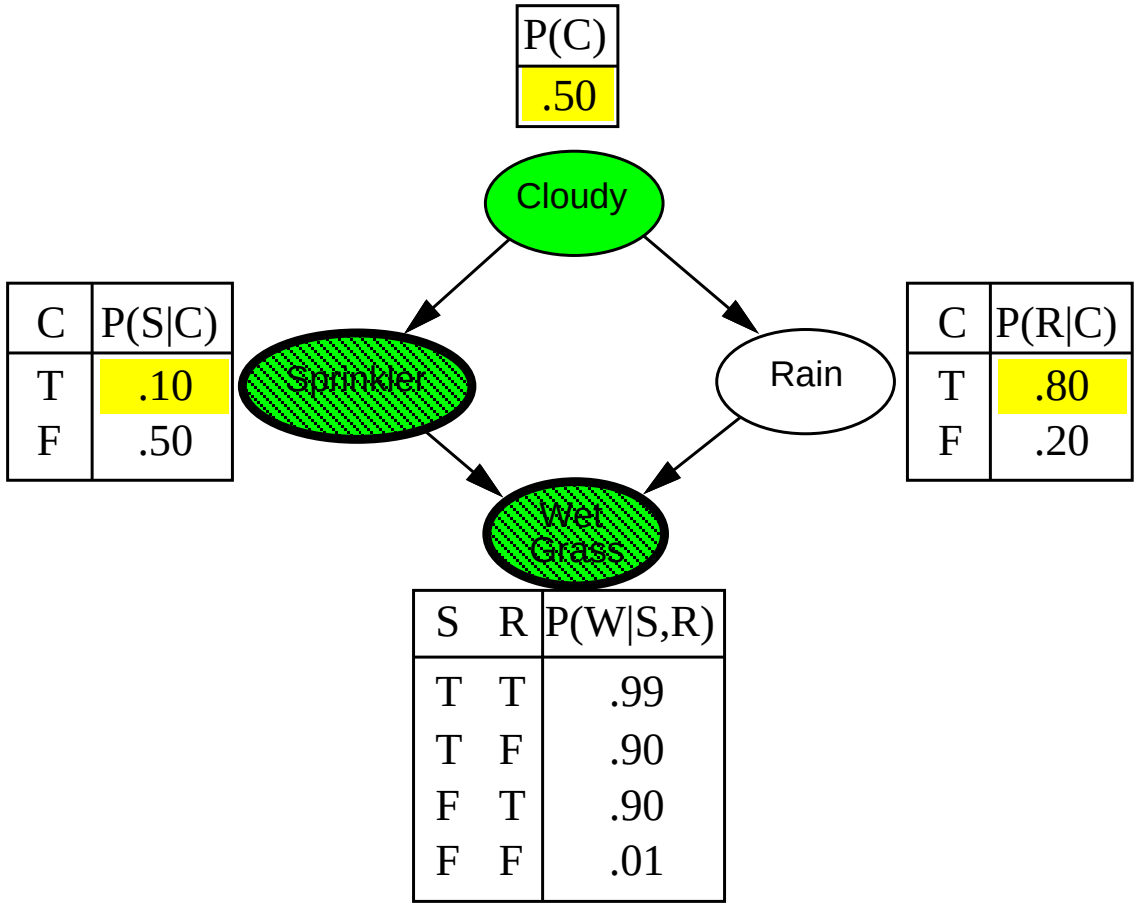
$w = 1.0$

Likelihood weighting example



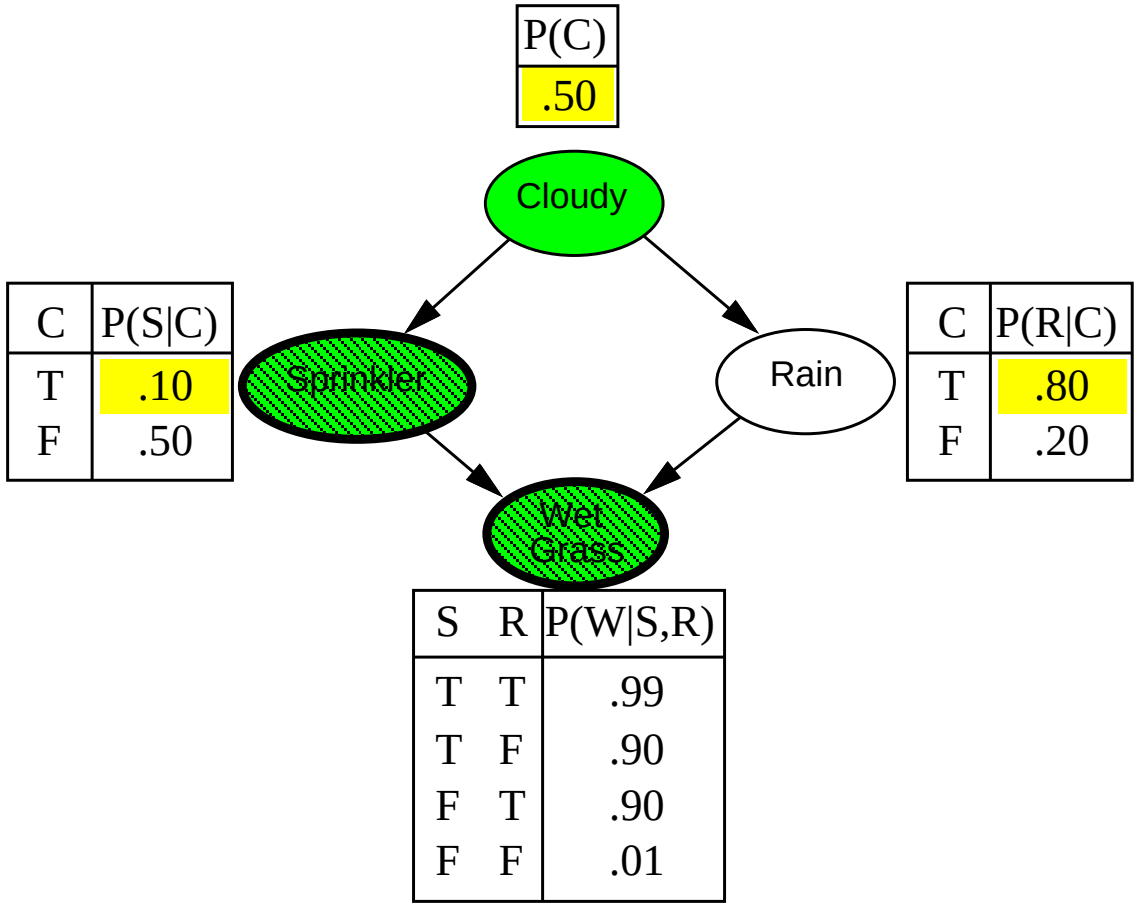
$w = 1.0$

Likelihood weighting example



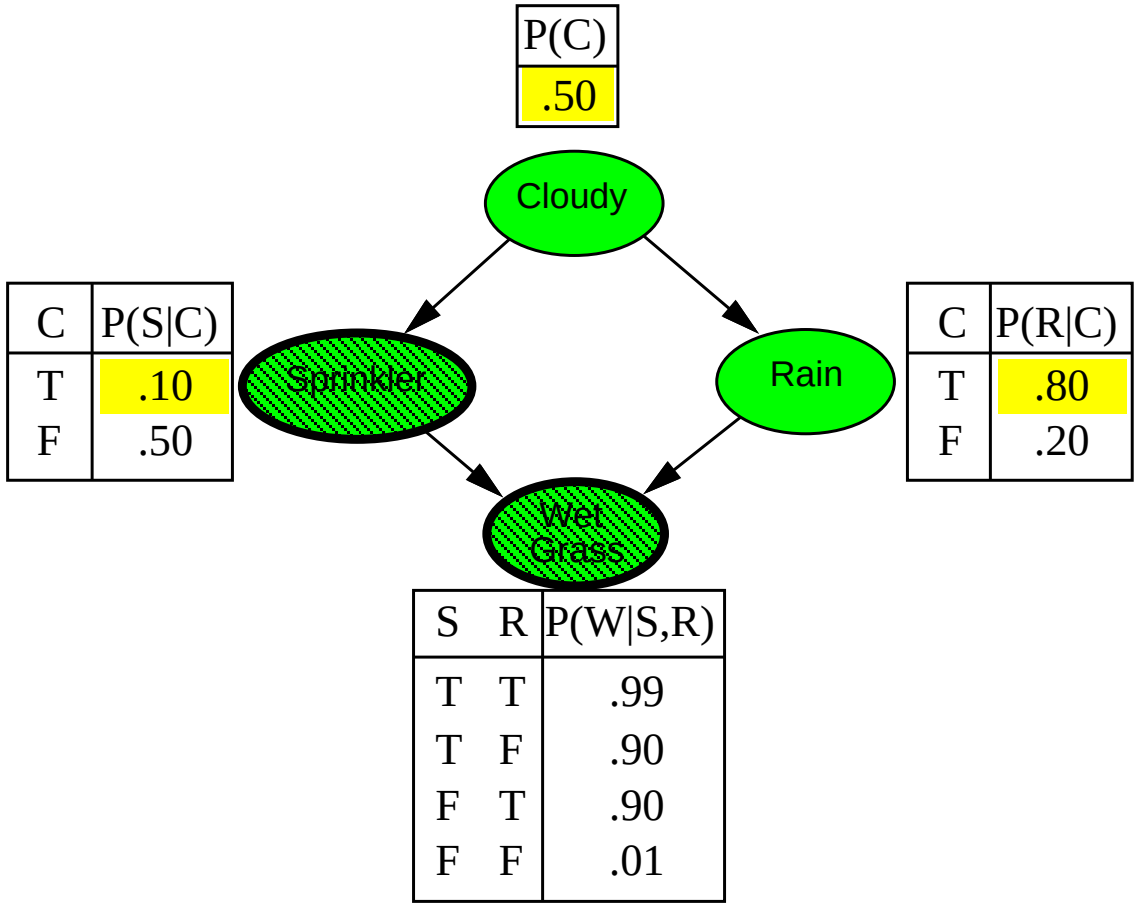
$w = 1.0$

Likelihood weighting example



