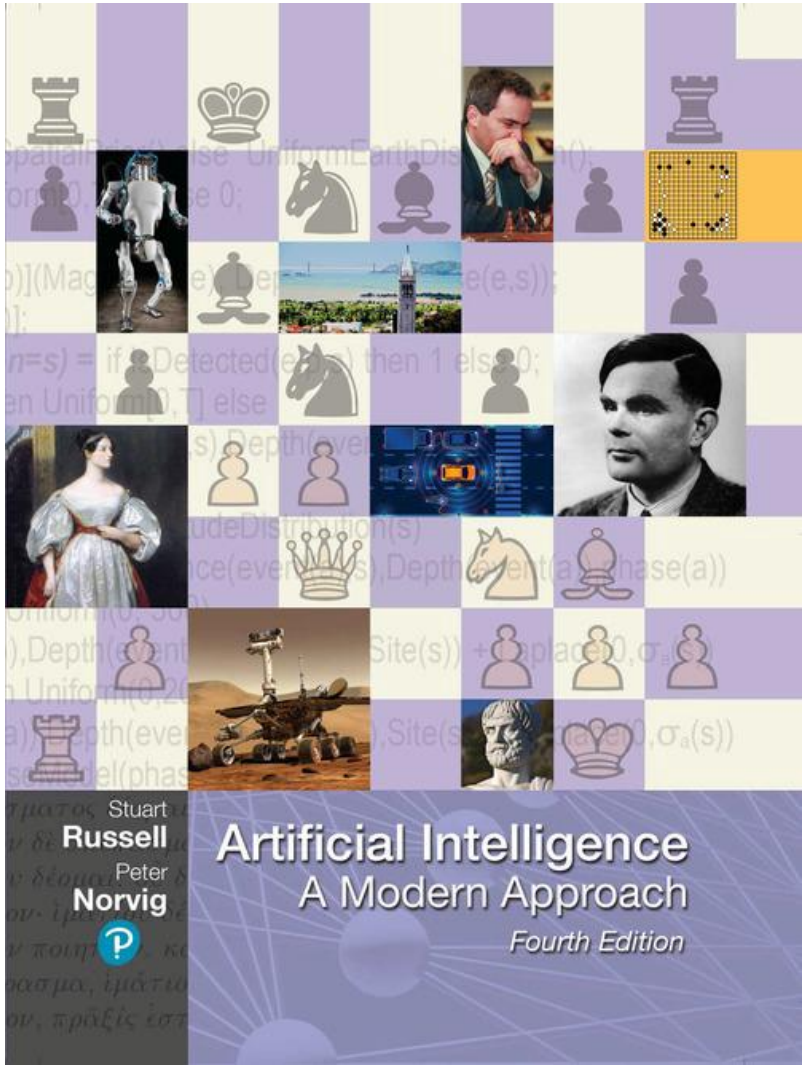
Artificial Intelligence Fundamentals

2023-2024



"The problem with monotonic logic lies not in the hardness of its truth values, but rather in its inability to process context-dependent information."

— **Judea Pearl**

Chapter 13

Probabilistic Reasoning

Outline

- Representing Knowledge in an Uncertain Domain
- Semantics of Bayesian Networks
- Exact Inference in Bayesian Networks
- Approximate Inference for Bayesian Networks
- Causal Networks

Representing Knowledge in an Uncertain Domain

Bayesian networks: represents dependencies among variables.

A simple, directed graph in which each node is annotated with quantitative probability information

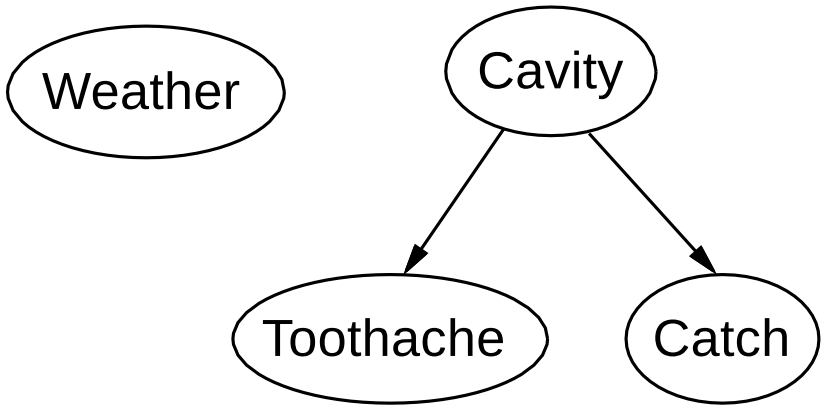
Syntax:

- a set of nodes, one per variable
- a directed, acyclic graph (link ≈ "*directly influences*")
- a conditional distribution for each node given its parents: $\mathbf{P}(X_i \mid Parents(X_i))$

In the simplest case, conditional distribution represented as a Conditional Probability Table (CPT) giving the distribution over X_i for each combination of parent values

Example

Topology of network encodes conditional independence assertions:



Weather is independent of the other variables

Toothache and *Catch* are conditionally independent given *Cavity*

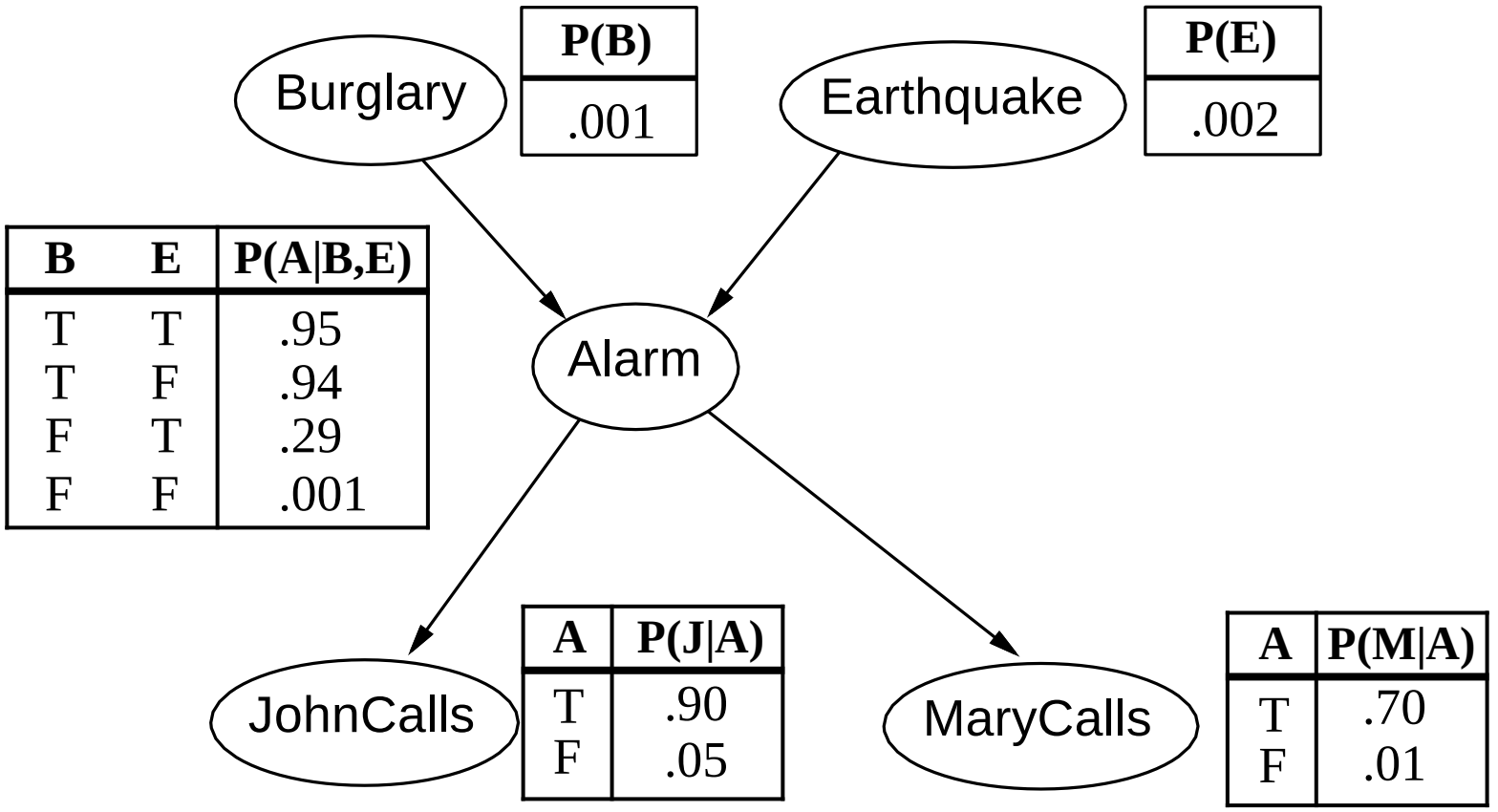
Example

I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

Variables: *Burglar*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*
Network topology reflects "causal" knowledge:

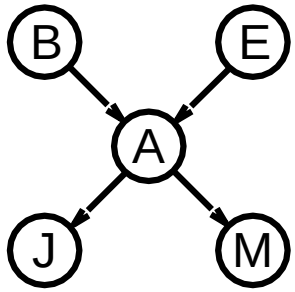
- A burglar can set the alarm off
- An earthquake can set the alarm off
- The alarm can cause Mary to call
- The alarm can cause John to call

Example contd.



