# Introduction to Artificial Intelligence

Lecture 6: Probabilistic reasoning II

LIÈGE université

# Today

- Exact inference
  - Inference by enumeration
  - Inference by variable elimination
  - Complexity of exact inference
- Bayesian networks with continuous variables
- Approximate inference
  - Stochastic simulation
  - Rejection sampling
  - Likelihood weighting
  - Gibbs sampling

# Exact inference

# Inference tasks

- Inference: computing a desired probability from a joint probability distribution.
- Examples:
  - Simple queries: $P(X_i|E=e)$
  - Conjunctive queries: $P(X_i, X_j|E=e) = P(X_i|E=e)P(X_j|X_i, E=e)$
  - Most likely explanation: $\arg\max_q P(Q=q|E=e)$
    - Do you need to necessarily know $P(Q=q|E=e)$ to answer this?
  - Optimal decisions: take the decision that maximizes the expected utility of the outcomes.
    - requires to $P(outcome|action, evidence)$ for weighting the corresponding utility.
  - Value of information: which evidence to seek next?

# Inference by enumeration

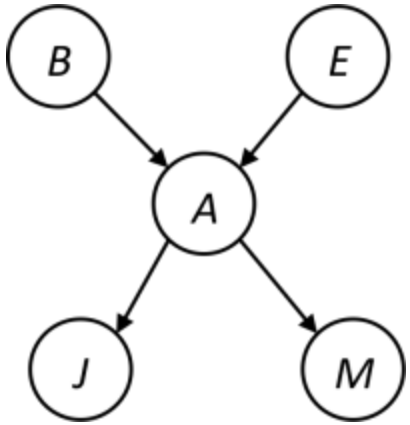
Start from the joint distribution $P(Q, E_1, ..., E_k, H_1, ..., H_r)$.

1. Select the entries consistent with the evidence $E_1, ..., E_k = e_1, ..., e_k$.
2. Marginalize out the hidden variables to obtain the joint of the query and the evidence values $P(Q, e_1, ..., e_k)$.
3. Normalize by $Z = P(e_1, ..., e_k) = \sum_q P(q, e_1, ..., e_k)$.

# Inference by enumeration in BNs

Consider the burglary network and the query $P(B|j,m)$:

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




Rewrite full joint entries using product of CPT entries:

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

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

# Enumeration algorithm

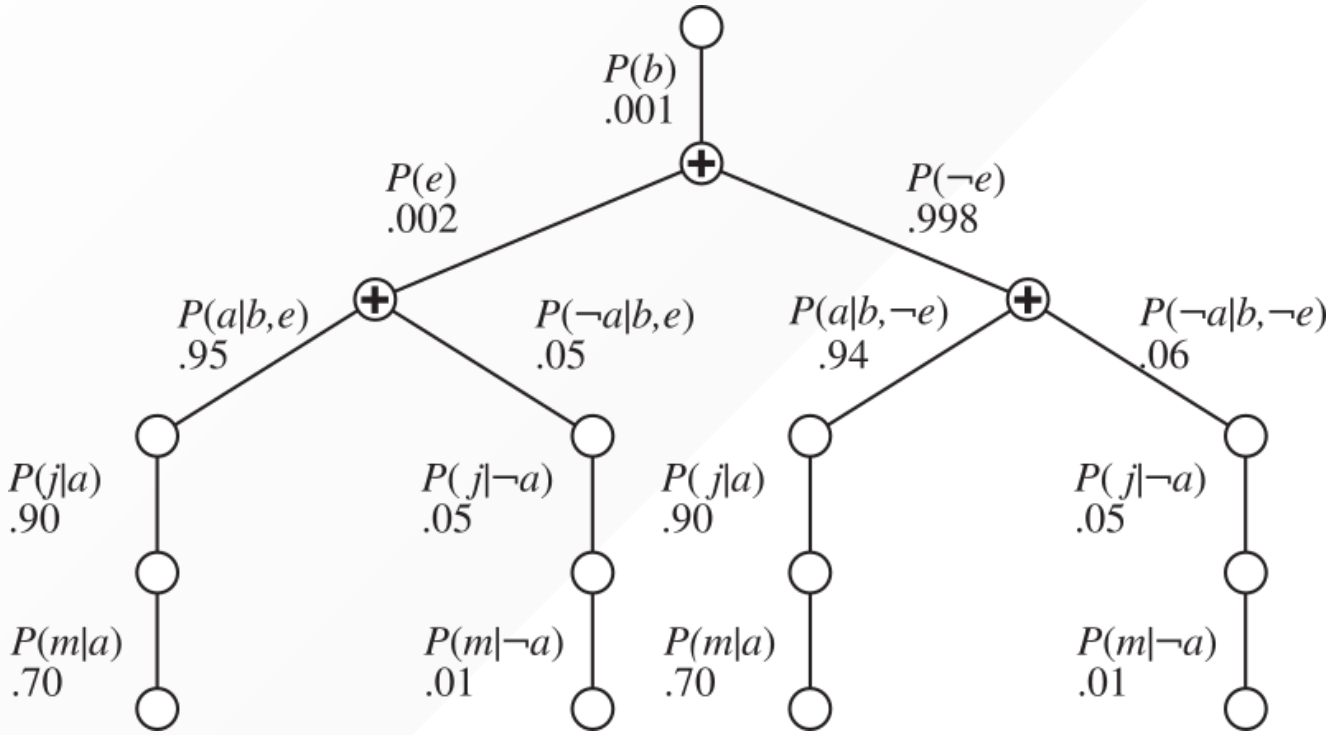
**function** ENUMERATION-ASK($X$, **e**, $bn$) **returns** a distribution over $X$
  **inputs**: $X$, the query variable
    **e**, observed values for variables **E**
    $bn$, a Bayes net with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$  /* **Y** = *hidden variables* */

  $\mathbf{Q}(X) \leftarrow$ a distribution over $X$, initially empty
  **for each** value $x_i$ of $X$ **do**
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL($bn$.VARS, $\mathbf{e}_{x_i}$)
      where $\mathbf{e}_{x_i}$ is **e** extended with $X = x_i$
  **return** NORMALIZE($\mathbf{Q}(X)$)

---

**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** $P(y \mid parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), **e**)
    **else return** $\sum_y P(y \mid parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_y$)
      where $\mathbf{e}_y$ is **e** extended with $Y = y$

# Evaluation tree




Enumeration is inefficient: there are repeated computations!

- e.g., $P(j|a)P(m|a)$ is computed twice, once for $e$ and once for $\neg e$.
- These can be avoided by storing intermediate results.

# Inference by variable elimination

- The variable elimination (VE) algorithm carries out summations right-to-left and stores intermediate results (called factors) to avoid recomputations.
- The algorithm interleaves:
  - Joining sub-tables
  - Eliminating hidden variables

# VE: factors

- Each factor $f_i$ is a matrix indexed by the values of its argument variables. E.g.:

$$\mathbf{f}_4(A) = \begin{pmatrix} P(j \mid a) \\ P(j \mid \neg a) \end{pmatrix} = \begin{pmatrix} 0.90 \\ 0.05 \end{pmatrix} \qquad \mathbf{f}_5(A) = \begin{pmatrix} P(m \mid a) \\ P(m \mid \neg a) \end{pmatrix} = \begin{pmatrix} 0.70 \\ 0.01 \end{pmatrix}$$

- Factors are initialized with the CPTs annotating the nodes of the Bayesian network, conditioned on the evidence.

# VE: join

The pointwise product, or join, of two factors $f_1$ and $f_2$ yields a new factor $f$.

- Exactly like a database join!
- The variables of $f$ are the union of the variables in $f_1$ and $f_2$.
- The elements of $f$ are given by the product of the corresponding elements in $f_1$ and $f_2$.

| $A$ | $B$ | $\mathbf{f}_1(A,B)$ |
|---|---|---|
| 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: $\mathbf{f}_1(A,B) \times \mathbf{f}_2(B,C) = \mathbf{f}_3(A,B,C)$.

# VE: elimination

Summing out, or eliminating, a variable from a sum of products of factors:

- move any constant factor outside the summation;
- add up submatrices of pointwise product of remaining factors.

---

Example (eliminate $E$):

$$\sum_e f_2(E) f_3(A, B, E) f_4(A) f_5(A)$$
$$= f_4(A) f_5(A) \sum_e f_2(E) f_3(A, B, E)$$
$$= f_4(A) f_5(A) f'_6(A, B)$$

# Variable elimination algorithm

Query: $P(Q|e_1, ..., e_n)$.

Algorithm:

- Start with initial factors:
  - Local CPTs (but instantiated by evidence).
- While there are still hidden variables (not Q nor evidence):
  - Pick a hidden variable $H$
    - The elimination ordering is a design parameter.
  - Join all factors mentioning $H$
  - Eliminate (sum out) $H$
- Join all remaining factors and normalize.