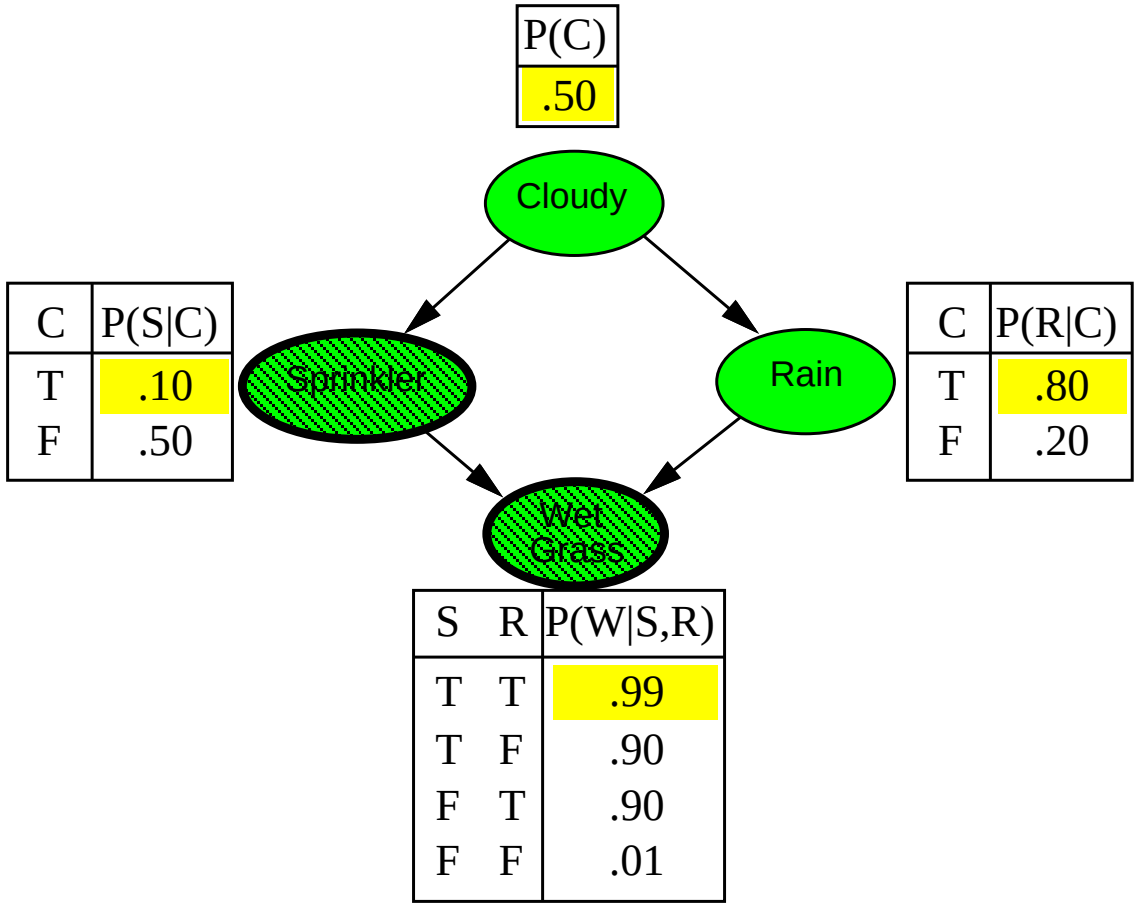
$w = 1.0 \times 0.1$

Likelihood weighting example



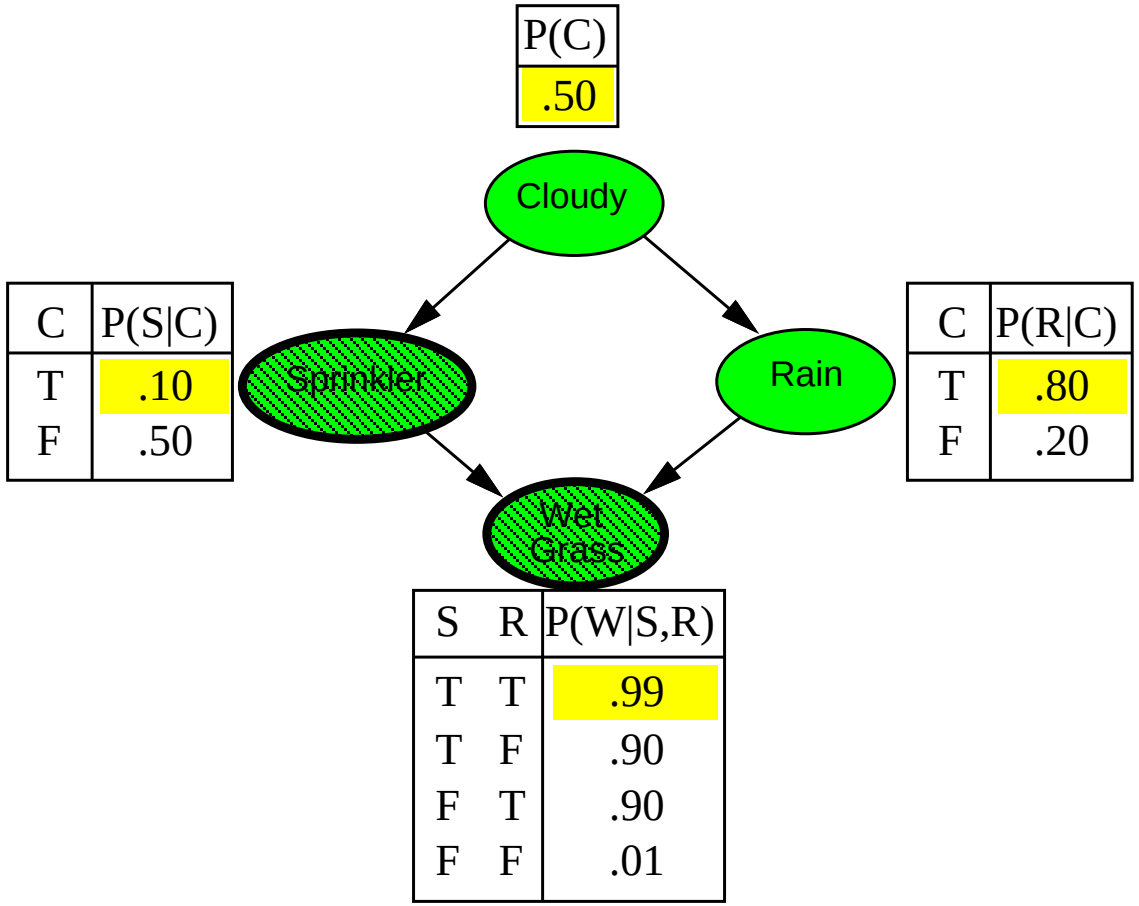
$w = 1.0 \times 0.1$

Likelihood weighting example



$w = 1.0 \times 0.1$

Likelihood weighting example



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

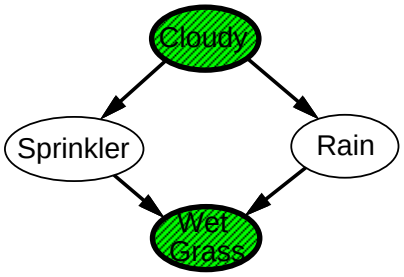
Likelihood weighting analysis

Sampling probability for WeightedSample is

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

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

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

Weighted sampling probability is

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

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