Compactness

A CPT for Boolean X_i with k Boolean parents has 2^k rows for the combinations of parent values



Each row requires one number p for $X_i = true$
(the number for $X_i = false$ is just $1 - p$)

If each variable has no more than k parents,
the complete network requires $O(n \cdot 2^k)$ numbers

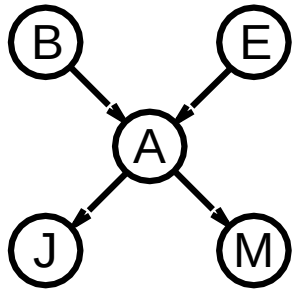
I.e., grows linearly with n, vs. $O(2^n)$ for the full joint distribution

For burglary net, $1 + 1 + 4 + 2 + 2 = 10$ numbers (vs. $2^5 - 1 = 31$)

The Semantics of Bayesian Networks

Global semantics defines the full joint distribution as the product of the local conditional distributions:

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

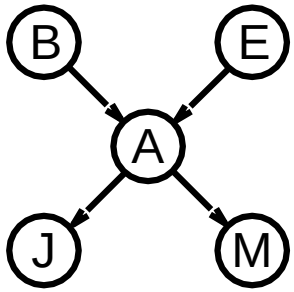


e.g., $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

$=$

Global semantics

"Global" semantics defines the full joint distribution as the product of the local conditional distributions:



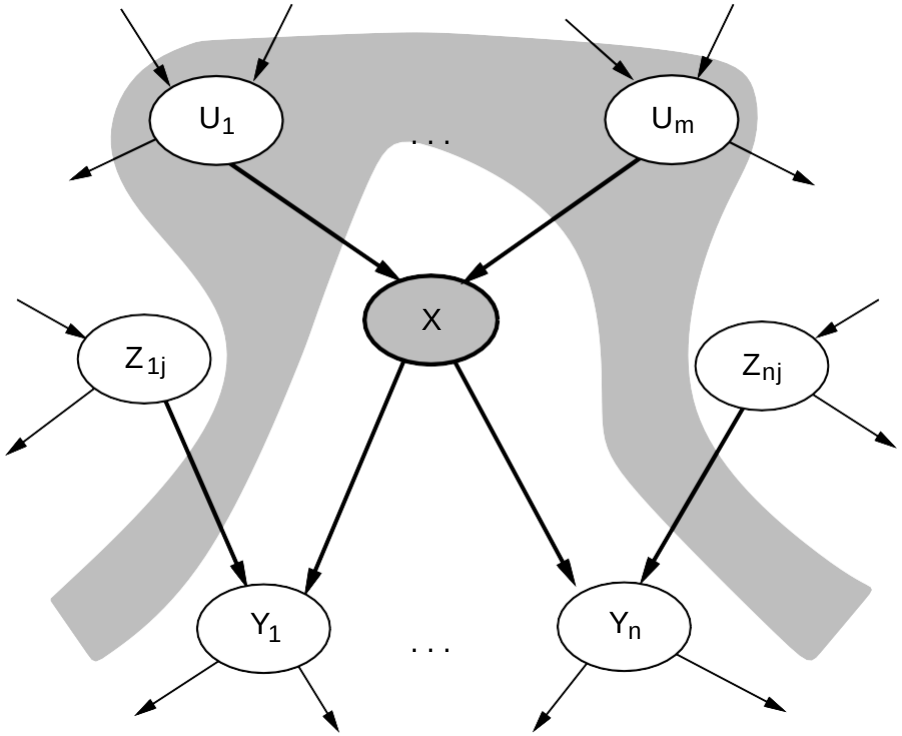
$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$

e.g., $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

$$\begin{aligned} &= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e) \\ &= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 \\ &\approx 0.00063 \end{aligned}$$

Local semantics

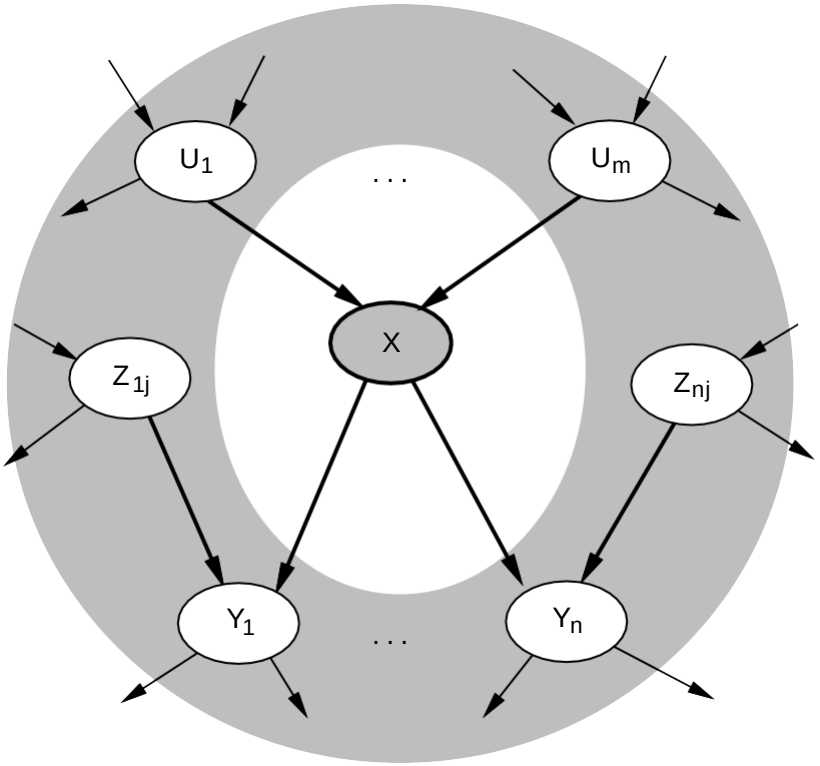
Local semantics: each node is conditionally independent of its nondescendants given its parents



Theorem: Local semantics $\Leftrightarrow$ global semantics

Markov blanket

Each node is conditionally independent of all others given its Markov blanket: parents + children + children's parents



Constructing Bayesian networks

Need a method such that a series of locally testable assertions of conditional independence guarantees the required global semantics

1. Choose an ordering of variables $X_1, \ldots, X_n$
2. For $i = 1$ to n
   add X_i to the network
   select parents from $X_1, \ldots, X_{i-1}$ such that
   $$\mathrm{P}(X_i | Parents(X_i)) = \mathrm{P}(X_i | X_1, \ldots, X_{i-1})$$

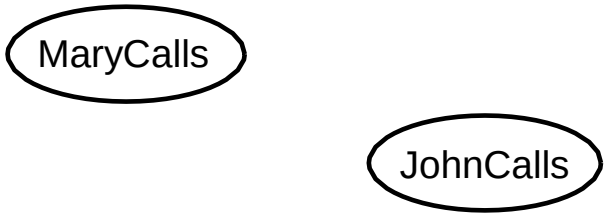
$P(a \wedge b) = P(a|b)\, P(b)$

This choice of parents guarantees the global semantics:

$$\mathrm{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathrm{P}(X_i | X_1, \ldots, X_{i-1}) \quad \text{(chain rule)}$$
$$= \prod_{i=1}^{n} \mathrm{P}(X_i | Parents(X_i)) \quad \text{(by construction)}$$

Example

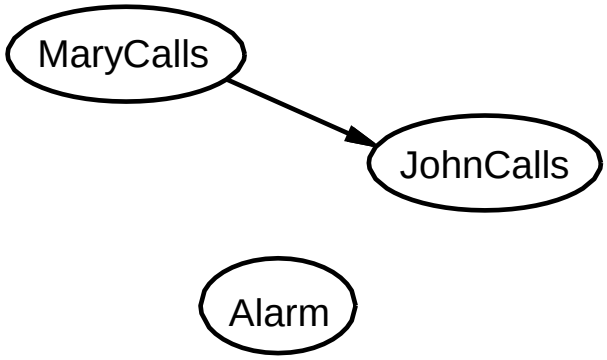
Suppose we choose the ordering M, J, A, B, E



$P(J|M) = P(J)$?

Example

Suppose we choose the ordering M, J, A, B, E

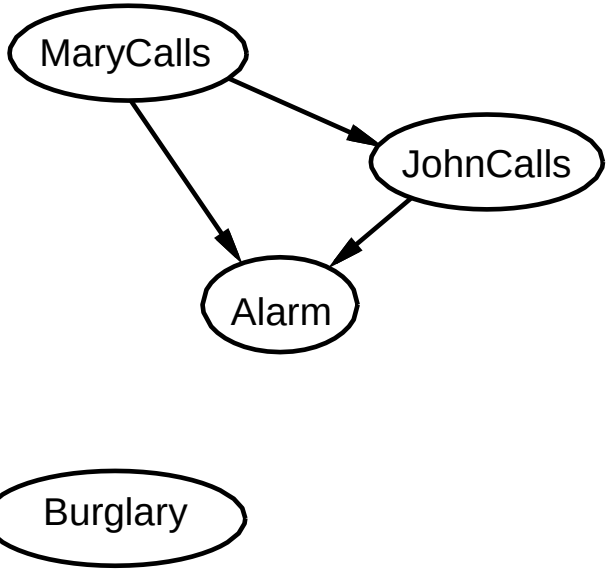


$P(J|M) = P(J)$? No

$P(A|J,M) = P(A|J)$? $P(A|J,M) = P(A)$?

Example

Suppose we choose the ordering M, J, A, B, E



$P(J|M) = P(J)$? No

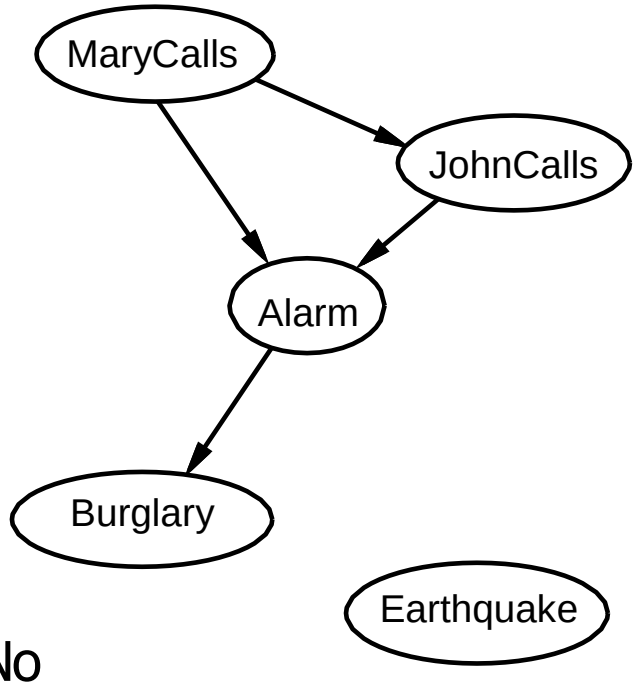
$P(A|J,M) = P(A|J)$? $P(A|J,M) = P(A)$? No

$P(B|A,J,M) = P(B|A)$?

$P(B|A,J,M) = P(B)$?

Example

Suppose we choose the ordering M, J, A, B, E



$P(J|M) = P(J)$? No

$P(A|J, M) = P(A|J)$? $P(A|J, M) = P(A)$? No

$P(B|A, J, M) = P(B|A)$? Yes

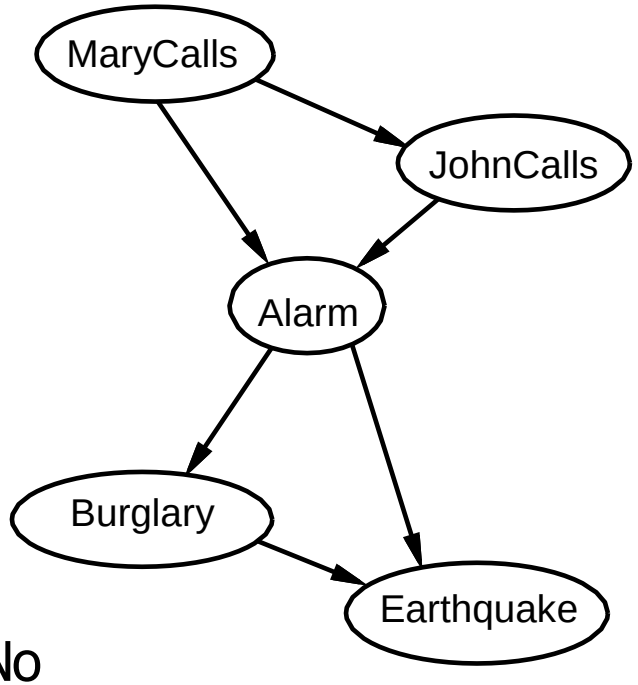
$P(B|A, J, M) = P(B)$? No

$P(E|B, A, J, M) = P(E|A)$?