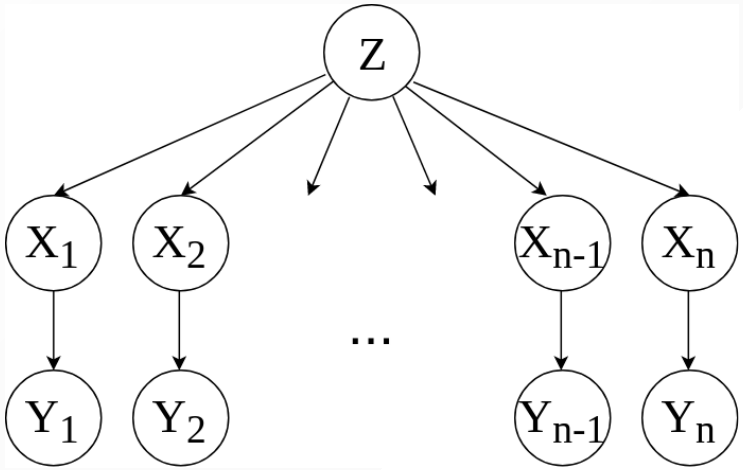
# Example

(blackboard example)

# Relevance

- Consider the query $P(JohnCalls|Burglar = true)$.
  - $P(J|b) = \alpha P(b) \sum_e P(e) \sum_a P(a|b,e) P(J|a) \sum_m P(m|a)$
- $\sum_m P(m|a) = 1$, therefore $M$ is irrelevant for the query.
- In other words, $P(J|b)$ remains unchanged if we remove $M$ from the network.
- Theorem: $H$ is irrelevant for $P(Q|E = e)$ unless $H \in \text{ancestors}(\{Q\} \cup E)$

# Elimination ordering

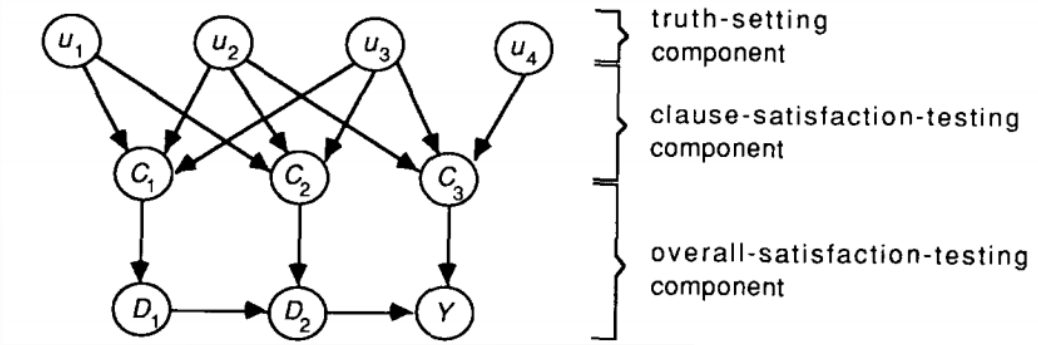



- Consider the query $P(X_n|y_1, ..., y_n)$.
- Work through the two elimination orderings:
  - $Z, X_1, ..., X_{n-1}$
  - $X_1, ..., X_{n-1}, Z$
- What is the size of the maximum factor generated for each of the orderings?
- Answer: $2^{n+1}$ vs. $2^2$ (assuming boolean values)

# Complexity of exact inference

- The computational and space complexity of variable elimination is determined by the largest factor.
- The elimination ordering can greatly affect the size of the largest factor.
- Does there always exist an ordering that only results in small factors? No!
- Singly connected networks (polytrees):
  - Any two nodes are connected by at most one (undirected path).
  - For these networks, time and space complexity of variable elimination are $O(nd^k)$.

# Worst case complexity?




3SAT is a special case of inference:

- CSP: $(u_1 \vee u_2 \vee u_3) \wedge (\neg u_1 \vee \neg u_2 \vee u_3) \wedge (u_2 \vee \neg u_3 \vee u_4)$
- $P(U_i = 0) = P(U_i = 1) = 0.5$
- $C_1 = U_1 \vee U_2 \vee U_3; C_2 = \neg U_1 \vee \neg U_2 \vee U_3; C_3 = U_2 \vee \neg U_3 \vee U_4$
- $D_1 = C_1; D_2 = D_1 \wedge C_2$
- $Y = D_2 \wedge C_3$

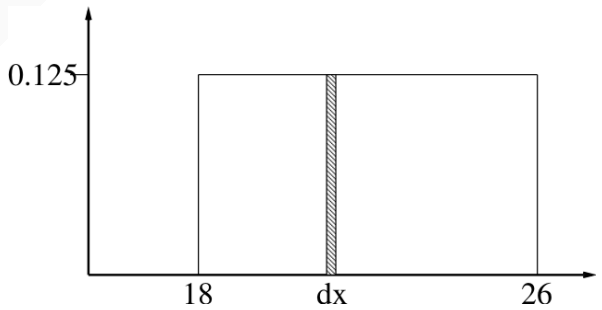
If we can answer whether $P(Y = 1) > 0$, then we answer whether 3SAT has a solution. By reduction, inference in Bayesian networks is therefore NP-hard.

- There is no known efficient probabilistic inference algorithm in general.

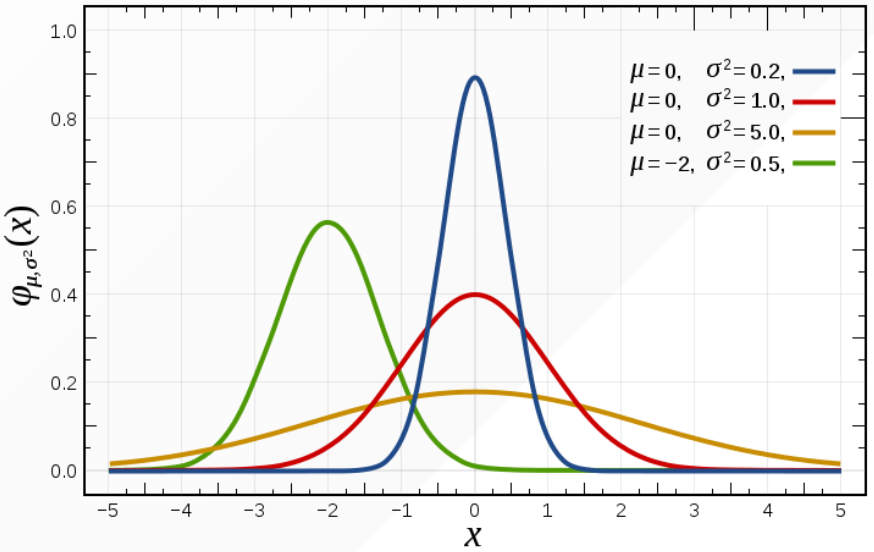
# Bayesian networks with continuous variables

# Continuous variables

- For continuous variables, the probability distribution can be described by a probability density function.
  - That is, the distribution is described by a continuous function of its value:
    - e.g., $P(X = x) = U[18, 26](x)$ for a uniform density between 18 and 26.
  - a density integrates to 1 and is non-negative everywhere.
- The absolute likelihood that a continuous variable $X$ takes value $x$ is $0$.
- The (integral of the) density provides the probability of falling within a particular range of values.
- E.g., $P(X = 20.5) = 0.125$ really means $\lim_{dx \to 0} P(20.5 \leq X \leq 20.5 + dx)/dx = 0.125$.



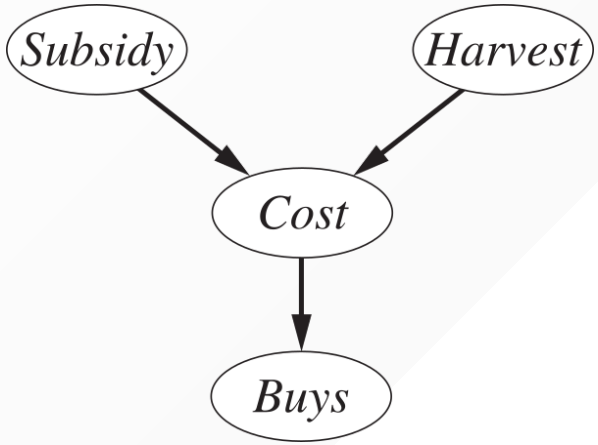
# Gaussian distribution




$$P(x) = \mathcal{N}(\mu, \sigma)(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

- $\mu$ and $\sigma$ are parameters of the distribution.
- The multivariate Gaussian distribution generalizes to $n \geq 1$ random variables.

