Artificial Intelligence

Probabilistic Reasoning (Probably the last part -- 4)

CS 444 – Spring 2019
Dr. Kevin Molloy
Department of Computer Science
James Madison University

Recall my question from last Thursday?

Given a coin, with potentially unknown bias, perform a fair coin toss.

```
def fairCoin(biasedCoin):
    coin1, coin2 = 0,0
    while coin1 == coin2:
        coin1, coin2 = biasedCoin(), biasedCoin()
    return coin1
```

Quick recap, why are we doing all this Probability stuff?

Recall we want to reason. And we know that:

$$\text{Toothache} \implies \text{Cavity}$$

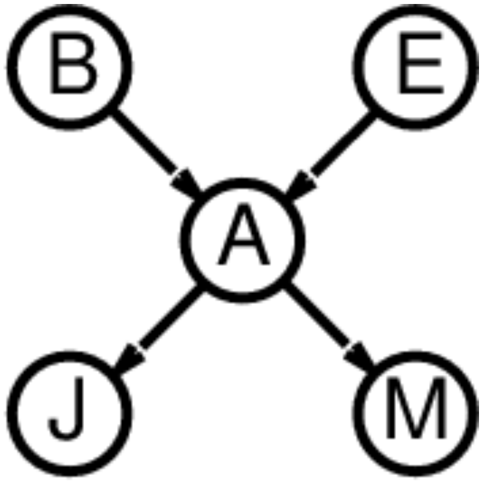
Is this correct?

Recall many things can cause a toothache? Gum disease for example, these people have Toothache = True, but may have cavity = false (not a valid implication).

Complexity of Exact Inference

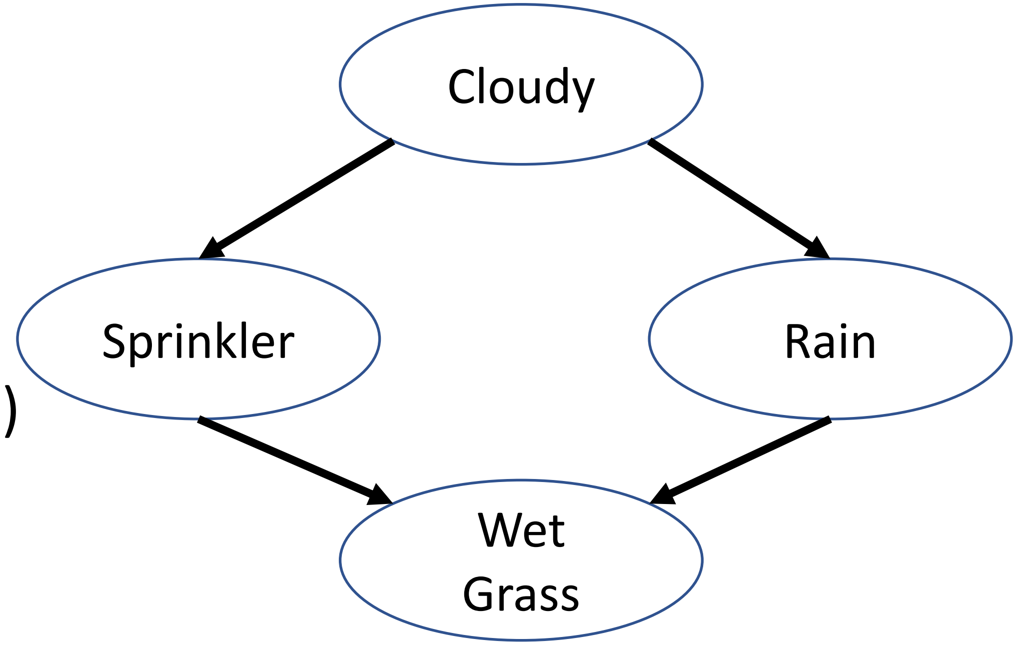
Singly connected BN (or polytrees):

- Any two nodes are connected by at most one (undirected path)
- Worst-case time and space complexity is O(n)
- Worst-case time and space cost of n queries is $O(n^2)$.



However, for multi connected networks:

- Worst-case time and space costs are expotential, $O(n \cdot d^n)$(n queries, d values per r.v.)
- NP-Hard (can reduce 3SAT to exact inference $\Longrightarrow$ NP-Hard)



Inference by Stochastic Simulation (Sampling-based)

Basic idea:

1. Draw N samples from a sampling distribution S. Can you draw N samples for the r.v. Coin from the probability distribution P(Coin) = [0.5, 0.5] ?
2. Compute an approximate posterior probability $\hat{P}$
3. Show this converges to the true probability P

Outline:

1. Direct sampling: Sampling from an empty network
2. Rejection sampling: reject samples disagreeing with the evidence
3. Likelihood weighting: use evidence to weight samples
4. Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior

Direct Sampling: Sampling from an Empty Network

Empty refers to the absence of any evidence: used to estimate joint probabailities

Main idea:

- Sample each r.v. in turn, in topological order, from parents to children
- Once parent is sampled, its value is fixed and used to sample the child
- Events generated via this direct sampling, observing joint probability distribution
- To get (prior) probability of an event, have to sample many times, so frequency of "observing" it among samples approaches it probability

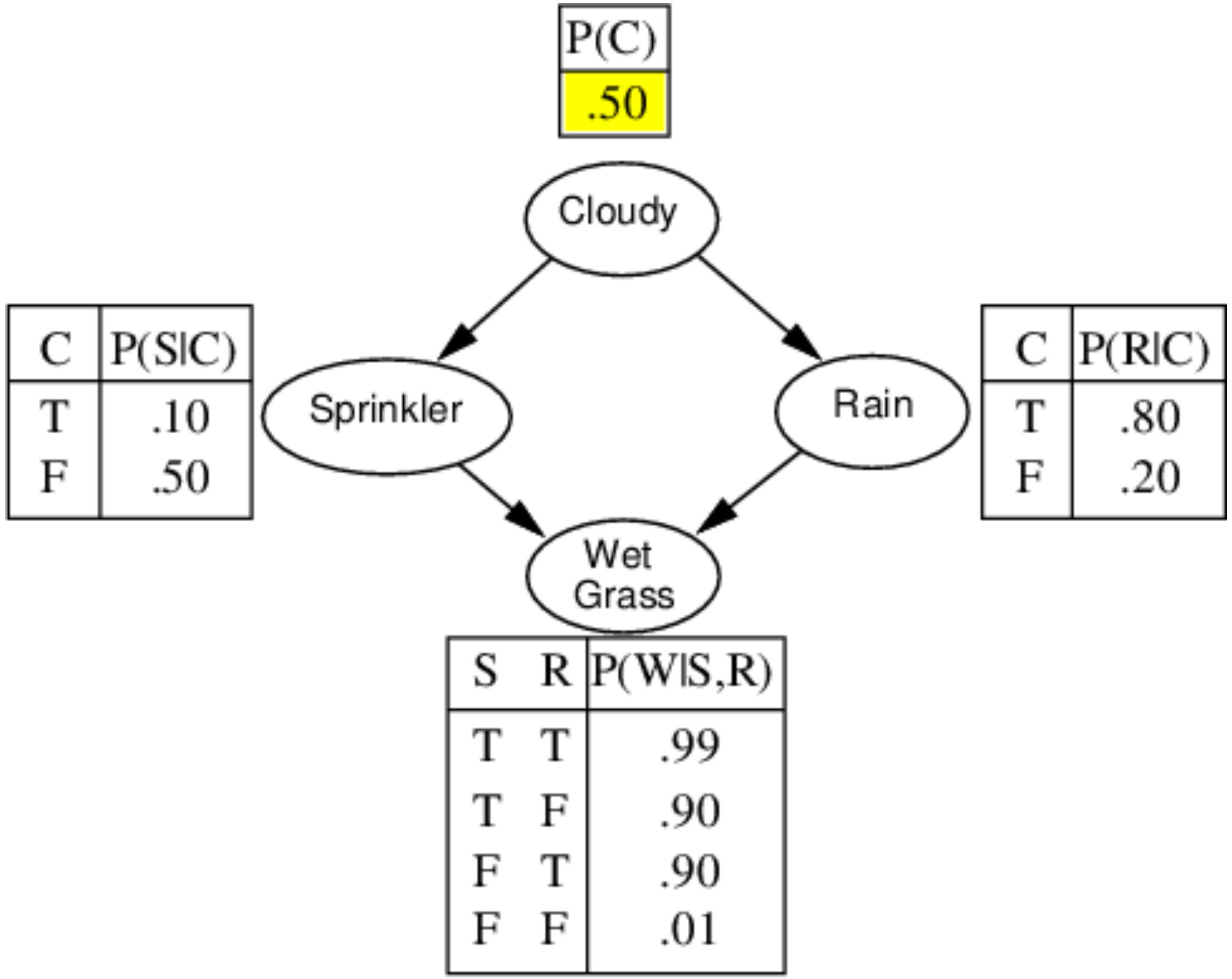
Direct Sampling Example

function Prior_Sample(bn) **returns** an event sampled from bn
Inputs: bn, a belief network specifying the joint distribution $P(X_1, \ldots, X_n)$

x $\leftarrow$ an event with n elements

for i = 1 to n do
$x_i \leftarrow$ a random sample from $P(X_i \mid parents(X_i))$ given the values of Parents(X_i) in x
return x