$P(E|B, A, J, M) = P(E|A, B)$?

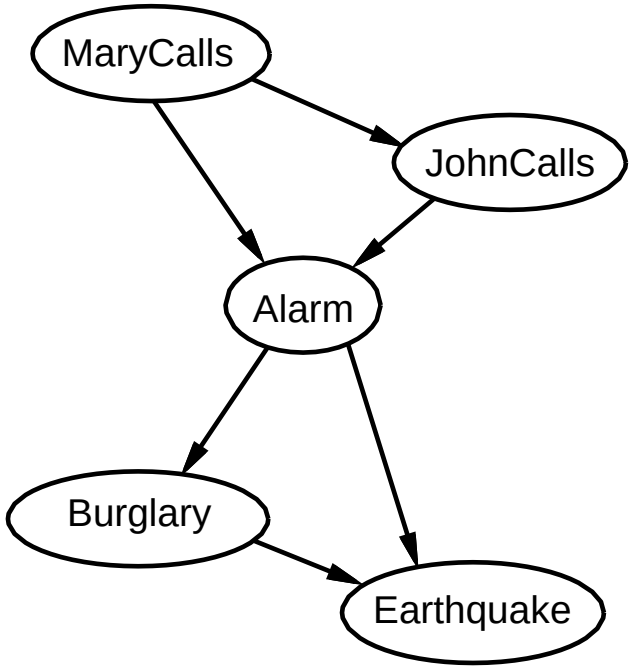
Example

Suppose we choose the ordering M, J, A, B, E



$P(J|M) = P(J)$? No
$P(A|J, M) = P(A|J)$? $P(A|J, M) = P(A)$? No
$P(B|A, J, M) = P(B|A)$? Yes
$P(B|A, J, M) = P(B)$? No
$P(E|B, A, J, M) = P(E|A)$? No
$P(E|B, A, J, M) = P(E|A, B)$? Yes

Example contd.



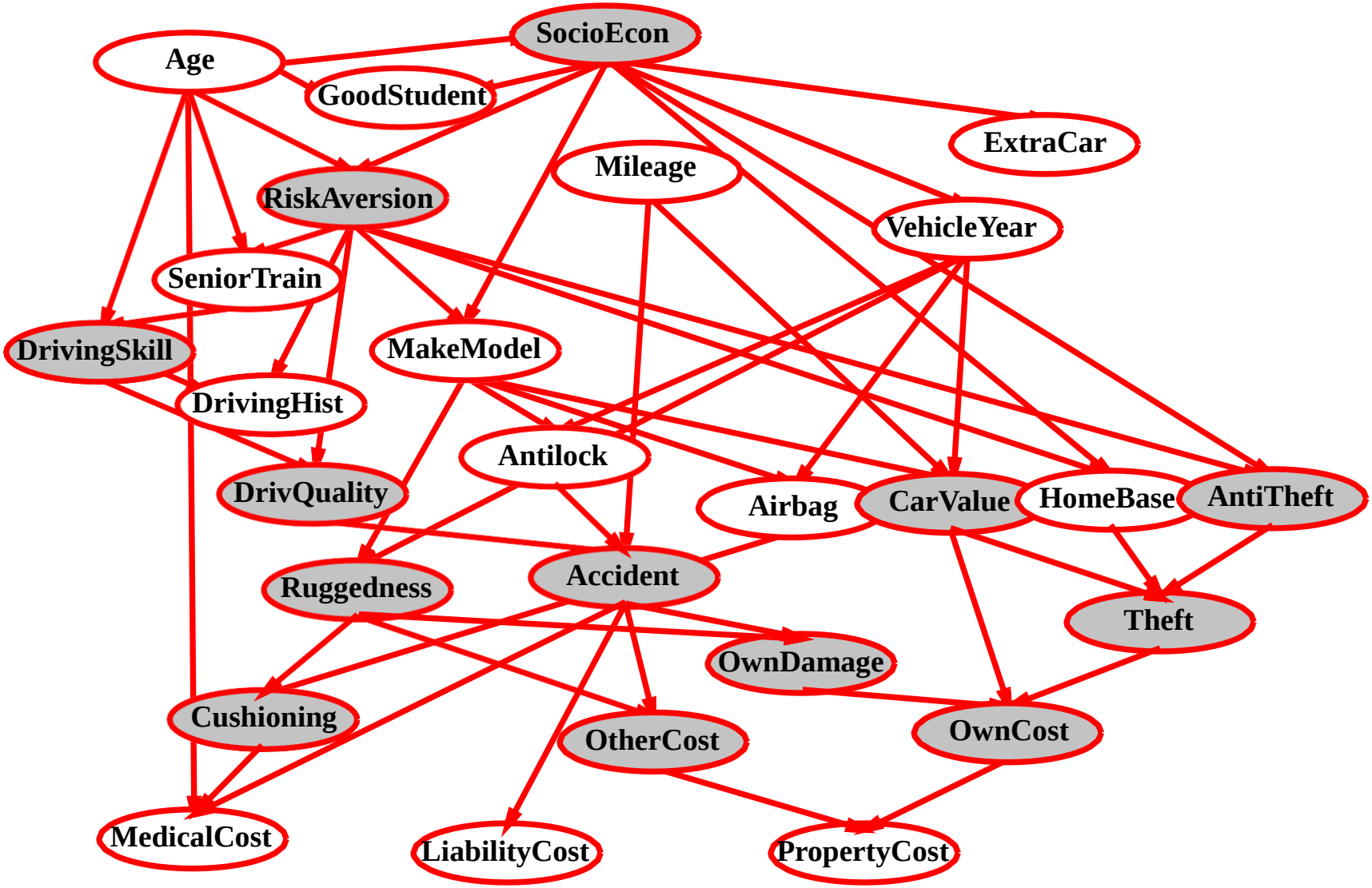
Deciding conditional independence is hard in noncausal directions

(Causal models and conditional independence seem hardwired for humans!)

Assessing conditional probabilities is hard in noncausal directions

Network is less compact: $1 + 2 + 4 + 2 + 4 = 13$ numbers needed

Example: Car insurance



Compact conditional distributions

CPT grows exponentially with number of parents
CPT becomes infinite with *continuous-valued* parent or child

Solution: canonical distributions that are defined compactly

Deterministic nodes are the simplest case:

$X = f(Parents(X))$ for some function f

E.g., Boolean functions

$NorthAmerican \Leftrightarrow Canadian \vee US \vee Mexican$

E.g., numerical relationships among continuous variables

$$\frac{\partial Level}{\partial t} = \text{inflow + precipitation - outflow - evaporation}$$

Compact conditional distributions contd.

Noisy-OR distributions model multiple noninteracting causes

1) Parents $U_1 \ldots U_k$ include all causes (can add leak node)
2) Independent failure probability q_i for each cause alone

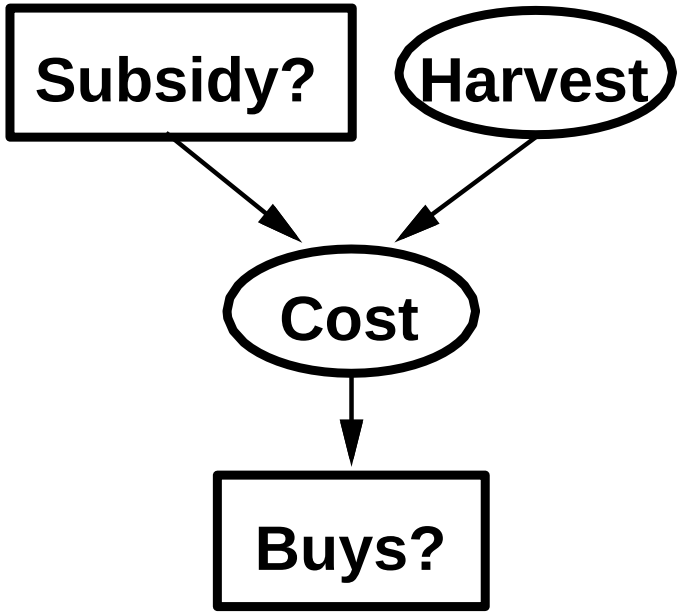
$$\Rightarrow \quad P(X \mid U_1 \ldots U_j, \neg U_{j+1} \ldots \neg U_k) = 1 - \prod_{i=1}^{j} q_i$$

Cold	*Flu*	*Malaria*	$P(Fever)$	$P(\neg Fever)$
F	F	F	0.0	1.0
F	F	T	0.9	**0.1**
F	T	F	0.8	**0.2**
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	**0.6**
T	F	T	0.94	$0.06 = 0.6 \times 0.1$
T	T	F	0.88	$0.12 = 0.6 \times 0.2$
T	T	T	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

Number of parameters linear in number of parents

Hybrid (discrete+continuous) networks

Discrete (*Subsidy?* and *Buys?*); continuous (*Harvest* and *Cost*)



Option 1: discretization—possibly large errors, large CPTs
Option 2: finitely parameterized canonical families

1) Continuous variable, discrete+continuous parents (e.g., *Cost*)
2) Discrete variable, continuous parents (e.g., *Buys?*)

Continuous child variables

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

Most common is the linear Gaussian model, e.g.,:

$$P(c \mid h, subsidy) = \mathcal{N}(c; a_t h + b_t, \sigma_t^2) = \frac{1}{\sigma_t \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{c-(a_t h+b_t)}{\sigma_t}\right)^2}$$

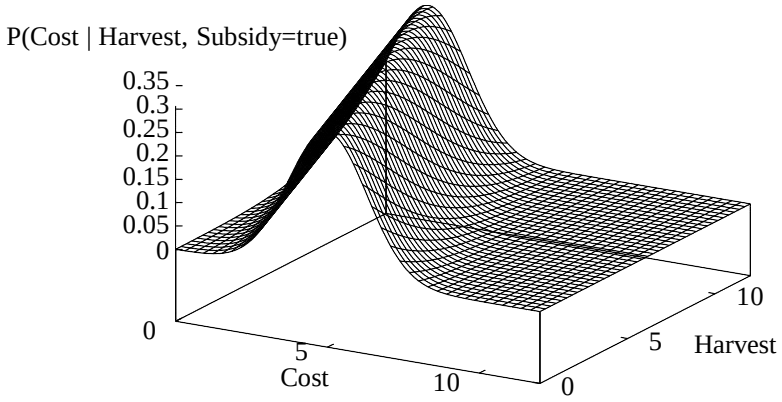
$$P(c \mid h, \neg subsidy) = \mathcal{N}(c; a_f h + b_f, \sigma_f^2) = \frac{1}{\sigma_f \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{c-(a_f h+b_f)}{\sigma_f}\right)^2} .$$