# Hybrid Bayesian networks




- What if we have both discrete (e.g., $\mathrm{subsidy}$ and $\mathrm{buys}$) and continuous variables (e.g., $\mathrm{harvest}$ and $\mathrm{cost}$) in a same network?
- Options:
  - discretization: transform continuous variables into discrete variables.
    - issues: possibly large errors due to precision loss, large CPTs.
  - define the conditional distribution with a finitely parameterized canonical distribution.
    - e.g., assume it is a gaussian distribution.
  - use a non-parametric representation.

# Continuous child variables

- We need to specify a conditional density function for each continuous child variable given continuous parents, for each possible assignment to discrete parents.
  - e.g., we need to specify both $P(c|h, s)$ and $P(c|h, \neg s)$
- Common choice: the linear Gaussian model (LG):
  - $P(c|h, s) = \mathcal{N}(a_t h + b_t, \sigma_t^2)(c)$
  - $P(c|h, \neg s) = \mathcal{N}(a_t f + b_f, \sigma_f^2)(c)$

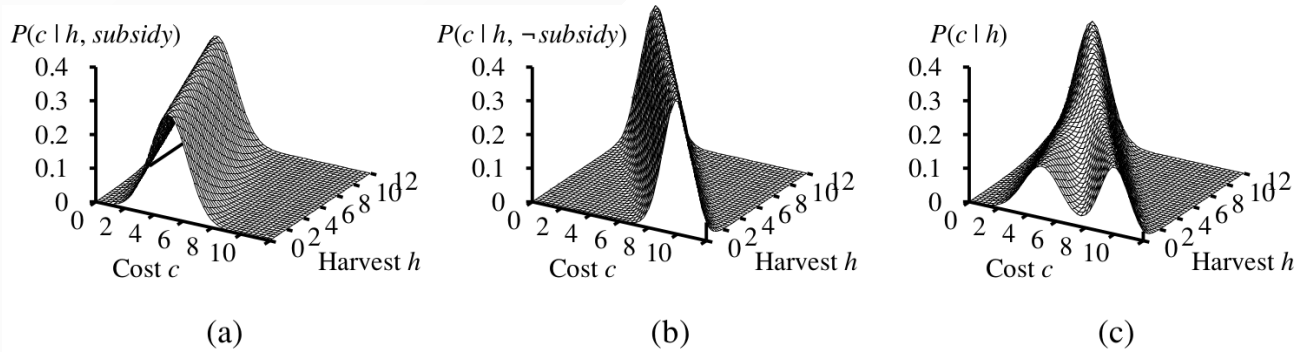


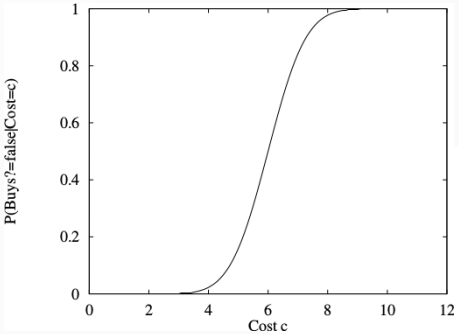

**Figure 14.6** The graphs in (a) and (b) show the probability distribution over *Cost* as a function of *Harvest* size, with *Subsidy* true and false, respectively. Graph (c) shows the distribution $P(Cost \mid Harvest)$, obtained by summing over the two subsidy cases.

# Conditional Gaussian network

- The joint distribution of an all-continuous network with LG distributions is a multivariate Gaussian.
- The joint distribution of a network with discrete+LG continuous variables is a conditional Gaussian network.
    - i.e., a multivariate Gaussian over all continuous variables for each combination of the discrete variable values.

# Discrete child variables, with continuous parents

- We need to specify a conditional distribution for each discrete child variable, given continuous parents.
- It is often reasonable to assume that the probability values of the discrete outcomes are almost piece-wise constant but vary smoothly in intermediate regions.
- E.g., $P(b|c)$ could be a "soft" threshold:




The probit distribution uses integral of Gaussian:

- $\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(0,1)(x)dx$
- $P(b|c) = \Phi((-c + \mu)/\sigma)$

# Variable elimination

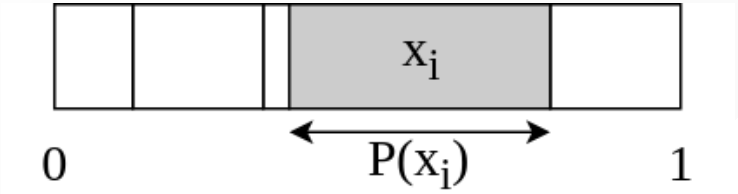
- Variable elimination in Hybrid Bayesian networks can be conducted similarly as in the discrete case, by replacing summations with integrations.
- Exact inference remains possible under some assumptions (e.g., linear Gaussian models).
  - in which case exact analytical computations can be derived.
- However, this often does not scale to arbitrary continuous distributions.
  - e.g., numerical approximations of integrals amount to discretize continuous variables.

# Approximate inference

# Approximate inference

- Exact inference is intractable for most probabilistic models of practical interest.
  - e.g., involving many variables, continuous and discrete, undirected cycles, etc.
- Solution: abandon exact inference and develop approximate but faster inference algorithms.
- Main families of approximate inference algorithms:
  - Sampling methods: produce answers by repeatedly generating random numbers from a distribution of interest.
    - This is the family of methods we will consider.
  - Variational methods: formulate inference as an optimization problem.
  - (Loopy) belief propagation methods: formulate inference as a message-passing algorithm.

# Sampling from a distribution




- How to sample from the distribution of a discrete variable $X$?
  - Assume $k$ discrete outcomes $x_1, \ldots, x_k$ with probability $P(x_i)$.
  - Assume sampling from $U[0, 1]$ is possible.
    - e.g., as enabled by a standard `rand()` function.
  - Divide the $[0, 1]$ interval into $d$ regions, with region $i$ having size $P(x_i)$.
  - Sample $u \sim U[0, 1]$ and return the value associated to the region in which $u$ falls.
- The same algorithm extends to continuous variables, assuming access to the inverse cumulative distribution function $F^{-1}$.
  - for $p \in [0, 1]$, $F^{-1}(p) = x$ such that $F(x) = p$, where $F$ is the CDF.
  - $F^{-1}$ is known analytically for most canonical distributions (e.g., Gaussian).

[Q] How to extend to arbitrary multivariate distributions?

# Ancestral sampling

Sampling from a Bayesian network, without observed evidence:

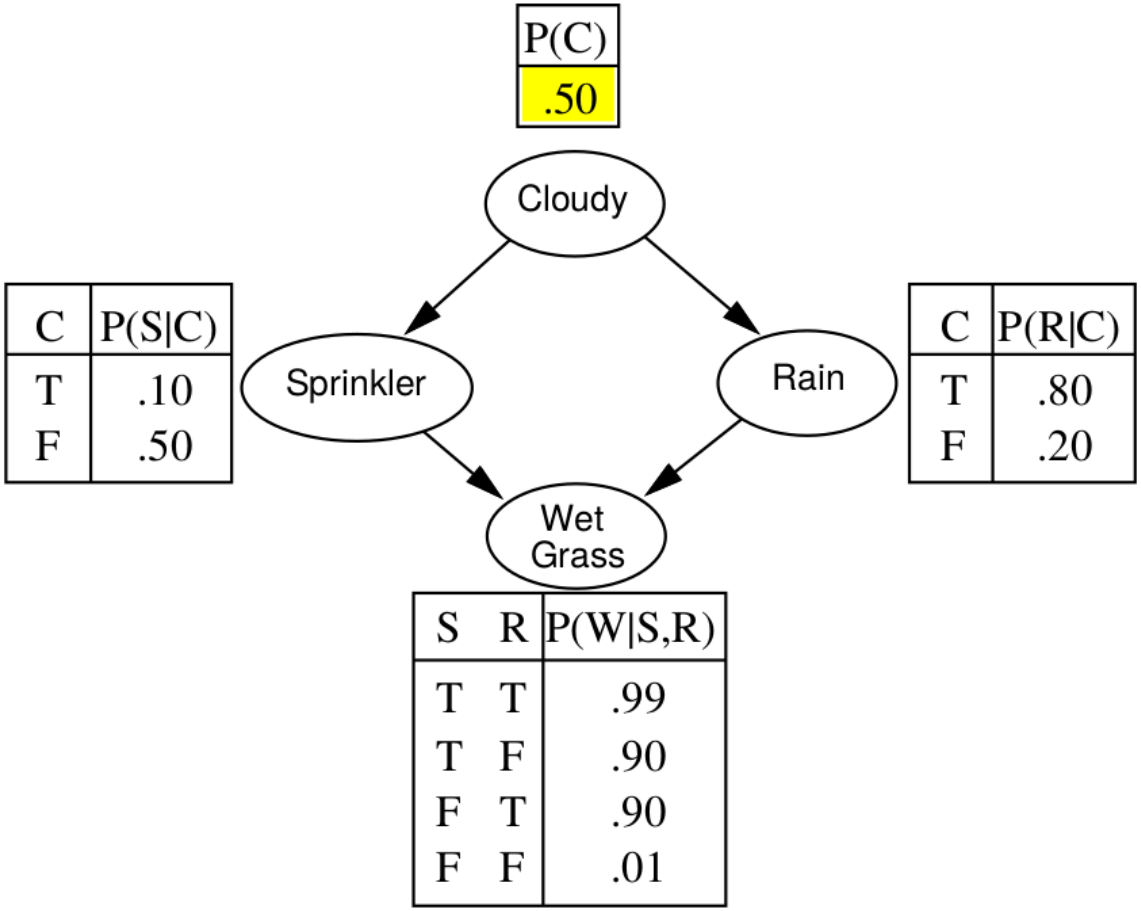
- Sample each variable in turn, in topological order.
- The probability distribution from which the value is sampled is conditioned on the values already assigned to the variable's parents.

