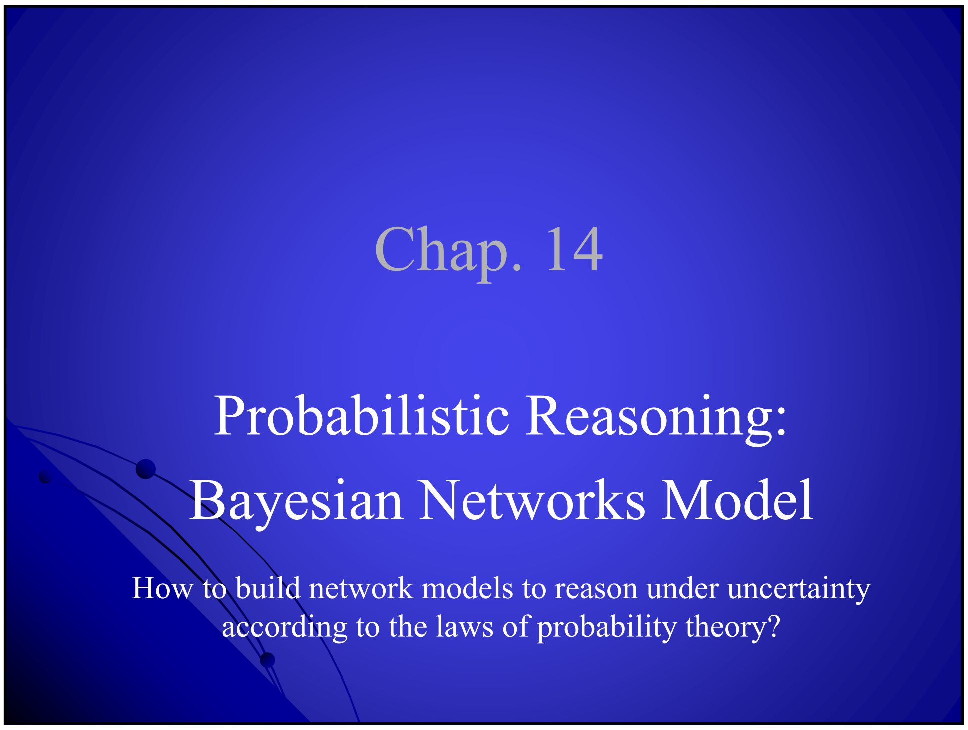
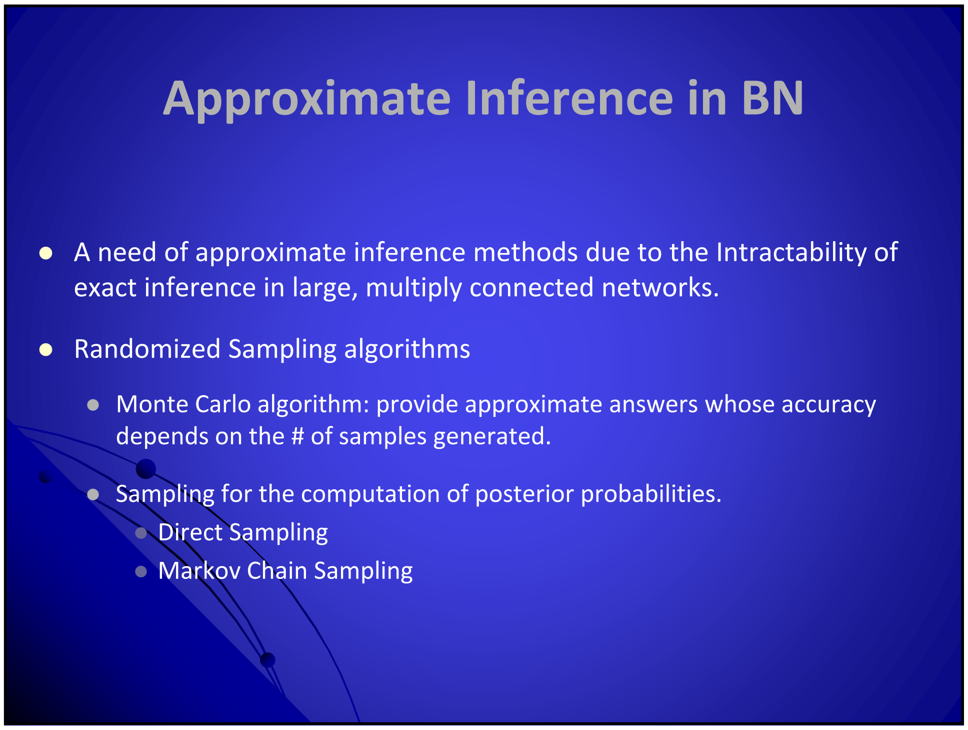


Chap. 14

Probabilistic Reasoning: Bayesian Networks Model

How to build network models to reason under uncertainty according to the laws of probability theory?



Approximate Inference in BN

- A need of approximate inference methods due to the Intractability of exact inference in large, multiply connected networks.
- Randomized Sampling algorithms
    - Monte Carlo algorithm: provide approximate answers whose accuracy depends on the # of samples generated.
    - Sampling for the computation of posterior probabilities.
        - Direct Sampling
        - Markov Chain Sampling

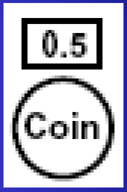
Inference by Stochastic Simulation

- Basic idea:

  1) Draw N samples from a sampling distribution S

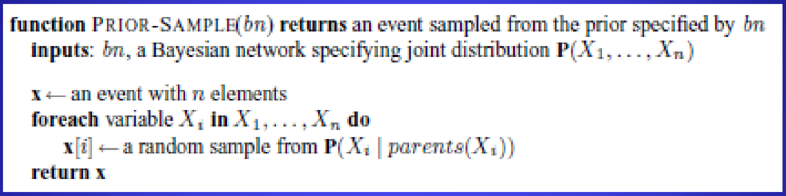
  2) Compute an approximate posterior probability

  3) Show this converges to the true probability P



- Outline:
  - Sampling from an empty network (Direct Sampling)
  - 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.

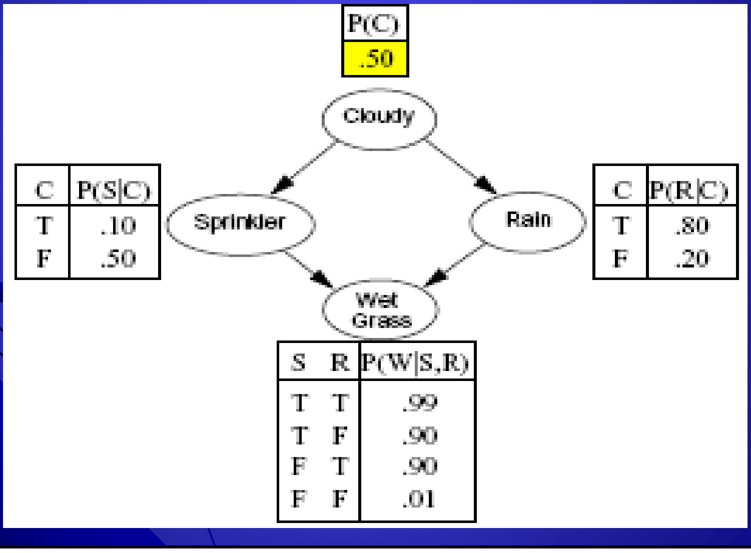
Direct Sampling method



```
function PRIOR-SAMPLE($bn$) returns an event sampled from the prior specified by $bn$
  inputs: $bn$, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $\mathbf{x} \leftarrow$ an event with $n$ elements
  foreach variable $X_i$ in $X_1, \ldots, X_n$ do
      $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return $\mathbf{x}$
```

- The simple kind of random sampling process for BN which generates events from a network with *no evidence* being associated with it.
- Sample each variable in turn, in topological order.
- The probability distribution from which the value is sampled is conditioned the values already assigned to the variable’s parents.