Gaussian distribution whose mean μ varies linearly with the value of the parent and whose standard deviation σ is fixed

Mean *Cost* varies linearly with *Harvest*, variance is fixed

Linear variation is unreasonable over the full range
but works OK if the likely range of *Harvest* is narrow

Continuous child variables



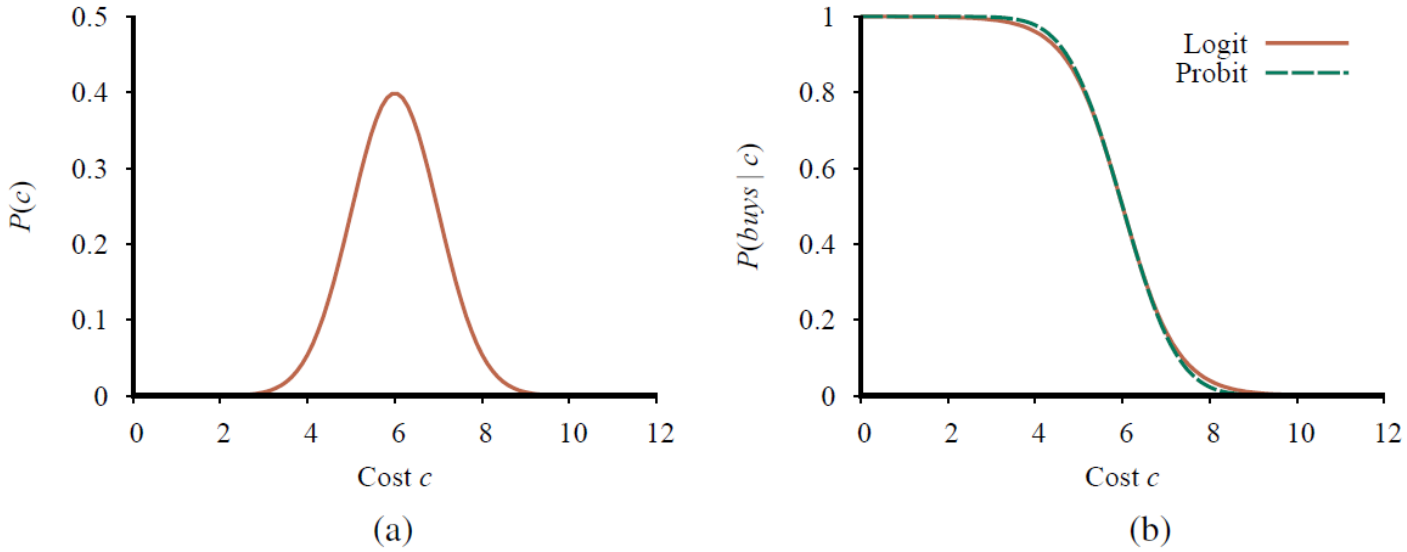
All-continuous network with LG distributions

⇒ full joint distribution is a multivariate Gaussian

Discrete + continuous LG network is a conditional Gaussian network i.e., a multivariate Gaussian over all continuous variables for each combination of discrete variable values

Discrete variable w/ continuous parents

Probability of *Buys?* given *Cost* should be a "soft" threshold:



(a) A normal (Gaussian) distribution for the cost threshold, centered on $\mu = 6.0$ with standard deviation $\sigma = 1.0$. (b) *Logit* and *Probit* models for the probability of *buys* given *cost*, for the parameters $\mu = 6.0$ and $\sigma = 1.0$.

Probit (probability unit):

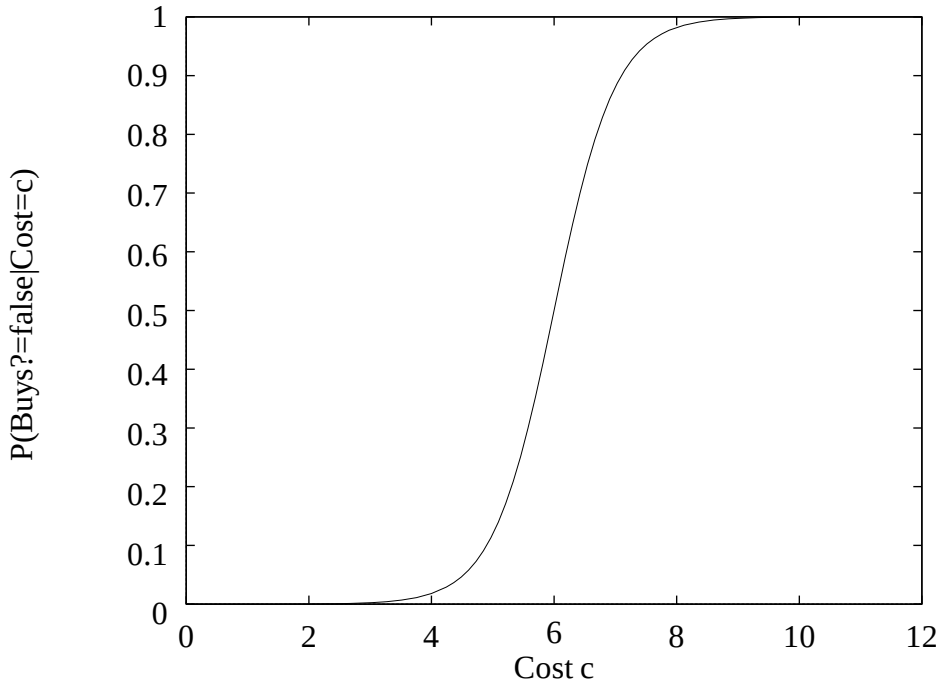
$$P(Buys? = true \mid Cost = c) = \Phi((-c + \mu)/\sigma)$$

Discrete variable contd.

Sigmoid (or logit) distribution also used in neural networks:

$$P(Buys? = true \mid Cost = c) = \frac{1}{1 + exp(-2\frac{-c+\mu}{\sigma})}$$

Sigmoid has similar shape to probit but much longer tails:



Exact Inference in Bayesian Networks

Simple queries: compute posterior marginal $\mathrm{P}(X_i|\mathrm{E}=\mathrm{e})$
e.g., $P(NoGas|Gauge=empty, Lights=on, Starts=false)$

Conjunctive queries: $\mathrm{P}(X_i, X_j|\mathrm{E}=\mathrm{e}) = \mathrm{P}(X_i|\mathrm{E}=\mathrm{e})\mathrm{P}(X_j|X_i, \mathrm{E}=\mathrm{e})$

Optimal decisions: decision networks include utility information; probabilistic inference required for $P(outcome|action, evidence)$

Within this context it is important to consider:

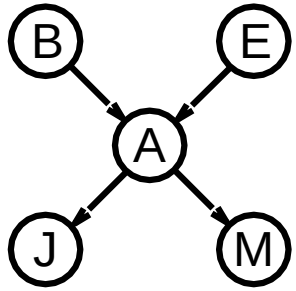
- Value of information: which evidence to seek next?
- Sensitivity analysis: which probability values are most critical?
- Explanation: why do I need a new starter motor?

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:

$$\begin{aligned}
&\mathrm{P}(B|j,m) \\
&= \mathrm{P}(B,j,m)/P(j,m) \\
&= a\mathrm{P}(B,j,m) \\
&= a\,\Sigma_e \Sigma_a\, \mathrm{P}(B,e,a,j,m)
\end{aligned}$$



Rewrite full joint entries using product of CPT entries:

$$\begin{aligned}
&\mathrm{P}(B|j,m) \\
&= a\,\Sigma_e \Sigma_a\, \mathrm{P}(B)P(e)\mathrm{P}(a|B,e)P(j|a)P(m|a) \\
&= a\mathrm{P}(B)\,\Sigma_e P(e)\,\Sigma_a\, \mathrm{P}(a|B,e)P(j|a)P(m|a)
\end{aligned}$$

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

Enumeration algorithm

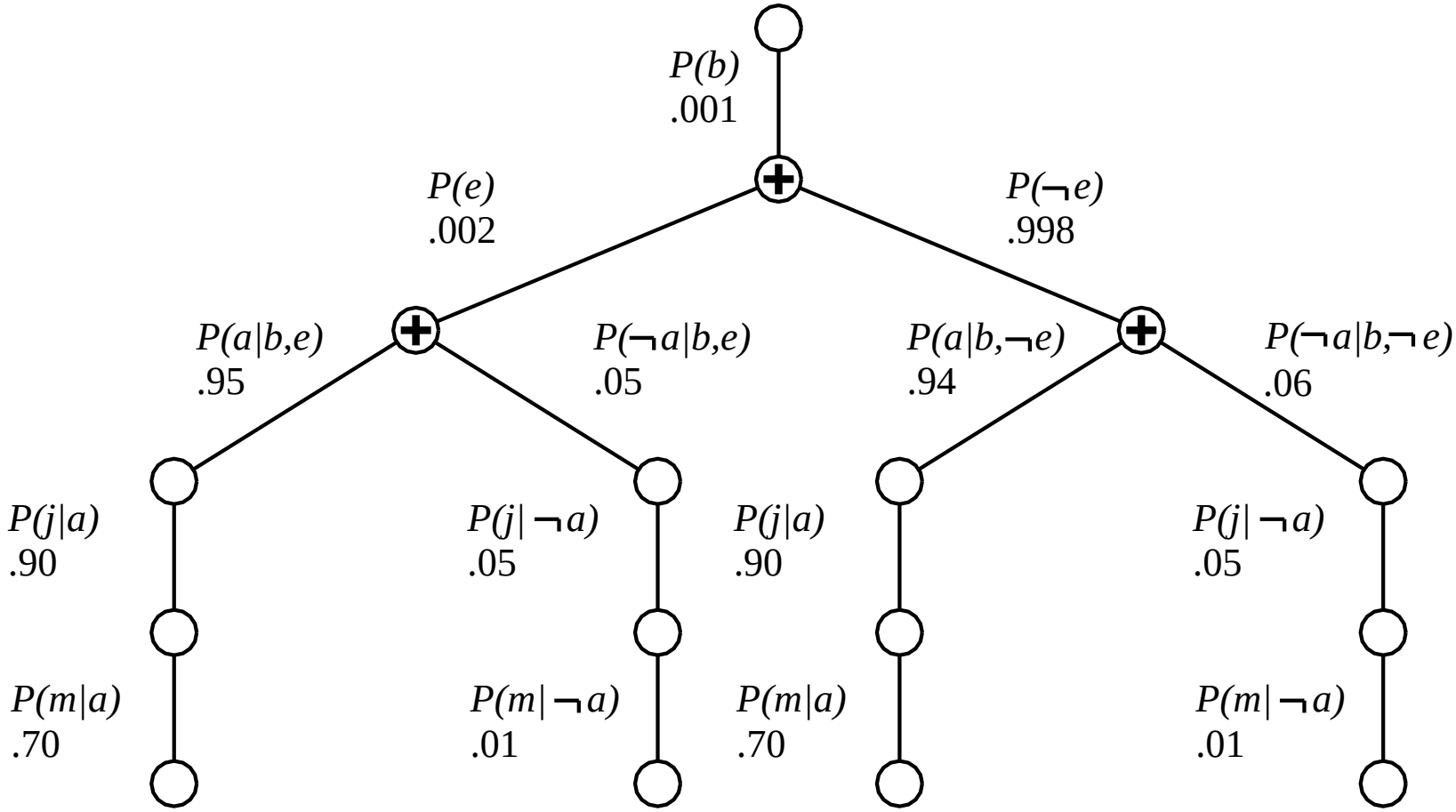
function ENUMERATION-ASK($X, \mathbf{e}, bn$) **returns** a distribution over X
  inputs: X, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayes net with variables $vars$

