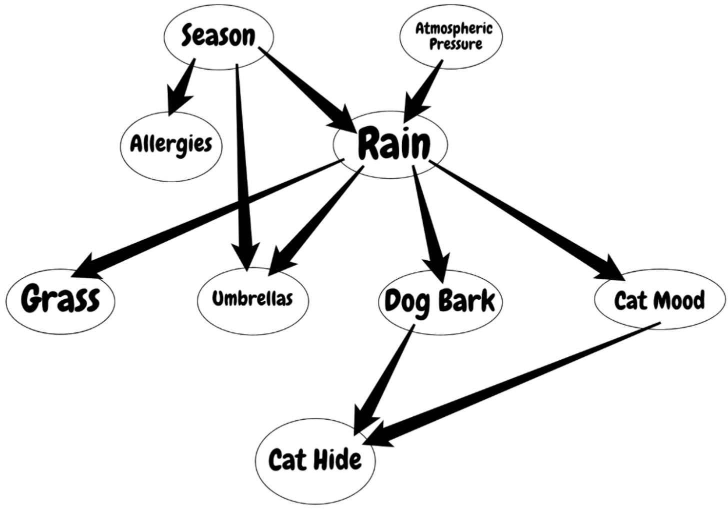
PROBABILISTIC REASONING SYSTEMS

In which we explain how to build reasoning systems that use network models to reason with uncertainty according to the laws of probability theory.

Outline



- Knowledge in uncertain domains
- Probabilistic Networks
- Semantic of Bayesian Networks
  - Global Semantic
  - Local Semantic
- Efficient representation of conditions distributions
- Exact inference in Bayesian Networks
- Approximate inference in Bayesian Networks
- Summary

Bayes'-Rule

Why is this rule useful?

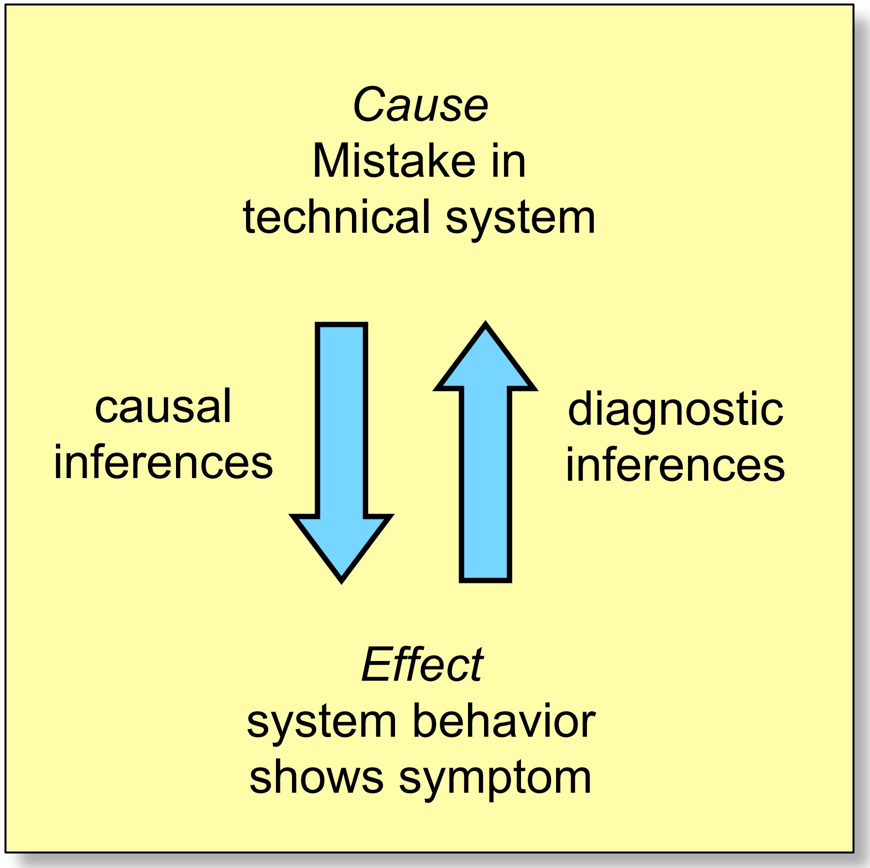
- Causal experiences
  C: cause, E: effect
- Diagnostic Inference

$$P(C\,|\,E) = \frac{P(E\,|\,C)\,P(C)}{P(E)}$$

This simple equation underlies

all modern AI systems for

probabilistic inference



Knowledge in uncertain domains

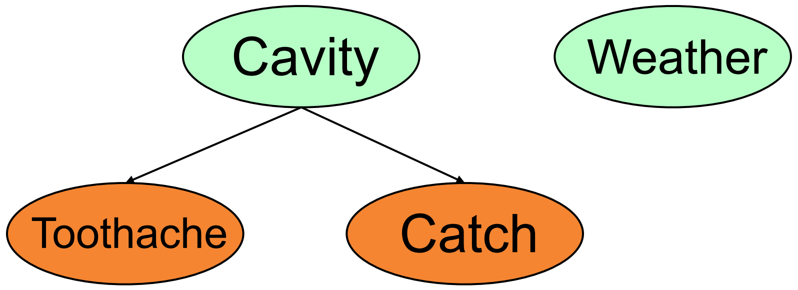
- Joint probability distribution
  - delivers answers to questions that exists in domain
  - Problem: intractable with large number of variables
  - Specification
    Probabilities difficult for atomic events

- Complexity
  - Independence and conditional dependence reduce complexity

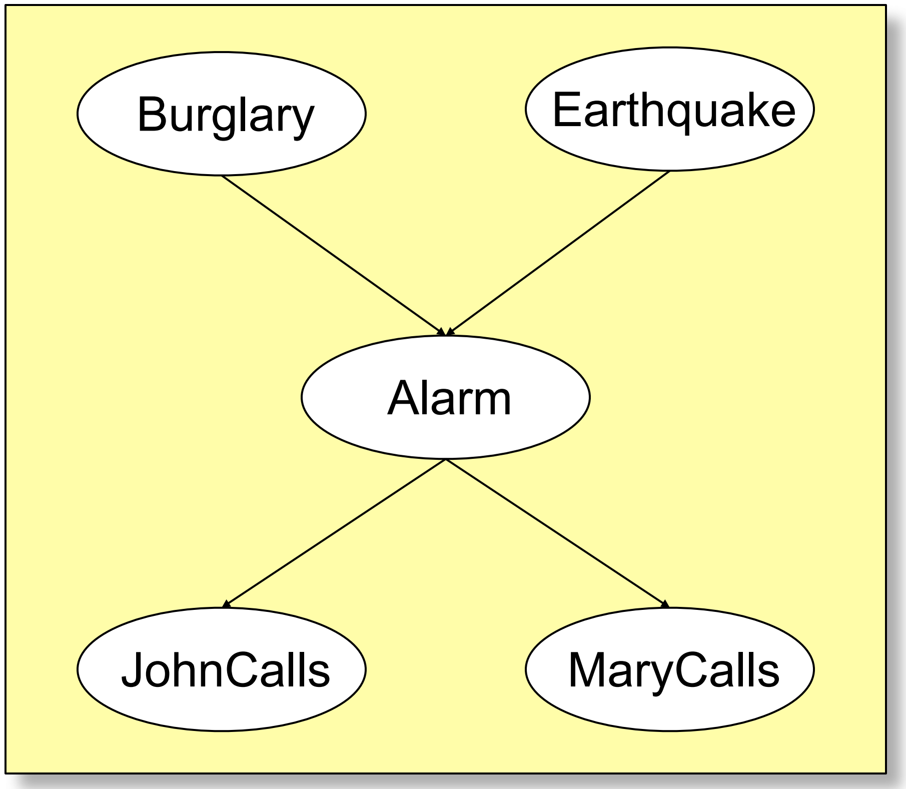
- Bayesian Networks
  - Data structure represents dependencies between variables
  - Specification of joint distribution