Example



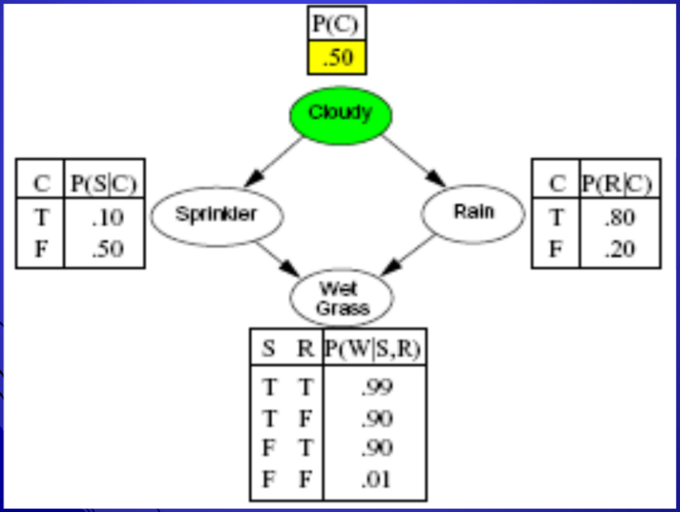
P(C)
.50

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

cont.



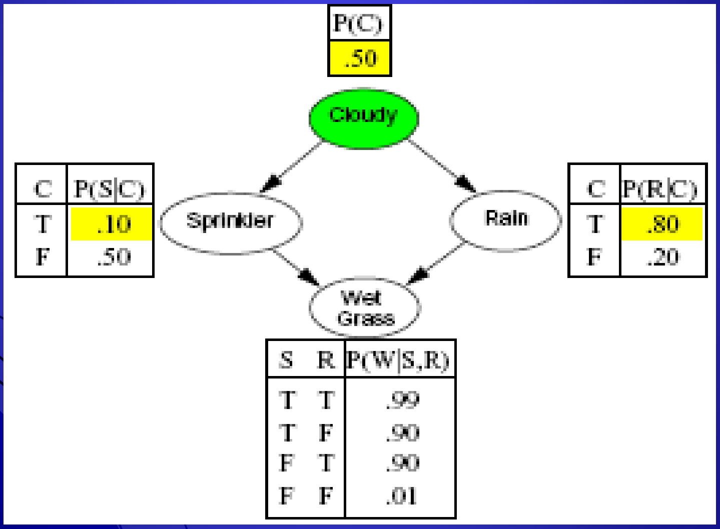
P(C)
.50

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

cont.



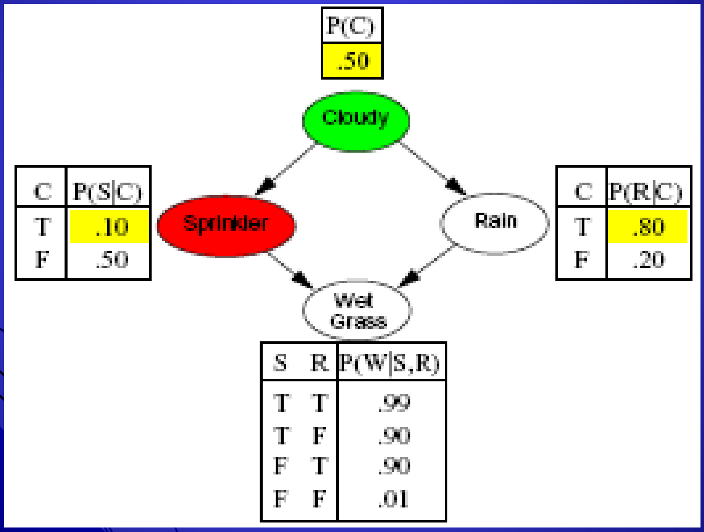
P(C)
.50

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

cont.



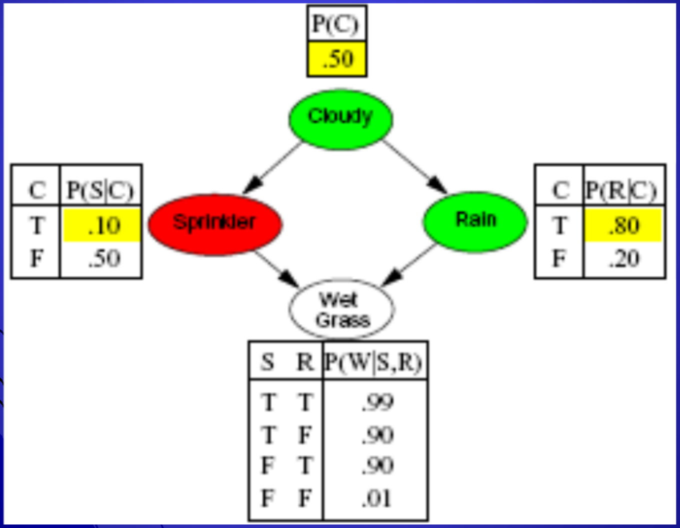
P(C)
.50

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

cont.



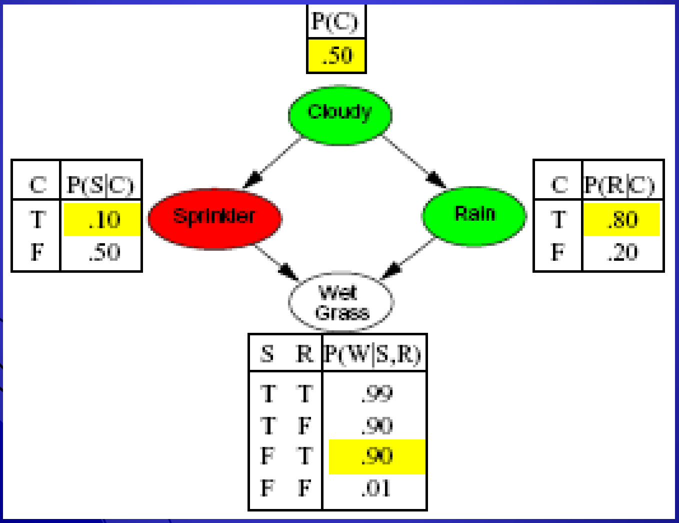
P(C)
.50

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

cont.



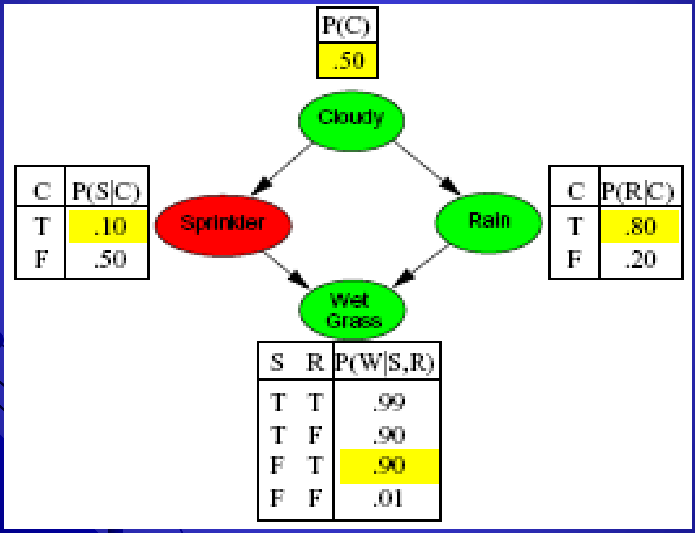
P(C)
.50

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

cont.



Returns the event [true, false, true, true] for [Cloudy, Sprinkler, Rain, WetGrass].