  $\mathbf{Q}(X) \leftarrow$ a distribution over X, initially empty
  for each value x_i of X **do**
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL($vars, \mathbf{e}_{x_i}$)
      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
  $V \leftarrow$ FIRST($vars$)
  if V is an evidence variable with value v in $\mathbf{e}$
    then return $P(v \mid parents(V)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}$)
    else return $\sum_v P(v \mid parents(V)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_v$)
      where $\mathbf{e}_v$ is $\mathbf{e}$ extended with $V = v$

Figure 13.11 The enumeration algorithm for exact inference in Bayes nets.

Evaluation tree



Enumeration is inefficient: repeated computation

e.g., computes $P(j|a)P(m|a)$ for each value of e

Inference by variable elimination

Variable elimination: carry out summations right-to-left, storing intermediate results (factors) to avoid recomputation

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

$$
\mathbf{f}_5(A) = \begin{pmatrix} P(m|a) \\ P(m|\neg a) \end{pmatrix} = \begin{pmatrix} 0.70 \\ 0.01 \end{pmatrix}
$$

Variable elimination: Basic operations

Summing out a variable from a product of factors:

move any constant factors outside the summation

add up submatrices in pointwise product of remaining factors

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

assuming $f_1, \ldots, f_i$ do not depend on X

Pointwise product of factors f_1 and f_2:

$$f_1(x_1, \ldots, x_j, y_1, \ldots, y_k) \times f_2(y_1, \ldots, y_k, z_1, \ldots, z_l)$$
$$= f(x_1, \ldots, x_j, y_1, \ldots, y_k, z_1, \ldots, z_l)$$

E.g., $f_1(a, b) \times f_2(b, c) = f(a, b, c)$

X	Y	$\mathbf{f}(X,Y)$	Y	Z	$\mathbf{g}(Y,Z)$	X	Y	Z	$\mathbf{h}(X,Y,Z)$
t	t	.3	t	t	.2	t	t	t	$.3 \times .2 = .06$
t	f	.7	t	f	.8	t	t	f	$.3 \times .8 = .24$
f	t	.9	f	t	.6	t	f	t	$.7 \times .6 = .42$
f	f	.1	f	f	.4	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 13.12 Illustrating pointwise multiplication: $\mathbf{f}(X,Y) \times \mathbf{g}(Y,Z) = \mathbf{h}(X,Y,Z)$.

Variable elimination algorithm

function ELIMINATION-ASK(X, **e**, bn) **returns** a distribution over X
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayesian network with variables $vars$

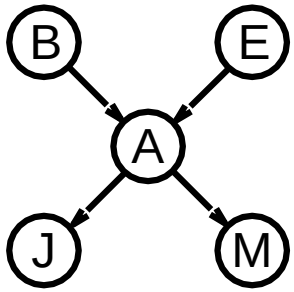
  $factors \leftarrow []$
  for each V **in** ORDER($vars$) **do**
    $factors \leftarrow [\text{MAKE-FACTOR}(V, \mathbf{e})] + factors$
    if V is a hidden variable **then** $factors \leftarrow \text{SUM-OUT}(V, factors)$
  return NORMALIZE(POINTWISE-PRODUCT($factors$))

Every choice of ordering yields a valid algorithm, but different orderings cause different intermediate factors to be generated during the calculation

Figure 13.13 The variable elimination algorithm for exact inference in Bayes nets.

Irrelevant variables

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



$$P(J|b) = \alpha P(b) \Sigma_e P(e) \Sigma_a P(a|b,e) P(J|a) \Sigma_m P(m|a)$$

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

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

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

(similarly to backward chaining from the query in Horn clause KBs)

Complexity of exact inference

Multiply connected networks:

- exponential time and space complexity in the worst case, even when the number of parents per node is bounded.
- inference in Bayes nets is NP-hard

Singly connected networks (or polytrees):

- any two nodes are connected by at most one (undirected) path
- time can be reduced to O(n)

Complexity of exact inference

Clustering algorithms (also known as join tree algorithms) are widely used in commercial Bayes net tools.

The basic idea of clustering is to join individual nodes of the network to form cluster nodes in such a way that the resulting network is a **polytree**.

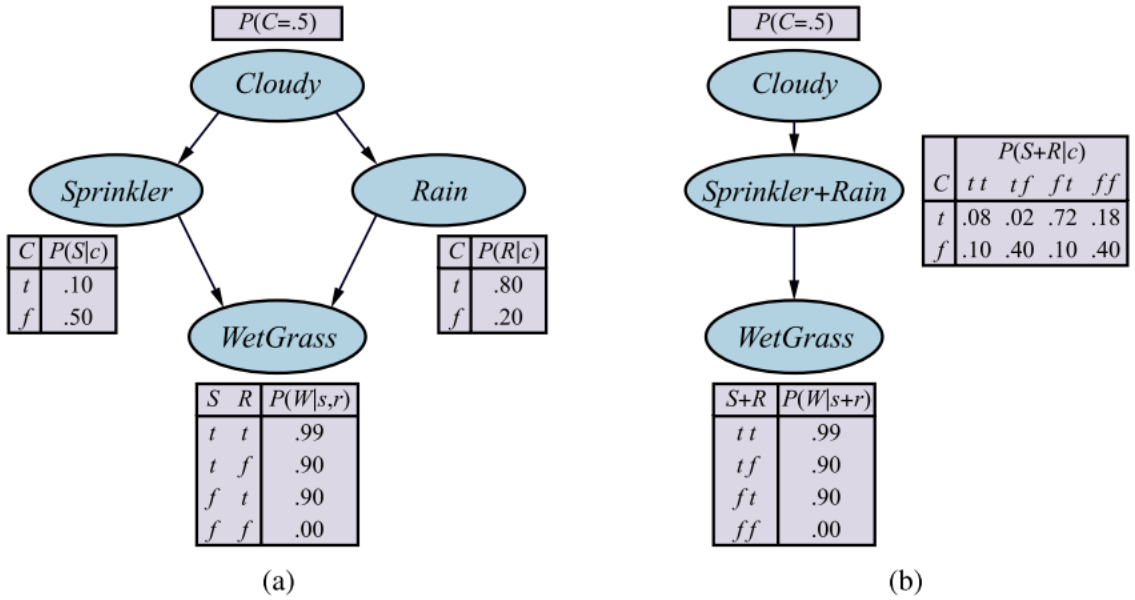
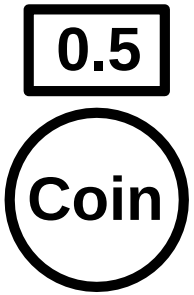


Figure 13.15 (a) A multiply connected network describing Mary's daily lawn routine: each morning, she checks the weather; if it's cloudy, she usually doesn't turn on the sprinkler; if the sprinkler is on, or if it rains during the day, the grass will be wet. Thus, *Cloudy* affects *WetGrass* via two different causal pathways. (b) A clustered equivalent of the multiply connected network.

Approximate Inference for Bayesian Networks

They work by *generating **random events** based on the probabilities in the Bayes net and **counting up** the different answers found in those random events*.



With enough samples, we can get **arbitrarily close to recovering the true probability distribution**—provided the Bayes net has no deterministic conditional distributions.

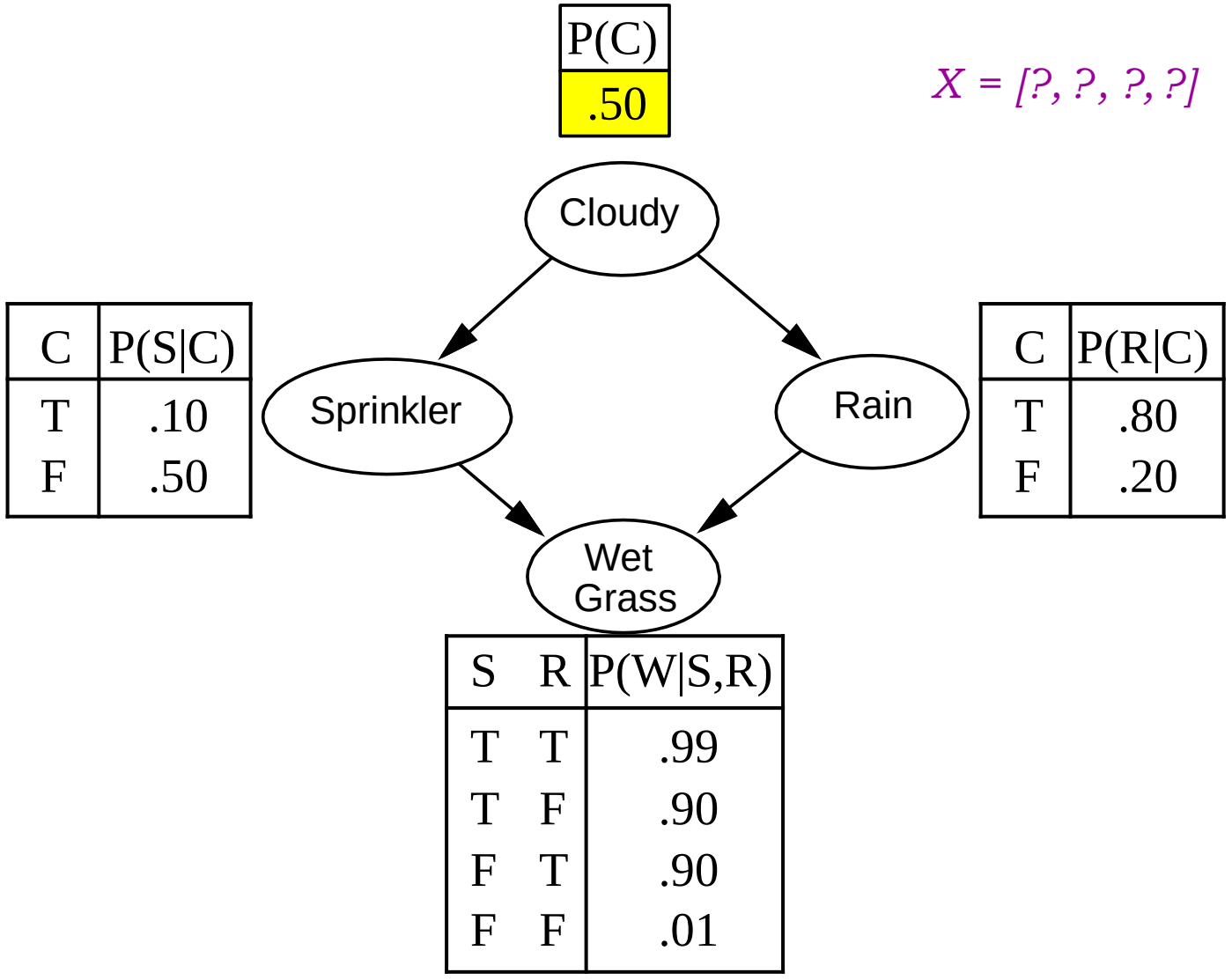
Basic procedure:

