Introduction to Artificial Intelligence

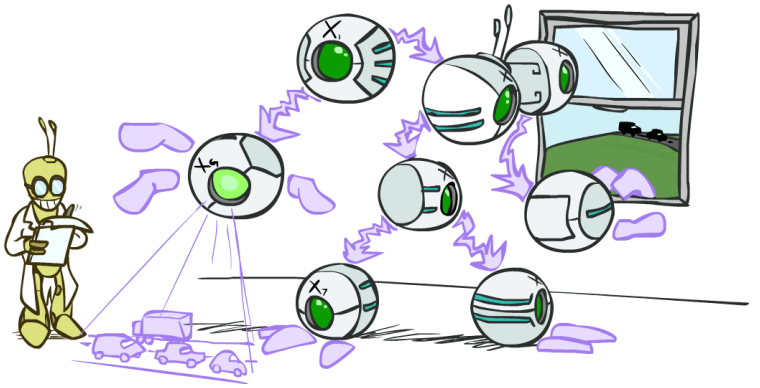
Lecture 5: Inference in Bayesian networks

Prof. Gilles Louppe
g.louppe@uliege.be

LIÈGE université

Today

- Exact inference
  - Inference by enumeration
  - Inference by variable elimination
- Approximate inference
  - Ancestral sampling
  - Rejection sampling
  - Likelihood weighting
  - Gibbs sampling



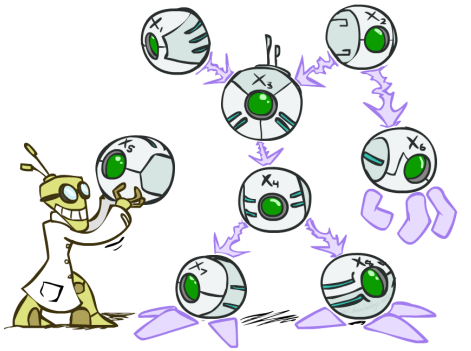
—

Image credits: CS188, UC Berkeley.

Bayesian networks

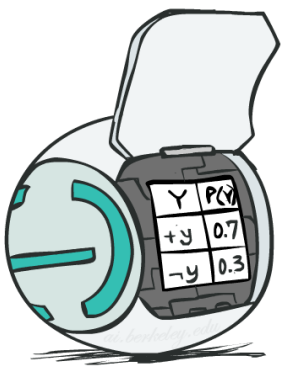
A Bayesian network is a directed acyclic graph in which:

- Each node corresponds to a random variable X_i.
- Each node X_i is annotated with a conditional probability distribution $\mathbf{P}(X_i|\text{parents}(X_i))$ that quantifies the effect of the parents on the node.



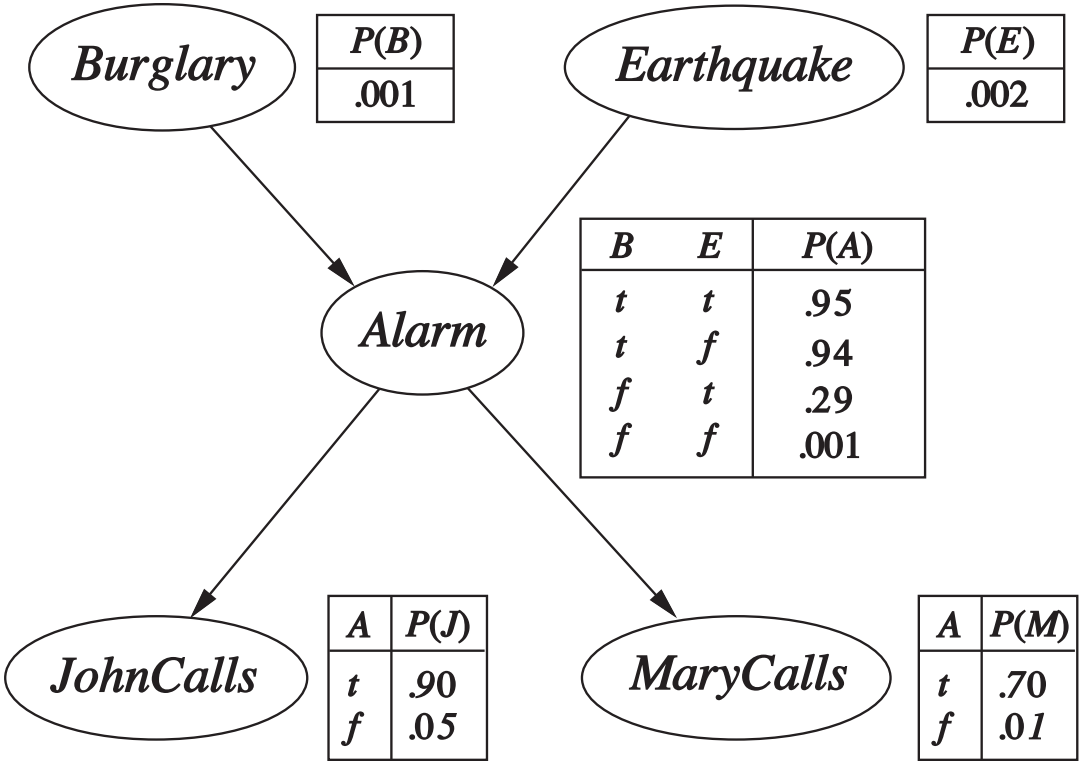
A Bayesian network implicitly encodes the full joint distribution as the product of the local distributions:

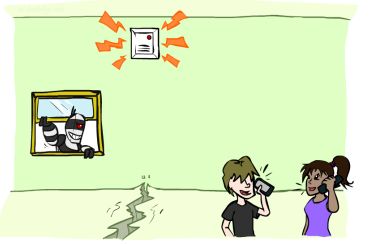
$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|\text{parents}(X_i))$$



—

Image credits: CS188, UC Berkeley.





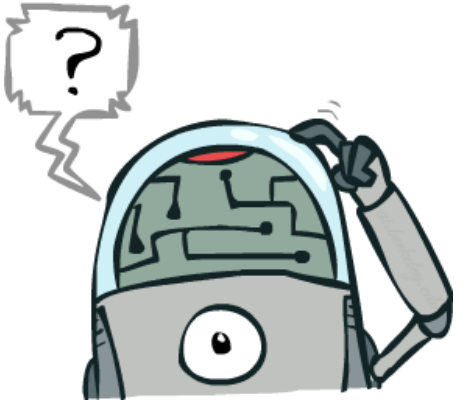
$$P(b, \neg e, a, \neg j, m) = P(b)P(\neg e)P(a|b, \neg e)P(\neg j|a)P(m, a)$$
$$= 0.001 \times 0.998 \times 0.94 \times 0.1 \times 0.7$$

Exact inference

Inference

Inference is concerned with the problem computing a marginal and/or a conditional probability distribution from a joint probability distribution:

Simple queries:	$\mathbf{P}(X_i \mid e)$
Conjunctive queries:	$\mathbf{P}(X_i, X_j \mid e) = \mathbf{P}(X_i \mid e)\mathbf{P}(X_j \mid X_i, e)$
Most likely explanation:	$\arg\max_q P(q \mid e)$
Optimal decisions:	$\arg\max_a \mathbb{E}_{p(s' \mid s,a)}\left[V(s')\right]$



—

Image credits: CS188, UC Berkeley.

Inference by enumeration

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

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 variables:

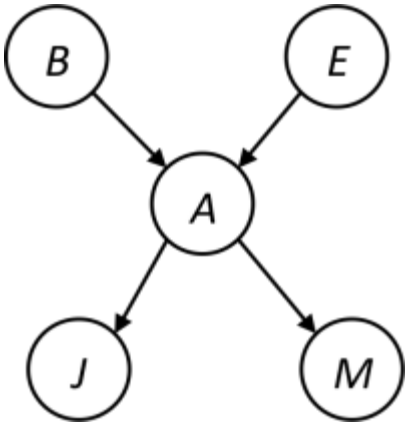
$$\mathbf{P}(Q, e_1, ..., e_k) = \sum_{h_1, ..., h_r} \mathbf{P}(Q, h_1, ..., h_r, e_1, ..., e_k).$$

3. Normalize:

$$Z = \sum_q P(q, e_1, ..., e_k)$$

$$\mathbf{P}(Q|e_1, ..., e_k) = \frac{1}{Z}\mathbf{P}(Q, e_1, ..., e_k)$$

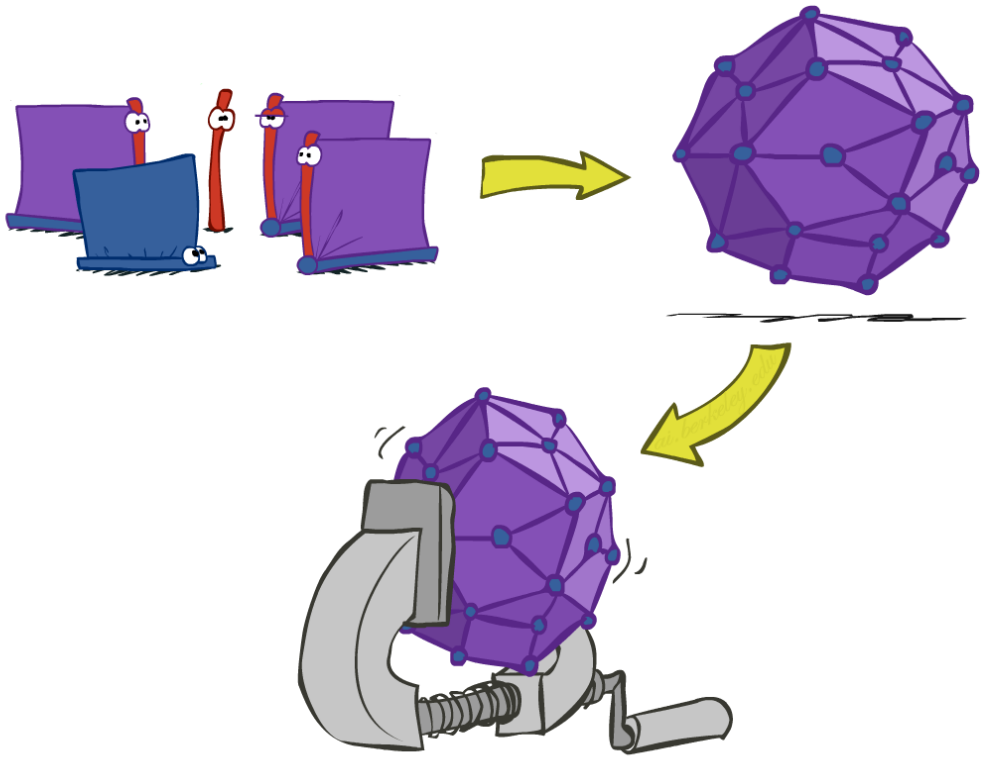
Consider the alarm network and the query $\mathbf{P}(B|j,m)$:



$$\mathbf{P}(B|j,m) = \frac{1}{Z}\sum_e\sum_a \mathbf{P}(B,j,m,e,a)$$
$$\propto \sum_e\sum_a \mathbf{P}(B,j,m,e,a)$$

Using the Bayesian network, the full joint entries can be rewritten as the product of CPT entries:

$$\mathbf{P}(B|j,m) \propto \sum_e\sum_a \mathbf{P}(B)P(e)\mathbf{P}(a|B,e)P(j|a)P(m|a)$$



Inference by enumeration is slow because the whole joint distribution is joined up before summing out the hidden variables.