Syntax

- Graph-theoretical structure
  - Set of variables as nodes (discrete, continuous)
  - Each node corresponds to random variable
  - Directed acyclic graph (DAG), links express causal dependencies between variables
- Conditional probability tables
  - For each node a table for conditional probabilities
  - Table consists of distribution of probabilities given their parents $\mathbf{P}(X_i|\text{Parents}(X_i))$

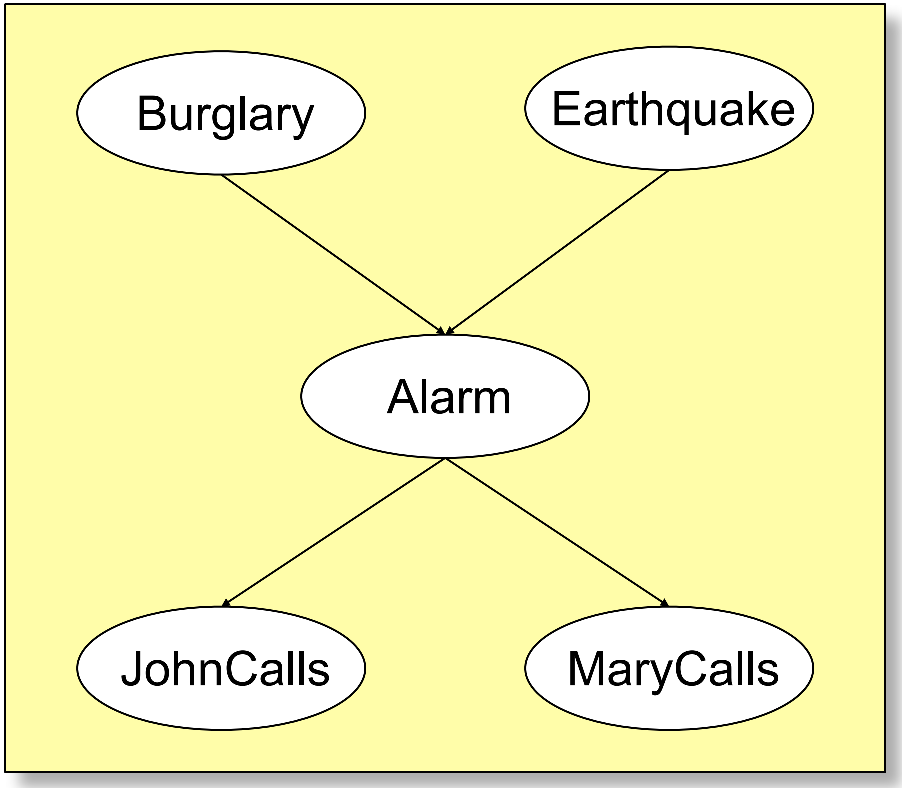


Simple Bayesian Network



- Example Alarm
  - new burglar alarm fairly reliable at detecting a burglary
  - also responds on occasion to minor earthquake
  - two neighbors, John and Mary have promised to call when they hear the alarm
  - John always calls when he hears alarm, but sometimes confuses the telephone ringing with the alarm and calls then too
  - Mary likes loud music and sometimes misses the alarm altogether
  - Given the evidence of who has or has not called, we would like to estimate the probability of a burglary.

Simple Bayesian Network



P(B)
0.001

P(E)
0.002

B	E	**P**(A)
1	1	0.95
1	0	0.94
0	1	0.29
0	0	0.001

A	**P**(J)
1	0.90
0	0.05

A	**P**(M)
1	0.70
0	0.01

conditional distributions

Semantics of Bayesian Networks

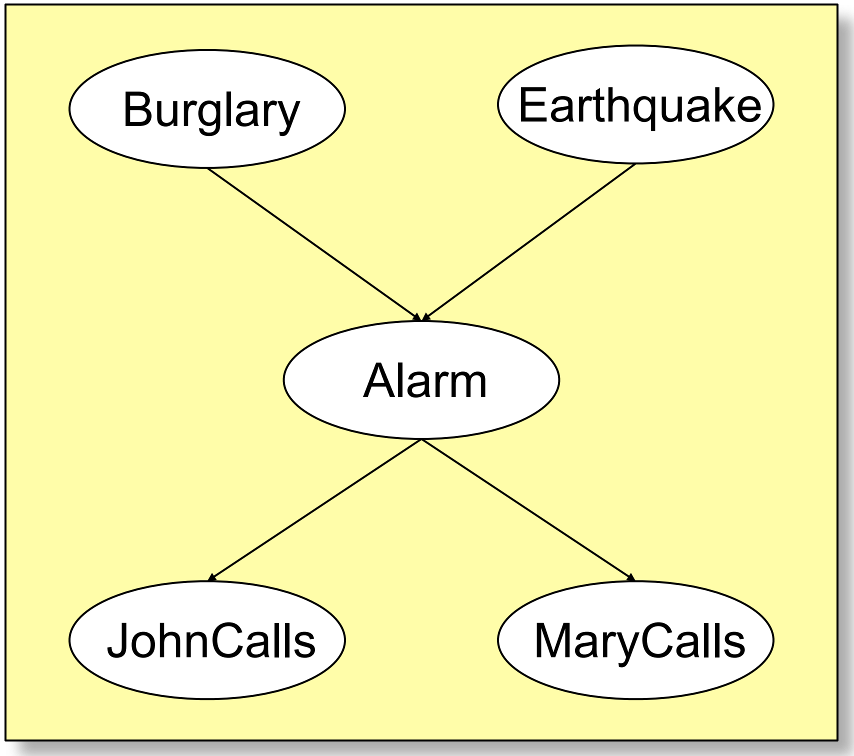
- Two views on semantics
  1. Global Semantics: The first is to see the network as a representation of the joint probability distribution
  2. Local Semantics: The second is to view it as an encoding of a collection of conditional independence statements

- Views are equivalent
  - first helpful in understanding how to *construct* networks
  - second helpful in designing inference procedures