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

Sampling from an empty network

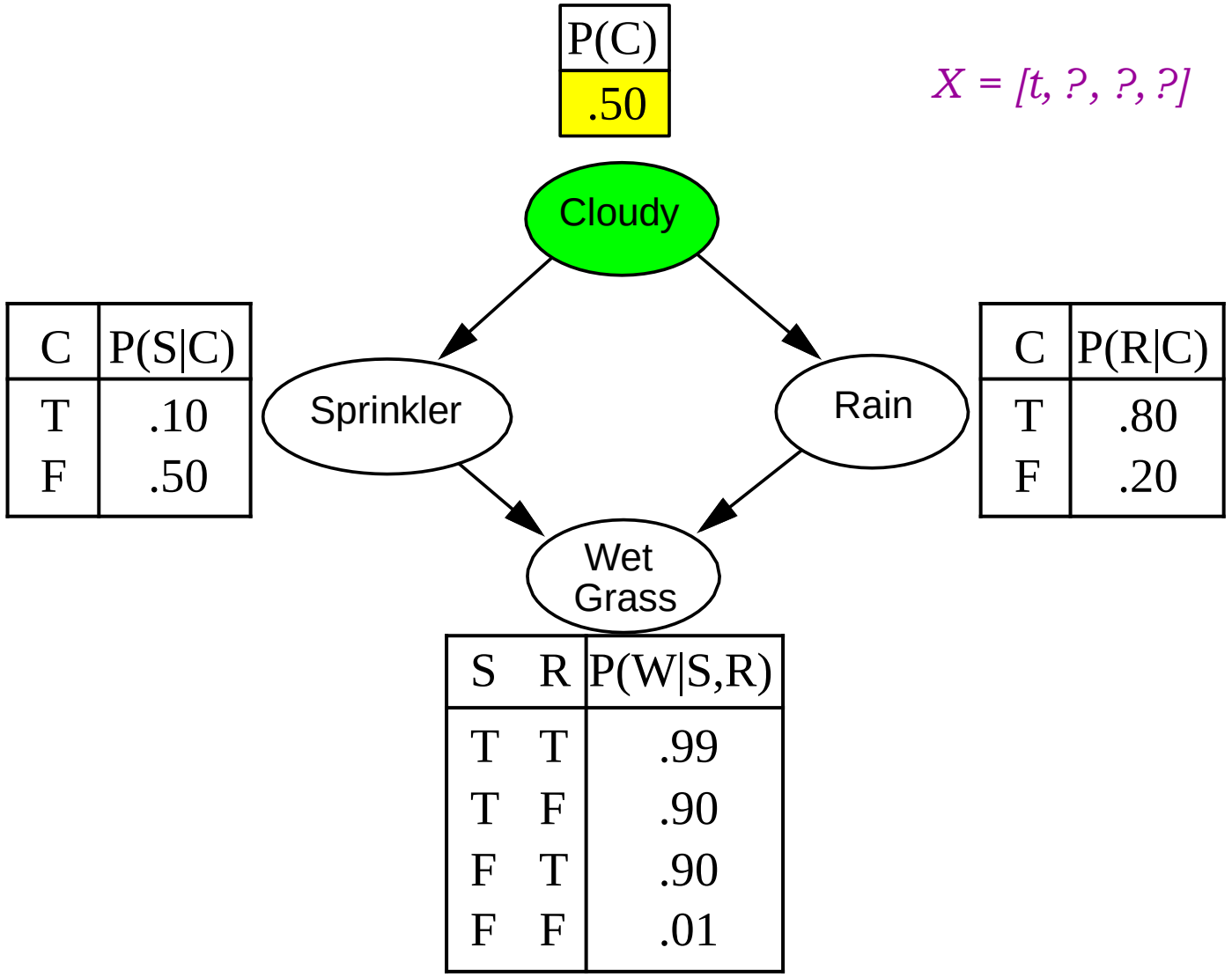
```
function Prior-Sample(bn) returns an event sampled from bn
    inputs: bn, a belief network specifying joint distribution P(X_1, ..., X_n)

    x ← an event with n elements
    for i = 1 to n do
        x_i ← a random sample from P(X_i | parents(X_i))
              given the values of Parents(X_i) in x
    return x
```

Example



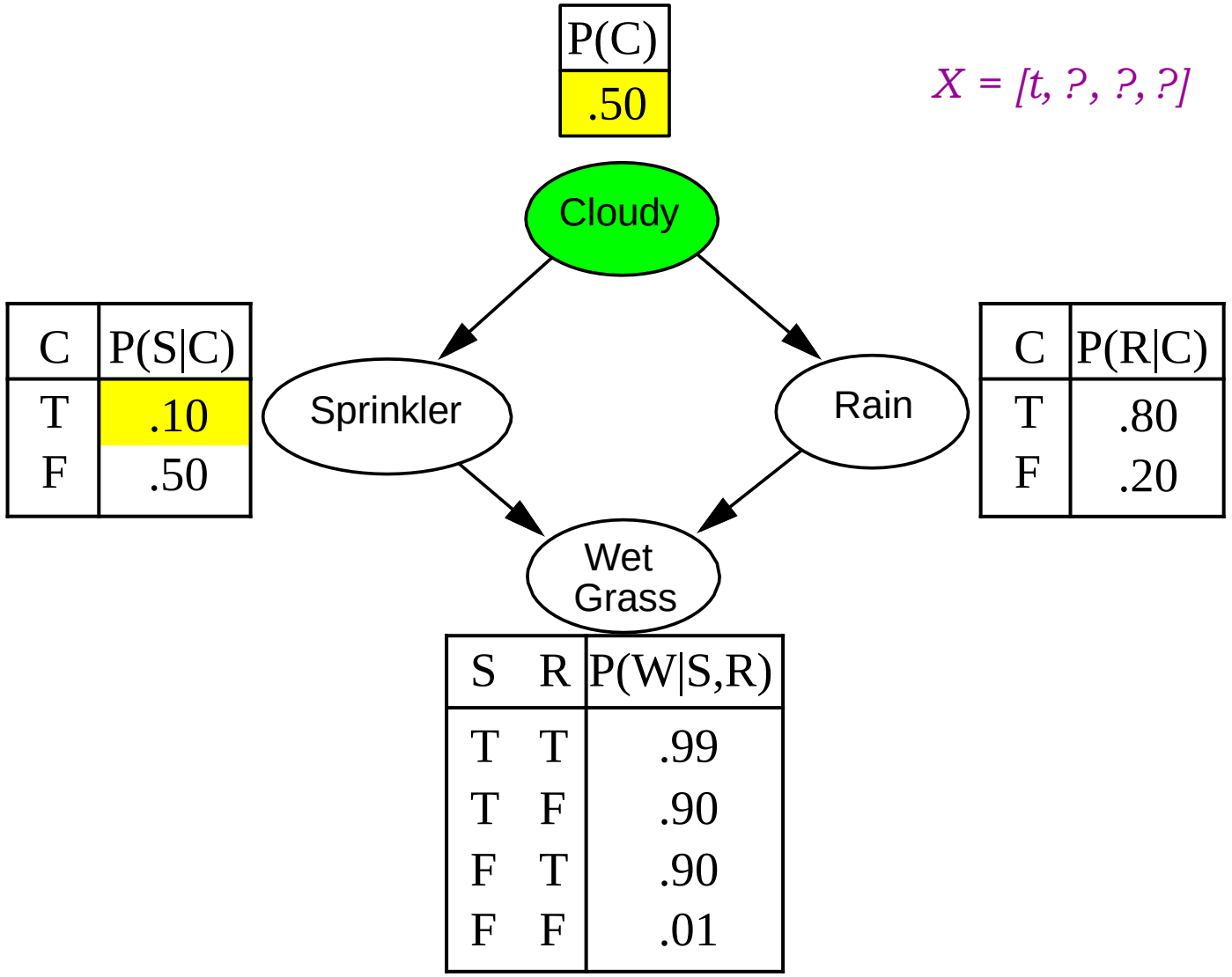
X = [?, ?, ?, ?]