Direct Sampling method

- *Sampling Probability* that PRIOR-SAMPLE generates a particular event

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

  i.e. the true prior joint probability
- E.g.) $S_{PS}(t, f, t, t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324 = P(t, f, t, t)$

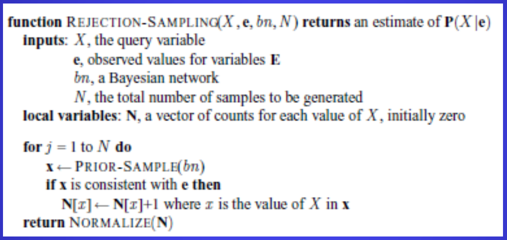
  - 32.4% of the samples to be of event (Cloudy=t, Sprinkler=f, Rain=t, Wet-Grass=t)
- Let $N_{PS}(x_1, .., x_n)$ be the frequency of samples generated for event $x_1, .., x_n$.
- Then we have

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

  That is, estimates derived from PRIOR-SAMPLE are consistent.
- Shorthand: $\hat{P}(x_1,\ldots,x_n) \approx P(x_1,\ldots,x_n)$

Rejection sampling

- A general method for producing samples from a hard-to-sample distribution given an easy-to-sample distribution.
- $\hat{P}(X|e)$ estimated from samples agreeing with e



```
function REJECTION-SAMPLING($X$, e, $bn$, $N$) returns an estimate of P($X$|e)
  inputs: $X$, the query variable
          e, observed values for variables E
          $bn$, a Bayesian network
          $N$, the total number of samples to be generated
  local variables: N, a vector of counts for each value of $X$, 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)
```

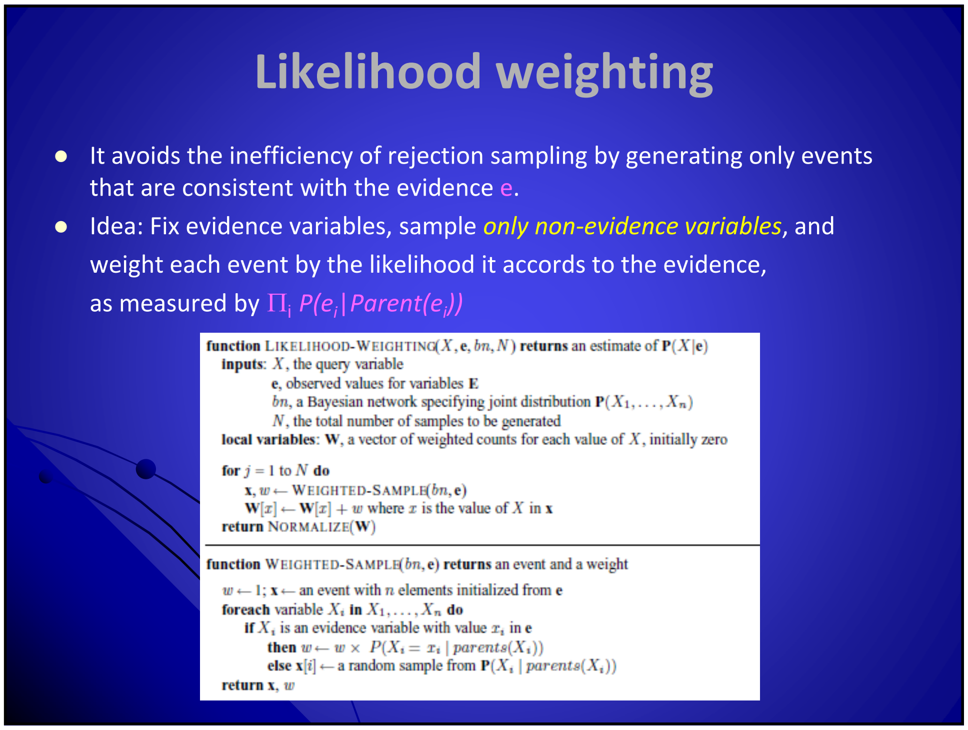
- E.g.) estimate *P (Rain | Sprinkler=true)* using 100 samples
  27 samples have *Sprinkler=true* (73 samples with *Sprinkler=false* are rejected)
  8 have *Rain=true* and 19 have *Rain=false.*

$$\hat{P}(Rain|Sprinkler = true) = \text{NORMALIZE}\,(< 8, 19 >) =< 0.296, 0.704)$$

  Cf) *P(Rain | Sprinkler=true) = 0.3*
- Similar to a basic real-world empirical estimation procedure

Analysis of rejection sampling

- $\hat{P}(X|e) = \alpha\, N_{PS}(X, e)$ (algorithm def.)
  $= N_{PS}(X, e) / N_{PS}(e)$ (normalized by $N_{PS}(e)$)
  $\approx P(X, e) / P(e)$ (property of PRIORSAMPLE)
  $= P(X | e)$ (def. of conditional probability)
- Hence rejection sampling returns consistent posterior estimates.
- Problem: hopelessly expensive if $P(e)$ is small
  - i.e. rejecting too many samples.
  - $P(e)$ drops off exponentially with a number of evidence variable
    -- the procedure is unusable for complex problem.

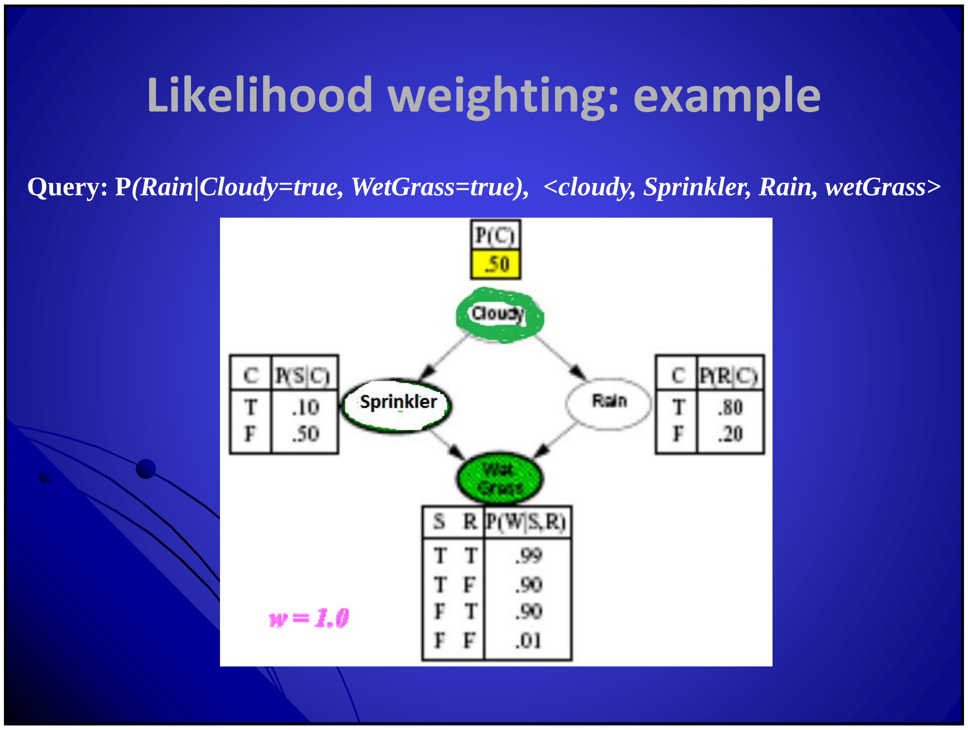


Likelihood weighting

- It avoids the inefficiency of rejection sampling by generating only events that are consistent with the evidence e.
- Idea: Fix evidence variables, sample *only non-evidence variables*, and weight each event by the likelihood it accords to the evidence, as measured by Π_i *$P(e_i|Parent(e_i))$*