—

Image credits: CS188, UC Berkeley.

Notice that factors that do not depend on the variables in the summations can be factored out, which means that marginalization does not necessarily have to be done at the end:

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

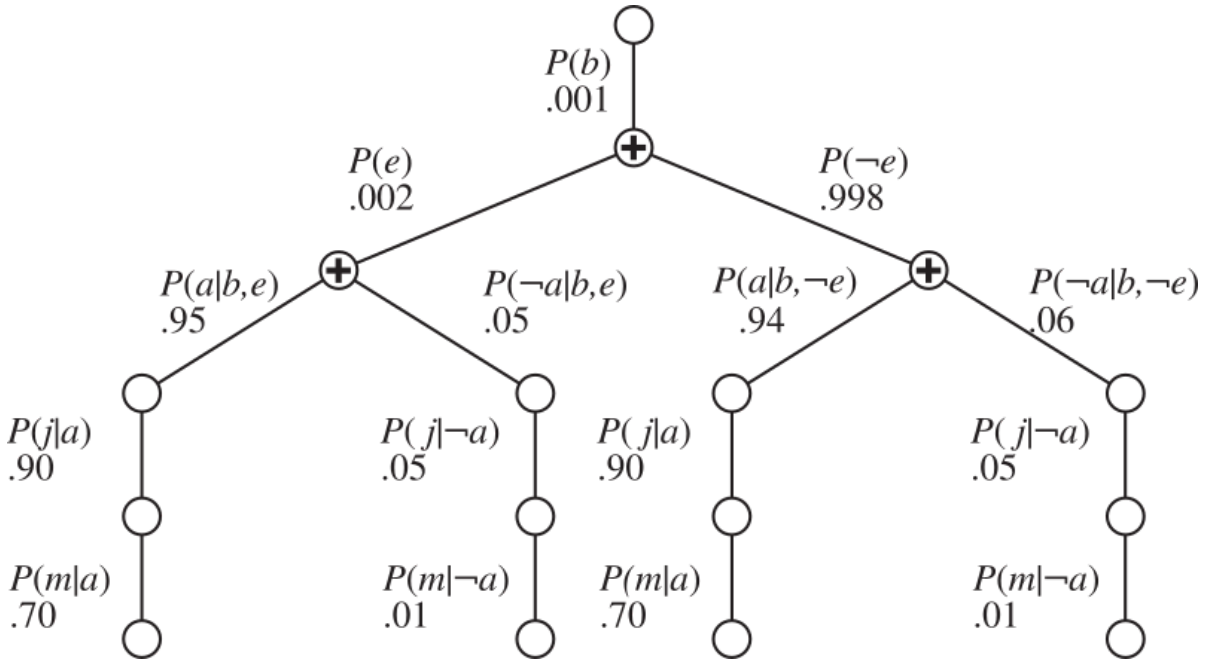
function ENUMERATION-ASK(X, $\mathbf{e}$, bn) **returns** a distribution over X
  inputs: X, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayes net with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$ /* $\mathbf{Y}$ = *hidden variables* */

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

Same complexity as DFS: $O(n)$ in space, $O(d^n)$ in time.

Evaluation tree for $P(b|j,m)$



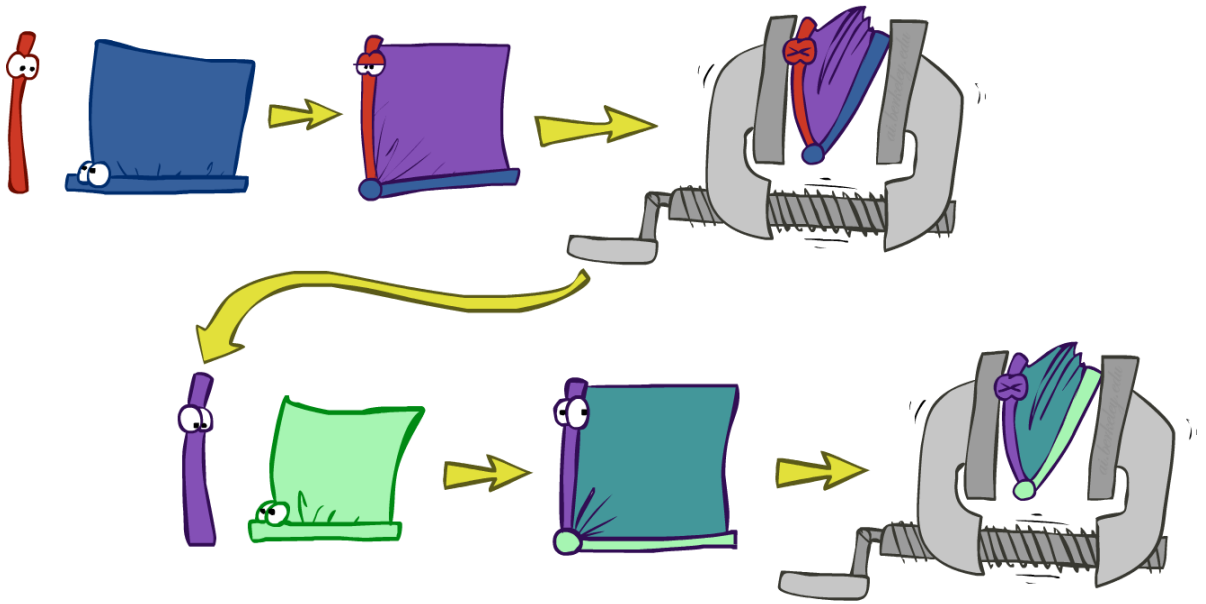
Enumeration is still inefficient: there are repeated computations!

- 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

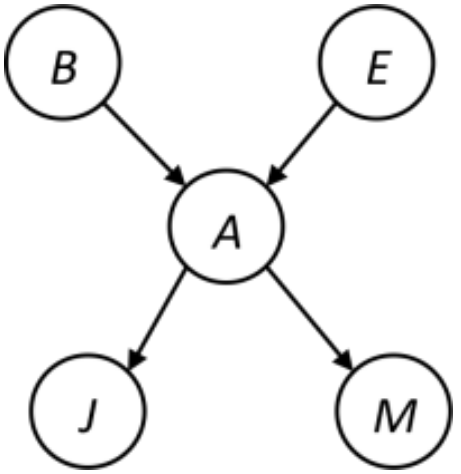
The variable elimination (VE) algorithm carries out summations right-to-left and stores intermediate results (called factors) to avoid recomputations. The algorithm interleaves:

- Joining sub-tables
- Eliminating hidden variables



—

Image credits: CS188, UC Berkeley.



Example

$$
\begin{aligned}
\mathbf{P}(B|j,m) &\propto \mathbf{P}(B,j,m) \\
&= \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e) P(j|a) P(m|a) \\
&= \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(e) \times \sum_a \mathbf{f}_3(a,B,e) \times \mathbf{f}_4(a) \times \mathbf{f}_5(a) \\
&= \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(e) \times \mathbf{f}_6(B,e) \quad (\text{sum out } A) \\
&= \mathbf{f}_1(B) \times \mathbf{f}_7(B) \quad (\text{sum out } E)
\end{aligned}
$$

Factors

- Each factor $\mathbf{f}_i$ is a multi-dimensional array indexed by the values of its argument variables. E.g.:

$$\mathbf{f}_4 = \mathbf{f}_4(A) = \begin{pmatrix} P(j|a) \\ P(j|\neg a) \end{pmatrix} = \begin{pmatrix} 0.90 \\ 0.05 \end{pmatrix}$$

$$\mathbf{f}_4(a) = 0.90$$

$$\mathbf{f}_4(\neg a) = 0.5$$

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

Join

The pointwise product $\times$, or join, of two factors $\mathbf{f}_1$ and $\mathbf{f}_2$ yields a new factor $\mathbf{f}_3$.

- Exactly like a database join!
- The variables of $\mathbf{f}_3$ are the union of the variables in $\mathbf{f}_1$ and $\mathbf{f}_2$.
- The elements of $\mathbf{f}_3$ are given by the product of the corresponding elements in $\mathbf{f}_1$ and $\mathbf{f}_2$.

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

Elimination

Summing out, or eliminating, a variable from a factor is done by adding up the sub-arrays formed by fixing the variable to each of its values in turn.

For example, to sum out A from $\mathbf{f}_3(A, B, C)$, we write:

$$\begin{aligned}\mathbf{f}(B, C) &= \sum_a \mathbf{f}_3(a, B, C) = \mathbf{f}_3(a, B, C) + \mathbf{f}_3(\neg a, B, C) \\ &= \begin{pmatrix} 0.06 & 0.24 \\ 0.42 & 0.28 \end{pmatrix} + \begin{pmatrix} 0.18 & 0.72 \\ 0.06 & 0.04 \end{pmatrix} = \begin{pmatrix} 0.24 & 0.96 \\ 0.48 & 0.32 \end{pmatrix}\end{aligned}$$

General Variable Elimination algorithm

Query: $\mathbf{P}(Q|e_1, ..., e_k)$.