Example



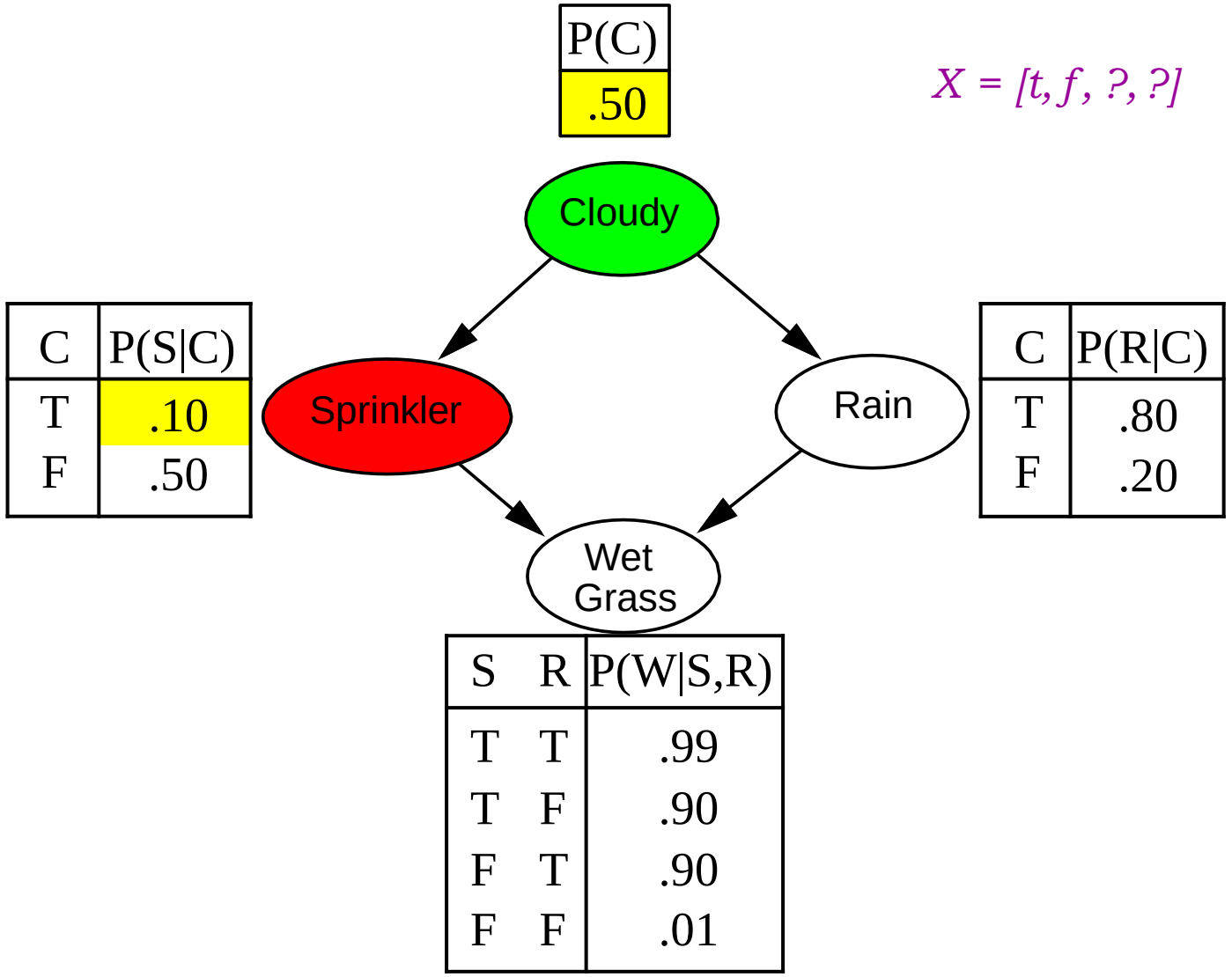
X = [t, ?, ?, ?]

Example



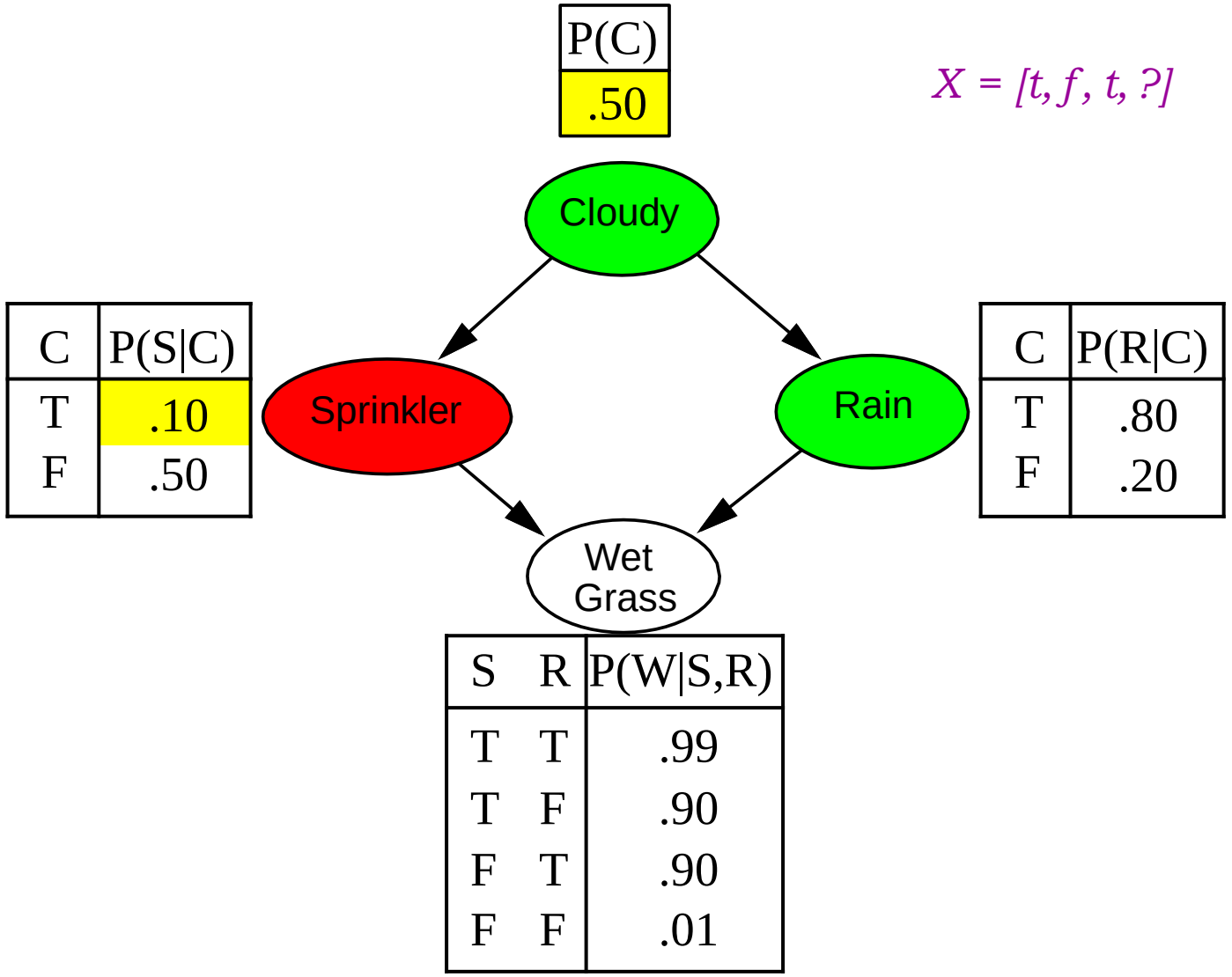
X = [t, ?, ?, ?]

Example



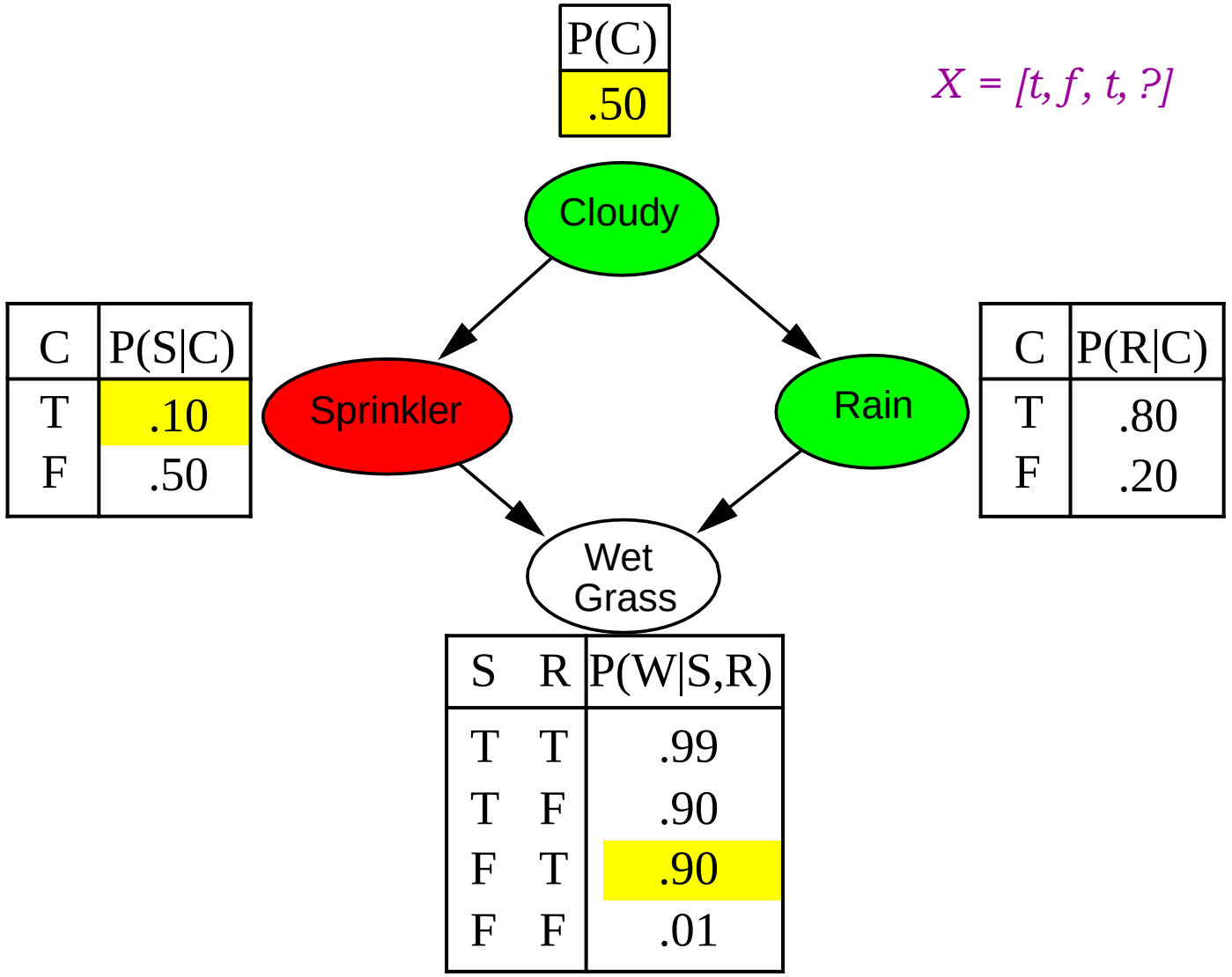
X = [t, f, ?, ?]