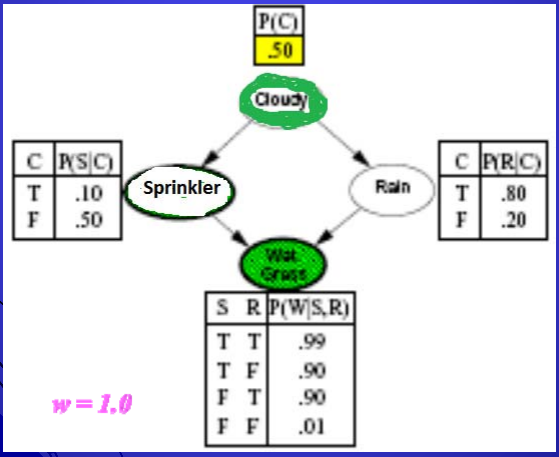
```
function LIKELIHOOD-WEIGHTING($X$, $\mathbf{e}$, $bn$, $N$) returns an estimate of $\mathbf{P}(X|\mathbf{e})$
  inputs: $X$, the query variable
          $\mathbf{e}$, observed values for variables $\mathbf{E}$
          $bn$, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$
          $N$, the total number of samples to be generated
  local variables: $\mathbf{W}$, a vector of weighted counts for each value of $X$, initially zero

  for $j = 1$ to $N$ do
     $\mathbf{x}, w \leftarrow$ WEIGHTED-SAMPLE($bn$, $\mathbf{e}$)
     $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where $x$ is the value of $X$ in $\mathbf{x}$
  return NORMALIZE($\mathbf{W}$)
```

```
function WEIGHTED-SAMPLE($bn$, $\mathbf{e}$) returns an event and a weight

  $w \leftarrow 1$; $\mathbf{x} \leftarrow$ an event with $n$ elements initialized from $\mathbf{e}$
  foreach variable $X_i$ in $X_1, \ldots, X_n$ do
     if $X_i$ is an evidence variable with value $x_i$ in $\mathbf{e}$
        then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
        else $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return $\mathbf{x}$, $w$
```



Likelihood weighting: example

Query: P*(Rain|Cloudy=true, WetGrass=true), <cloudy, Sprinkler, Rain, wetGrass>*



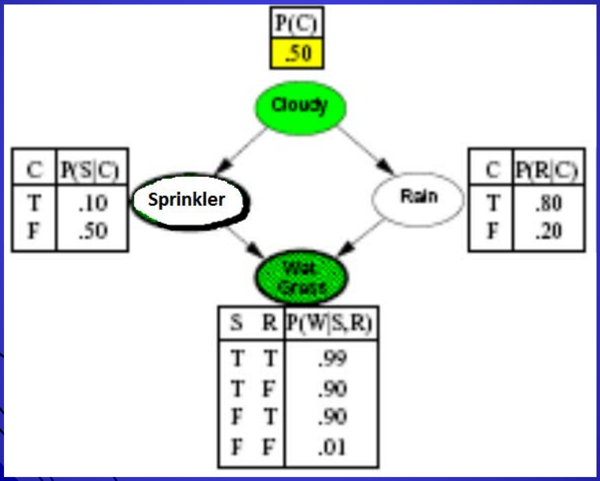
P(C)
.50

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

cont.



P(C)
.50

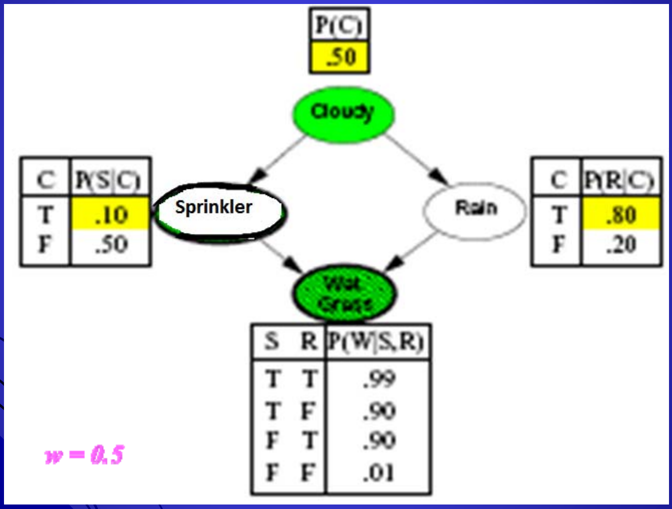
C	P(S|C)
T	.10
F	.50

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×P(cloudy) = 0.5 -- [Cloudy = true] is an evidence.

cont.



P(C)
.50

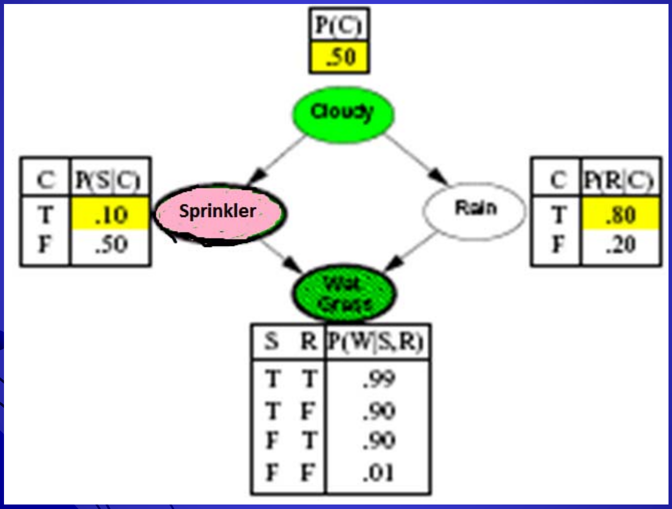
C	P(S|C)
T	.10
F	.50

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 = 0.5

cont.



P(C)
.50

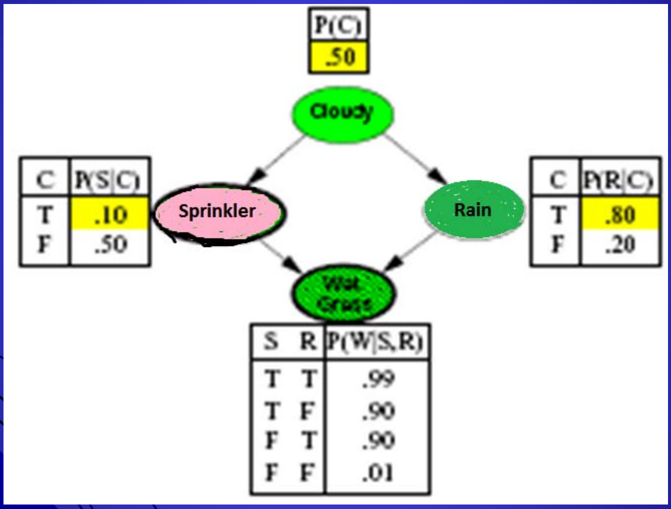
C	P(S|C)
T	.10
F	.50

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

[Sprinkler = false]

cont.



P(C)
.50

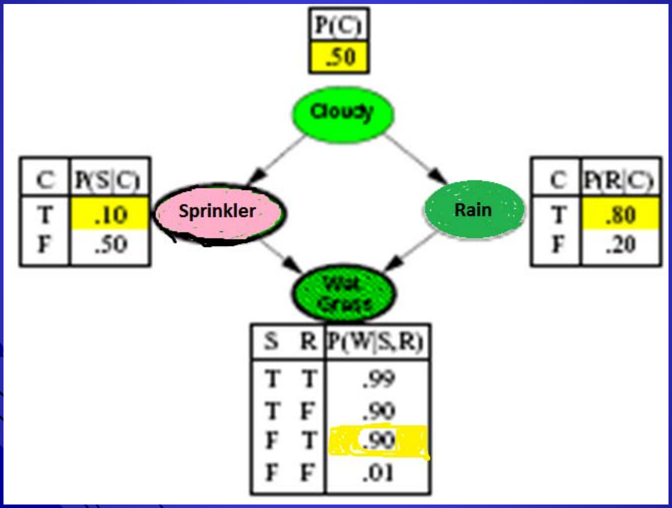
C	P(S|C)
T	.10
F	.50

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

[Rain = true]

cont.



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

$w = w \times P(WetGrass{=}true \mid \neg sprinkler, rain) = 0.5 \times 0.9 = 0.45$

- A weight for a sample [cloudy, ¬sprinkler, rain, wetgrass]

WEIGHTED-SAMPLE returns the event [*true*, false, *true*, true] with weight=0.45.