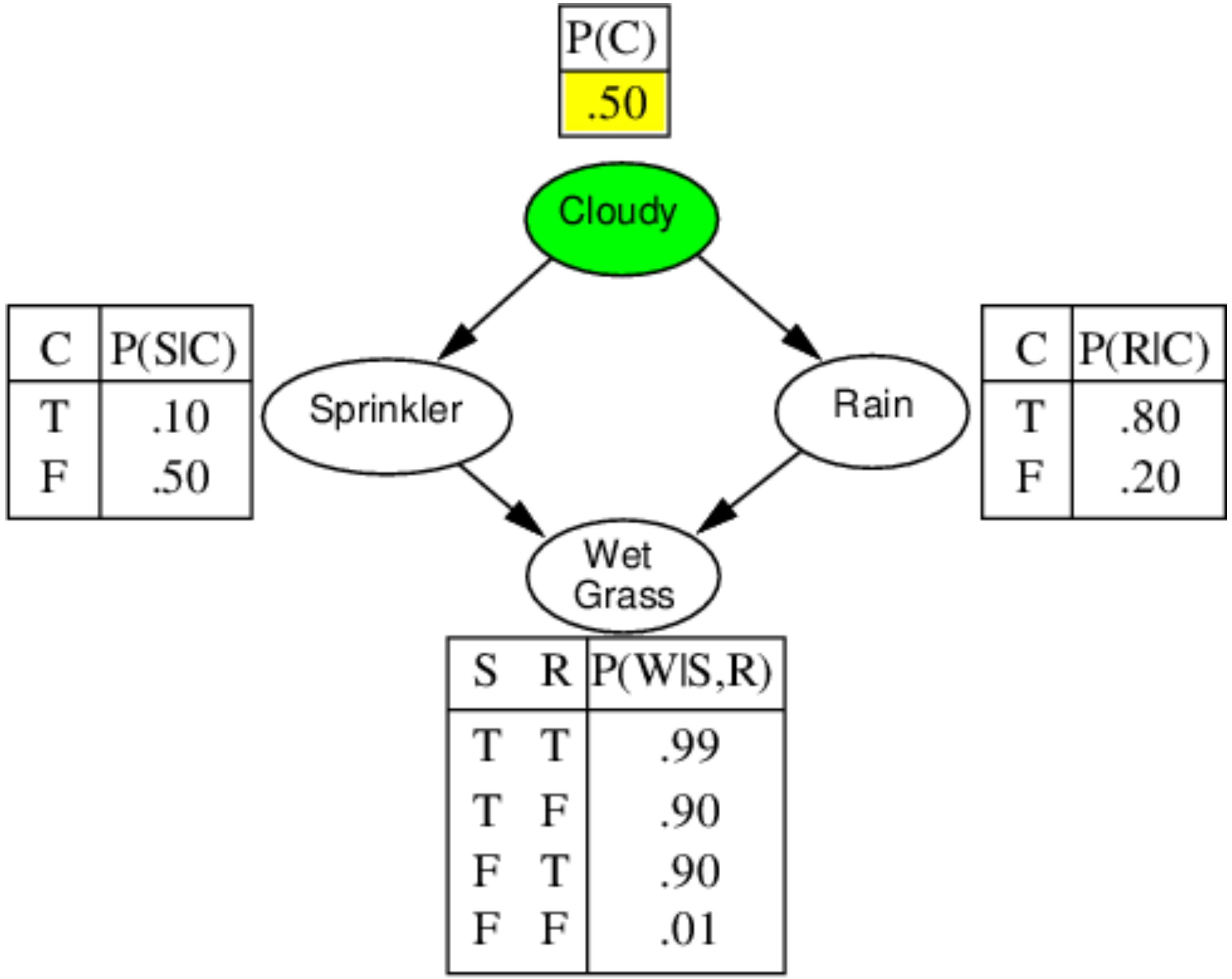
Direct Sampling Example



P(WetGrass).

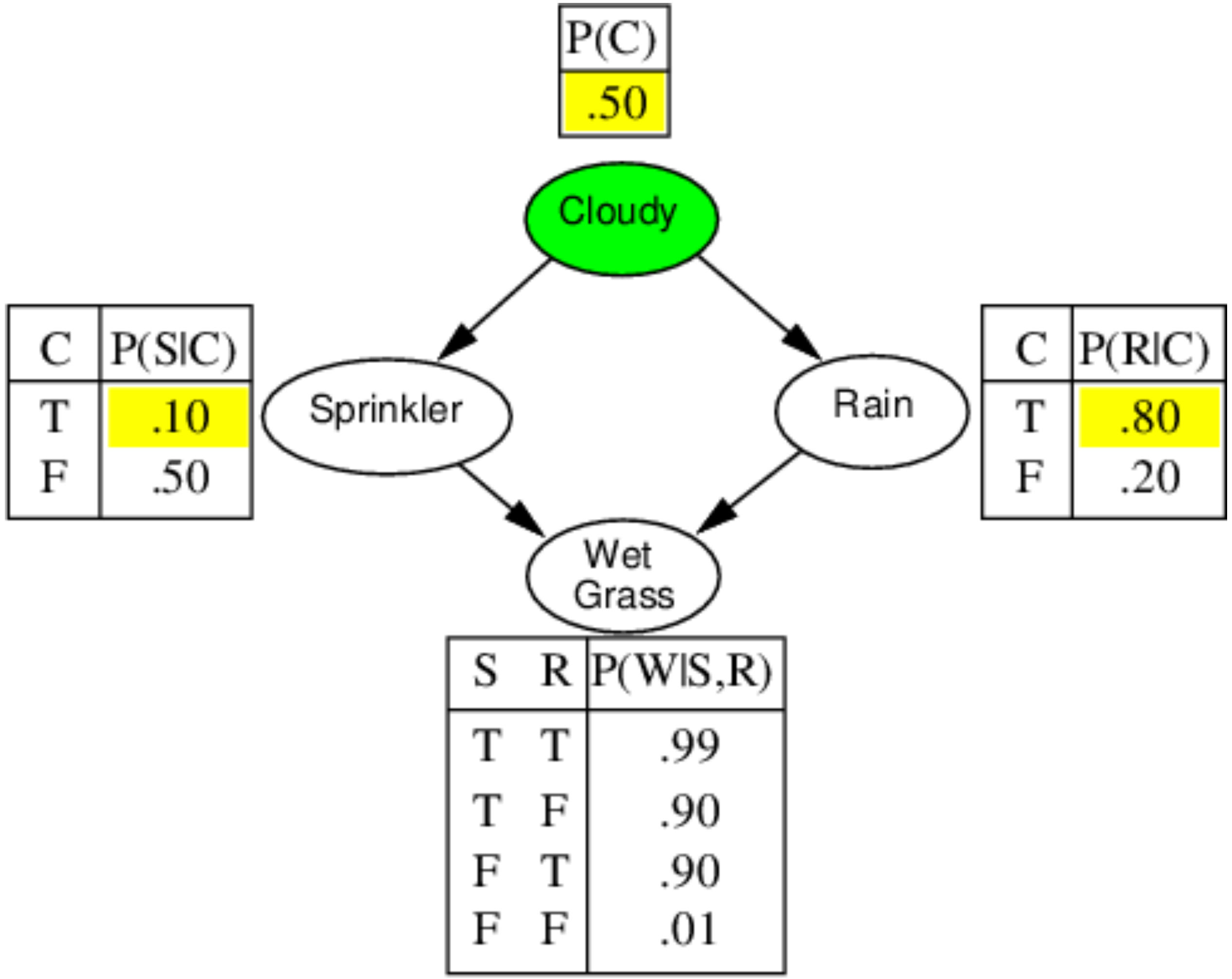
Given the form $\sum_z P(\text{WetGrass}|\boldsymbol{e}, \boldsymbol{z})$