Representing the full joint distribution

- General idea
  - Joint distribution can be expressed as product of local conditional probabilities
  - Every entry in the joint probability distribution can be calculated from the information in the network.
  - Generic entry

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



Representing the full joint distribution

- Example
  - Alarm has sounded but neither a burglary nor an earthquake has occurred, and both John and Mary call
  - $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$
    $= P(j|a)\, P(m|a)\, P(a|\neg b,\neg e)\, P(\neg b)\, P(\neg e)$
    $= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998$
    ≈ 0.00063

P(B)
0.001

P(E)
0.002

B	E	**P**(A)
1	1	0.95
1	0	0.94
0	1	0.29
0	0	0.001

A	**P**(J)
1	0.90
0	0.05

A	**P**(M)
1	0.70
0	0.01

Method for Constructing Bayesian Networks

- Generic Rule

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

  - Semantic
    but: not how to construct a network
  - Implicitly: conditional independence
    → Help for Knowledge Engineer

- Reformulate the rule
  - Use of conditional probabilities
    → Product rule

$$P(x_1,\ldots,x_n) = P(x_n \mid x_{n-1},\ldots,x_1)\, P(x_{n-1},\ldots,x_1)$$

- Repeat process
  - Reduction of conjunctive probabilities to a conditional dependency and a smaller conjunction
  - Final: a big product

$$P(x_1,\ldots,x_n) = P(x_n \mid x_{n-1},\ldots,x_1)\, P(x_{n-1} \mid x_{n-2},\ldots,x_1) \ldots P(x_2 \mid x_1) P(x_1) = \prod_{i=1}^{n} P(x_i \mid x_{i-1},\ldots,x_1)$$

Chain rule

$$P(x_1,\ldots,x_n) = P(x_n \mid x_{n-1},\ldots,x_1)\,P(x_{n-1} \mid x_{n-2},\ldots,x_1)\ldots P(x_2 \mid x_1)P(x_1) = \prod_{i=1}^{n} P(x_i \mid x_{i-1},\ldots,x_1)$$

- Compare with

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

  reveals that specification is equivalent to general assertion

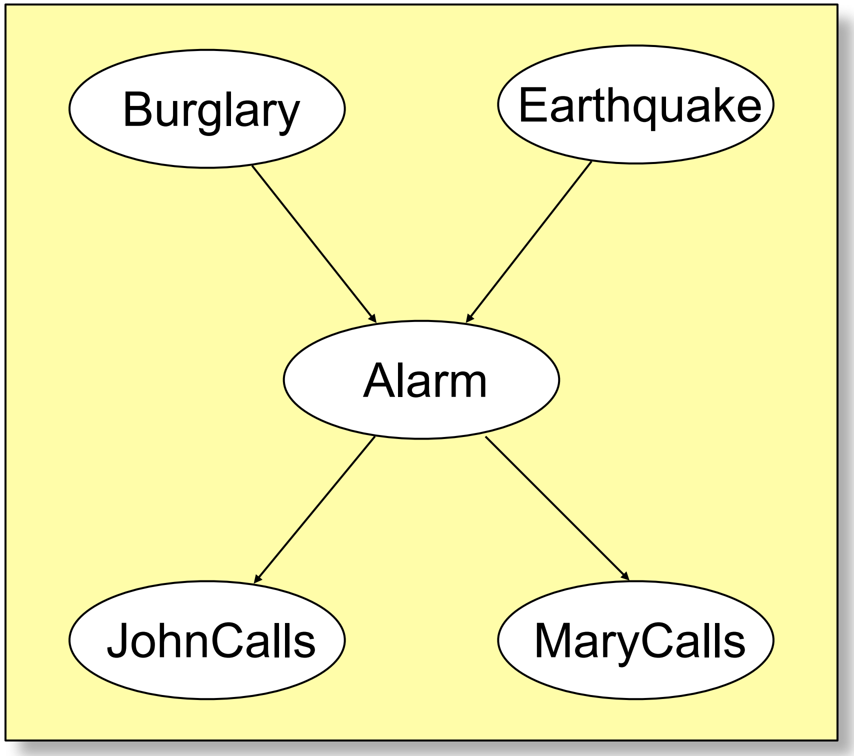
  $\mathbf{P}(X_i|X_{i-1},\ldots,x_1) = \mathbf{P}(X_i|\text{Parents}(X_i))$

  (as long as $\text{Parents}(X_i) \subseteq \{X_{i-1},\ldots,X_1\}$)

- I.e.:
  - This last condition is satisfied by labeling the nodes in any order that is consistent with the partial order implicit in the graph structure.
  - The Bayesian network is a correct representation of the domain only if each node is conditionally independent of its predecessors in the node ordering, given its parents.

Construction of Bayesian Networks

- Important while constructing
  - We need to choose parents for each node such that this property holds.
- Intuitive
  - Parents of node X_i should contain all those nodes in $X_1, \ldots X_{i-1}$ that directly influence Xi
  - Example.:
    - M (is influenced by B or E but not directly)
    - Influenced by A, and J calls are not evident



$$\mathbf{P}(M|J,A,E,B) = \mathbf{P}(M|A)$$

General Procedure

1. Choose the set of relevant variables X_i that describe the domain.
2. Choose an ordering for the variables.
   (Any ordering works, but some orderings work better than others, as we will see.)
3. While there are variables left:
   a) Pick a variable X_i and add a node to the network for it.
   b) Set *Parents*(X_i) to some minimal set of nodes already in the net such that the conditional independence property is satisfied.
   c) Define the conditional distribution $\mathbf{P}(X_i|Parents(X_i))$

Notes

- Construction method guarantees that the network is acyclic
  - Because each node is connected only to earlier nodes.

- Redundancies
  - No redundant probability values
  - Exception: for one entry in each row of each conditional probability table, if $(P(x_2|x_1)\ P(\neg x_2|x_1))$ is redundant

- *This means that it is impossible for the knowledge engineer or domain expert to create a Bayesian network that violates the axioms of probability!*

Compactness

- Compactness
  - A Bayesian Network is a complete and not-redundant representation of a domain
  - Can be more compact as a joint distribution
  - This is important in practice
  - Compactness is an example for property that we call in local structure (or sparse coded) in general

- Local Structures (also: sparse)
  - Each sub-component is connected to a limited number of other components
  - Complexity: linear instead of exponential
  - With BN: in most domains one variable is influenced by k others, with n variables 2^k conditional probabilities, the whole network $n2^k$
  - In contrast, the full joint distribution contains 2^n numbers