---

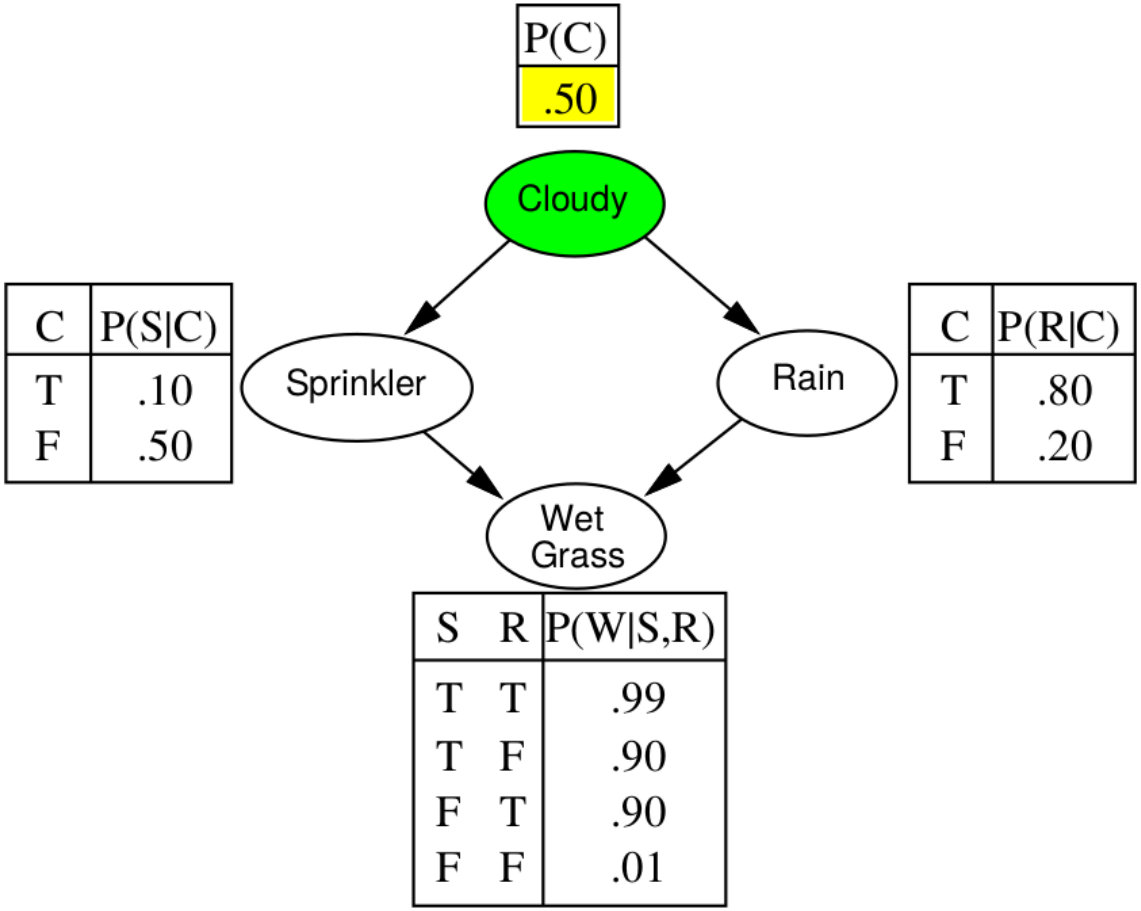
**function** PRIOR-SAMPLE($bn$) **returns** an event sampled from the prior specified by $bn$
  **inputs**: $bn$, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $\mathbf{x} \leftarrow$ 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** $\mathbf{x}$

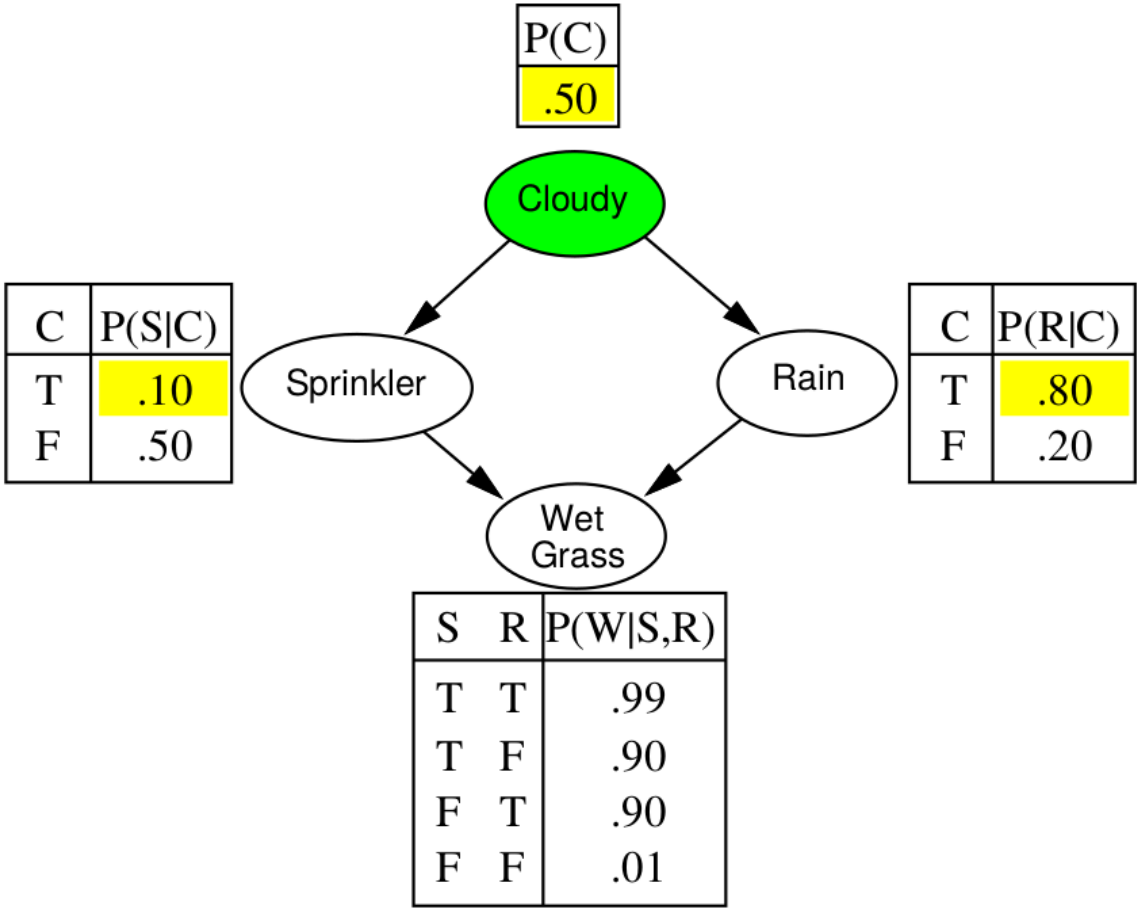
# Example (1)