Direct Sampling Example



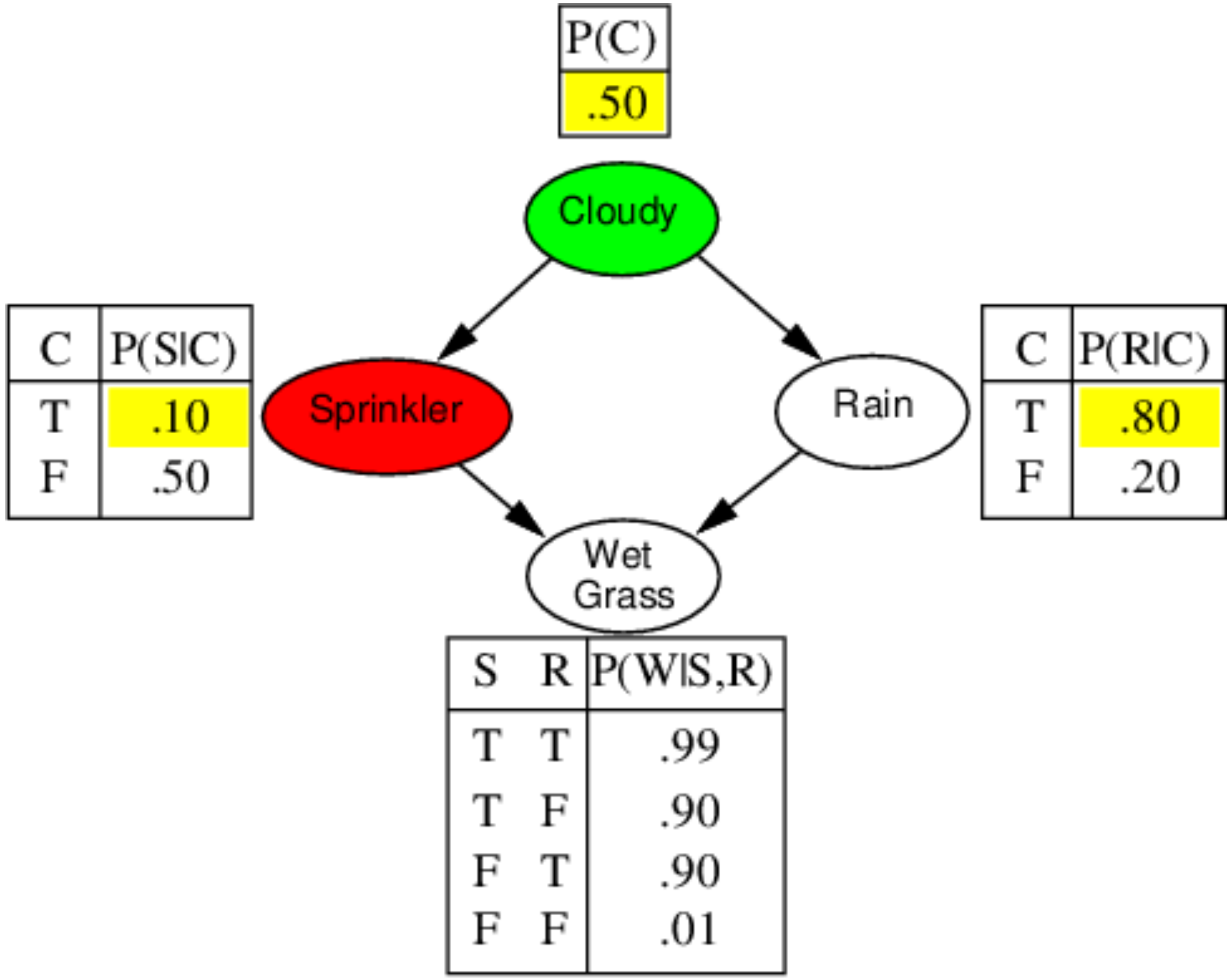
P(WetGrass) = 0.5 x ….

Direct Sampling



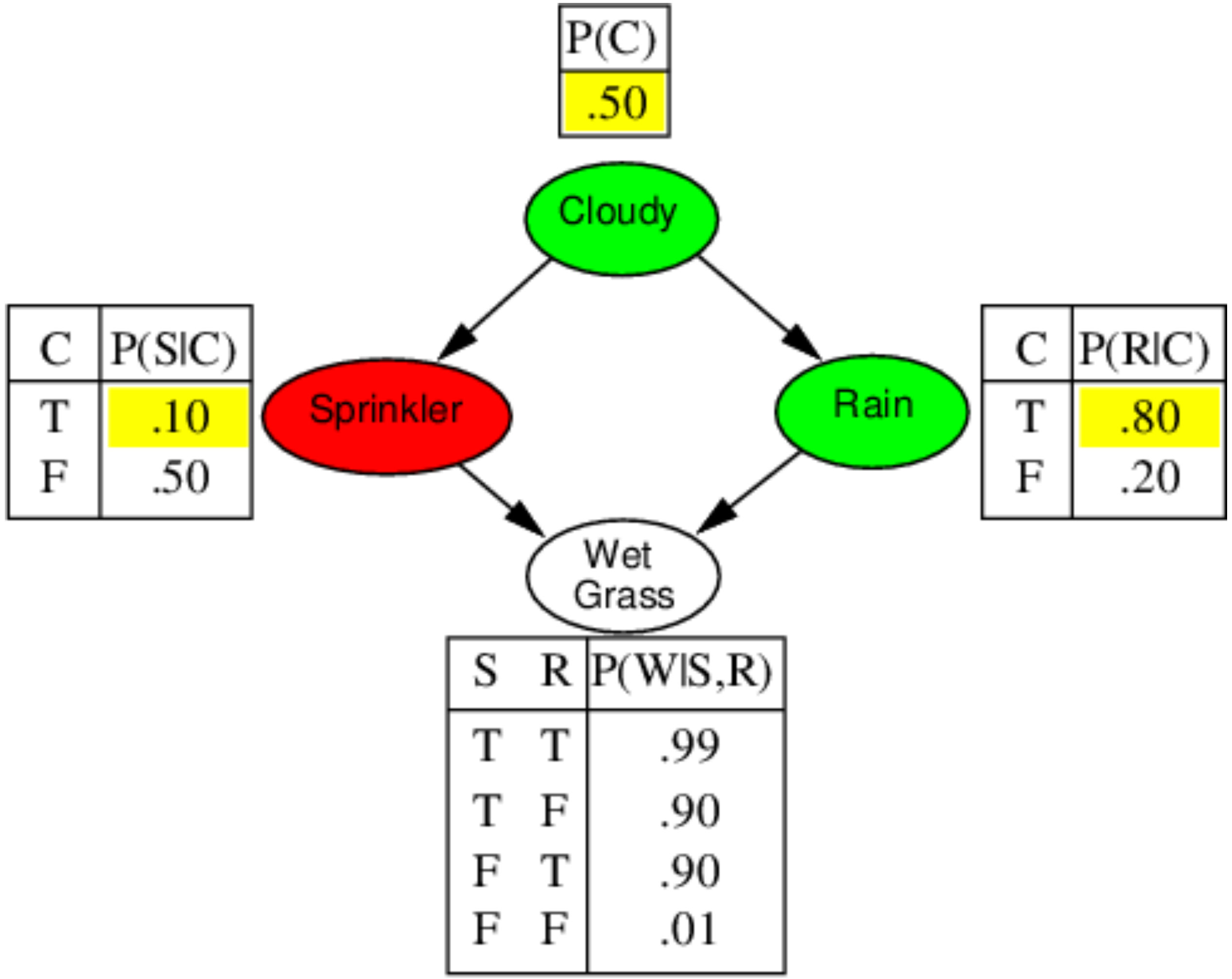
P(WetGrass) = 0.5 x ….