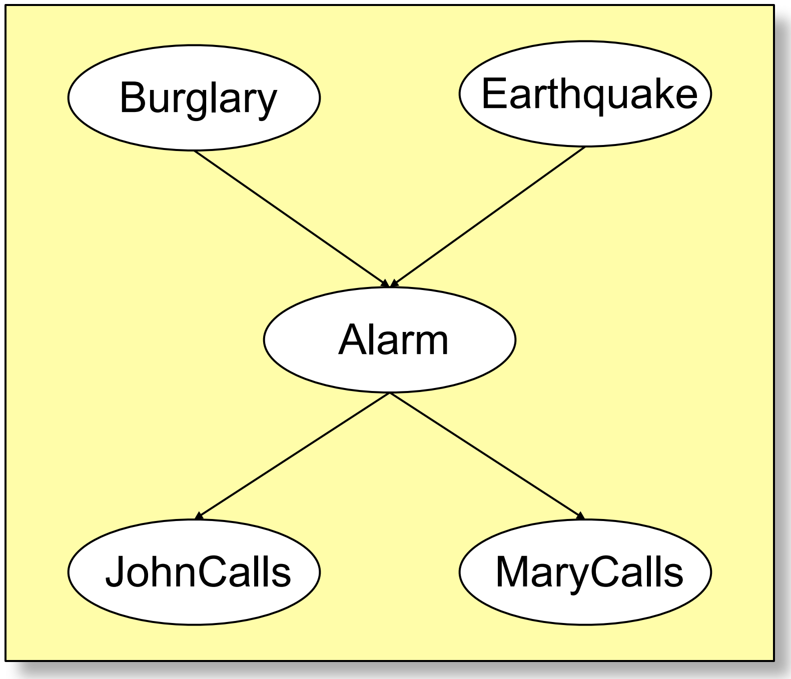
Node Ordering

- Local structures (example)
  - 30 nodes, each max. 5 parents
  - 960 for BN, > 1 billion with joint distribution
- Construction
  - Not trivial
  - Variable directly influenced only from a few others
  - Set parent node "appropriately" → Network topology
  - "Direct influencers" first
  - Thus: correct order important
- Order
  - Add:
    - root first
    - then direct influencers
    - then down to leaves
  - What happens with "wrong" order?

Example ordering

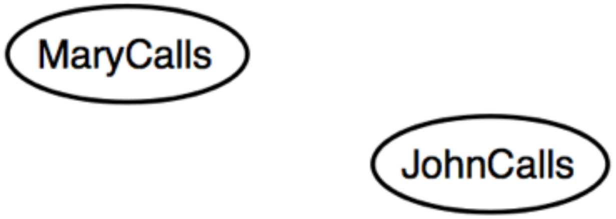
- Let us consider the burglary example again.

- Suppose we decide to add the nodes in the order
  - M, J, A, B, E
  - M, J, E, B, A



Example

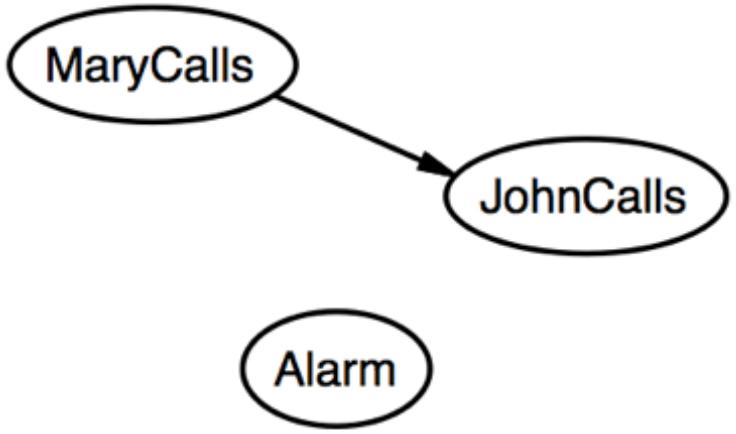
Suppose we choose the ordering M, J, A, B, E



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

Example contd.

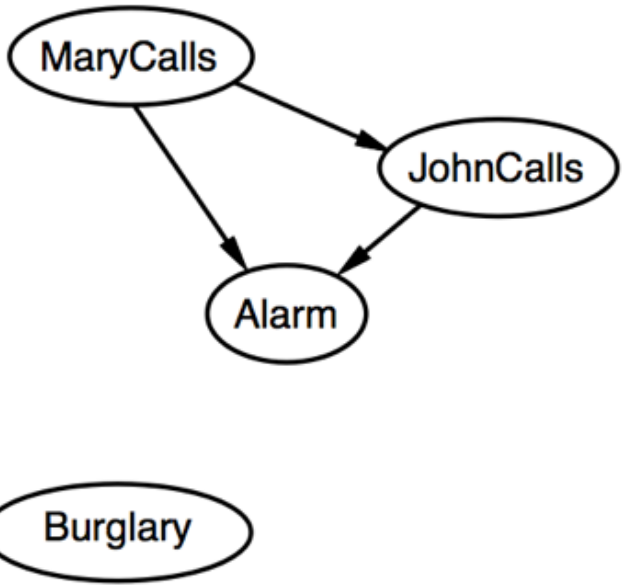
Suppose we choose the ordering M, J, A, B, E



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

Example contd.

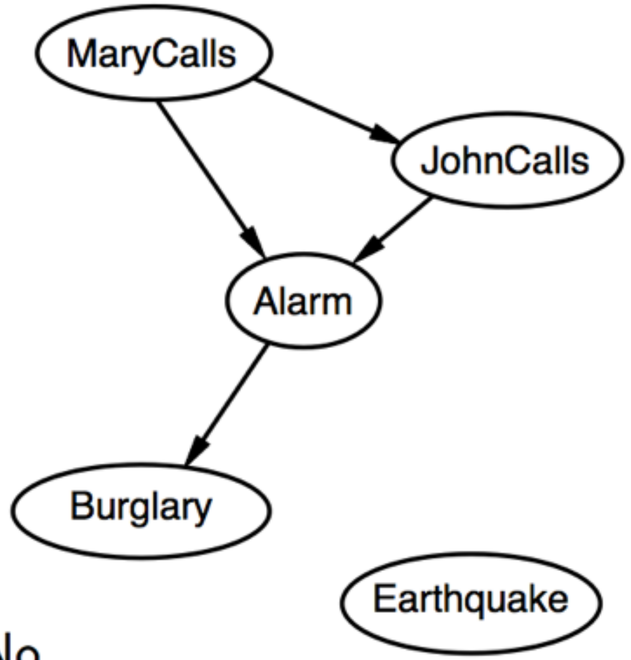
Suppose we choose the ordering M, J, A, B, E