1. Start with the initial factors (the local CPTs, instantiated by the evidence).
2. While there are still hidden variables:
    1. Pick a hidden variable H
    2. Join all factors mentioning H
    3. Eliminate H
3. Join all remaining factors
4. Normalize

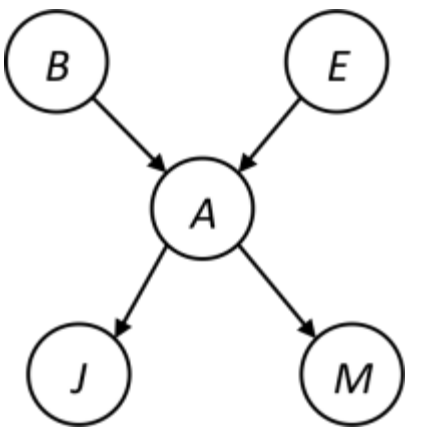
Relevance

Consider the query $\mathbf{P}(J|b)$:

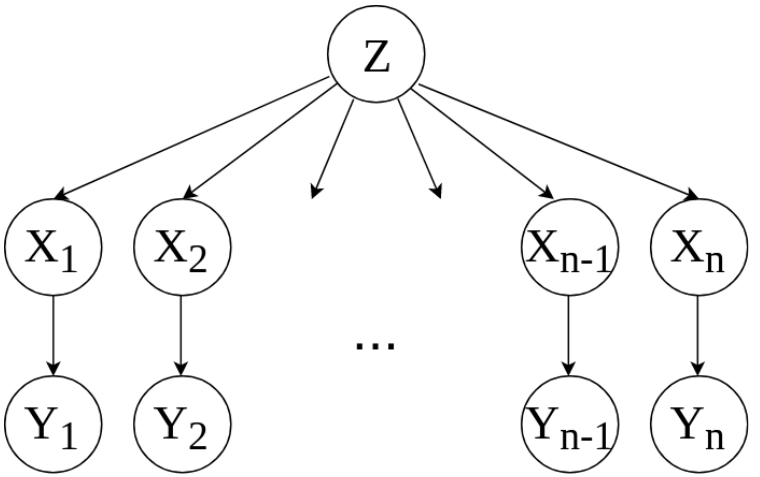
$$\mathbf{P}(J|b) \propto P(b) \sum_e P(e) \sum_a P(a|b,e)\mathbf{P}(J|a) \sum_m P(m|a)$$

- $\sum_m P(m|a) = 1$, therefore M is irrelevant for the query.
- In other words, $\mathbf{P}(J|b)$ remains unchanged if we remove M from the network.

Theorem. H is irrelevant for $\mathbf{P}(Q|e)$ unless $H \in \text{ancestors}(\{Q\} \cup E)$.



Complexity



Consider the query $\mathbf{P}(X_n|y_1, ..., y_n)$.

Work through the two elimination orderings:

- $Z, X_1, ..., X_{n-1}$
- $X_1, ..., X_{n-1}, Z$

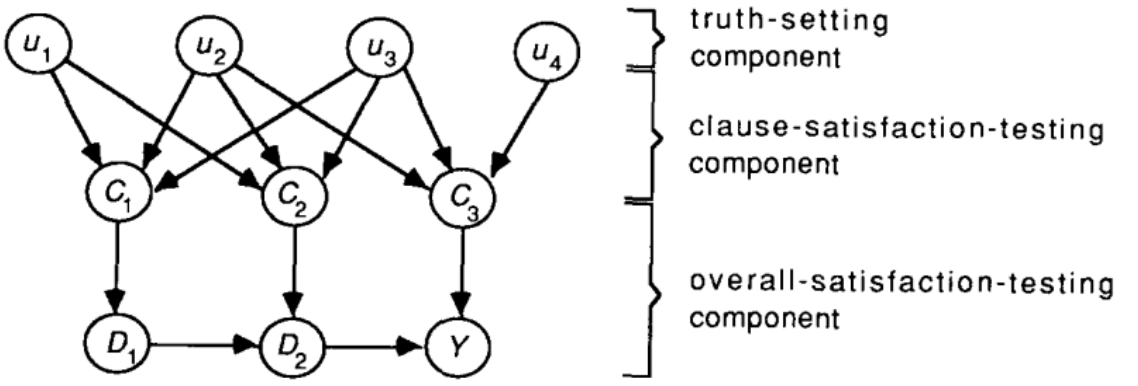
What is the size of the maximum factor generated for each of the orderings?

- Answer: 2^{n+1} vs. 2^2 (assuming boolean values)

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!
  - Greedy heuristic: eliminate whichever variable minimizes the size of the factor to be constructed.
  - 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:

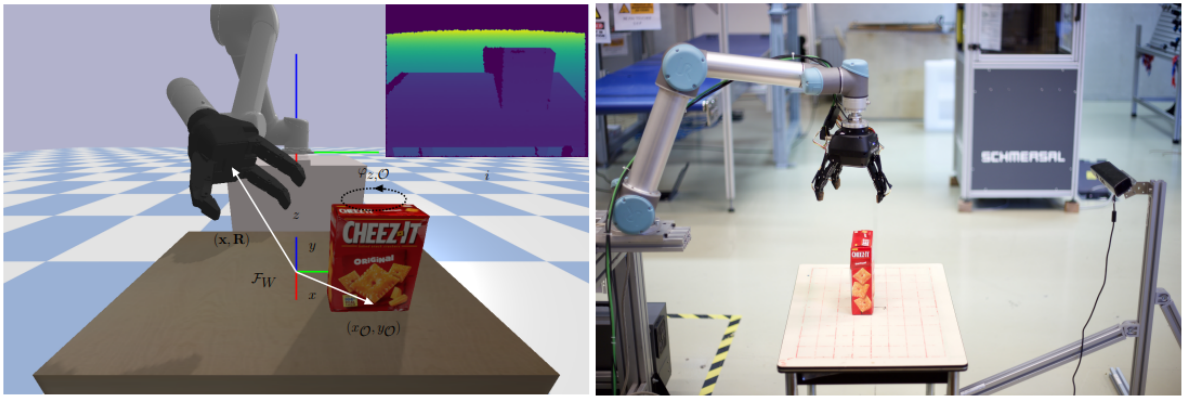
- CSP: $(u_1 \vee u_2 \vee u_3) \wedge (\neg u_1 \vee \neg u_2 \vee u_3) \wedge (u_2 \vee \neg u_3 \vee u_4)$
- $P(U_i = 0) = P(U_i = 1) = 0.5$
- $C_1 = U_1 \vee U_2 \vee U_3; C_2 = \neg U_1 \vee \neg U_2 \vee U_3; C_3 = U_2 \vee \neg U_3 \vee U_4$
- $D_1 = C_1; D_2 = D_1 \wedge C_2$
- $Y = D_2 \wedge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

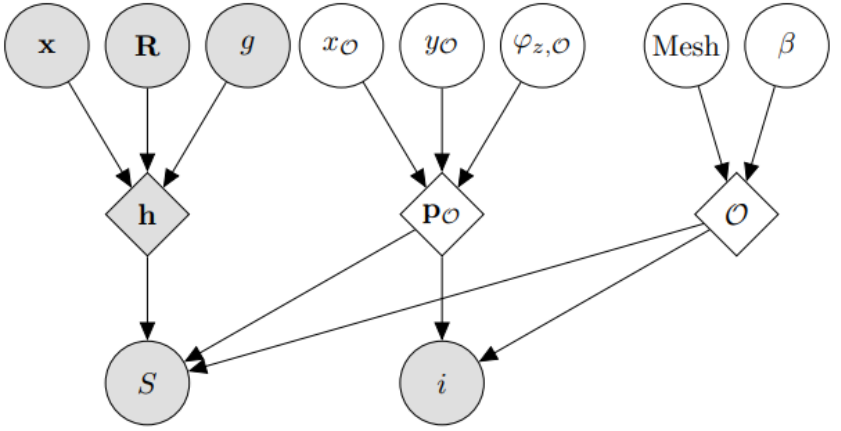
If we can answer whether $P(Y = 1) > 0$, then we answer whether 3SAT has a solution.

- By reduction, inference in Bayesian networks is therefore NP-complete.
- There is no known efficient probabilistic inference algorithm in general.

Approximate inference

Exact inference is intractable for most probabilistic models of practical interest. (e.g., involving many variables, continuous and discrete, undirected cycles, etc).





Variable	Prior
x	uniform$(-0.15, 0.15)$
y	uniform$(-0.15, 0.15)$
z	uniform$(0.12, 0.34)$
$\mathbf{R}$	mixture of power spherical(μ_i, κ)
g	categorical$(\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\})$
x_O	uniform$(-0.05, 0.05)$
y_O	uniform$(-0.05, 0.05)$
$\varphi_{z,O}$	uniform$(-\pi, \pi)$
Mesh	uniform in the set of objects
β	uniform(0.9, 1.1)

Solution

Abandon exact inference and develop approximate but faster inference algorithms:

- Sampling methods: produce answers by repeatedly generating random numbers from a distribution of interest.
- Variational methods: formulate inference as an optimization problem.
- Belief propagation methods: formulate inference as a message-passing algorithm.
- Machine learning methods: learn an approximation of the target distribution from training examples.

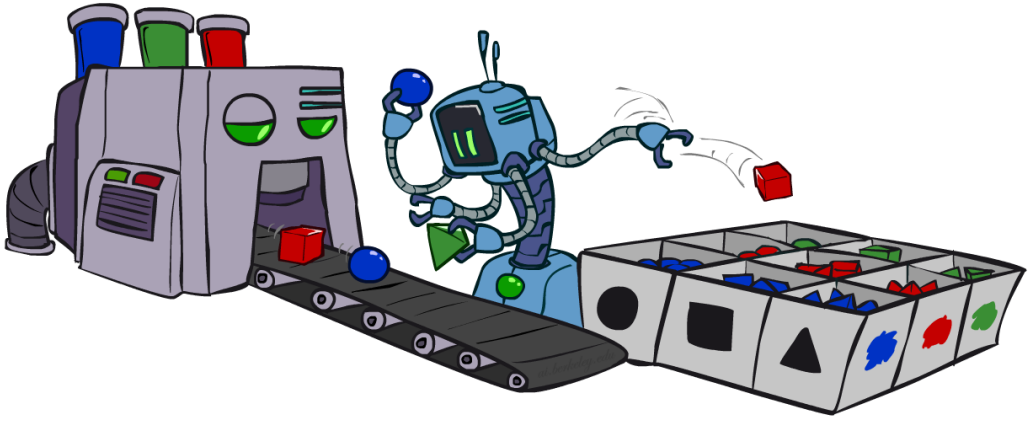
Sampling methods

Basic idea:

- Draw N samples from a sampling distribution S.
- Compute an approximate posterior probability $\hat{P}$.
- Show this approximate converges to the true probability distribution P.

Why sampling?

Generating samples is often much faster than computing the right answer (e.g., with variable elimination).



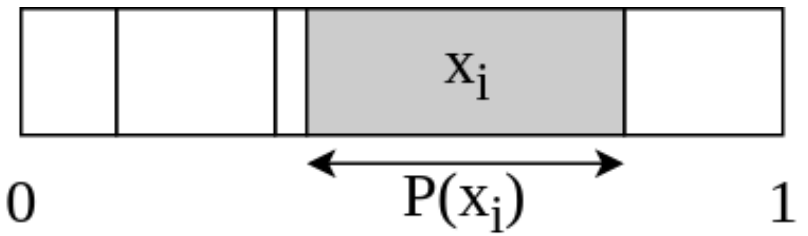
—

Image credits: CS188, UC Berkeley.