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

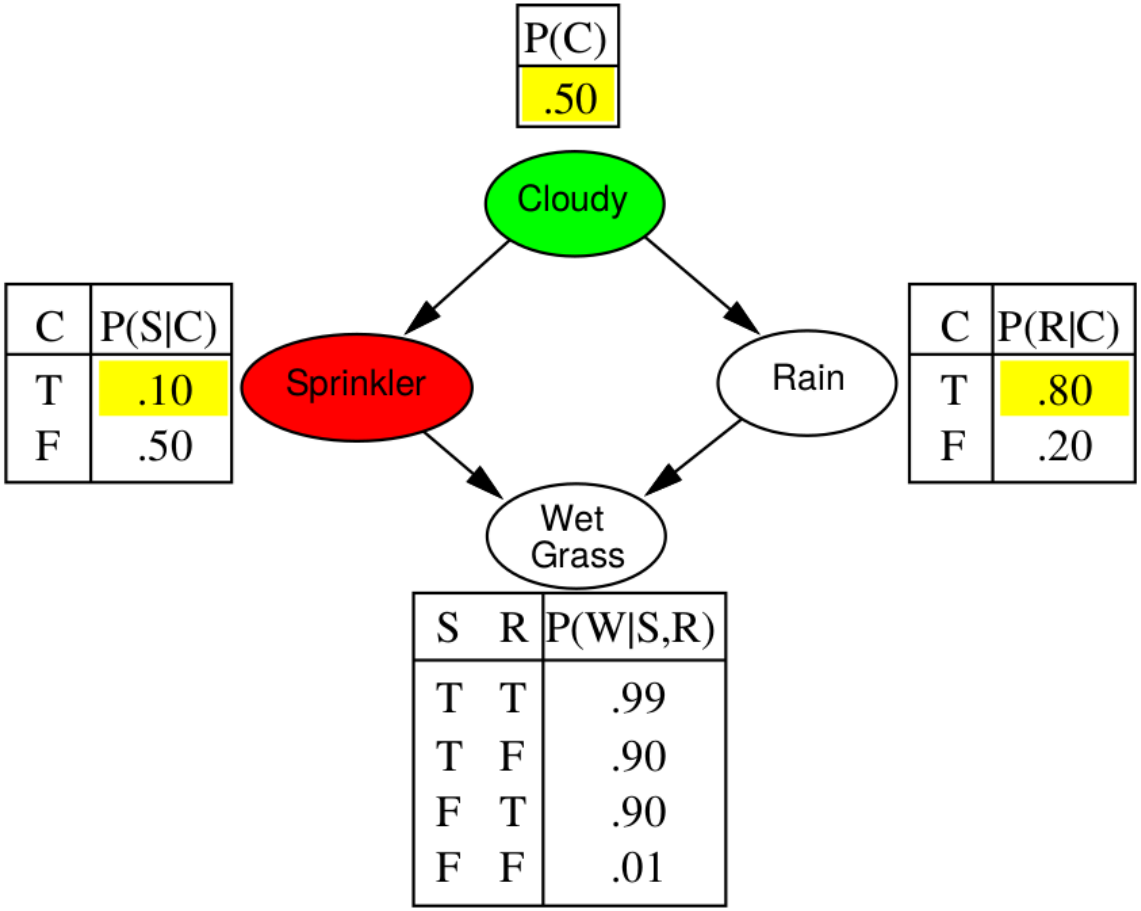
# Example (2)


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

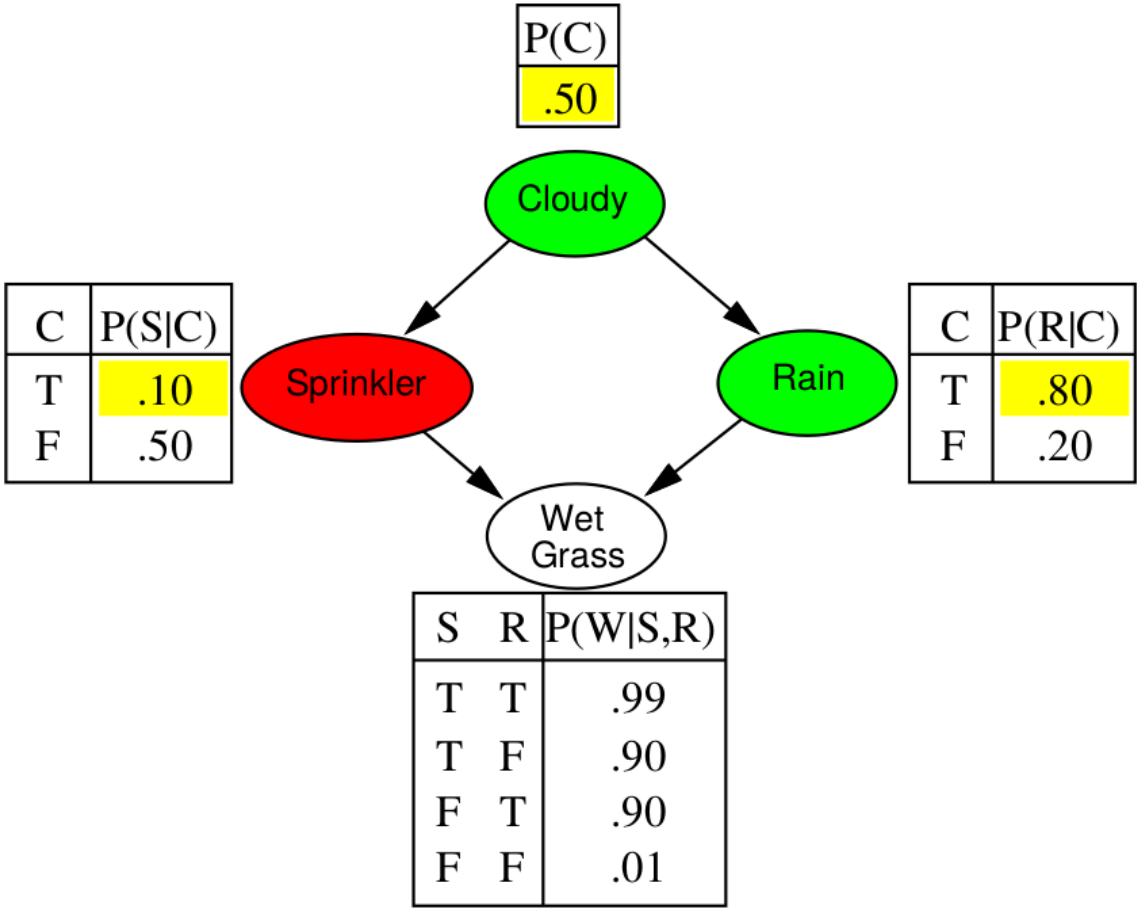
# Example (3)


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

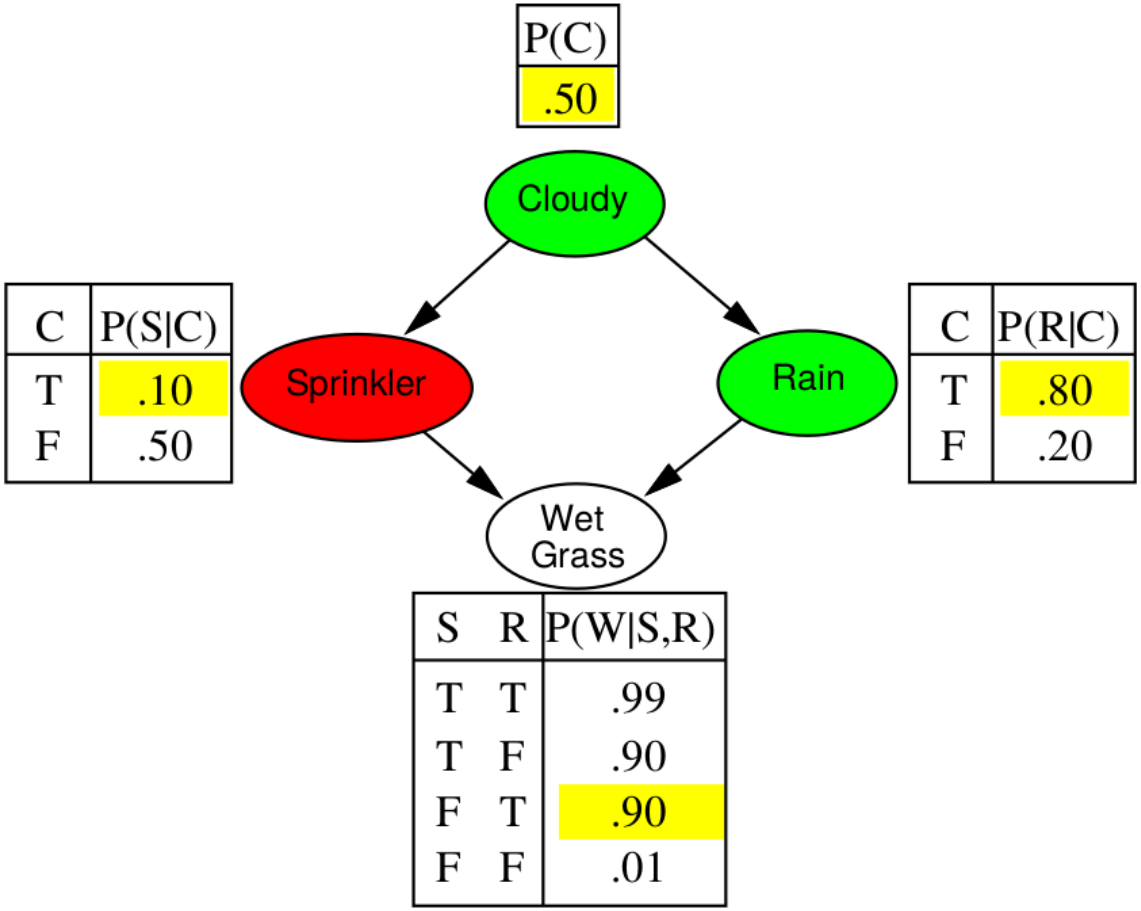
# Example (4)