Likelihood weighting: analysis

- *Sampling probability* for WEIGHTED-SAMPLE is

  $S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1..l} P(z_i \mid parents(Z_i))$

  where z is the non-evidence var.+Query var.
- Note: pays attention to evidence in ancestors only

  $\Rightarrow \quad P(\mathbf{z}) \le S_{WS}(\mathbf{z}, \mathbf{e}) \le P(\mathbf{z}|\mathbf{e})$

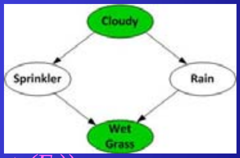
  somewhere "in between" prior and posterior distribution
- *Weight* for a given sample x composed from z, e is

  $w(\mathbf{z}, \mathbf{e}) = \prod_{i=1..m} P(e_i \mid parents(E_i))$
- *Weighted Sampling Probability* is

  $S_{WS}(\mathbf{z}, \mathbf{e}) \cdot w(\mathbf{z}, \mathbf{e})$

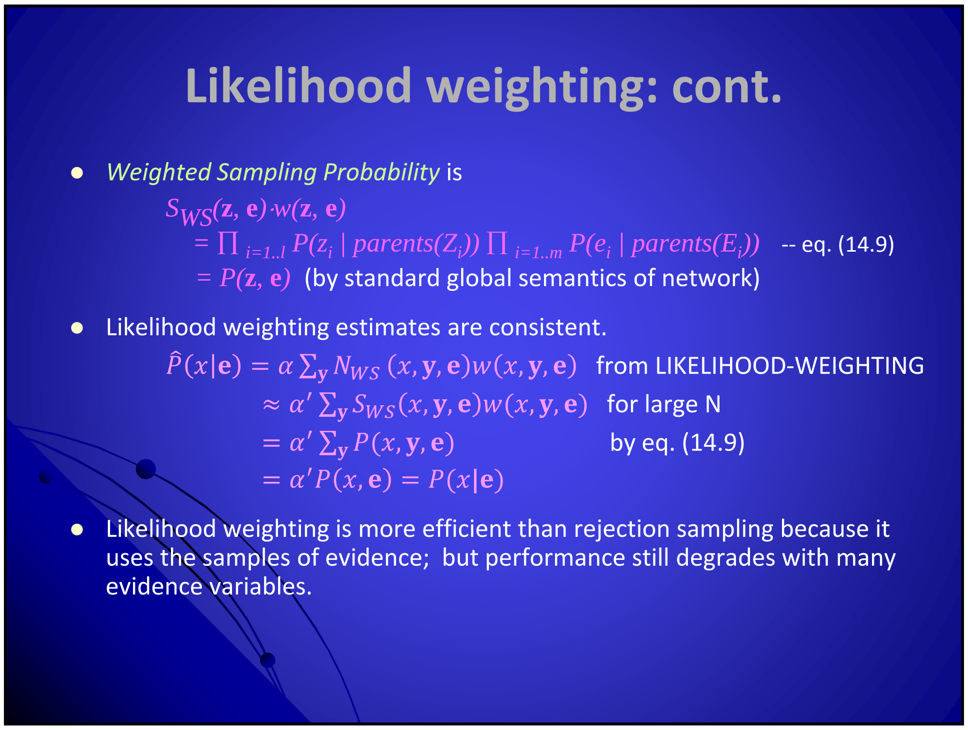
  $= \prod_{i=1..l} P(z_i \mid parents(Z_i)) \prod_{i=1..m} P(e_i \mid parents(E_i))$

  $= P(\mathbf{z}, \mathbf{e})$ (by standard global semantics of network)



- Likelihood weighting also returns consistent estimates and more efficient than rejection sampling because it uses all the samples generated;

  but performance still degrades with many evidence variables because a few samples have nearly all the total weight.



Likelihood weighting: cont.

- *Weighted Sampling Probability* is

  $S_{WS}(\mathbf{z}, \mathbf{e}) \cdot w(\mathbf{z}, \mathbf{e})$

  $= \prod_{i=1..l} P(z_i \mid parents(Z_i)) \prod_{i=1..m} P(e_i \mid parents(E_i))$ -- eq. (14.9)

  $= P(\mathbf{z}, \mathbf{e})$ (by standard global semantics of network)

- Likelihood weighting estimates are consistent.

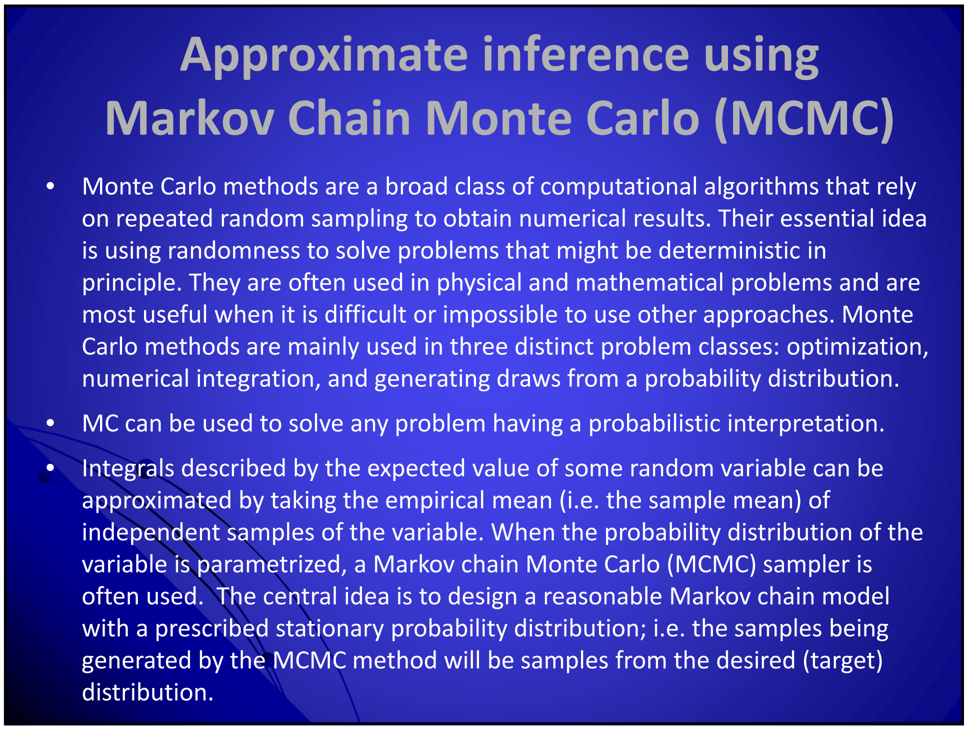
  $\hat{P}(x|\mathbf{e}) = \alpha \sum_{\mathbf{y}} N_{WS}(x, \mathbf{y}, \mathbf{e}) w(x, \mathbf{y}, \mathbf{e})$ from LIKELIHOOD-WEIGHTING

  $\approx \alpha' \sum_{\mathbf{y}} S_{WS}(x, \mathbf{y}, \mathbf{e}) w(x, \mathbf{y}, \mathbf{e})$ for large N

  $= \alpha' \sum_{\mathbf{y}} P(x, \mathbf{y}, \mathbf{e})$ by eq. (14.9)

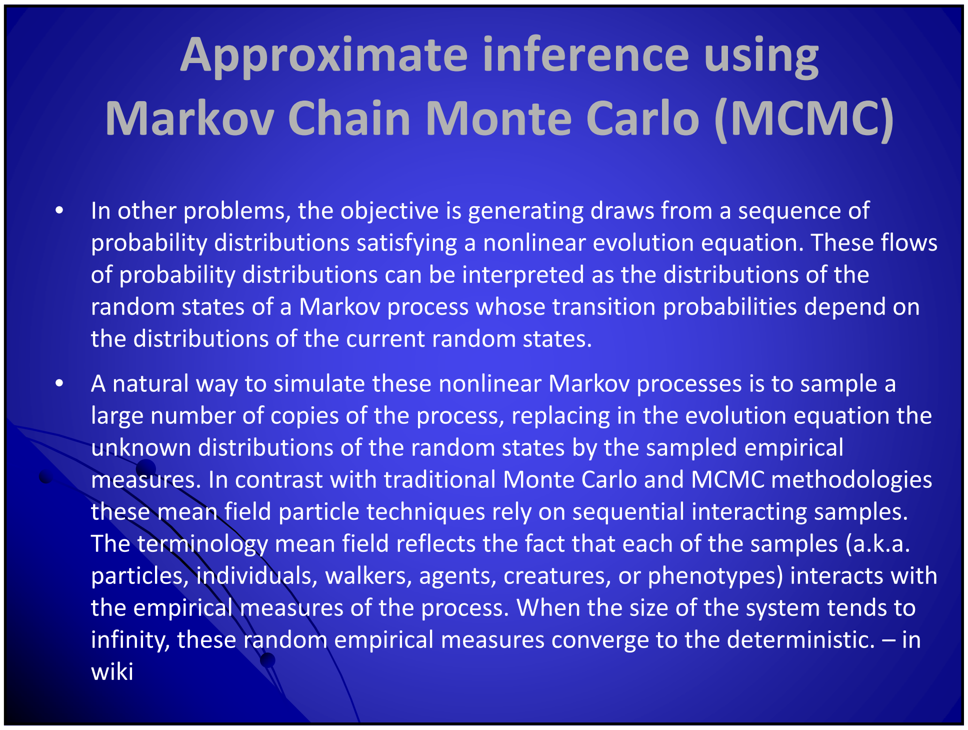
  $= \alpha' P(x, \mathbf{e}) = P(x|\mathbf{e})$