Sampling

How to sample from the distribution of a discrete variable X?

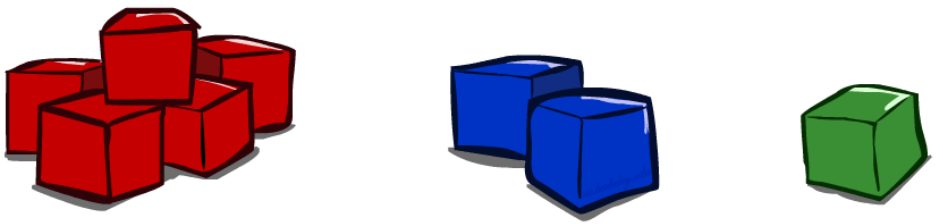
- Assume k discrete outcomes $x_1, \ldots, x_k$ with probability $P(x_i)$.
- Assume sampling from the uniform $\mathcal{U}(0, 1)$ is possible.
  - e.g., as enabled by a standard `rand()` function.
- Divide the $[0, 1]$ interval into k regions, with region i having size $P(x_i)$.
- Sample $u \sim \mathcal{U}(0, 1)$ and return the value associated to the region in which u falls.



$$P(C)$$

C	P
red	0.6
green	0.1
blue	0.3

$$0 \leq u < 0.6 \rightarrow C = \text{red}$$
$$0.6 \leq u < 0.7 \rightarrow C = \text{green}$$
$$0.7 \leq u < 1 \rightarrow C = \text{blue}$$



—
Image credits: CS188, UC Berkeley.

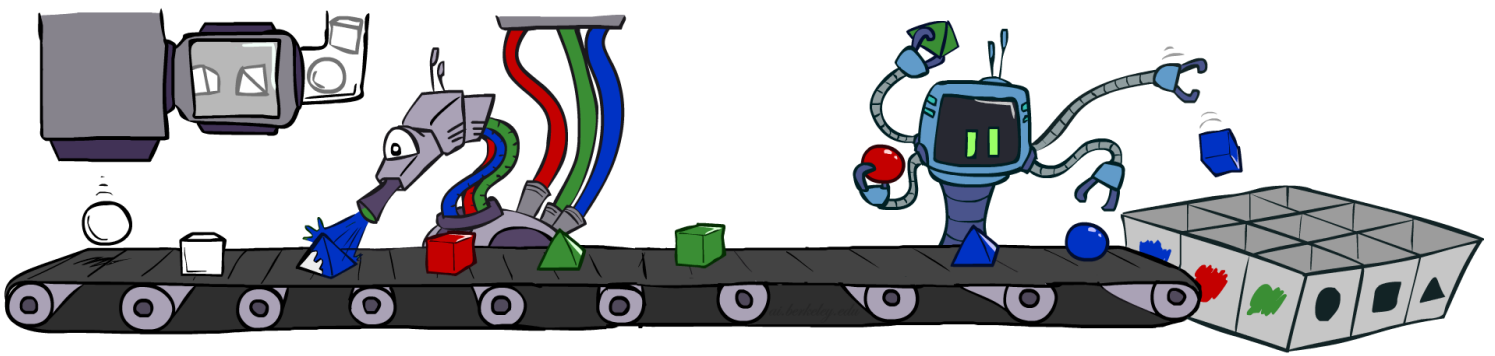
Prior sampling

Sampling from a Bayesian network, without observed evidence:

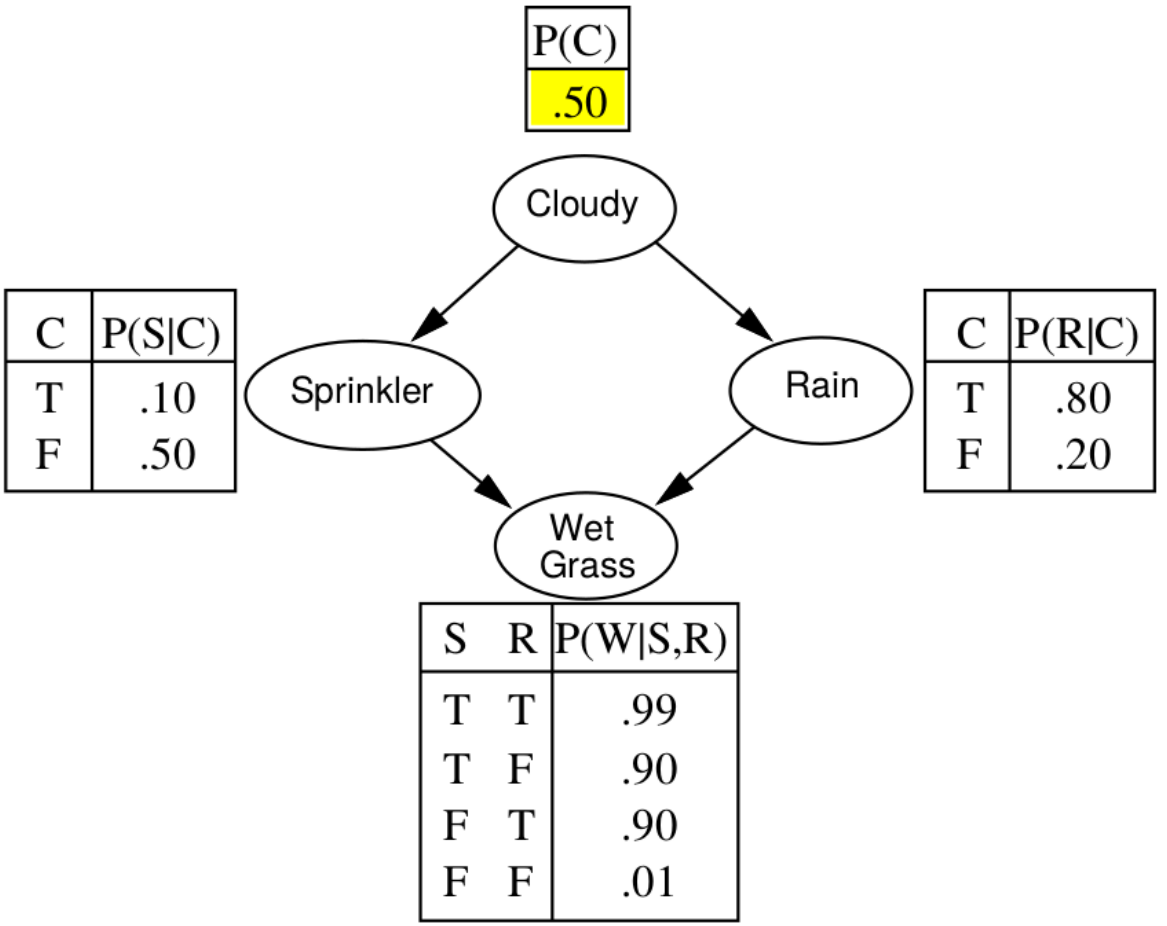
- Sample each variable in turn, in topological order.
- The probability distribution from which the value is sampled is conditioned on the values already assigned to the variable's parents.

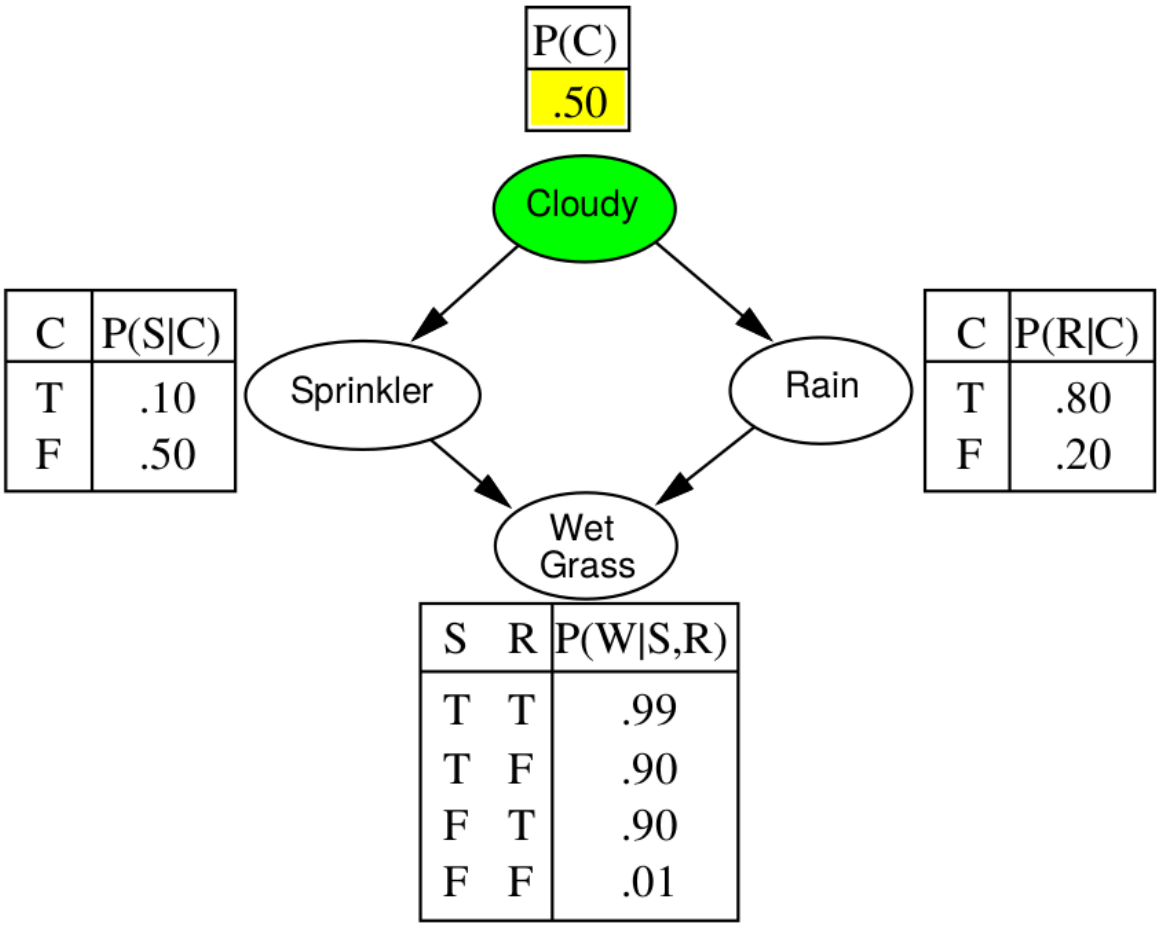
function PRIOR-SAMPLE(bn) **returns** an event sampled from the prior specified by bn
inputs: bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