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

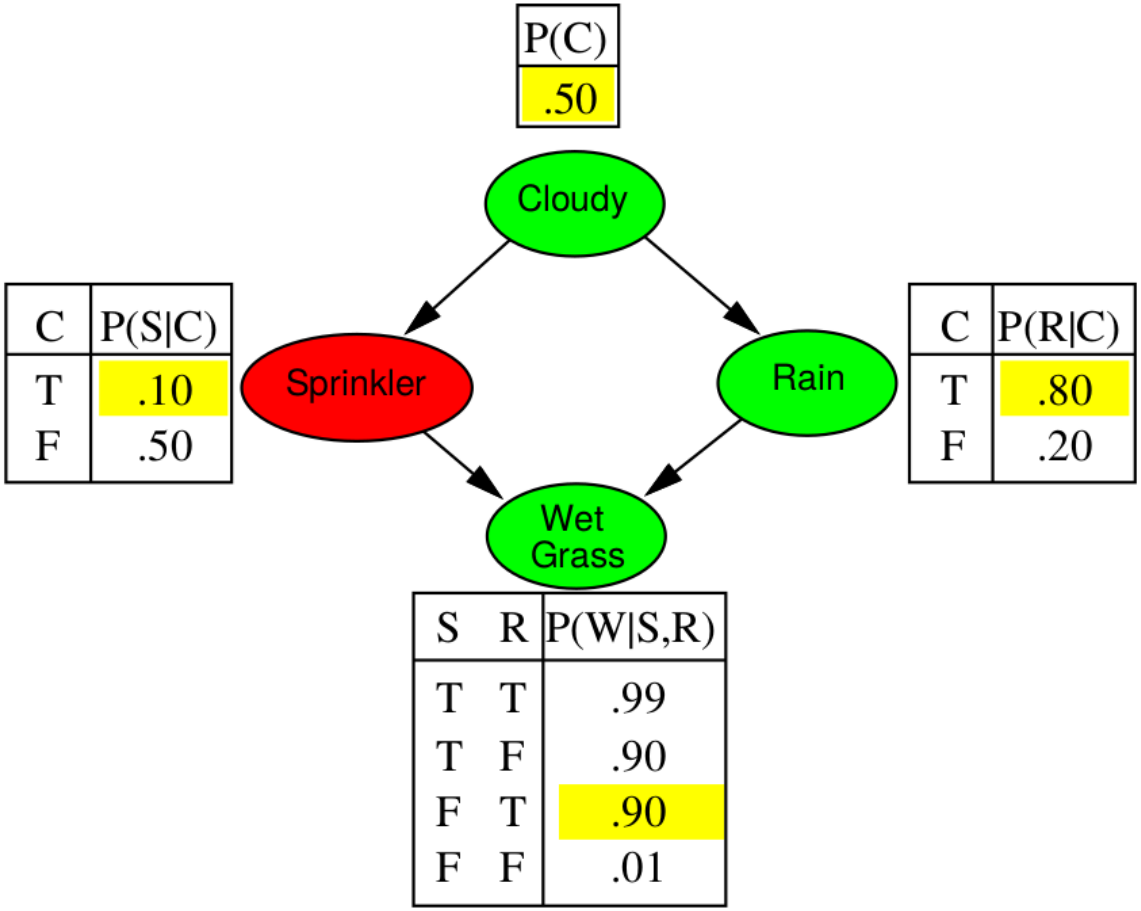
# Example (5)


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 (6)


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 (7)


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

# Analysis of ancestral sampling

- The probability that ancestral sampling generates a particular event is

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

  i.e., the Bayesian network's joint probability.

- Let the number of samples of an event be $N_{PS}(x_1, ..., x_n)$. We define the probability estimate

$$\hat{P}(x_1, ..., x_n) = N_{PS}(x_1, ..., x_n)/N.$$

- Then:

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

- That is, the sampling procedure is consistent:
  $P(x_1, ..., x_n) \approx N_{PS}(x_1, ..., x_n)/N.$

# Rejection sampling

Using ancestral sampling, an estimate $\hat{P}(x|e)$ can be formed from the samples agreeing with the evidence.

---

**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** $\leftarrow$ 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**)

[Q] Can we use a similar idea to sample continuous variables for which $P$ is known but $F^{-1}$ isn't?

# Analysis of rejection sampling

- Let consider the posterior probability estimate $\hat{P}(x|e)$ formed by rejection sampling:

  $\hat{P}(x|e) = \alpha N_{PS}(x, e)$ (by definition of the algorithm)
  $= N_{PS}(x, e)/N_{PS}(e)$
  $\approx P(x, e)/P(e)$
  $= P(x|e)$

- Therefore, rejection sampling returns consistent posterior estimates.
- The standard deviation of the error in each probability is $O(1/\sqrt{n})$.
- Problem: many samples are rejected!
  - Hopelessly expensive if $P(e)$ is small.
  - Evidence is not exploited when sampling.

# Likelihood weighting

Idea: fix evidence variables, sample the rest.

- Problem: the resulting sampling distribution is not consisent.
- Solution: weight by probability of evidence given parents.

---

**function** LIKELIHOOD-WEIGHTING($X$, **e**, $bn$, $N$) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  **inputs**: $X$, the query variable
    **e**, observed values for variables **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**: **W**, a vector of weighted counts for each value of $X$, initially zero

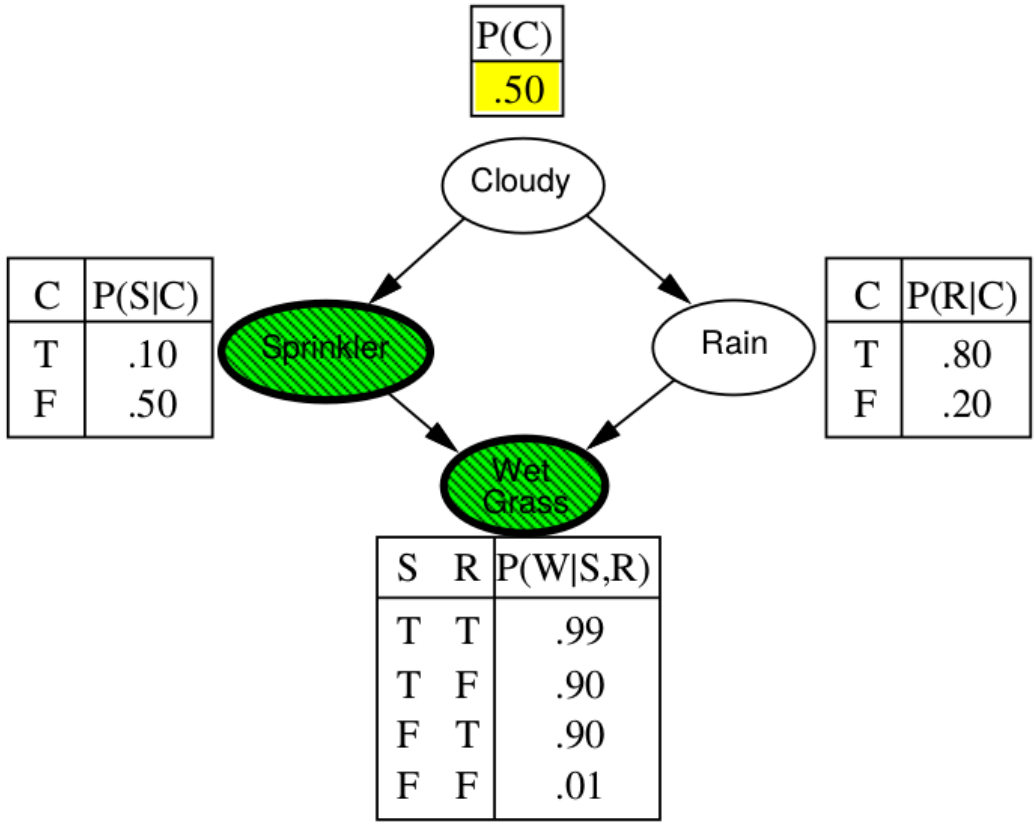
  **for** $j = 1$ to $N$ **do**
    **x**, $w \leftarrow$ WEIGHTED-SAMPLE($bn$, **e**)
    $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where $x$ is the value of $X$ in **x**
  **return** NORMALIZE(**W**)