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

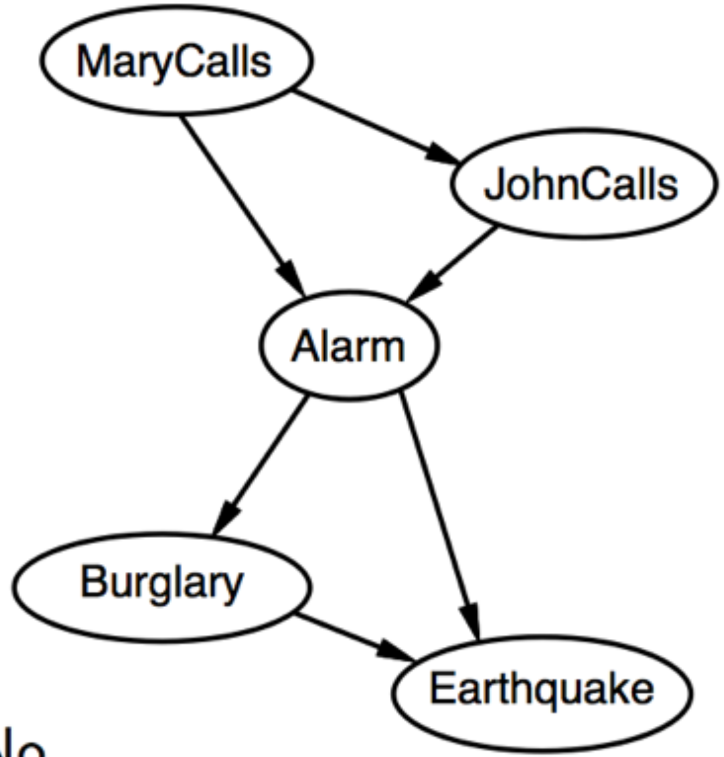
Suppose we choose the ordering M, J, A, B, E



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

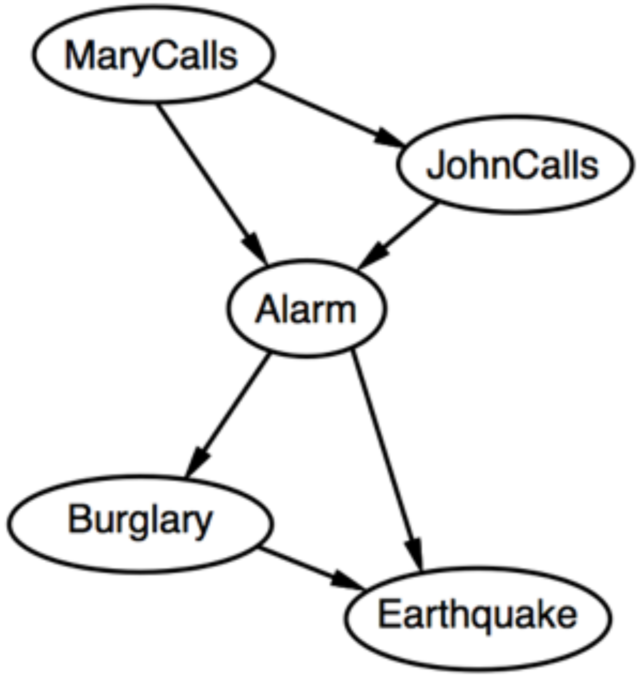
Example contd.

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.



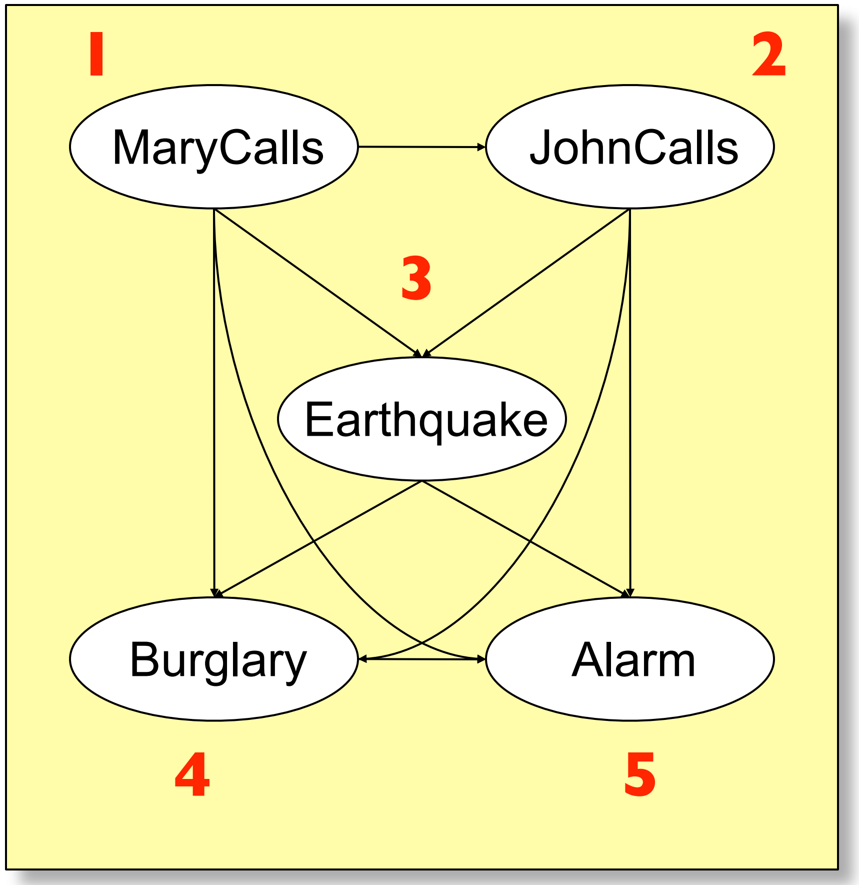
Deciding conditional independence is hard in noncausal directions

(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

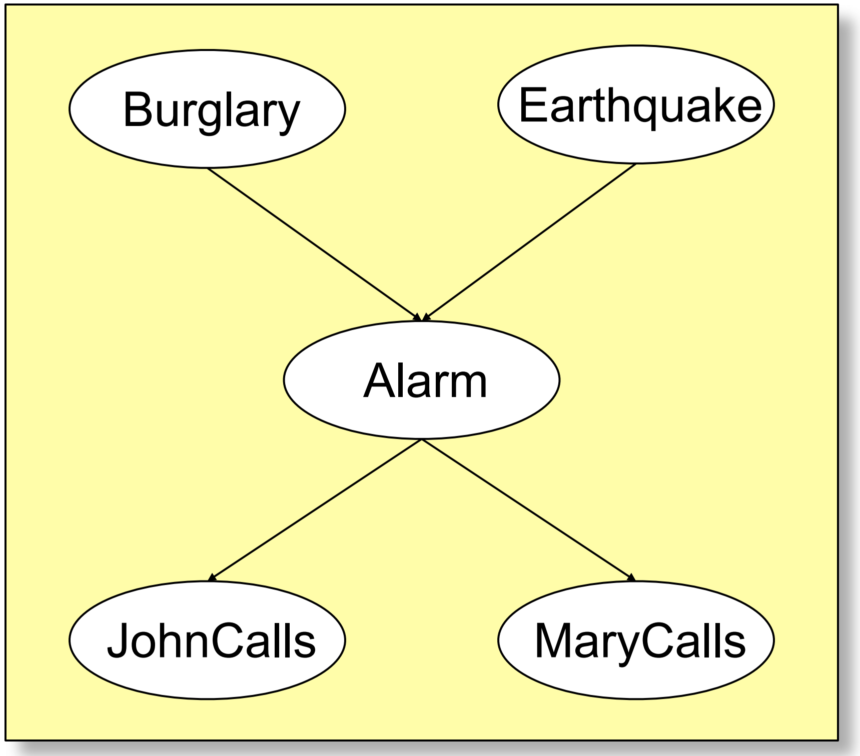
Example ordering (2)



- Order
  - M,J,E,B,A
- Network
  - 31 probabilities
  - like full joint distribution
  - thus: bad choice
- All three networks represent same probability distribution
- Last two versions
  - simply fail to represent all the conditional independence relationships
  - end up specifying a lot of unnecessary numbers instead.