Example



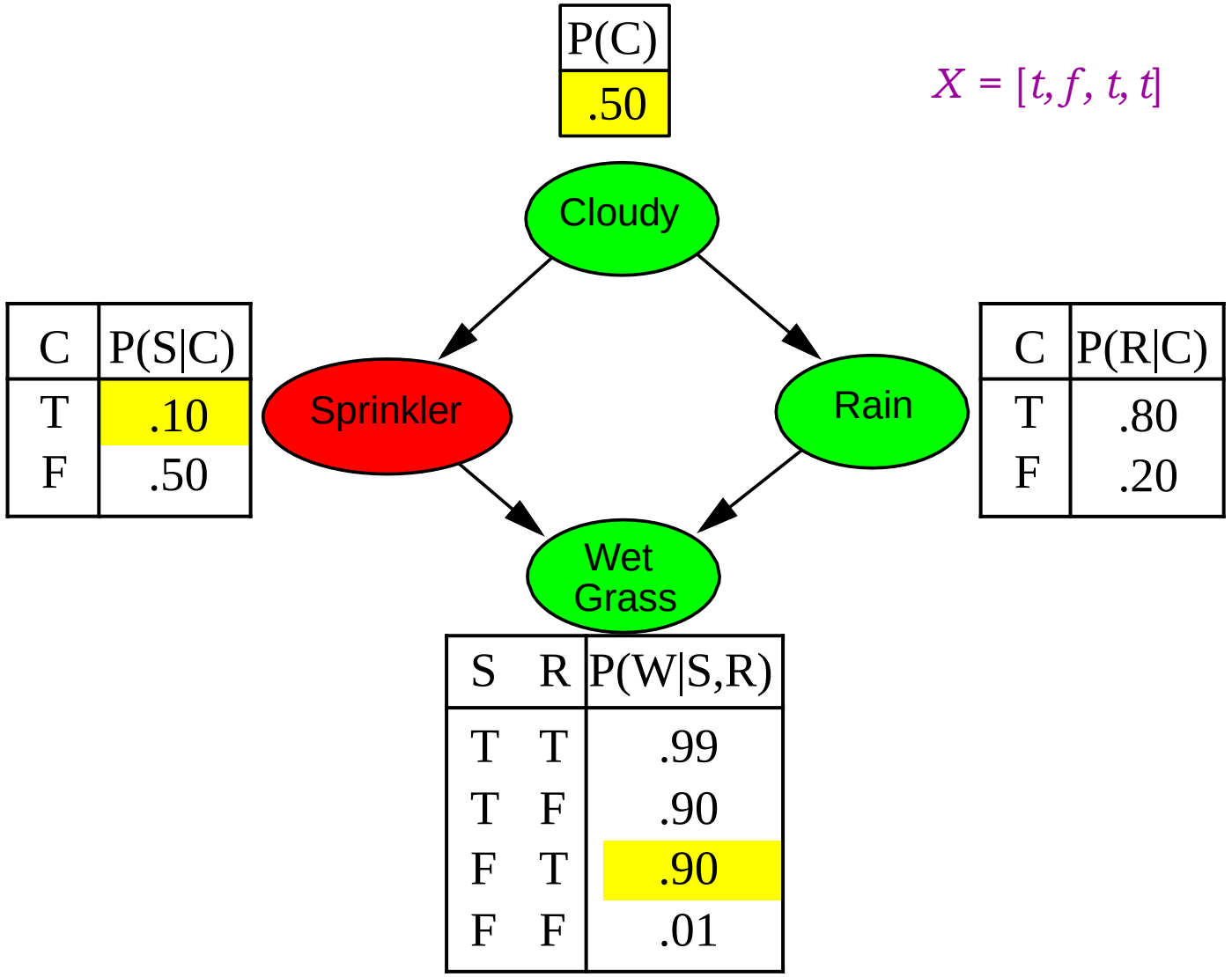
$X = [t, f, t, ?]$

Example



$X = [t, f, t, ?]$

Example



$X = [t, f, t, t]$

Sampling from an empty network contd.

Probability that PriorSample generates a particular event

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

i.e., the true prior probability

E.g., $S_{PS}(t, f, t, t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324 = P(t, f, t, t)$

Let $N_{PS}(x_1 \ldots x_n)$ be the number of samples generated for event $x_1, \ldots, x_n$

Then we have

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

That is, estimates derived from PriorSample are consistent

Shorthand: $\hat{P}(x_1, \ldots, x_n) \approx P(x_1 \ldots x_n)$

Rejection sampling (using some evidence!)

$\hat{\mathbf{P}}(X|\mathbf{e})$ estimated from samples agreeing with $\mathbf{e}$

```
function Rejection-Sampling(X,e,bn,N) returns an estimate of P(X|e)
  local variables: N, a vector of counts over 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[X])
```

E.g., estimate $\mathbf{P}(Rain|Sprinkler = true)$ using 100 samples

27 samples have $Sprinkler = true$

Of these, 8 have $Rain = true$ and 19 have $Rain = false$.

$\hat{\mathbf{P}}(Rain|Sprinkler = true) = \text{Normalize}(\langle 8, 19\rangle) = \langle 0.296, 0.704\rangle$

Similar to a basic real-world empirical estimation procedure

Analysis of rejection sampling

$\hat{\mathrm{P}}(X|\mathrm{e}) = \alpha \mathrm{N}_{PS}(X,\mathrm{e})$ (algorithm defn.)

$= \mathrm{N}_{PS}(X,\mathrm{e})/N_{PS}(\mathrm{e})$ (normalized by $N_{PS}(\mathrm{e})$)

$\approx \mathrm{P}(X,\mathrm{e})/P(\mathrm{e})$ (property of PriorSample)

$= \mathrm{P}(X|\mathrm{e})$ (defn. of conditional probability)

Hence rejection sampling returns consistent posterior estimates

Problem: hopelessly expensive if $P(\mathrm{e})$ is small -> convergence is quite slow.

$P(\mathrm{e})$ drops off exponentially with number of evidence variables

Causal Networks

Causal Networks: a restricted class of Bayesian networks that forbids *all but causally compatible* orderings.

$$P(c, r, s, w, g) = P(c)\, P(r\,|c)\, P(s|c)\, P(w|r, s)\, P(g|w)$$

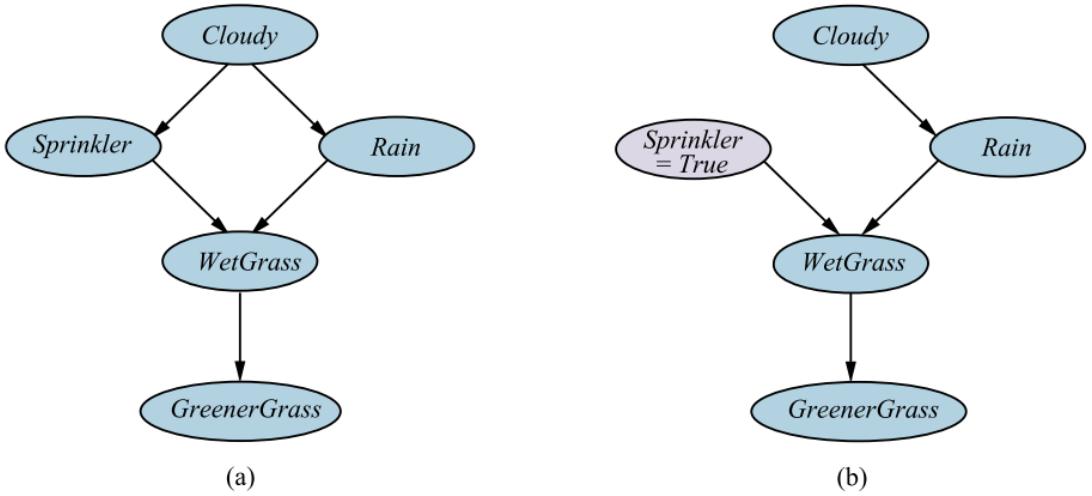


Figure 13.23 (a) A causal Bayesian network representing cause–effect relations among five variables. (b) The network after performing the action "turn *Sprinkler* on."

Note the difference between conditioning on the action **do(Sprinkler=true)** in the original network and conditioning on the observation **Sprinkler=true**.

The original network tells us that the sprinkler is less likely to be on when the weather is cloudy, so if we observe the sprinkler to be on, that reduces the probability that the weather is cloudy. But common sense tells us that if we reach in and turn on the sprinkler, that doesn't affect the weather or provide new information about what the weather is like that day.

Intervening breaks the normal causal link between the weather and the sprinkler. This prevents any influence flowing backward from Sprinkler to Cloudy.

For example, suppose we turn the sprinkler on—$do(Sprinkler = true)$

$$P(c, r, w, g|do(S = true)) = P(c)\, P(r\,|c)\, P(w|r, s = true)\, P(g|w)$$

Causal Networks

Example:

Predict the effect of turning on the sprinkler on a downstream variable such as *GreenerGrass*, but the adjustment formula must take into account not only the direct route from Sprinkler, but also the "back door" route via Cloudy and Rain.

$$P(g \mid do(S = true)) = \sum_r P(g|S = true, r)P(r)$$

we wish to find the effect of $do(X_j = x_{jk})$ on a variable X_i,

Back-door criterion

allows us to write an adjustment formula that conditions on any set of variables **Z** that closes the back door, so to speak

AIMA-python - “Probability.ipynb”

6536 lines (6536 sloc) | 397 KB

Raw Blame

Probability

This IPy notebook acts as supporting material for topics covered in **Chapter 13 Quantifying Uncertainty**, **Chapter 14 Probabilistic Reasoning**, **Chapter 15 Probabilistic Reasoning over Time**, **Chapter 16 Making Simple Decisions** and parts of **Chapter 25 Robotics** of the book* Artificial Intelligence: A Modern Approach*. This notebook makes use of the implementations in probability.py module. Let us import everything from the probability module. It might be helpful to view the source of some of our implementations. Please refer to the Introductory IPy file for more details on how to do so.

In [1]:

```
from probability import *
from utils import print_table
from notebook import psource, pseudocode, heatmap
```

CONTENTS

- Probability Distribution
  - Joint probability distribution
  - Inference using full joint distributions

- Bayesian Networks - BayesNode - BayesNet - Exact Inference in Bayesian Networks - Enumeration - Variable elimination - Approximate Inference in Bayesian Networks - Prior sample - Rejection sampling - Likelihood weighting - Gibbs sampling
- Hidden Markov Models - Inference in Hidden Markov Models - Forward-backward - Fixed lag smoothing - Particle filtering

- Monte Carlo Localization - Decision Theoretic Agent - Information Gathering Agent

[aima-python/probability.ipynb at master · aimacode/aima-python (github.com)]

Summary

Bayes nets provide a natural representation for (causally induced) conditional independence:

- **Topology + CPTs** = compact representation of joint distribution
- Generally easy for (non)experts to construct

Canonical distributions (e.g., noisy-OR) = compact representation of CPTs

Continuous variables ⇒ parameterized distributions (e.g., linear Gaussian)

Exact inference by variable elimination:

- polytime on polytrees, NP-hard on general graphs
- space = time, very sensitive to topology

Random sampling techniques can give reasonable estimates of the true posterior probabilities in a network and can cope with much larger networks than can exact algorithms.

In the next lecture...

- Time and Uncertainty
- Inference in Temporal Models
- Hidden Markov Models
- Dynamic Bayesian Networks