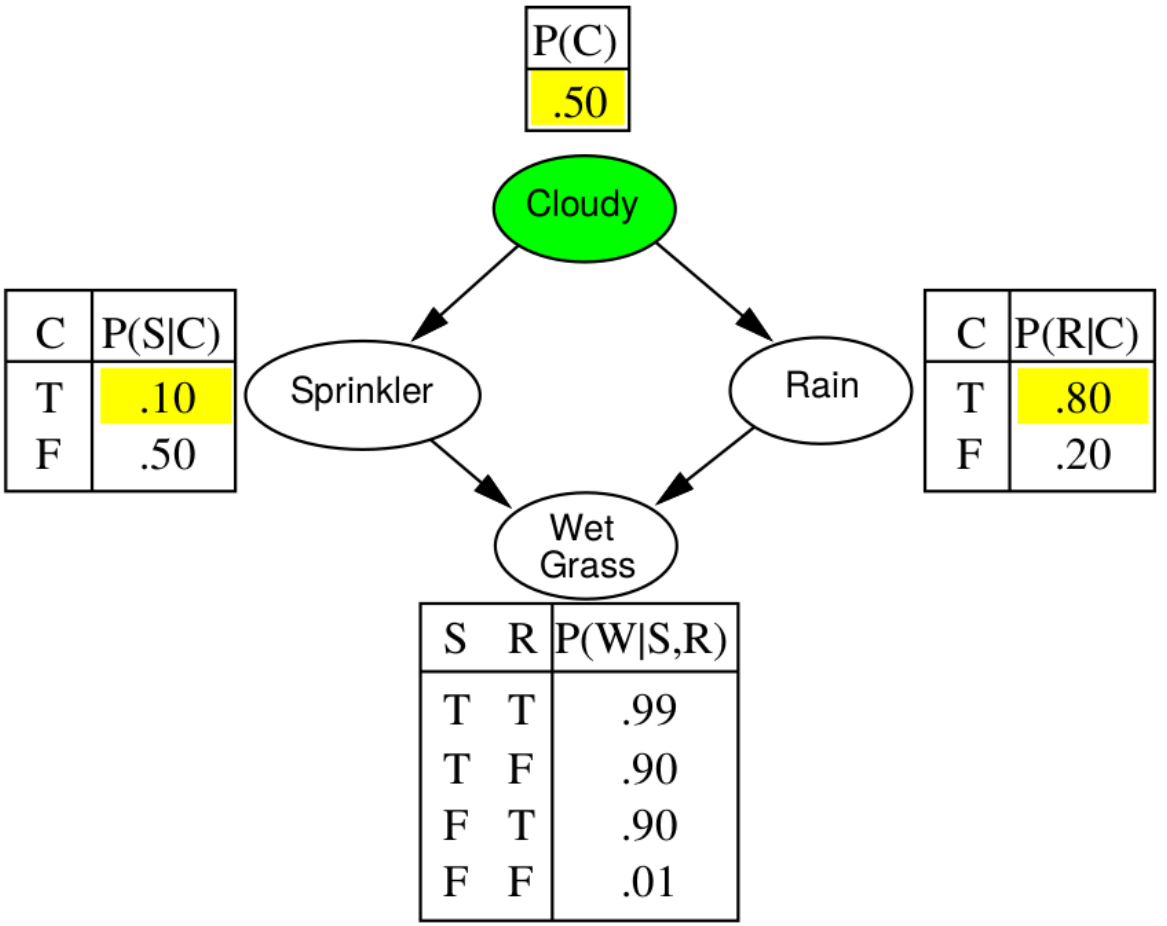
$\mathbf{x} \leftarrow$ an event with n elements
foreach variable X_i **in** $X_1, \ldots, X_n$ **do**
    $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
return $\mathbf{x}$

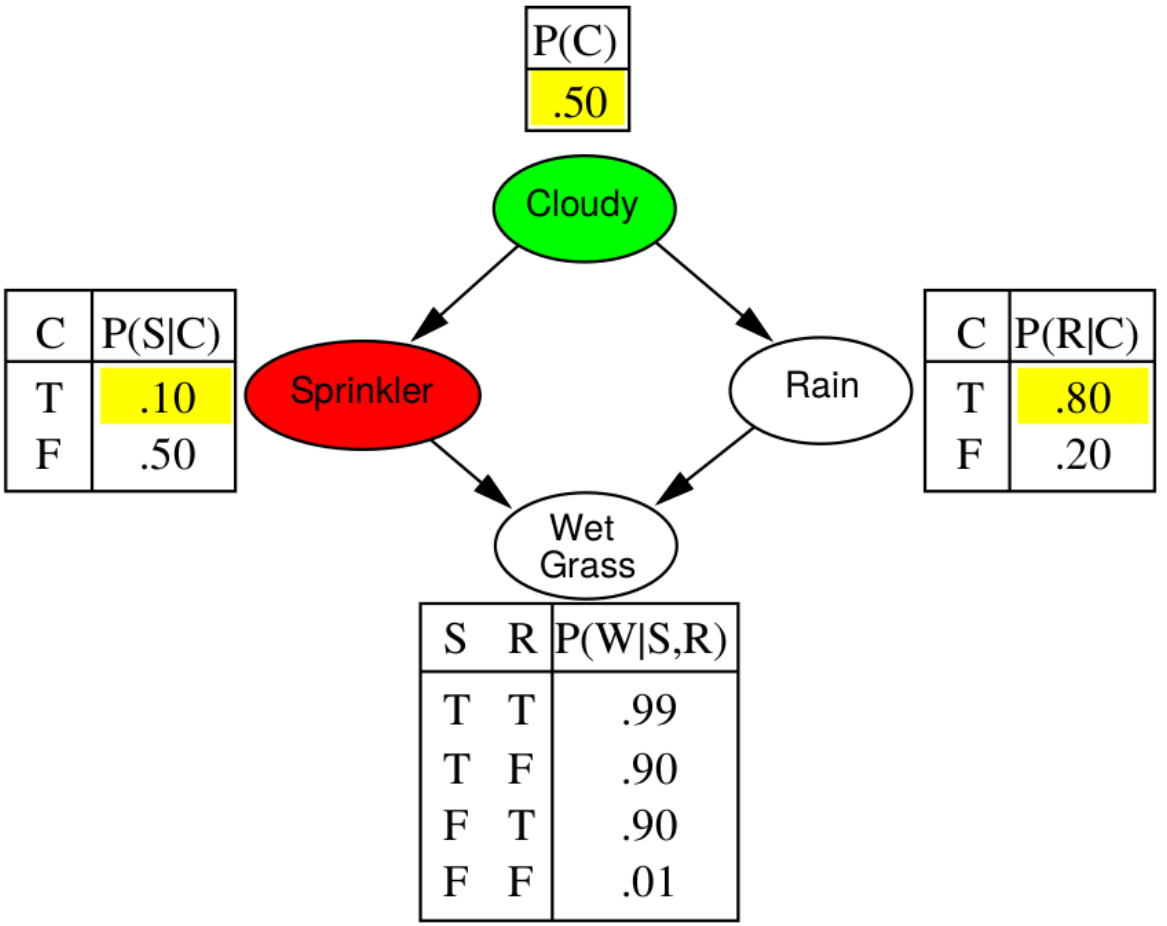


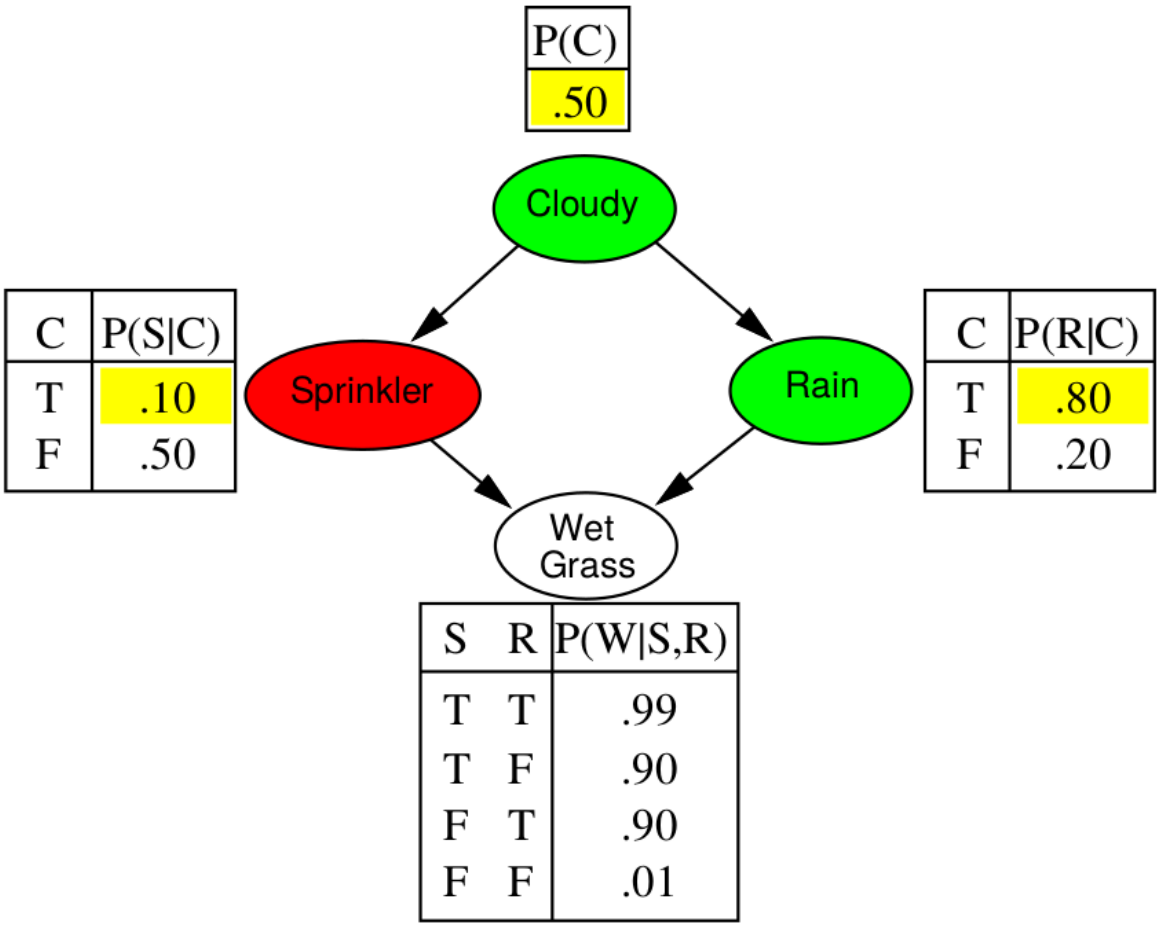
—
Image credits: CS188, UC Berkeley.