---

**function** WEIGHTED-SAMPLE($bn$, **e**) **returns** an event and a weight

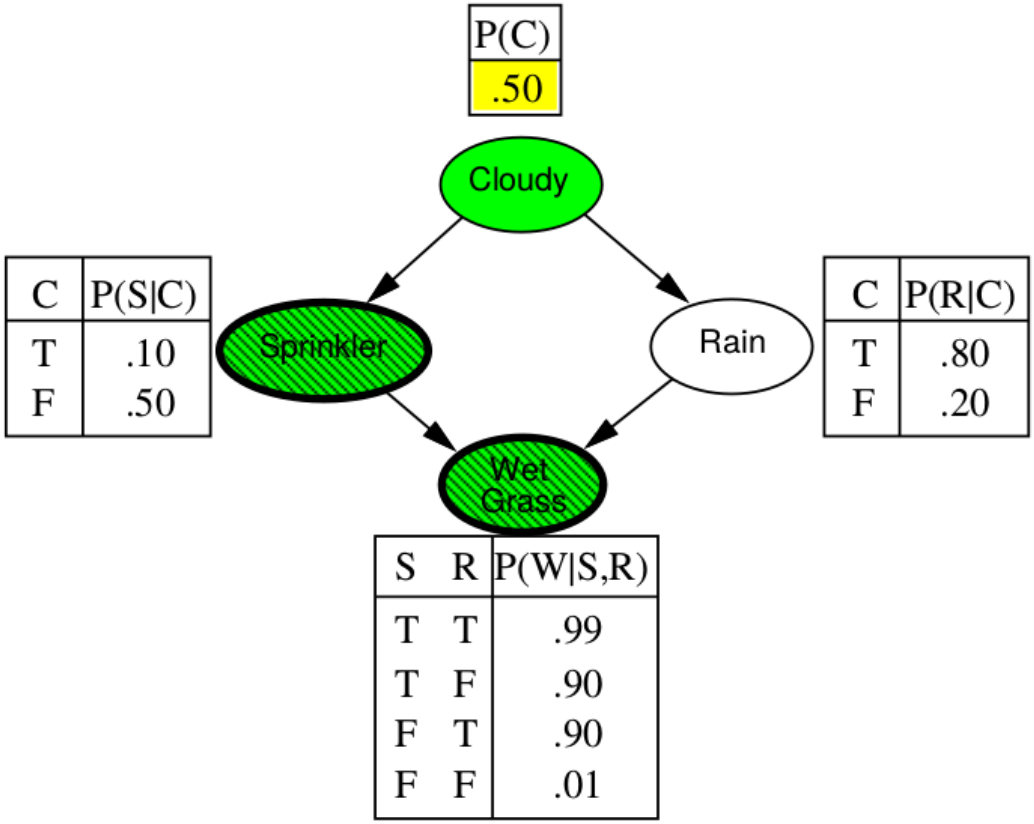
  $w \leftarrow 1$; **x** $\leftarrow$ an event with $n$ elements initialized from **e**
  **foreach** variable $X_i$ **in** $X_1, \ldots, X_n$ **do**
    **if** $X_i$ is an evidence variable with value $x_i$ in **e**
      **then** $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
      **else**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