Direct Sampling Example



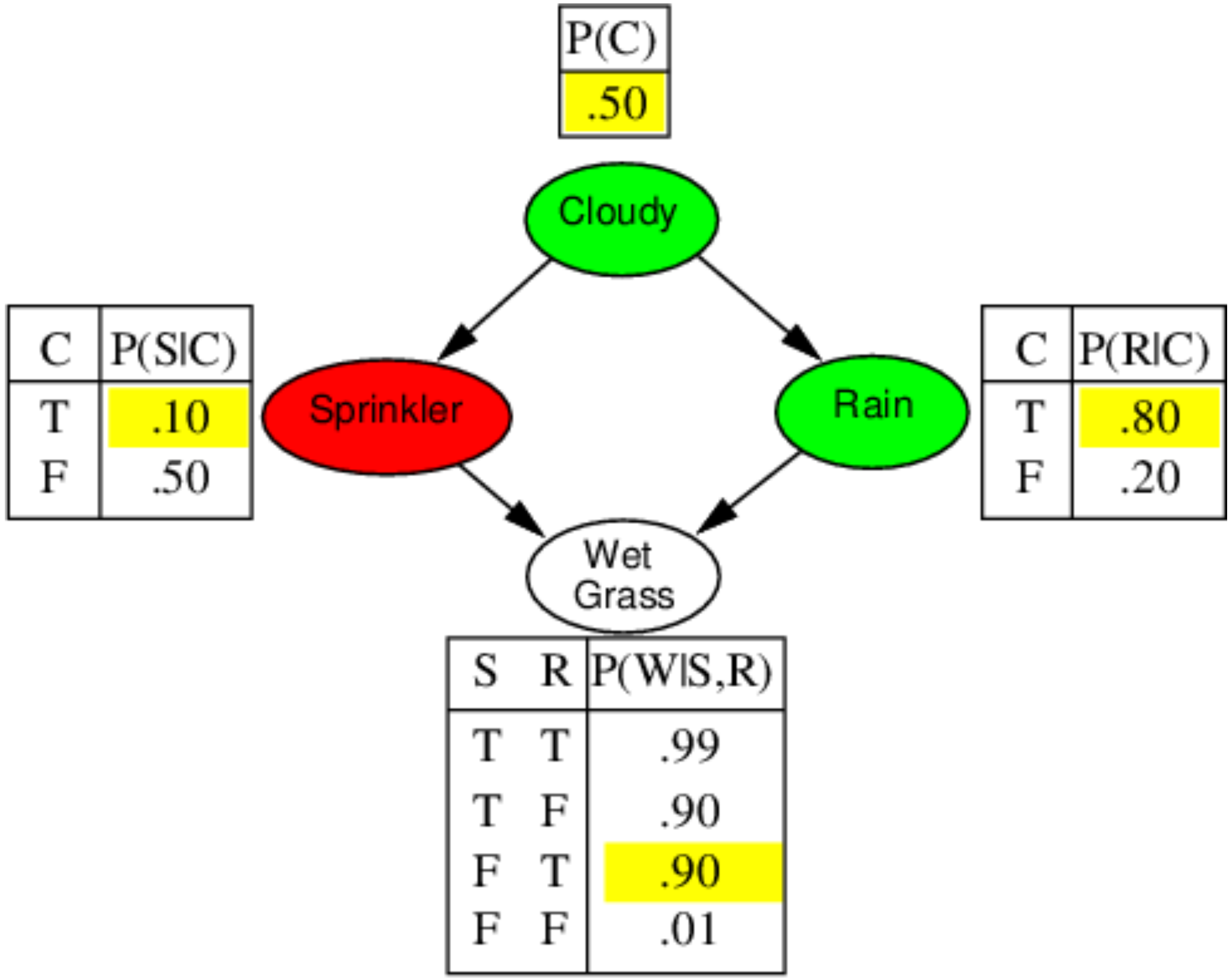
P(WetGrass) = 0.5 x 0.9 …

Direct Sampling Example



P(WetGrass) = 0.5 x 0.9 x 0.8 x …

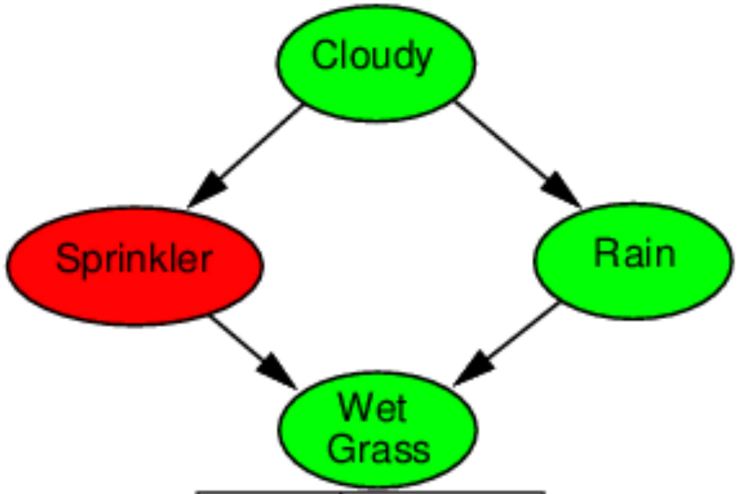
Direct Sampling Example



P(WetGrass) = 0.5 x 0.9 x 0.8 x …

Direct Sampling 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

P(WetGrass) = 0.5 x 0.9 x 0.8 x 0.9

P(c, ¬s, r, wg) ≈ 0.324

Rejection Sampling (for conditional probabilities P(X | e))

Main idea:

Given distribution too hard to sample directly from it, use an easy-to-sample distribution for direct sampling, and then reject samples based on hard-to-sample distribution.

1. Direct sampling to sample (X, E) events from prior distribution in BN
2. Determine whether (X, E) is consistent with given evidence e
3. Get $\hat{P}$ (X | E = e) by counting how often (E = e) and (X, E = e) occur as per Bayes' rule:
   $\hat{P}(X \mid E = e) = \frac{N(X,E=e)}{N(E=e)}$

Example: estimate P(Rain | Sprinkler = true) using 100 samples

Generate 100 samples for Cloudy, Sprinkler, Rain, WetGrass via direct sampling event of interest.

27 samples have Sprinkler = true, of these, 8 have Rain = true and 19 have Rain = false.

$\hat{P}$ (Rain | Sprinkler = true) = Normalize(⟨8, 19⟩) = ⟨8/27, 19/27⟩ = ⟨0.296, 0.704⟩

Similar to a basic real-world empirical estimation

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 l e)
Local Vars: N, a vector of counts over X, initially zero

for j = 1 to N do
$x_i \leftarrow$ 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 Normalized(N)

Analysis of Rejection Sampling

$\hat{P}$ (X|e) = αN_{ps} (X, e) algorithm definition)

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

≈ P(X, e)/P€

= P(X | e)

Hence, rejection sampling returns consistent posterior estimates.

Standard deviation of error in each probability proportional to $\frac{1}{\sqrt{n}}$ $(number\ of\ r.v.s)$

Problem:

If e is a very rare event, most samples are rejected; hopelessly expensive if P(e)is small.

P(e) drops off exponentially with number of evidence variables!

Rejection sampling is unusable for complex problems

Likelihood Weighting

A form of important sampling (for BNs)

Main idea:

Generate only events that are consistent with given values *e* of evidence variables *E*.

Fix evidence variables to given values, sample only nonevidence variables.

Weight each sample by the likelihood it accords the evidence (how likely *e* is).

Example: Query P(Rain | Cloudy = true, WetGrass = true)

Consider r.v.s in some topological ordering: Set w = 1.0 (weight will be a running product)