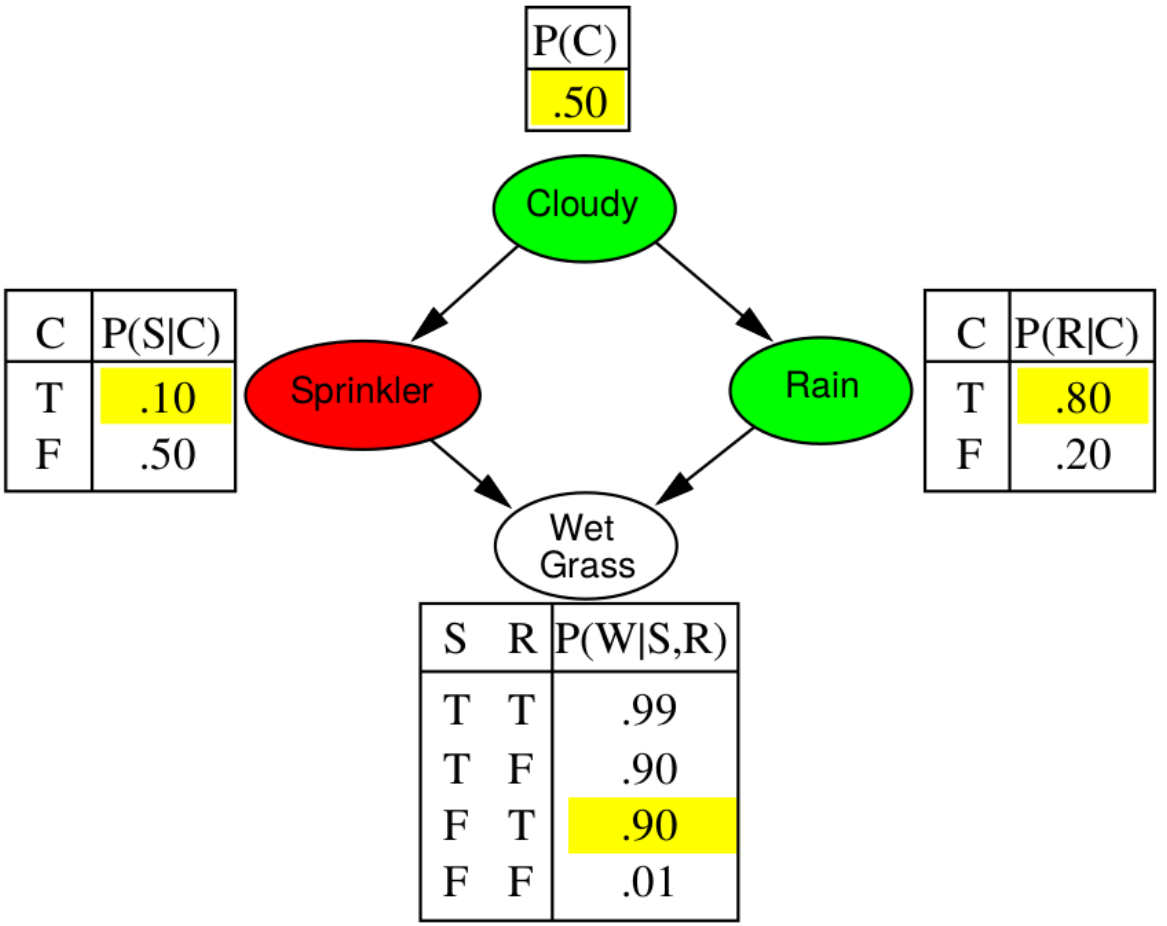

P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01

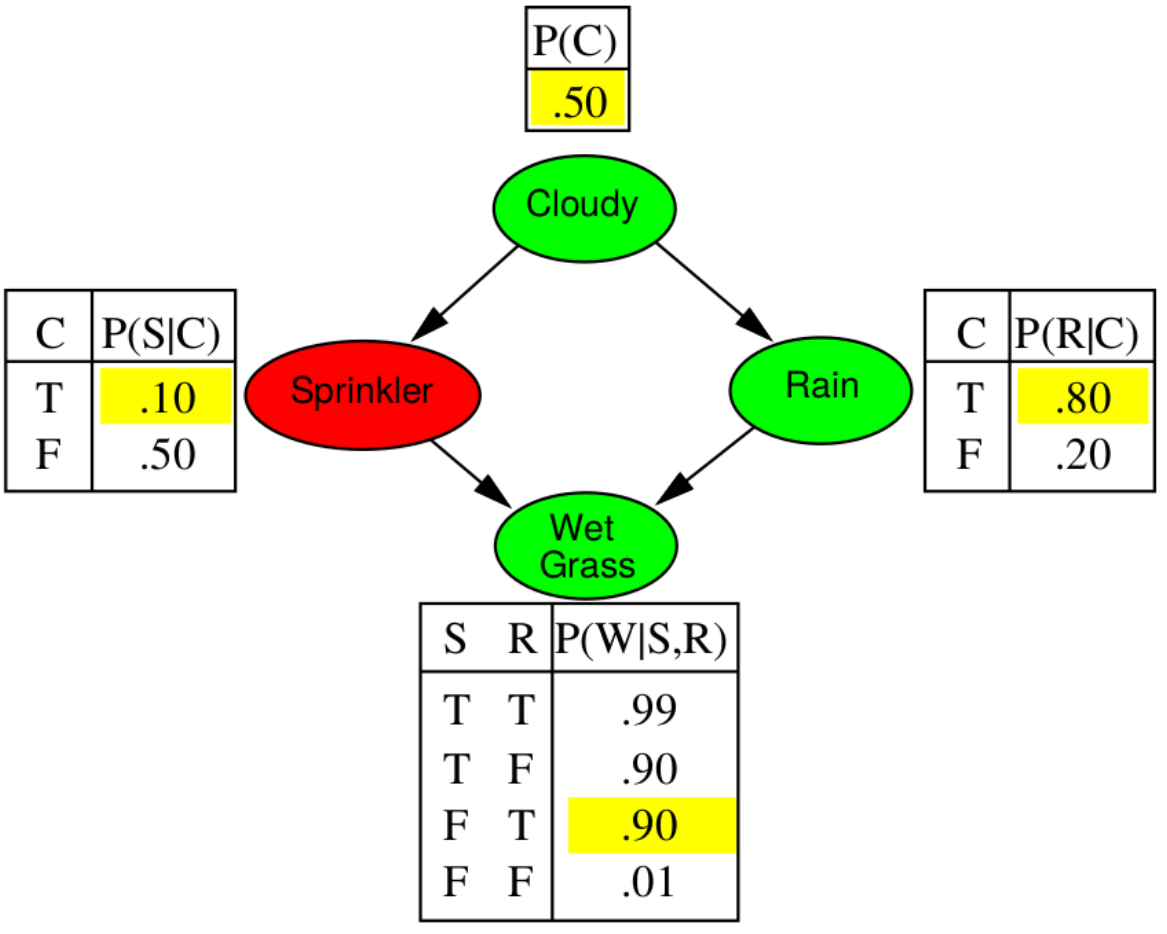

P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01


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


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


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


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


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

We will collect a bunch of samples from the Bayesian network:

$c, \neg s, r, w$
c, s, r, w
$\neg c, s, r, \neg w$
$c, \neg s, r, w$
$\neg c, \neg s, \neg r, w$

If we want to know $\mathbf{P}(W)$:

- We have counts $\langle w : 4, \neg w : 1 \rangle$
- Normalize to obtain $\hat{\mathbf{P}}(W) = \langle w : 0.8, \neg w : 0.2 \rangle$
- $\hat{\mathbf{P}}(W)$ will get closer to the true distribution $\mathbf{P}(W)$ as we generate more samples.

Analysis

The probability that prior sampling generates a particular event is

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

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

Let $N_{\mathrm{PS}}(x_1, ..., x_n)$ denote the number of samples of an event. We define the probability estimator

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

Then,

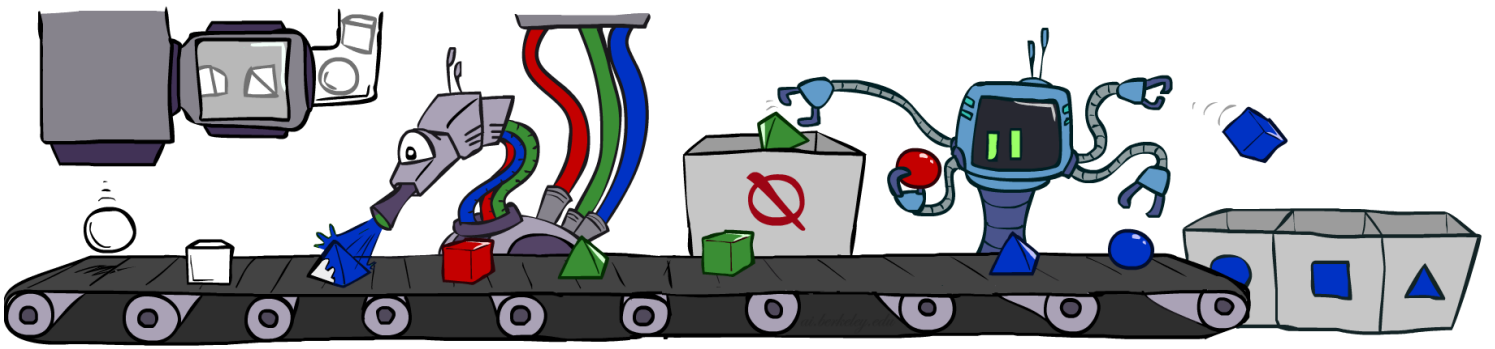
$$\begin{aligned}\lim_{N\to\infty} \hat{P}(x_1, ..., x_n) &= \lim_{N\to\infty} N_{\text{PS}}(x_1, ..., x_n)/N \\ &= S_{\text{PS}}(x_1, ..., x_n) \\ &= P(x_1, ..., x_n)\end{aligned}$$

Therefore, prior sampling is consistent:

$$P(x_1, ..., x_n) \approx N_{\text{PS}}(x_1, ..., x_n)/N$$

Rejection sampling

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



—

Image credits: CS188, UC Berkeley.

function REJECTION-SAMPLING(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
    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**)