- Likelihood weighting is more efficient than rejection sampling because it uses the samples of evidence; but performance still degrades with many evidence variables.



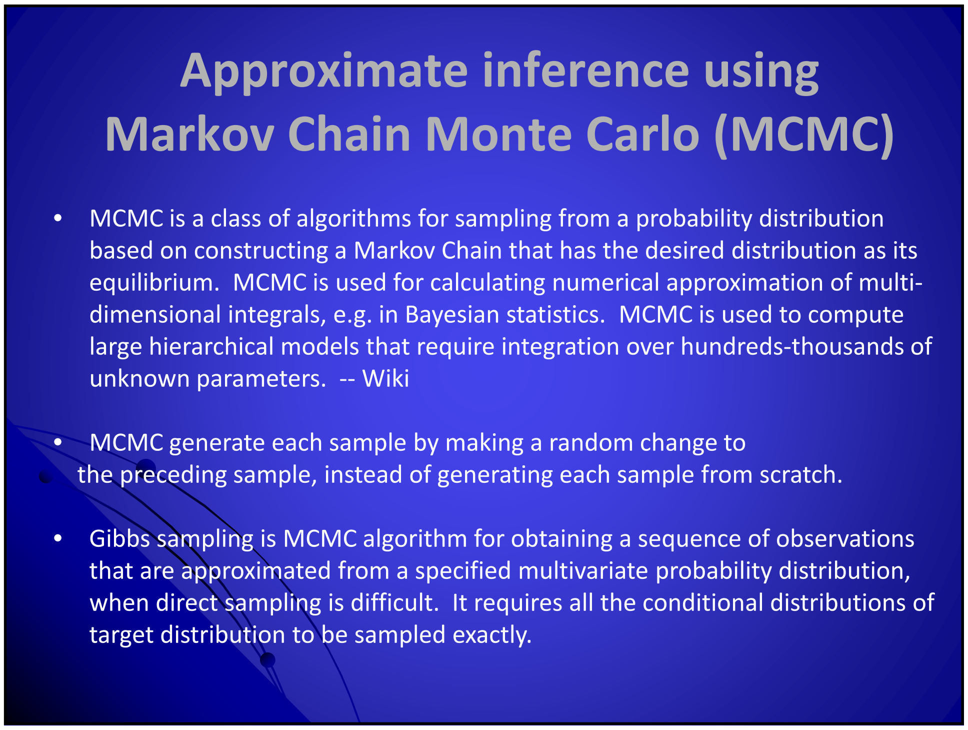
Approximate inference using Markov Chain Monte Carlo (MCMC)

- Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. Their essential idea is using randomness to solve problems that might be deterministic in principle. They are often used in physical and mathematical problems and are most useful when it is difficult or impossible to use other approaches. Monte Carlo methods are mainly used in three distinct problem classes: optimization, numerical integration, and generating draws from a probability distribution.
- MC can be used to solve any problem having a probabilistic interpretation.
- Integrals described by the expected value of some random variable can be approximated by taking the empirical mean (i.e. the sample mean) of independent samples of the variable. When the probability distribution of the variable is parametrized, a Markov chain Monte Carlo (MCMC) sampler is often used. The central idea is to design a reasonable Markov chain model with a prescribed stationary probability distribution; i.e. the samples being generated by the MCMC method will be samples from the desired (target) distribution.



Approximate inference using Markov Chain Monte Carlo (MCMC)

- In other problems, the objective is generating draws from a sequence of probability distributions satisfying a nonlinear evolution equation. These flows of probability distributions can be interpreted as the distributions of the random states of a Markov process whose transition probabilities depend on the distributions of the current random states.
- A natural way to simulate these nonlinear Markov processes is to sample a large number of copies of the process, replacing in the evolution equation the unknown distributions of the random states by the sampled empirical measures. In contrast with traditional Monte Carlo and MCMC methodologies these mean field particle techniques rely on sequential interacting samples. The terminology mean field reflects the fact that each of the samples (a.k.a. particles, individuals, walkers, agents, creatures, or phenotypes) interacts with the empirical measures of the process. When the size of the system tends to infinity, these random empirical measures converge to the deterministic. – in wiki

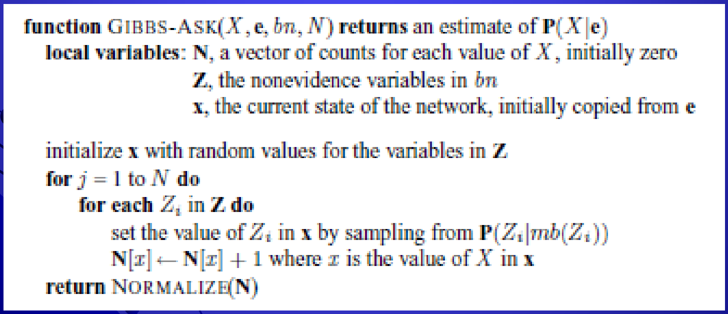


Approximate inference using Markov Chain Monte Carlo (MCMC)

- MCMC is a class of algorithms for sampling from a probability distribution based on constructing a Markov Chain that has the desired distribution as its equilibrium. MCMC is used for calculating numerical approximation of multi-dimensional integrals, e.g. in Bayesian statistics. MCMC is used to compute large hierarchical models that require integration over hundreds-thousands of unknown parameters. -- Wiki
- MCMC generate each sample by making a random change to
the preceding sample, instead of generating each sample from scratch.
- Gibbs sampling is MCMC algorithm for obtaining a sequence of observations that are approximated from a specified multivariate probability distribution, when direct sampling is difficult. It requires all the conditional distributions of target distribution to be sampled exactly.

Approximate inference using Markov Chain Monte Carlo (MCMC)

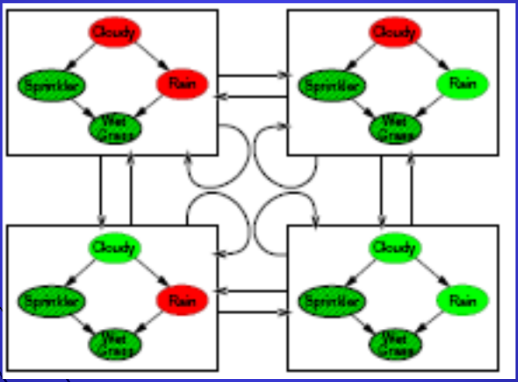
- Gibbs Sampling in BN:
  Start with an arbitrary state (with a fixed evidence) and generate a next state by randomly sampling a value for one of the non-evidence variables X_i. The sampling for X_i is done conditioned on the current values of the variables in MB(X_i)