Conditional independence relations in Bayesian networks

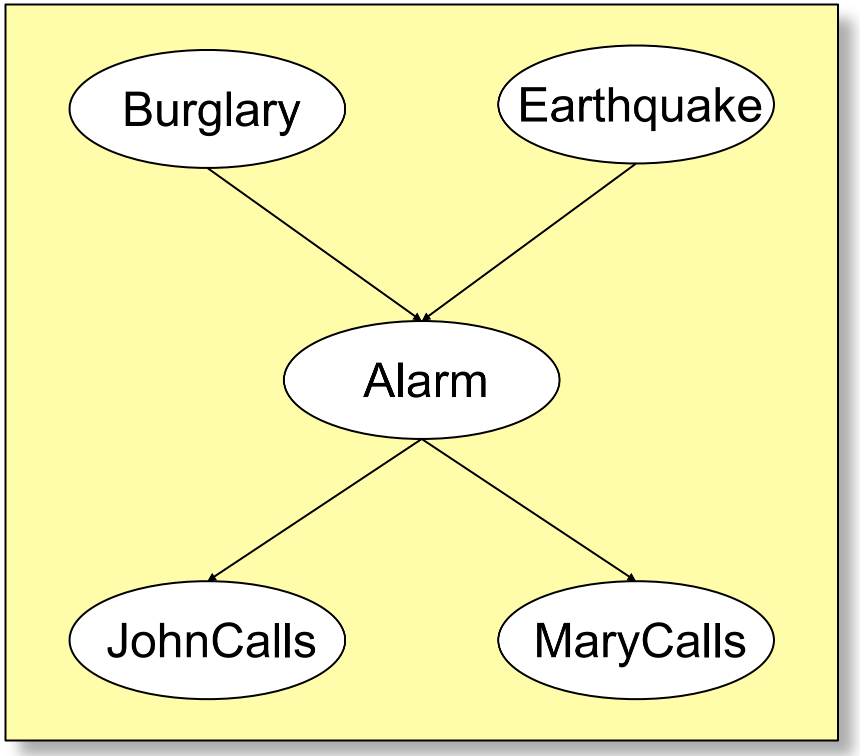
- Before
  - “numerical (global) semantics with probability distribution
  - from this derive conditional independencies
- Idea now
  - opposite direction: topological (local) semantics
  - specifies conditional independencies
  - from this derive numerical semantics



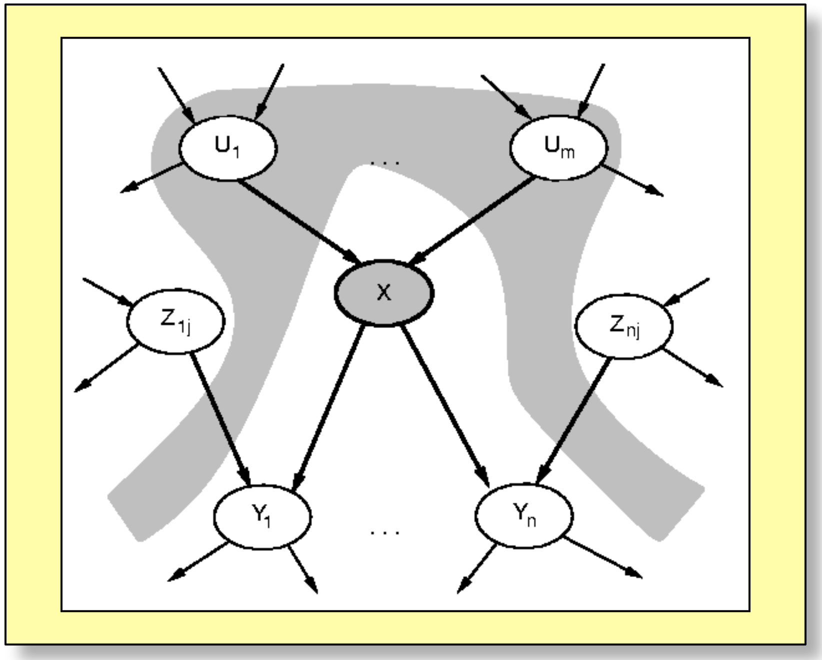
Conditional independence relations in Bayesian networks

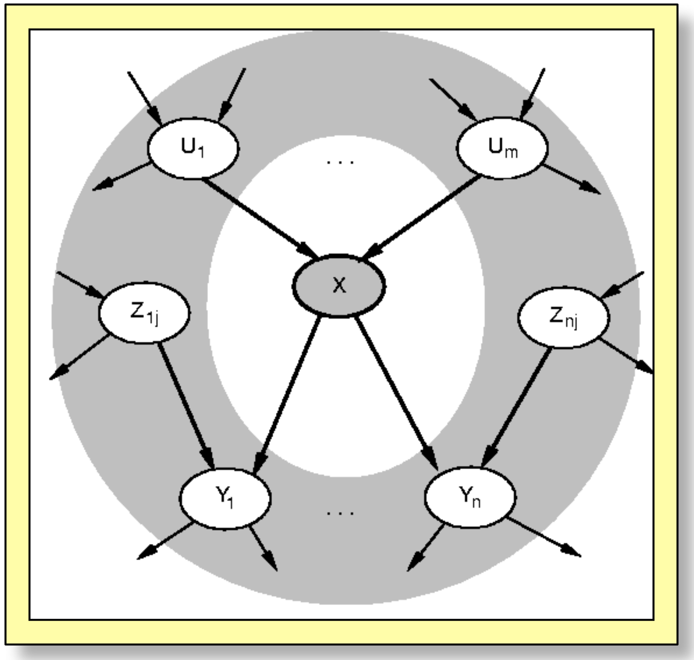
- General idea
  - A node is conditionally independent of its non-**descendants** given its parents
  - A node is conditionally independent of all other nodes in the network given its parents, children, and children's parents—that is, given its **Markov blanket**
- Examples
  J is independent of B and E given A, i.e.
  P(J|A,B,E) = P(J|A)

  B is independent of J and M given A and E, i.e.
  P(B|A,E,J,M) = P(B|A,E)



Conditional independence relations in Bayesian networks





- Node X is conditionally independent of its non-descendants (e.g., the *Zij* s) given its parents (the *Uij* s)

- A node X is conditionally independent of all other nodes in the network given its Markov blanket.