Analysis

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

$$\begin{aligned}\hat{P}(x|e) &= N_{\text{PS}}(x,e)/N_{\text{PS}}(e) \\ &= \frac{N_{\text{PS}}(x,e)}{N} \Big/ \frac{N_{\text{PS}}(e)}{N} \\ &\approx P(x,e)/P(e) \\ &= P(x|e)\end{aligned}$$

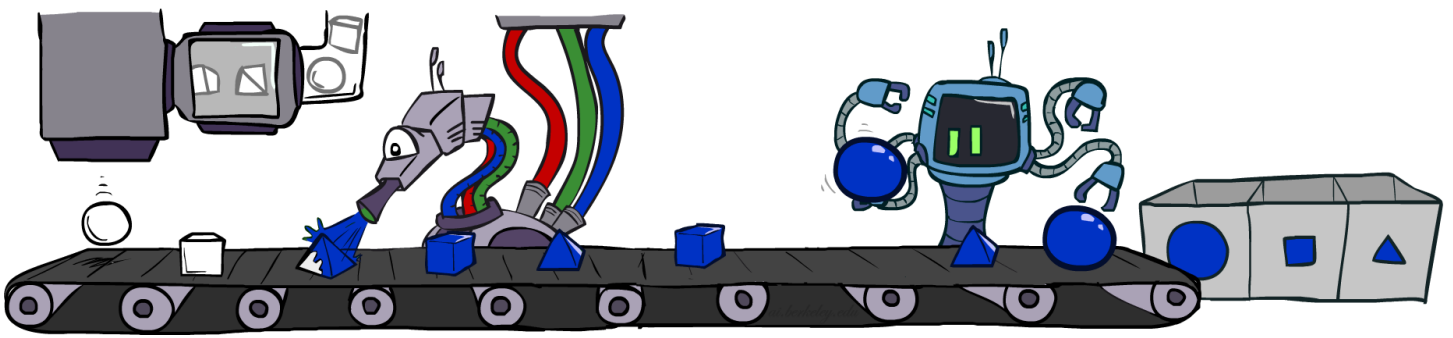
Therefore, rejection sampling is consistent.

- The standard deviation of the error in each probability is $O(1/\sqrt{n})$, where n is the number of samples used to compute the estimate.
- Problem: many samples are rejected!
  - Hopelessly expensive if the evidence is unlikely, i.e. if $P(e)$ is small.
  - Evidence is not exploited when sampling.

Likelihood weighting

Idea: clamp the evidence variables, sample the rest.

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



—

Image credits: CS188, UC Berkeley.

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

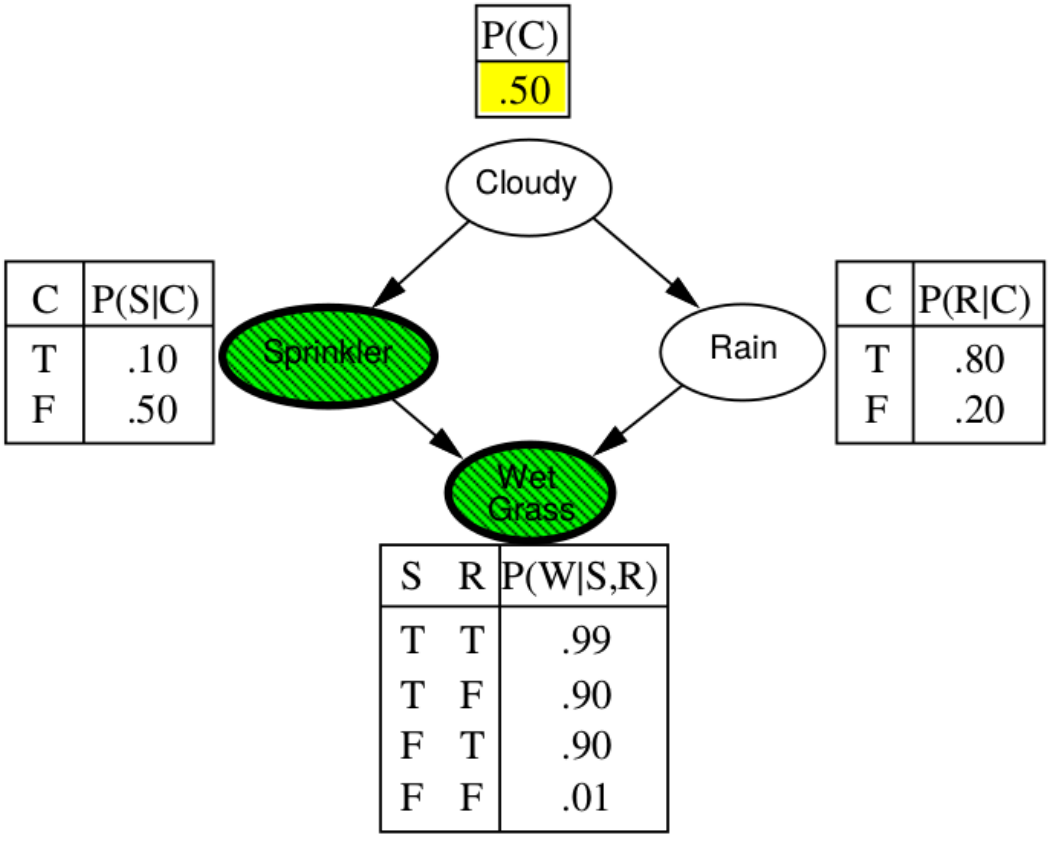
local variables: **W**, a vector of weighted counts for each value of X, initially zero

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

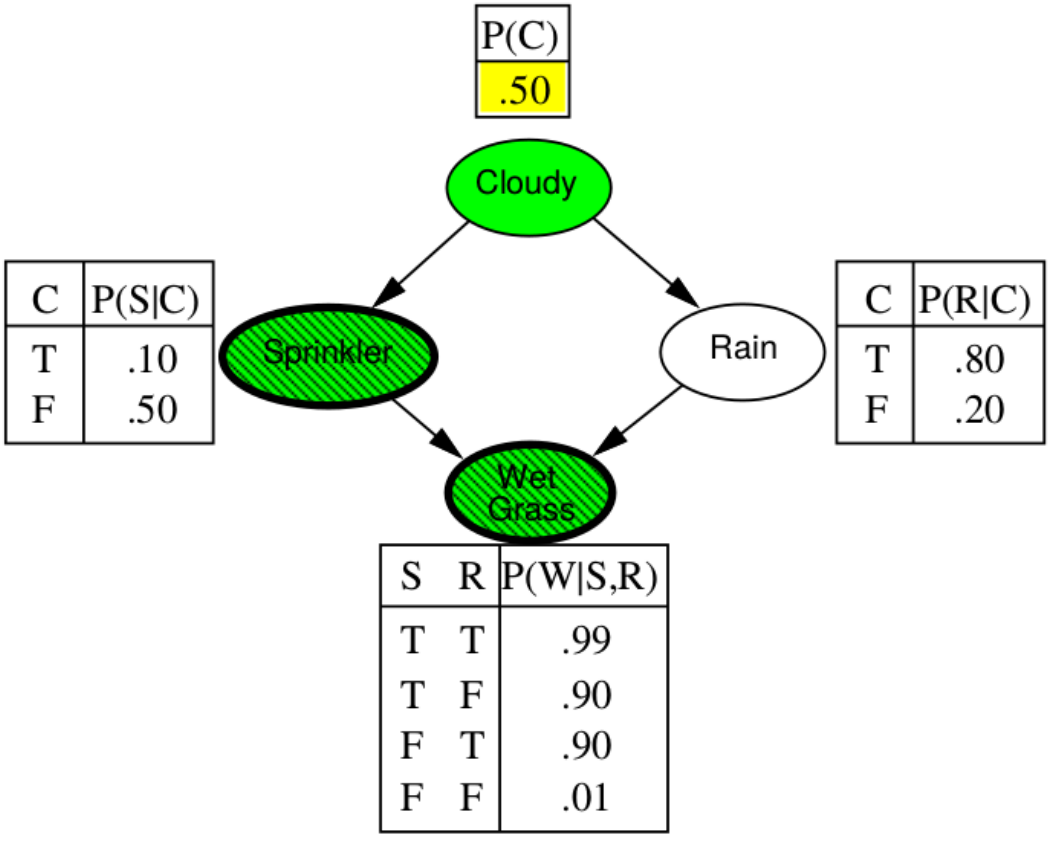
---

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

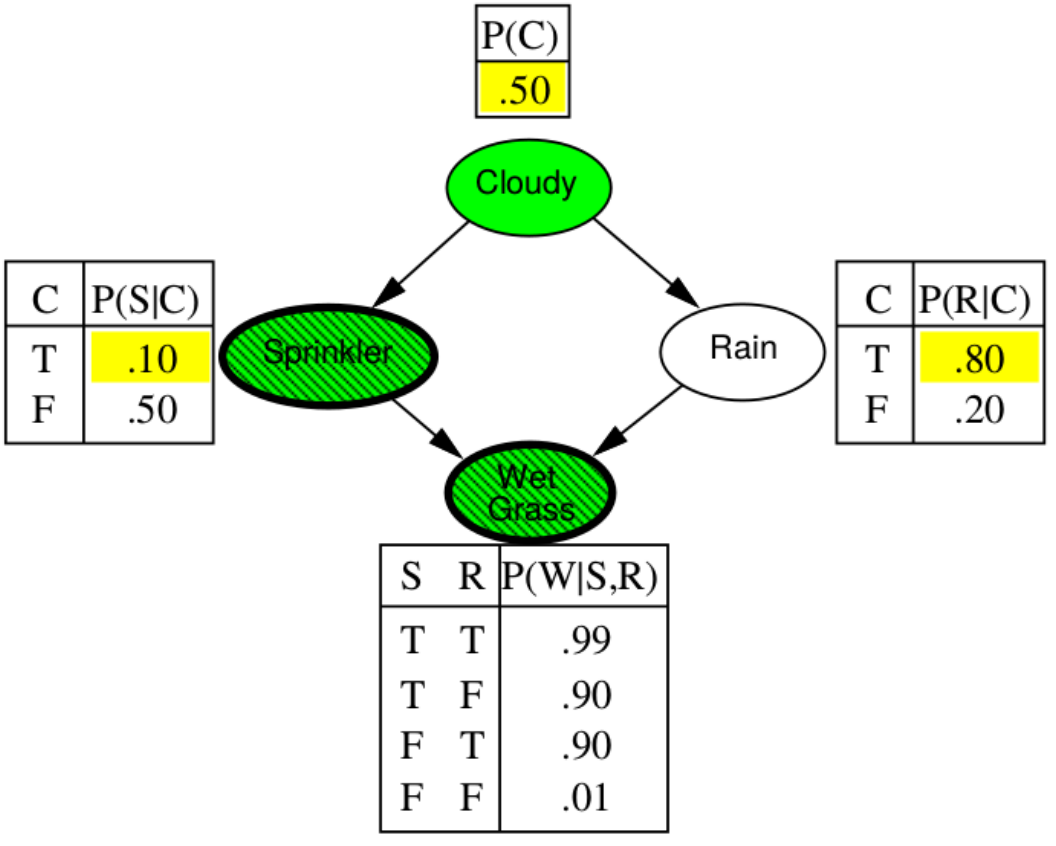
$w \leftarrow 1$; **x** $\leftarrow$ an event with n elements initialized from **e**
foreach variable X_i **in** $X_1, \ldots, X_n$ **do**
  if X_i is an evidence variable with value x_i in **e**
    then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
    else $\mathbf{x}[i] \leftarrow$ a random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