If r.v. Xi is in given evidence variables (Cloudy or WetGrass in this example),

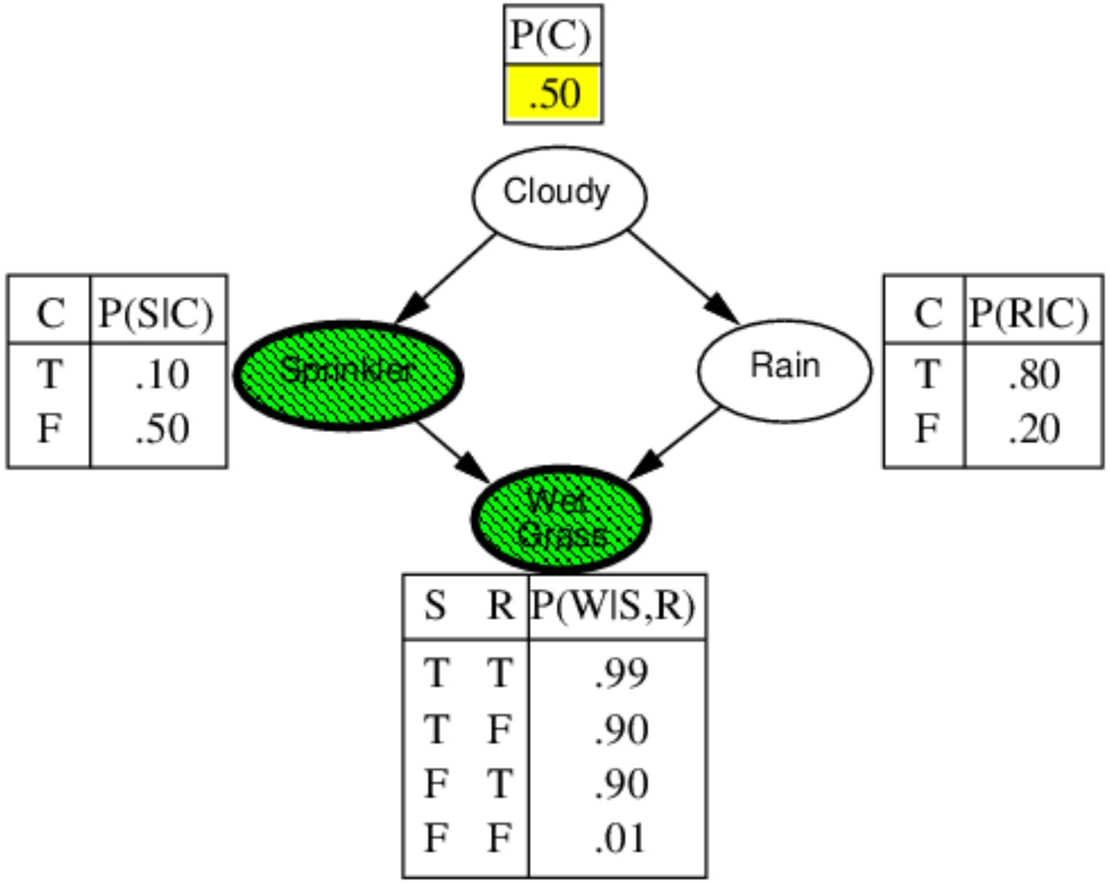
$w = w \times P(X_i \mid Parents(X_i))$

Else, sample X_i from $P(X_i \mid evidence)$. Normalize weights to turn to probabilities.

Likelihood Weighting Example: P(Rain|Sprinkler = t, WetGrass =t)

Cloudy considered first, sample, w= 1.0 (because not in evidence)

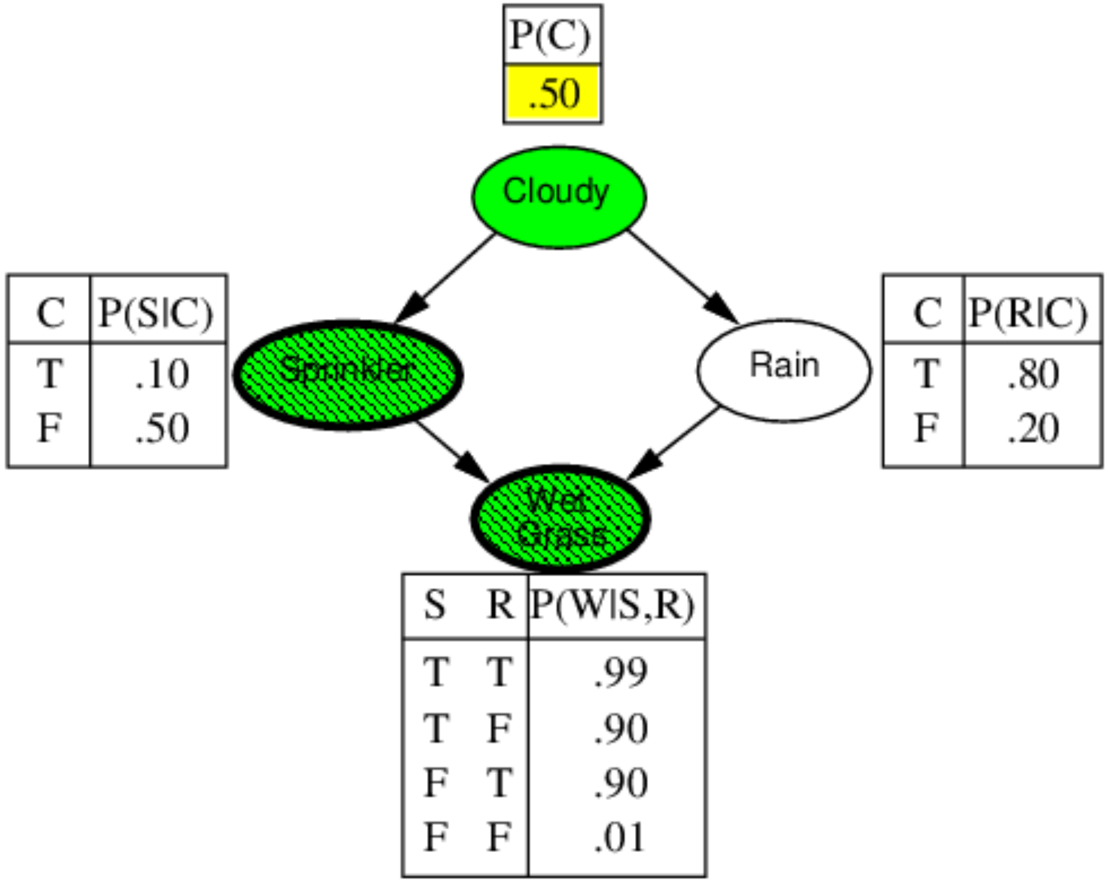
Lets assume that Cloudy = T is sampled



Importance Sampling

Cloudy considered first, sample, w= 1.0 (because not in evidence)

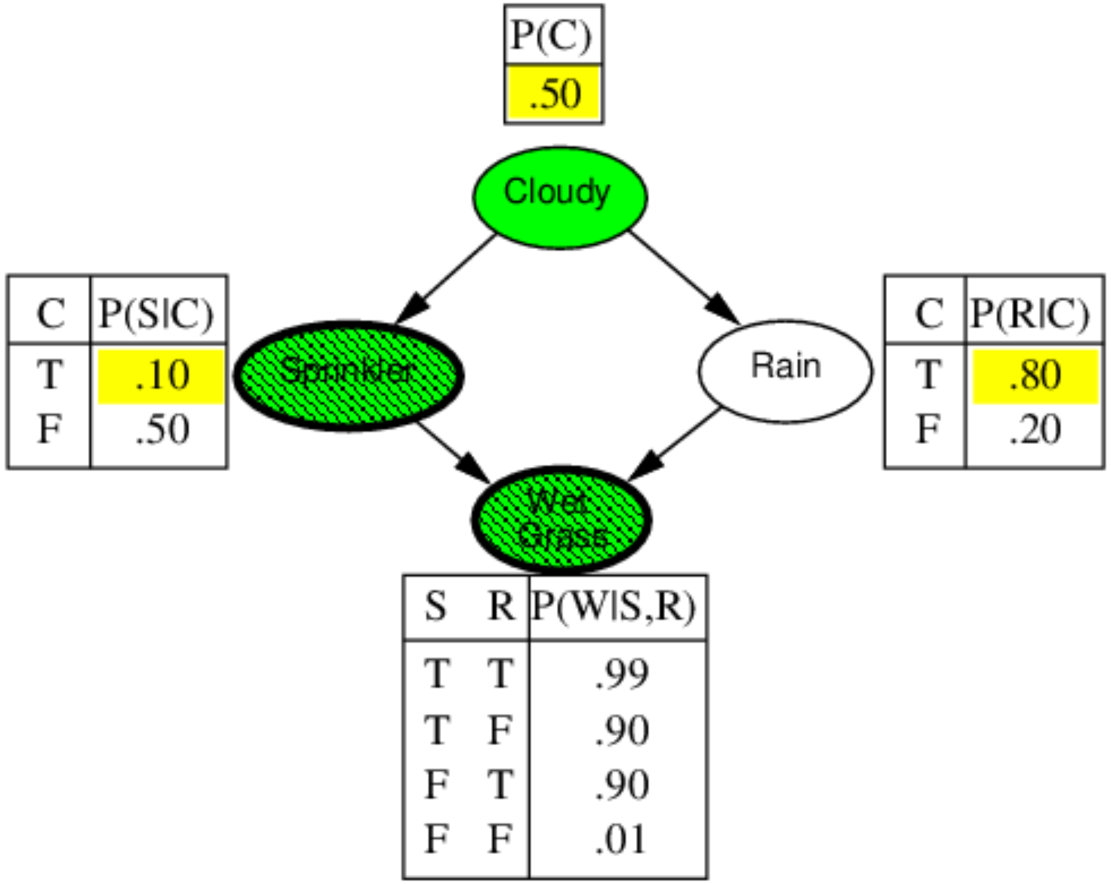
Lets assume that Cloudy = T is sampled



JAMES MADISON UNIVERSITY

Importance Sampling

Need one conditional density function for child variables given continuous parents, for each possible assignment to discrete parents.