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

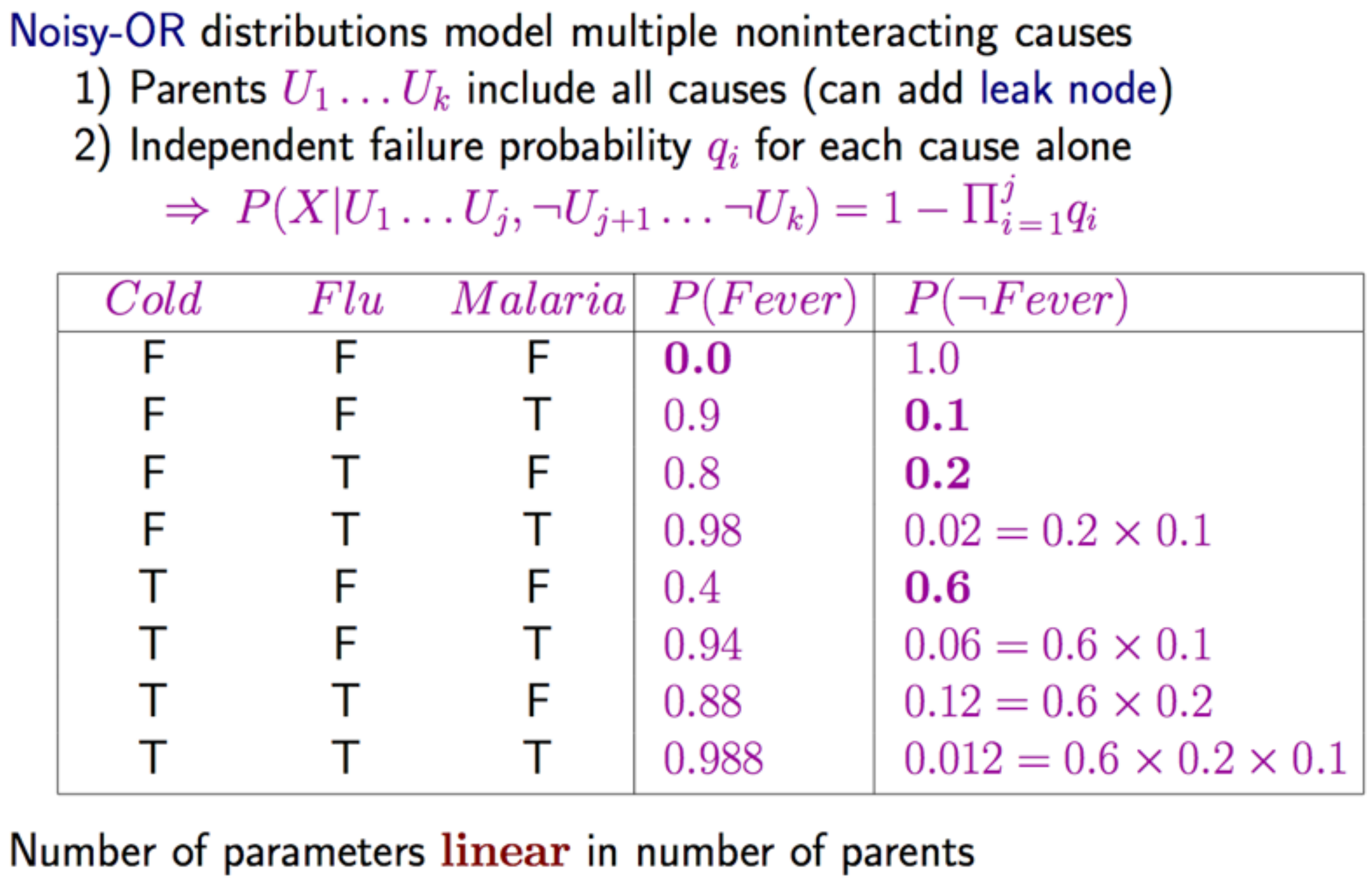
$NorthAmerican \Leftrightarrow Canadian \lor US \lor Mexican$

E.g., numerical relationships among continuous variables

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

$$P(\neg fever|cold, \neg flu, \neg malaria) = 0.6$$
$$P(\neg fever|\neg cold, flu, \neg malaria) = 0.2$$
$$P(\neg fever|\neg cold, \neg flu, malaria) = 0.1$$

Compact conditional distributions



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

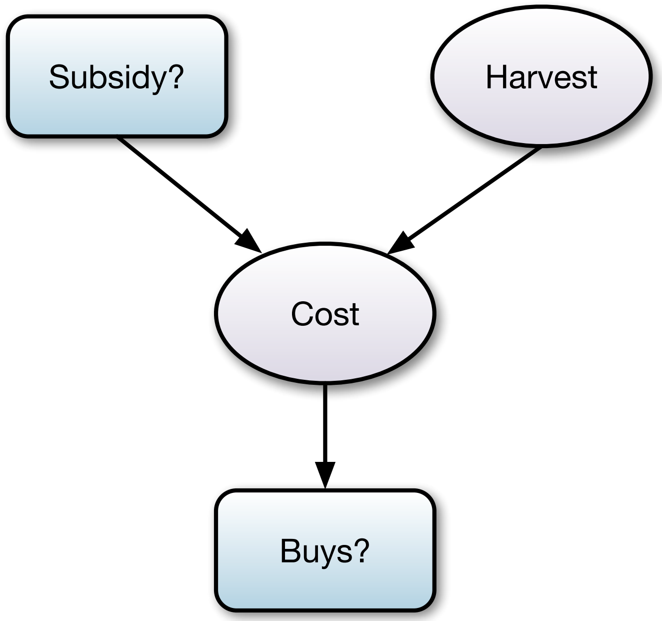
$$\Rightarrow P(X|U_1 \ldots U_j, \neg U_{j+1} \ldots \neg U_k) = 1 - \Pi_{i=1}^{j} q_i$$