# Example (1)



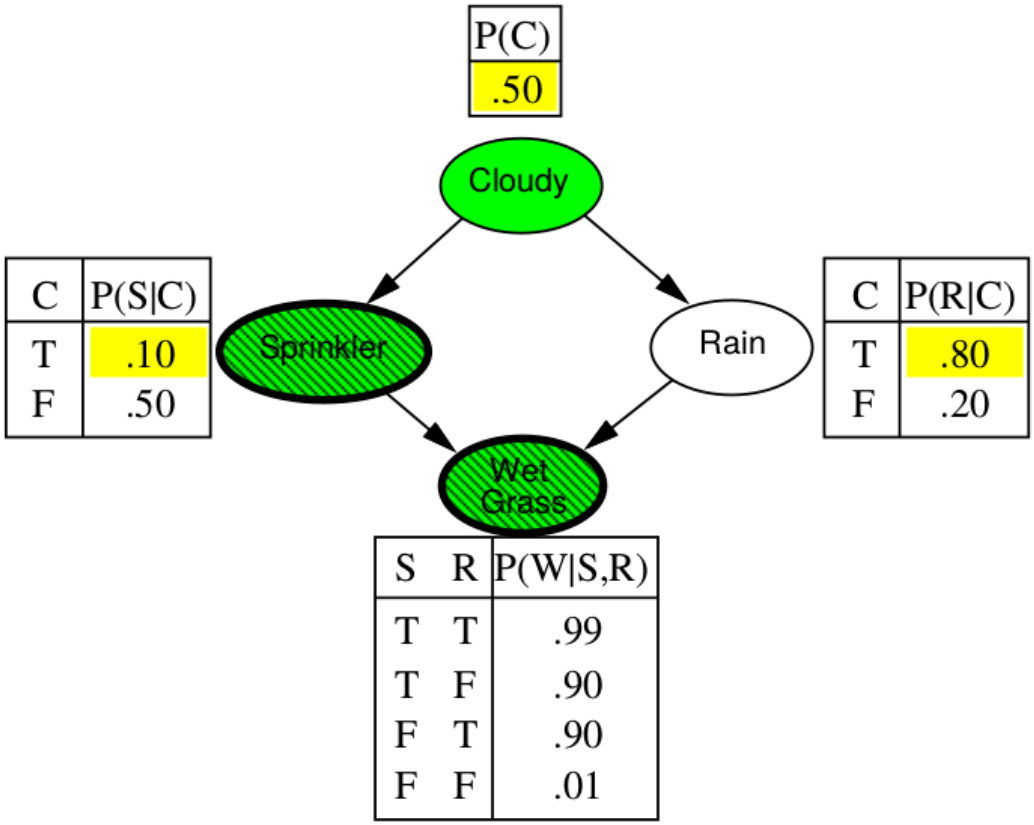

$w = 1.0$

# Example (2)



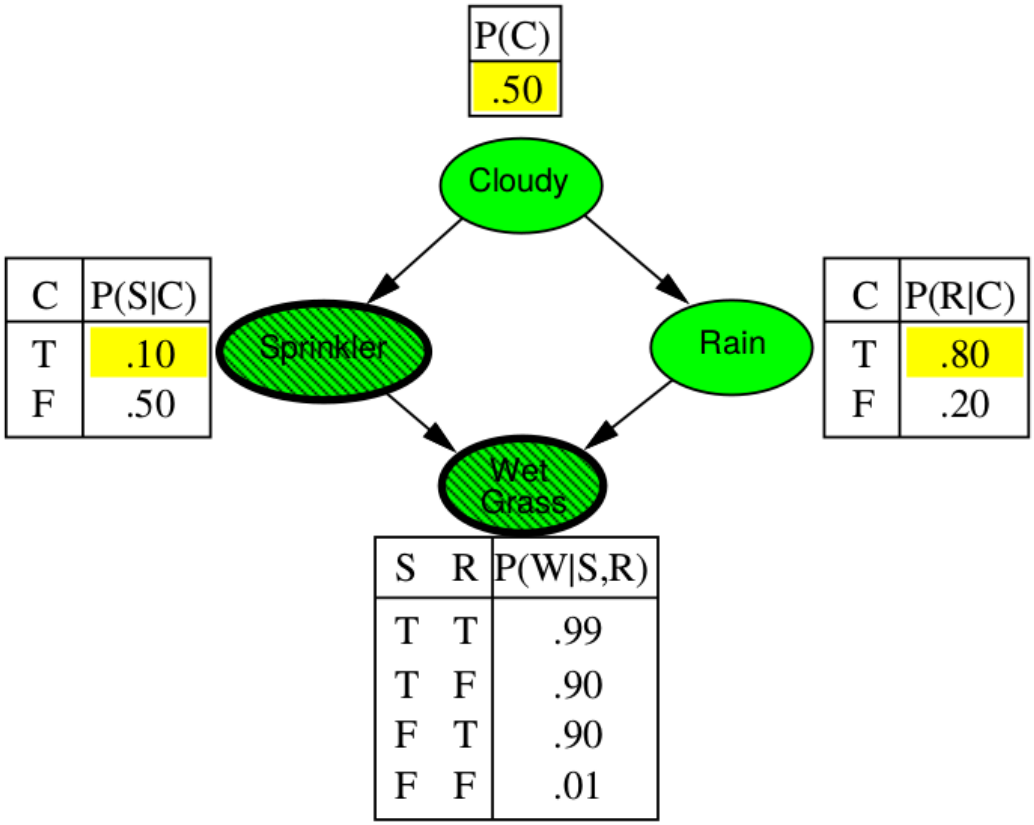

$w = 1.0$

# Example (3)



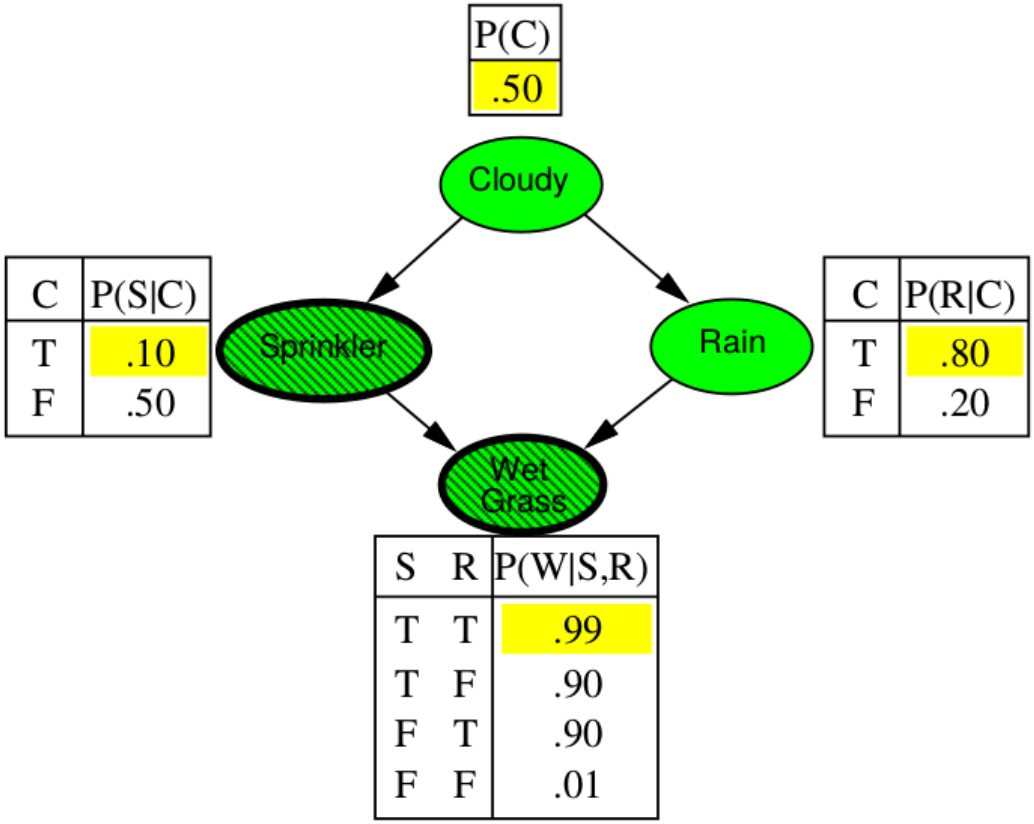

$w = 1.0$

# Example (4)




$w = 1.0 \times 0.1$

# Example (5)




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

# Analysis of likelihood weighting (1)

- The sampling probability for an event with likelihood weighting is

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

  where the product is over the non-evidence variables.

- The weight for a given sample $z, e$ is

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

  where the product is over the evidence variables.

- The weighted sampling probability is

$$\begin{aligned} S_{WS}(z, e) w(z, e) &= \prod_{i=1}^{l} P(z_i | \text{parents}(Z_i)) \prod_{i=1}^{m} P(e_i | \text{parents}(E_i)) \\ &= P(z, e). \end{aligned}$$

# Analysis of likelihood weighting (2)

- The estimated posterior probability is computed as follows:

$$\begin{aligned}\hat{P}(x|e) &= \alpha \sum_y N_{WS}(x,y,e) w(x,y,e) \\ &\approx \alpha' \sum_y S_{WS}(x,y,e) w(x,y,e) \\ &= \alpha' \sum_y P(x,y,e) \\ &= \alpha' P(x,e) = P(x|e)\end{aligned}$$

- Hence likelihood weighting returns consistent estimates.
- Performance still degrades with many evidence variables.
  - A few samples have nearly all the total weight.

[Q] What should be the normalization constants $\alpha$ and $\alpha'$ to obtain correct results?

# Likelihood weighting

- Likelihood weighting is good:
  - The evidence is taken into account to generate a sample.
  - More of the samples will reflect the state of the world suggested by the evidence.
- Likelihood weighting does not solve all problems:
  - The evidence influences the choice of downstream variables, but not upstream ones.
- We would like to consider evidence when we sample every variable.

# Gibbs sampling

- Procedure:
  - Keep track of a full instance $x_1, \ldots, x_n$. Start with an arbitrary instance consistent with the evidence.
  - Sample one variable at a time, conditioned on all the rest.
    - Keep the evidence fixed.
  - Keep repeating this for a long time.
- Property:
  - 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.
- Rationale:
  - Both upstream and downstream variables condition on evidence.
  - In contrast, likelihood weighting only conditions on upstream evidence, and hence the resulting weights might be very small.

# Gibbs sampling

**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**

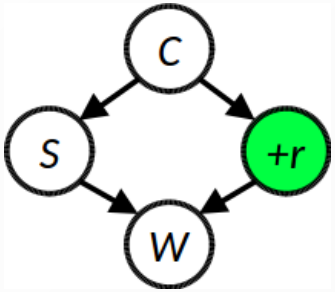
  initialize **x** with random values for the variables in **Z**
  **for** $j = 1$ to $N$ **do**
    **for each** $Z_i$ in **Z do**
      set the value of $Z_i$ in **x** by sampling from $\mathbf{P}(Z_i|mb(Z_i))$
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where $x$ is the value of $X$ in **x**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

Note that we need to derive $P(Z_i|mb(Z_i))$:

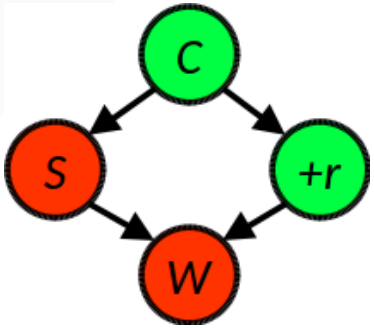
- $mb(Z_i)$ is the Markov blanket of $Z_i$.
- i.e., the set of $Z_i$'s parents, children and children's parents.

# Example

1) Fix the evidence

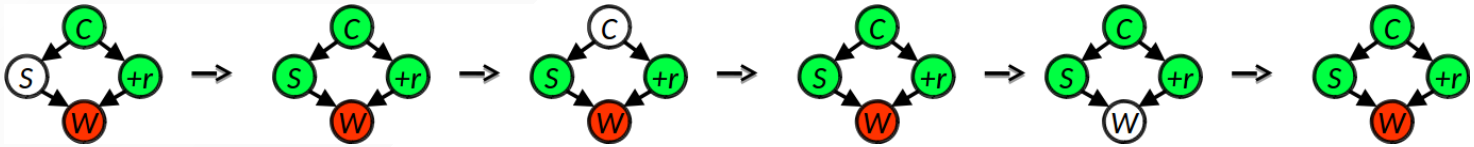



2) Randomly initialize the other variables




3) Repeat

- Choose a non-evidence variable $X$
- Resample $X$ from $P(X|\text{all other variables})$



# Further reading on Gibbs sampling

- Gibbs sampling produces samples from the query distribution $P(X|e)$ in the limit of re-sampling infinitely often.
- Gibbs sampling is a special case of a more general methods called Markov chain Monte Carlo (MCMC) methods.
  - Metropolis-Hastings is one of the more famous MCMC methods.
    - In fact, Gibbs sampling is a special case of Metropolis-Hastings.
- You may read about Monte Carlo methods: they are just sampling.

# (Gibbs sampling demo)

# Summary

- Exact inference by variable elimination .
  - NP-hard on general graphs, but polynomial on polytrees.
  - space = time, very sensitive to topology.
- Approximate inference gives reasonable estimates of the true posterior probabilities in a network and can cope with much larger networks than can exact algorithms.
  - LW does poorly when there is lots of evidence.
  - LW and GS generally insensitive to topology.
  - Convergence can be slow with probabilities close to 1 or 0.
  - Can handle arbitrary combinations of discrete and continuous variables.
- Want to know more about sampling?
  - Follow MATH2022 Large sample analysis: theory and practice.

# References


- Cooper, Gregory F. "The computational complexity of probabilistic inference using Bayesian belief networks." Artificial intelligence 42.2-3 (1990): 393-405.