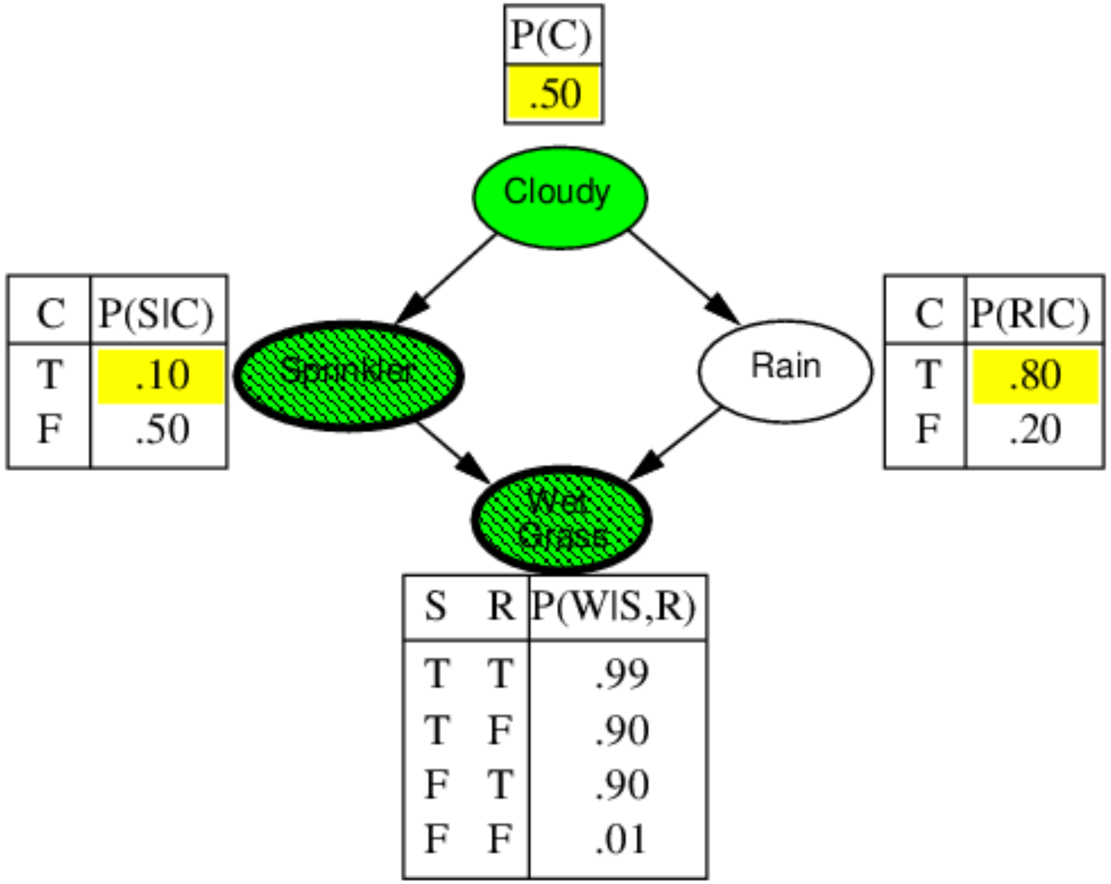
Sprinkler considered next, evidence variable, so we need to update w.

w = w × P(Sprinkler = t | Parents (Sprinkler))

w = 1.0

Importance Sampling

Need one conditional density function for child variables given continuous parents, for each possible assignment to discrete parents.



Sprinkler considered next, evidence variable, so we need to update w.

w = w × P(Sprinkler = t | Parents (Sprinkler))

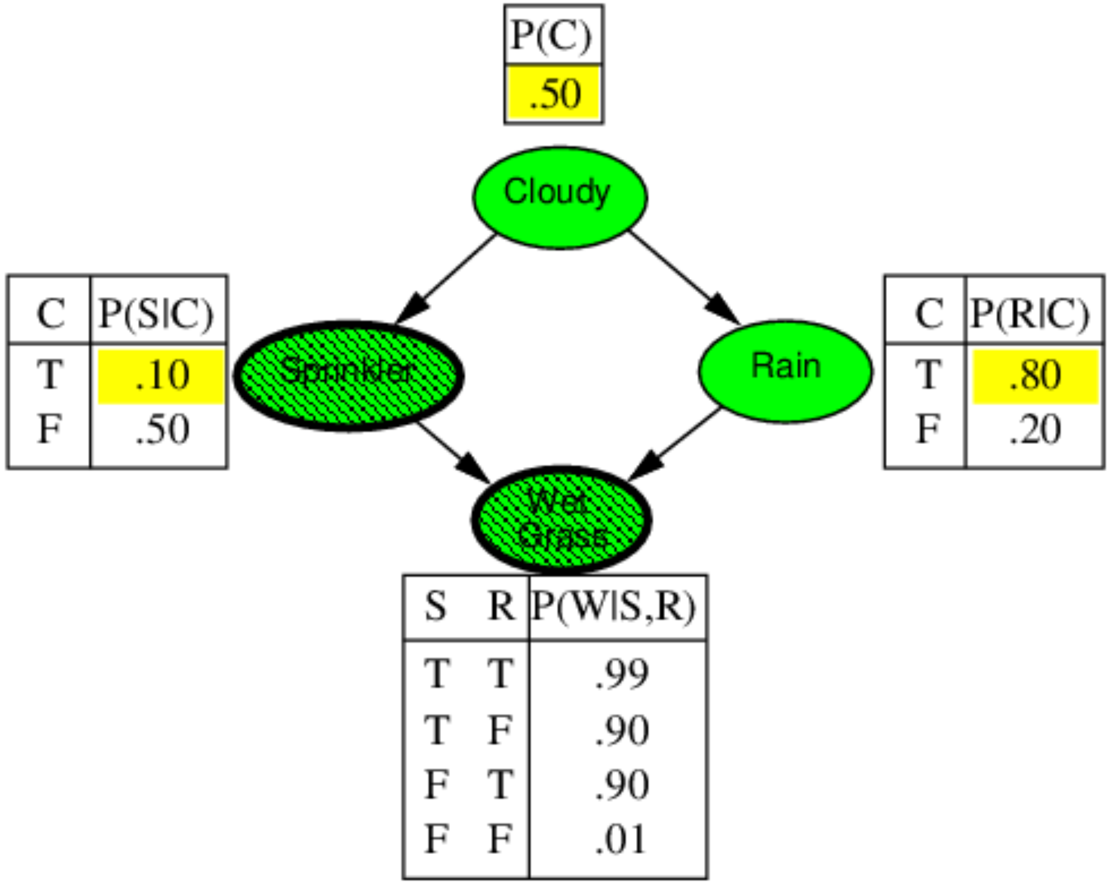
w = 1.0 × 0.1

Importance Sampling

Need one conditional density function for child variables given continuous parents, for each possible assignment to discrete parents.

Rain considered next, nonevidence, so sample from BN, w does not change.

$w = 1.0 \times 0.1$



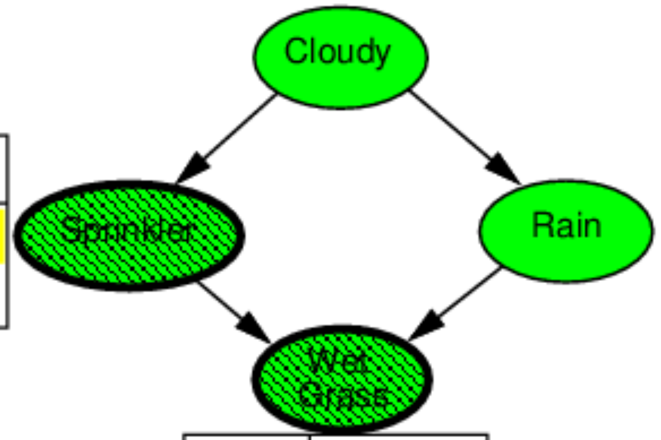
Importance Sampling

Need one conditional density function for child variables given continuous parents, for each possible assignment to discrete parents.