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

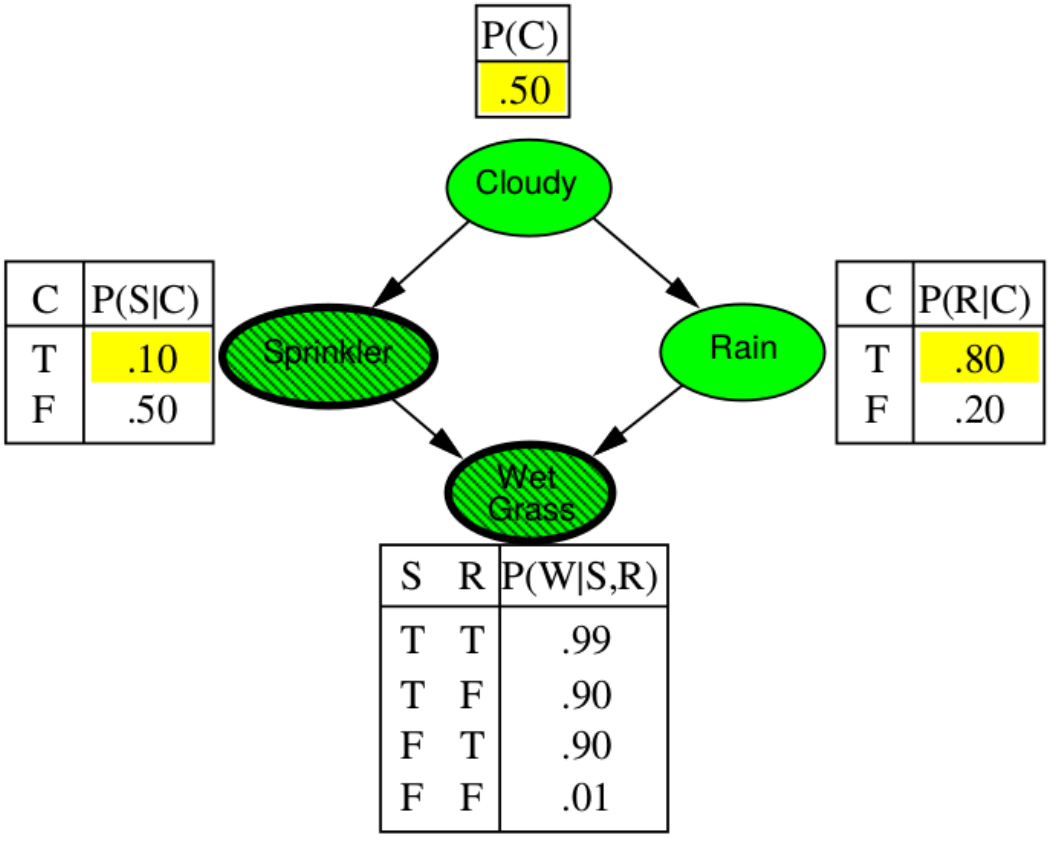
$w = 1.0$


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

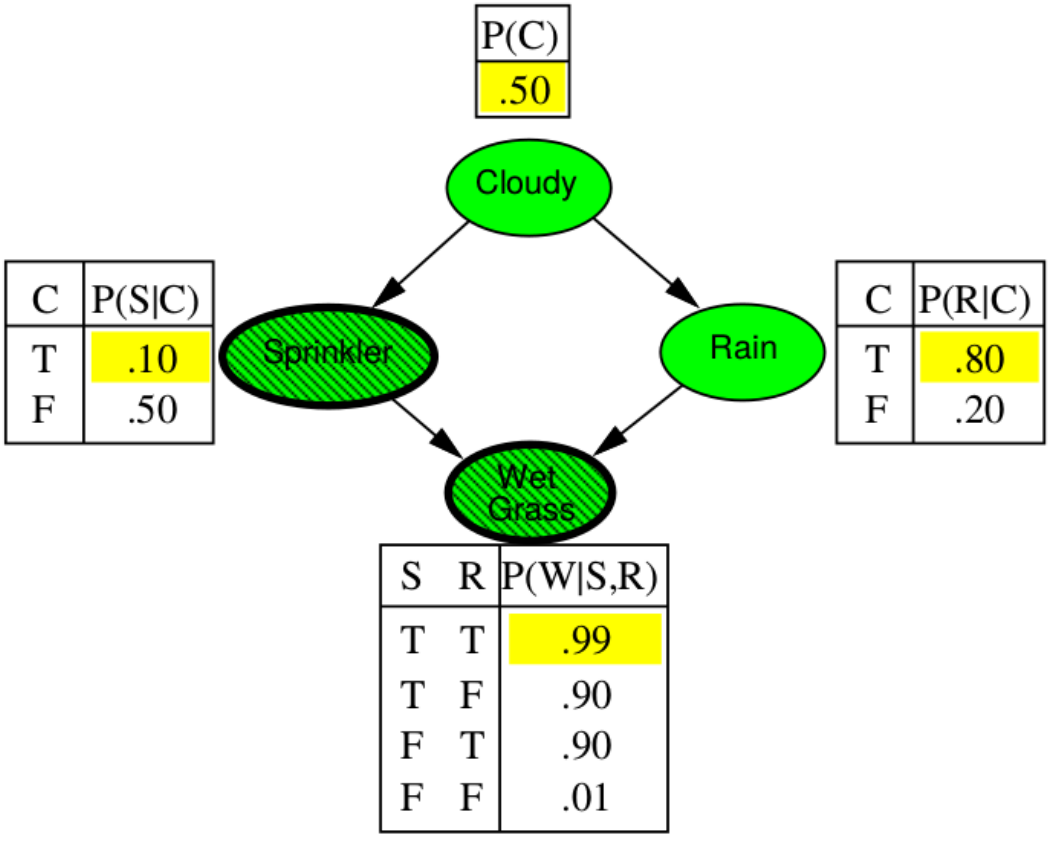
$w = 1.0$


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

$w = 1.0$


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

$w = 1.0 \times 0.1$



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

Analysis

The sampling probability for an event with likelihood weighting is

$$S_{\text{WS}}(x, e) = \prod_{i=1}^{l} P(x_i | \text{parents}(X_i)),$$

where the product is over the non-evidence variables. The weight for a given sample x, e is

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

where the product is over the evidence variables.

The weighted sampling probability is

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

The estimated posterior probability is computed as follows:

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

where α and α' are normalization constants.

Hence likelihood weighting returns consistent estimates.

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

Inference by Markov chain simulation

- Markov chain Monte Carlo (MCMC) algorithms are a family of sampling algorithms that generate samples through a Markov chain.
- They generate a sequence of samples by making random changes to a preceding sample, instead of generating each sample from scratch.
- Helpful to think of a Bayesian network as being in a particular current state specifying a value for each variable and generating a next state by making random changes to the current state.
- Metropolis-Hastings is one of the most famous MCMC methods, of which Gibbs sampling is a special case.

Gibbs sampling

- Start with an arbitrary instance $x_1, \ldots, x_n$ consistent with the evidence.
- Sample one variable at a time, conditioned on all the rest, but keep the evidence fixed.
- Keep repeating this for a long time.

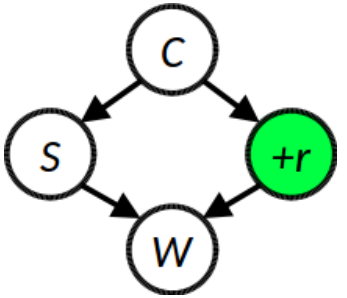
function GIBBS-ASK(X, **e**, bn, N) **returns** an estimate of **P**(X|**e**)
  local variables: **N**, a vector of counts for each value of X, initially zero
    Z, the nonevidence variables in bn
    x, the current state of the network, initially copied from **e**

  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 **P**($Z_i|mb(Z_i)$)
      N$[x] \leftarrow$ **N**$[x] + 1$ where x is the value of X in **x**
  return NORMALIZE(**N**)

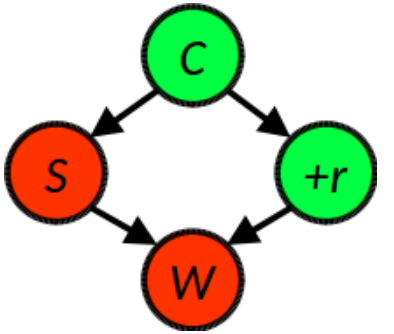
- Both upstream and downstream variables condition on evidence.
- In contrast, likelihood weighting only conditions on upstream evidence, and hence the resulting weights might be very small.

Example

1) Fix the evidence

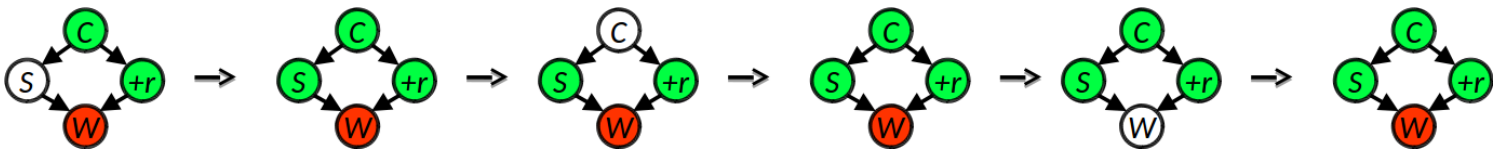


2) Randomly initialize the other variables



3) Repeat

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



Demo

See `code/lecture5-gibbs.ipynb`.

Rationale

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.

See 14.5.2 for a technical proof.

Summary

- Exact inference by variable elimination .
  - NP-complete 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.
  - Likelihood weighting does poorly when there is lots of evidence.
  - Likelihood weighting and Gibbs sampling are generally insensitive to topology.
  - Convergence can be slow with probabilities close to 1 or 0.
  - Can handle arbitrary combinations of discrete and continuous variables.

The end.

References

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