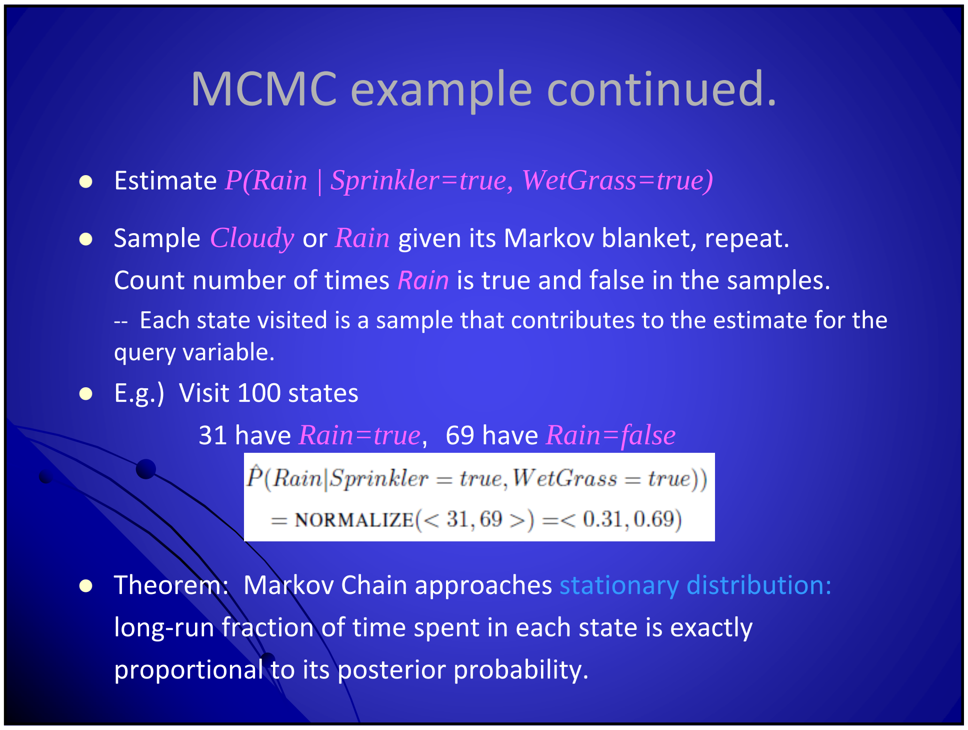
```
function GIBBS-ASK($X$, e, $bn$, $N$) returns an estimate of $\mathbf{P}(X|\mathbf{e})$
  local variables: N, a vector of counts for each value of $X$, initially zero
                   Z, the nonevidence variables in $bn$
                   x, the current state of the network, initially copied from e

  initialize x with random values for the variables in Z
  for $j = 1$ to $N$ do
      for each $Z_i$ in Z do
          set the value of $Z_i$ in x by sampling from $\mathbf{P}(Z_i|mb(Z_i))$
          $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where $x$ is the value of $X$ in x
  return NORMALIZE(N)
```

The Markov chain

- Initial state = [true, *true*, false, *true*].
- With *Sprinkler=true, WetGrass=true*, there are four states:



- Wander about for a while, average what you see.
- The transition probability $q(x \rightarrow x')$ is called a Markov Chain on the state space.

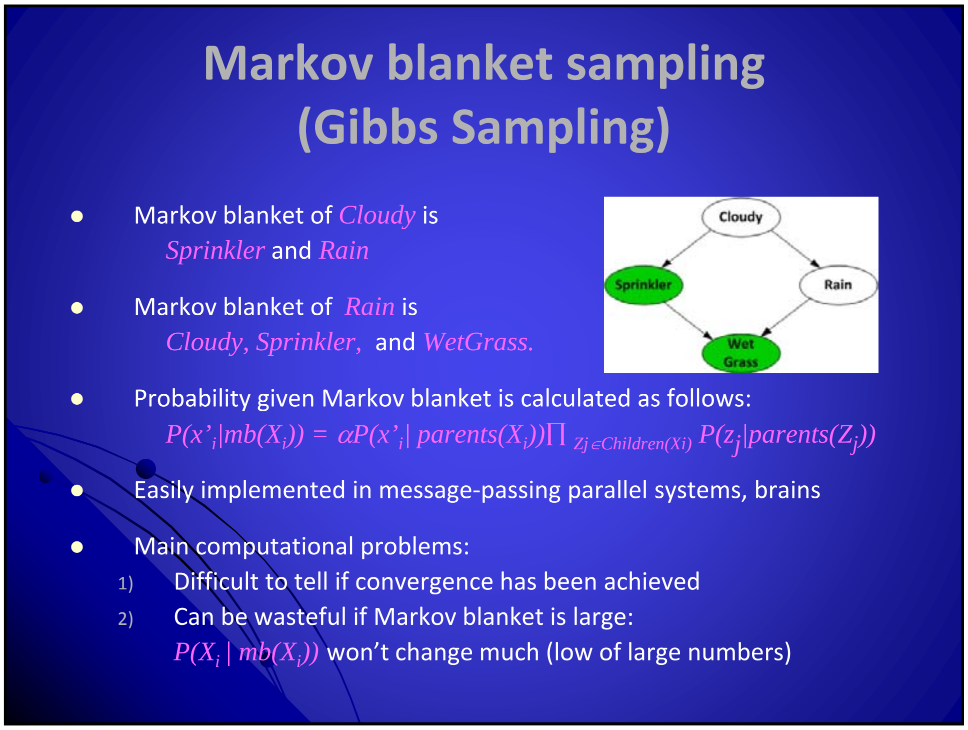


MCMC example continued.

- 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.
  -- Each state visited is a sample that contributes to the estimate for the query variable.
- E.g.) Visit 100 states
  31 have *Rain=true*, 69 have *Rain=false*

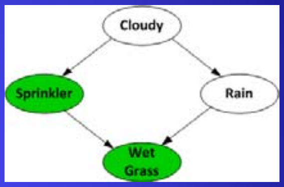
$$\hat{P}(Rain|Sprinkler = true, WetGrass = true))$$
$$= \text{NORMALIZE}(<31, 69>) =< 0.31, 0.69)$$

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



Markov blanket sampling (Gibbs Sampling)

- Markov blanket of *Cloudy* is
  Sprinkler and *Rain*
- Markov blanket of *Rain* is
  Cloudy, Sprinkler, and *WetGrass.*



- Probability given Markov blanket is calculated as follows:
  $P(x'_i|mb(X_i)) = \alpha P(x'_i|\, parents(X_i))\prod_{Z_j \in Children(X_i)} P(z_j|parents(Z_j))$
- 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:
     $P(X_i \mid mb(X_i))$ won't change much (low of large numbers)

Why Gibbs Sampling works?

- Gibbs sampling returns consistent estimates for posterior probability. Why?
- The sampling process settles into a dynamic equilibrium in which the long-run fraction of time spent in each state is exactly proportional to its posterior probability. – a transition probability is defined by the conditional distribution given MB of the sampled variable.
- $q(\mathrm{x} \rightarrow \mathrm{x}')$: transition probability that defines a Markov chain.
- $\pi_t(\mathrm{x})$, $\pi_{t+1}(\mathrm{x}')$: the probability that the system is in state x and x′, at time *t* and t+1, respectively, after running M.C. for *t*-steps.
- Given $\pi_t(\mathrm{x})$, $\pi_{t+1}(\mathrm{x}') = \sum_{\mathbf{x}} \pi_t(\mathrm{x}) \cdot q(\mathrm{x} \rightarrow \mathrm{x}')$.
- Markov chain has reached its stationary distribution if $\pi_t = \pi_{t+1}$.
  - $\pi(\mathrm{x}') = \sum_{\mathbf{x}} \pi(\mathrm{x}) \cdot q(\mathrm{x} \rightarrow \mathrm{x}')$ for all x′ .

the *expected outflow* from each state = the *expected inflow* from all the states.

← when *detailed balance* with $\pi(\mathrm{x})$ if $\pi(\mathrm{x}) \cdot q(\mathrm{x} \rightarrow \mathrm{x}') = \pi(\mathrm{x}') \cdot q(\mathrm{x}' \rightarrow \mathrm{x})$, $\forall \mathrm{x}, \mathrm{x}'$.