Sample Rain, note Cloudy = t from before

Say, Rain = t sampled

w = 1.0 × 0.1

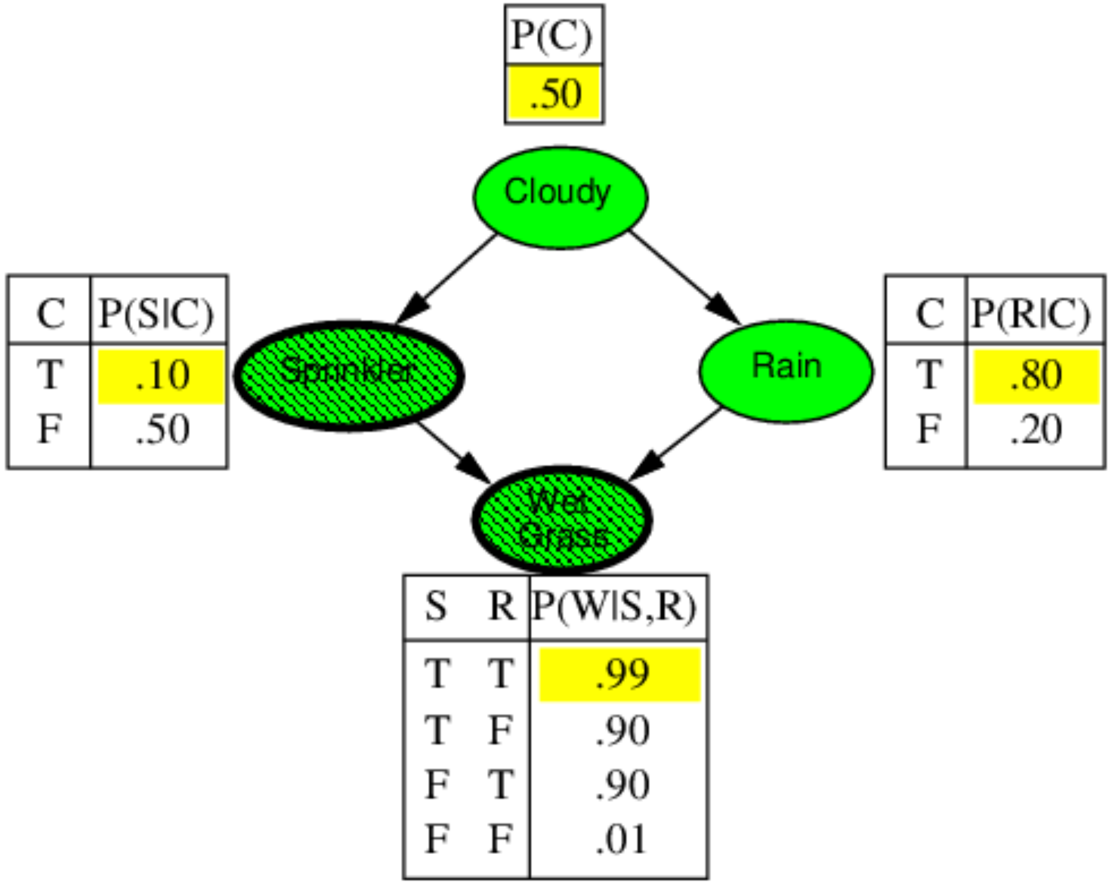


C	P(S|C)
T	.10
F	.50

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

Importance Sampling

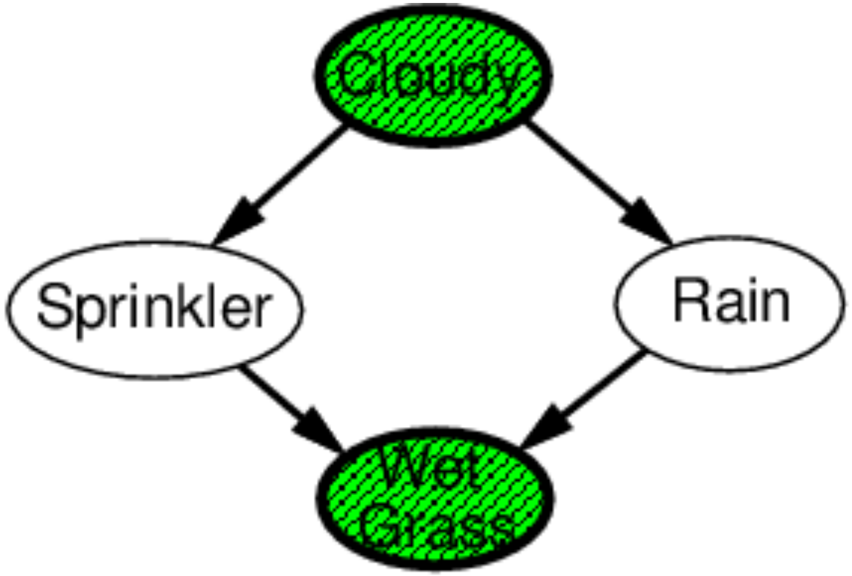


Last r.v. WetGrass, evidence variable, so update w

w = w x P(WetGrass = t| Parents(WetGrass)) = P(W = t | S = t, R = t)

w = 1.0 x 0.1 x 0.99 = 0.099 (this is NOT a probability, but the weight of this sample).

Summary of Likelihood Sampling



Sampling probability for WeightedSample is:

$$S_{ws}(z,e) = \prod_{i=1}^{I} P(z_i | parents(Z_i))$$

Note: pays attention to evidence in ancestors only

⟹somewhere "in between" prior and posterior distributions

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

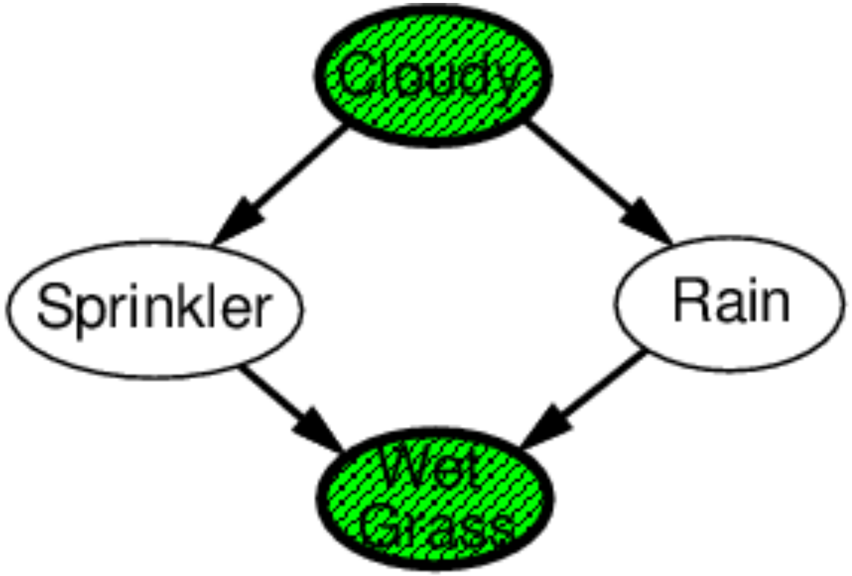
Summary of Likelihood Sampling

Sampling probability for WeightedSample is:

$$S_{ws}(z,e) = \prod_{i=1}^{I} P(z_i | parents(Z_i))$$

Note: pays attention to evidence in ancestors only

⟹somewhere "in between" prior and posterior distributions



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

Likelihood Weighting

- Likelihood weighting returns consistent estimates.
- Order actually matters
- Degradation in performance as number of evidence variables increases
- A few samples have nearly all the total weight
- Most samples will have very low weights, and weight estimate will be dominated by tiny fraction of samples that contribute little likelihood to evidence.
- Exacerbated when evidence variables occur late in the ordering
- Nonevidence variables will have no evidence in their parents to guide generation of samples

Idea: Change framework: do not directly sample (from scratch), but modify preceding sample

Approximate Inference using MCMC

Main idea:

Markov Chain Monte Carlo (MCMC) algorithm(s) generate each sample by making a random change to a preceding sample

Concept of *current state*: specifies value for every r.v.

"State" of the network = current assignment to all variables

Random change to current state yields next state

A form of MCMC: Gibbs sampling

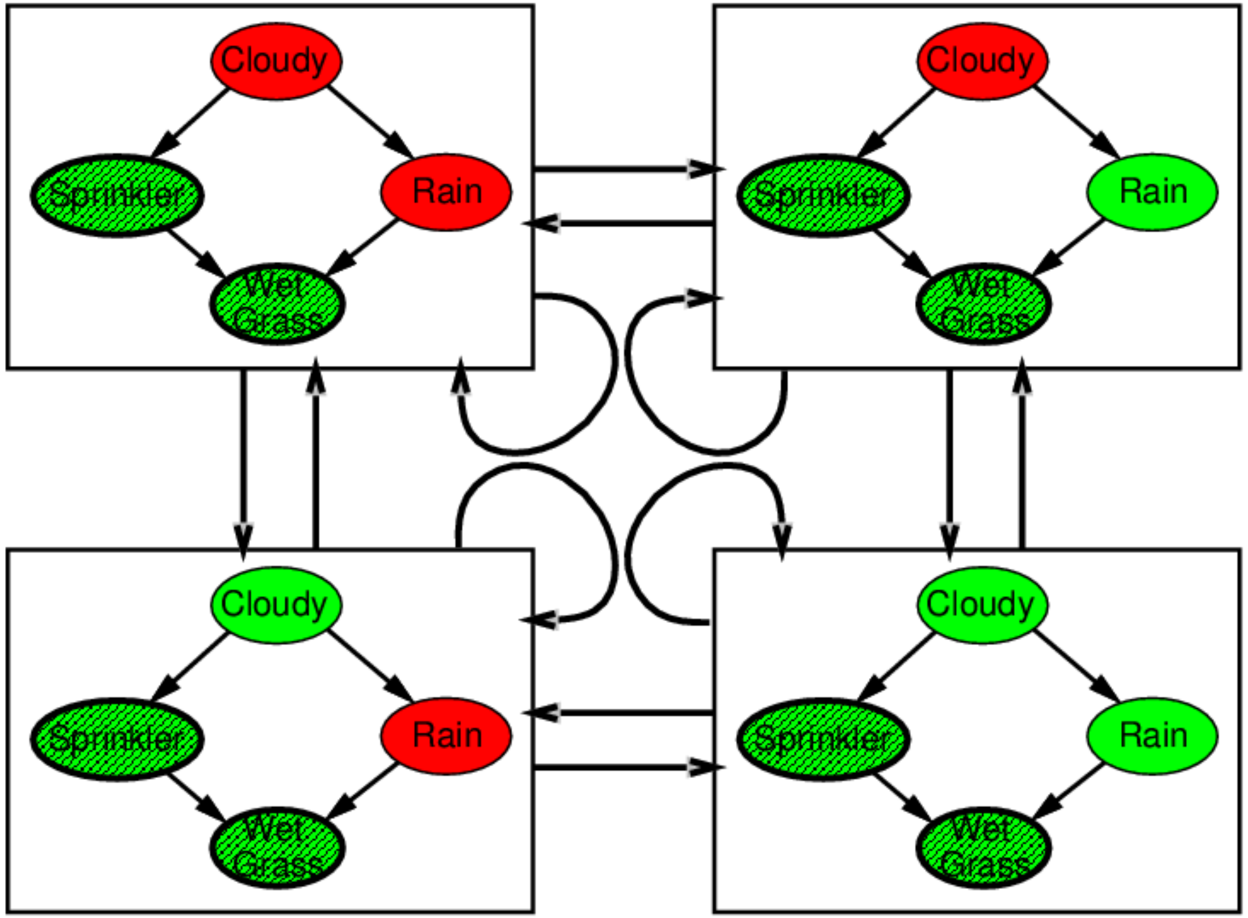
Gibbs Sampling to Estimate P(X | e)

- Initial state has evidence variables assigned as provided
- Next state generated by randomly sampling values for nonevidence variables
- Each nonevidence variable *Z* sampled in turn, given its Markov blanket (mb).

function Gibbs-Ask(X, **e**, bn, N, mb) **returns** an estimate of P(X l **e**)
Local var: *N[X],* a vector of counts over *X,* initially zero
Z, nonevidence variables in bn
X, current state of network, initially copied from ***e***
x, current state of network, initially copied from **e**
Initialize ***x*** *with random values for the variables in* ***Z***
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 l *mb*(Z_i)) given the values of MB(Z_i)
N[x] ← N[x] + 1 where x is the value of X in x
return Normalized(N)

The Markov Chain

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



Wander about for while (random walk), average what you see

MCMC Example Continued

Estimate 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

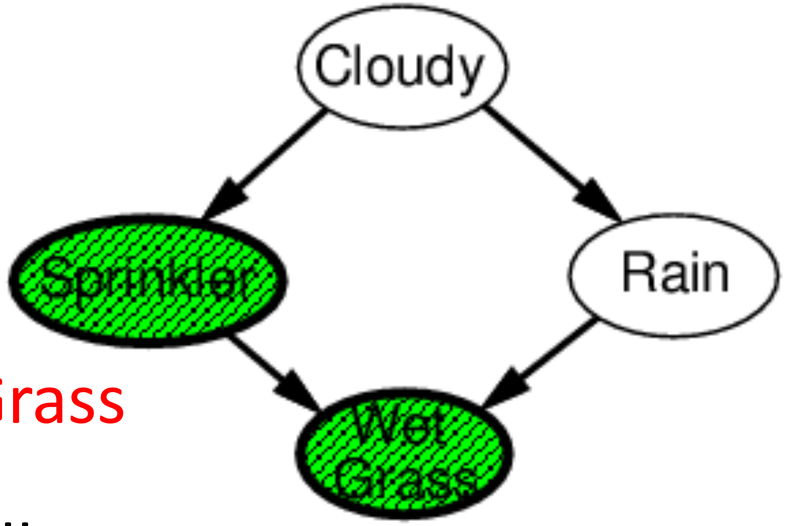
31 have Rain = true, 69 have Rain = false

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

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

MCMC Analysis

Transition probability $q(x \rightarrow x')$

Occupancy probability is $\pi_t(x)$ at time t

Equilibrium condition on πt defines stationary distribution $\pi(x)$

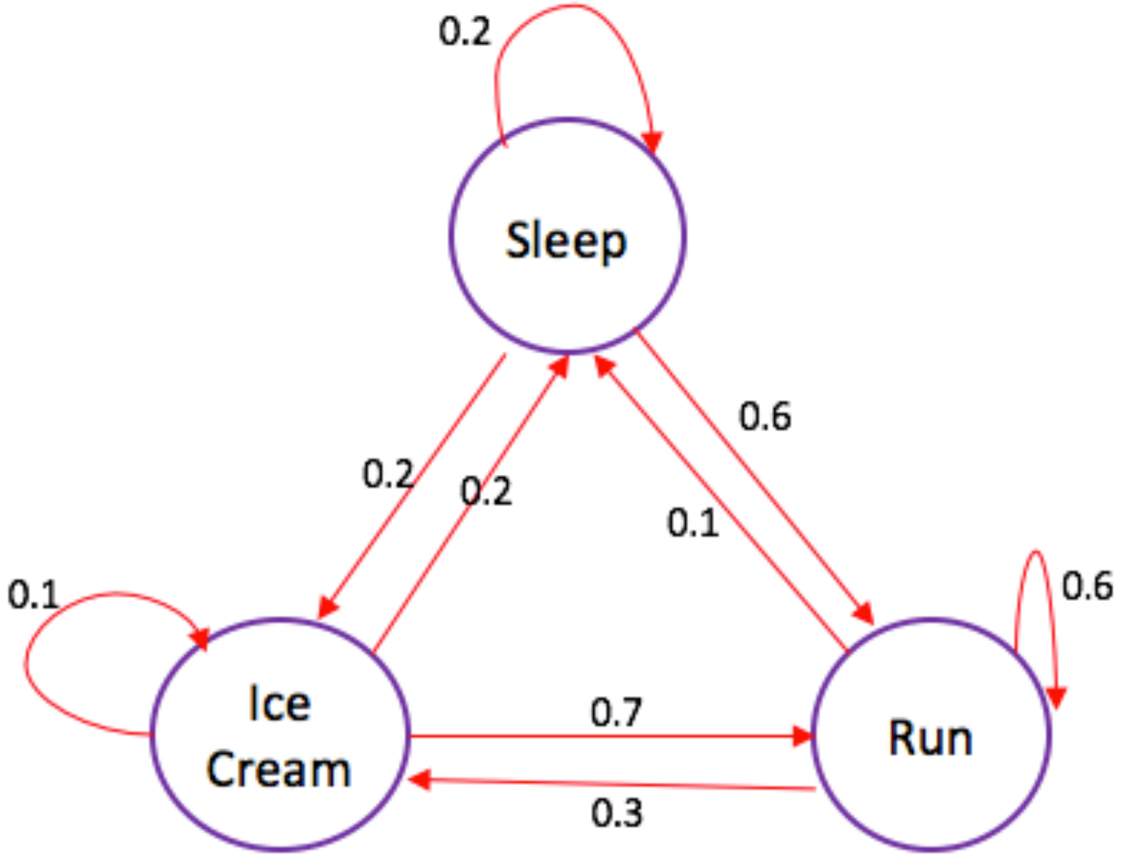
Pairwise detailed balance on states guarantees equilibrium.

.

Gibbs sampling transition probability:

Sample each variable given current values of all others

$\Longrightarrow$detailed balance with true posterior



JAMES MADISON UNIVERSITY

Summary on Inference on Bayesian Networks

Exact inference by variable elimination: good for polytrees (but NP-Hard in general)

As a result, approximate inference by LW, MCMC is common:

- LW does poorly when there is lots of downstream evidence
- LW, MCMC generally insensitive to topology
- Convergence can be very slow in some cases