- Detailed balance implies stationarity

$$\sum_{\mathbf{x}} \pi(\mathrm{x}) \cdot q(\mathrm{x} \rightarrow \mathrm{x}') = \sum_{\mathbf{x}} \pi(\mathrm{x}') \cdot q(\mathrm{x}' \rightarrow \mathrm{x}) = \pi(\mathrm{x}') \cdot \sum_{\mathbf{x}} q(\mathrm{x}' \rightarrow \mathrm{x}) = \pi(\mathrm{x}').$$

- Transition probability, $q(\mathrm{x} \rightarrow \mathrm{x}')$, defined by GIBBS-ASK is a special case of Gibbs sampling.

Cont: Why Gibbs Sampling works?

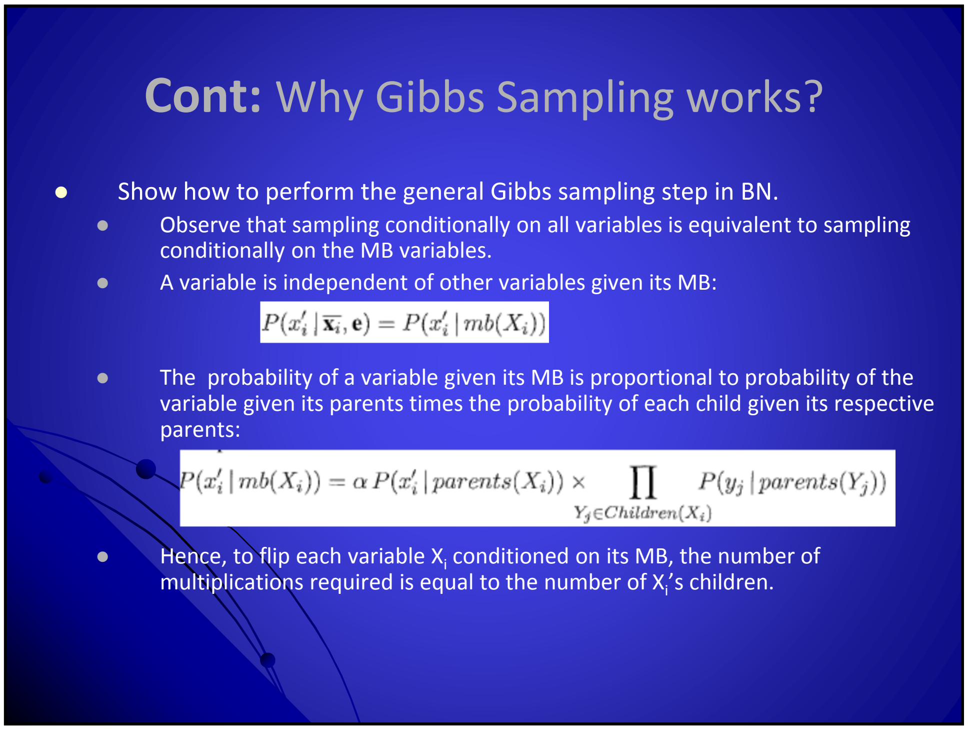
- Transition probability, $q(\mathrm{x} \rightarrow \mathrm{x}')$, defined by GIBBS-ASK is a special case of Gibbs sampling.
- Show that general Gibbs sampling satisfies the detailed balance eq. with a stationary distribution equal to P(x|e).
  - As general Gibbs sampler samples each X_i in turn with q_i, define $\overline{X_i}$ be other variables except e. If we sample a new value x_i for X_i conditionally on all other variables (including e),

$$q_i(\mathbf{x} \rightarrow \mathbf{x}') = q_i\big((x_i, \overline{\mathbf{x}_i}) \rightarrow (x_i', \overline{\mathbf{x}_i})\big) = P(x_i' | \overline{\mathbf{x}_i}, \mathbf{e})$$

  - Transition probability for each step of Gibbs sampler is in detailed balance with the true posterior:

$$\begin{aligned}
\pi(\mathbf{x}) q_i(\mathbf{x} \rightarrow \mathbf{x}') &= P(\mathbf{x} \,|\, \mathbf{e}) P(x_i' \,|\, \overline{\mathbf{x}_i}, \mathbf{e}) = P(x_i, \overline{\mathbf{x}_i} \,|\, \mathbf{e}) P(x_i' \,|\, \overline{\mathbf{x}_i}, \mathbf{e}) \\
&= P(x_i \,|\, \overline{\mathbf{x}_i}, \mathbf{e}) P(\overline{\mathbf{x}_i} \,|\, \mathbf{e}) P(x_i' \,|\, \overline{\mathbf{x}_i}, \mathbf{e}) \quad \text{(using the chain rule on the first term)} \\
&= P(x_i \,|\, \overline{\mathbf{x}_i}, \mathbf{e}) P(x_i', \overline{\mathbf{x}_i} \,|\, \mathbf{e}) \quad \text{(using the chain rule backward)} \\
&= \pi(\mathbf{x}') q_i(\mathbf{x}' \rightarrow \mathbf{x}) \,.
\end{aligned}$$

  - The samples generated by Gibbs sampling will eventually be drawn from the true posterior distribution.



Cont: Why Gibbs Sampling works?

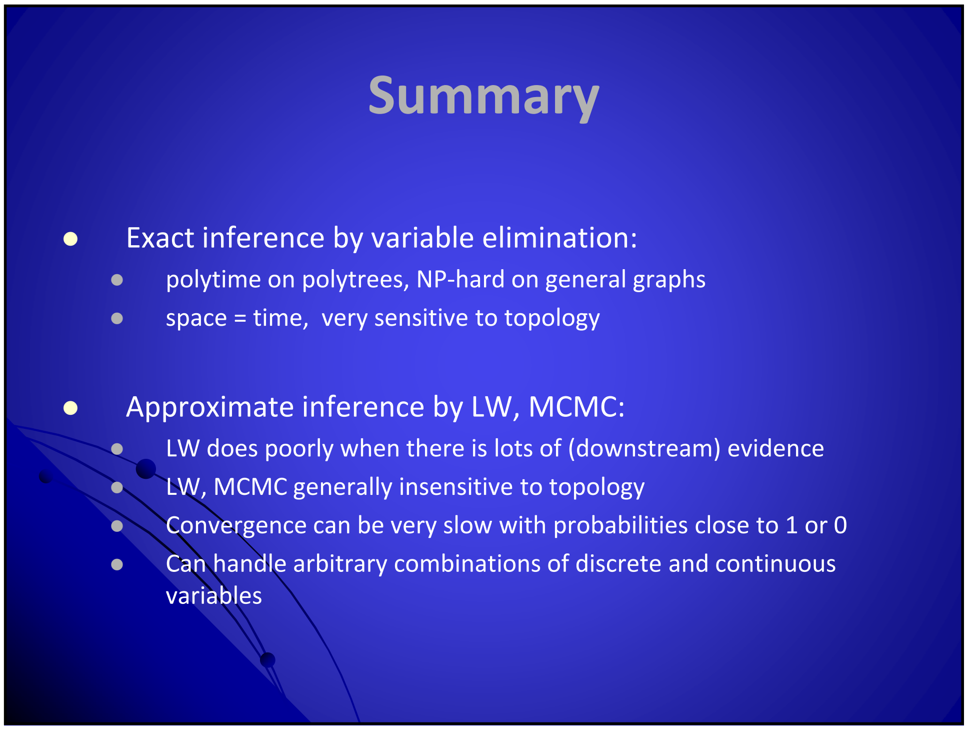
- Show how to perform the general Gibbs sampling step in BN.
  - Observe that sampling conditionally on all variables is equivalent to sampling conditionally on the MB variables.
  - A variable is independent of other variables given its MB:

$$P(x_i' \mid \overline{\mathbf{x}_i}, \mathbf{e}) = P(x_i' \mid mb(X_i))$$

  - The probability of a variable given its MB is proportional to probability of the variable given its parents times the probability of each child given its respective parents:

$$P(x_i' \mid mb(X_i)) = \alpha \, P(x_i' \mid parents(X_i)) \times \prod_{Y_j \in Children(X_i)} P(y_j \mid parents(Y_j))$$

  - Hence, to flip each variable X_i conditioned on its MB, the number of multiplications required is equal to the number of X_i's children.



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