$Cold$	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

Bayesian nets with continuous variables

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

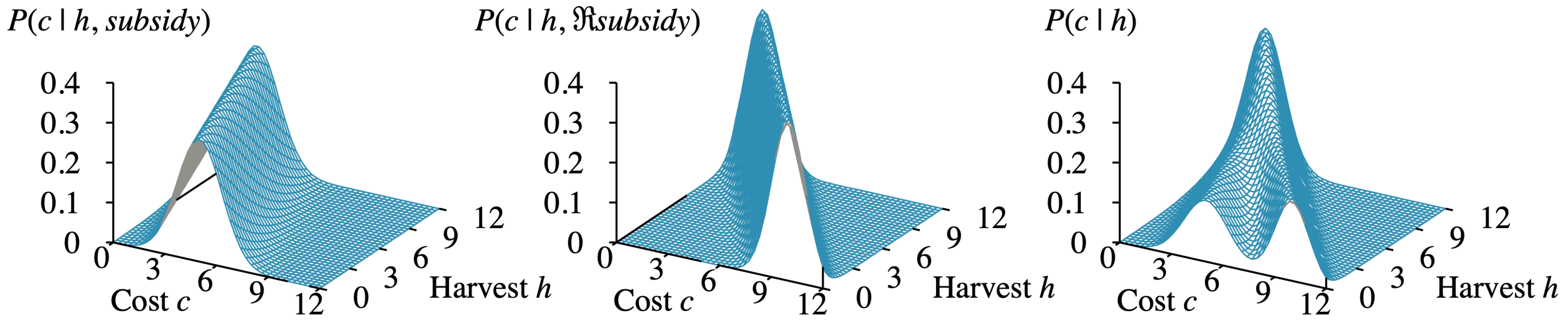
Most common is the linear Gaussian model, e.g.,:

$$\begin{aligned} & P(Cost = c | Harvest = h, Subsidy? = true) \\ & = N(a_t h + b_t, \sigma_t^2)(c) \\ & = \frac{1}{\sigma_t \sqrt{2\pi}} exp\left(-\frac{1}{2} \left(\frac{c - (a_t h + b_t)}{\sigma_t} \right)^2 \right) \end{aligned}$$

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



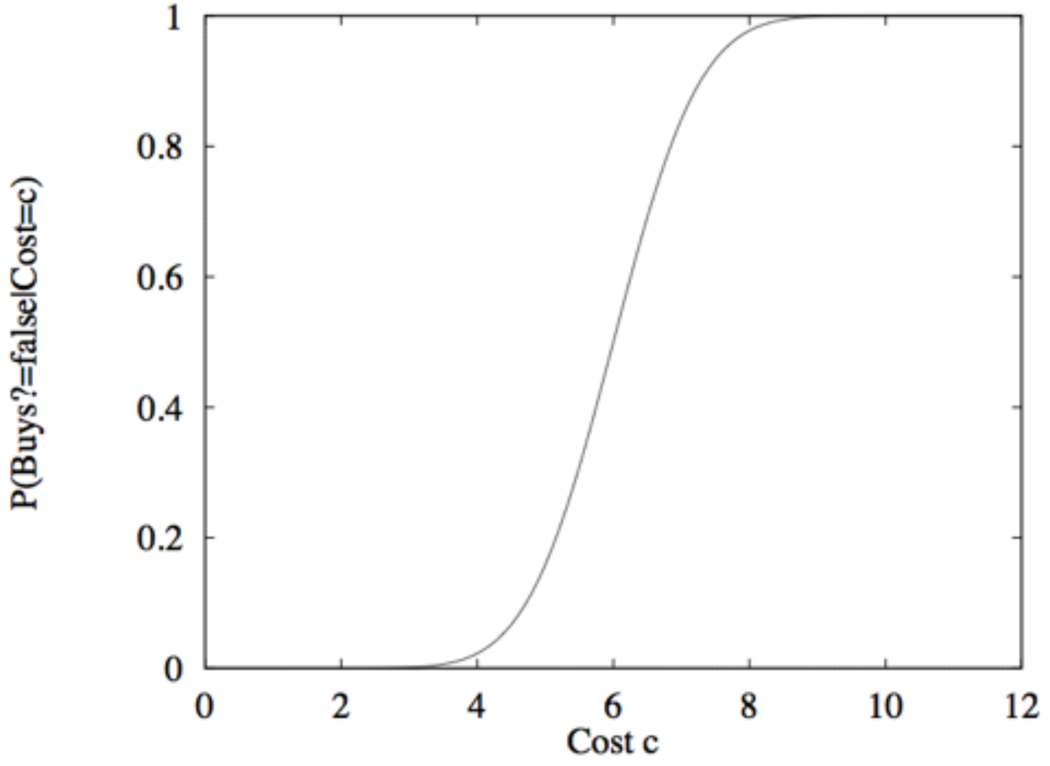
(a) (b) (c)

All-continuous network with LG distributions
  $\Rightarrow$ 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

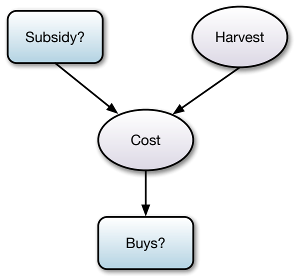
Probability of *Buys?* given *Cost* should be a "soft" threshold:



Probit distribution uses integral of Gaussian:

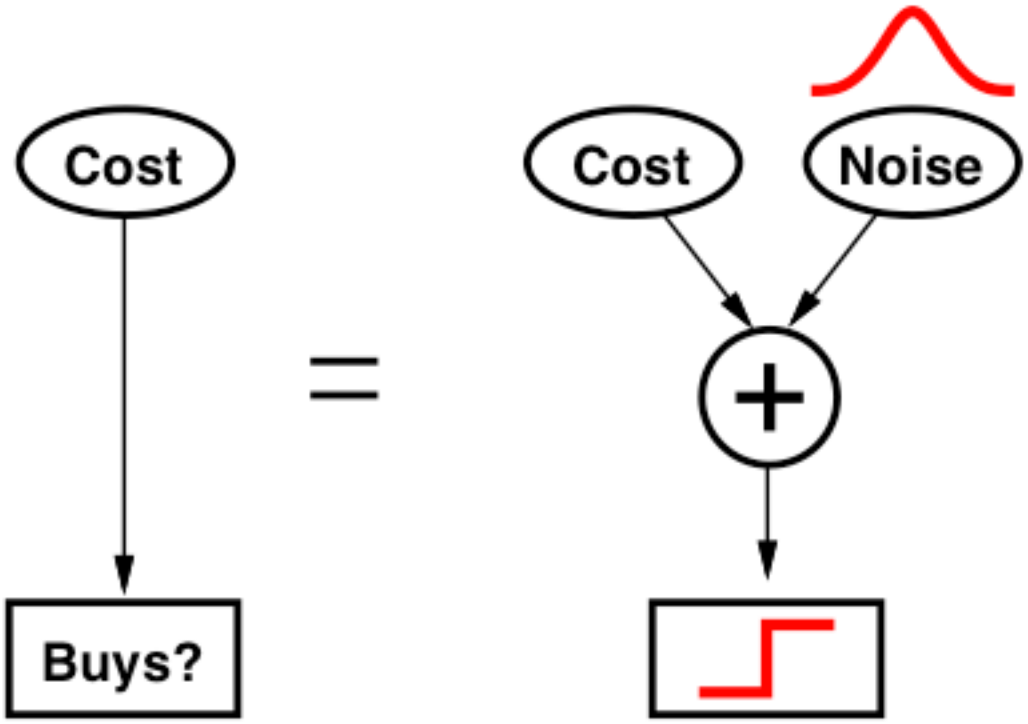
$\Phi(x) = \int_{-\infty}^{x} N(0,1)(x)dx$

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



Discrete variable w/ continuous parents

1. It's sort of the right shape
2. Can view as hard threshold whose location is subject to noise

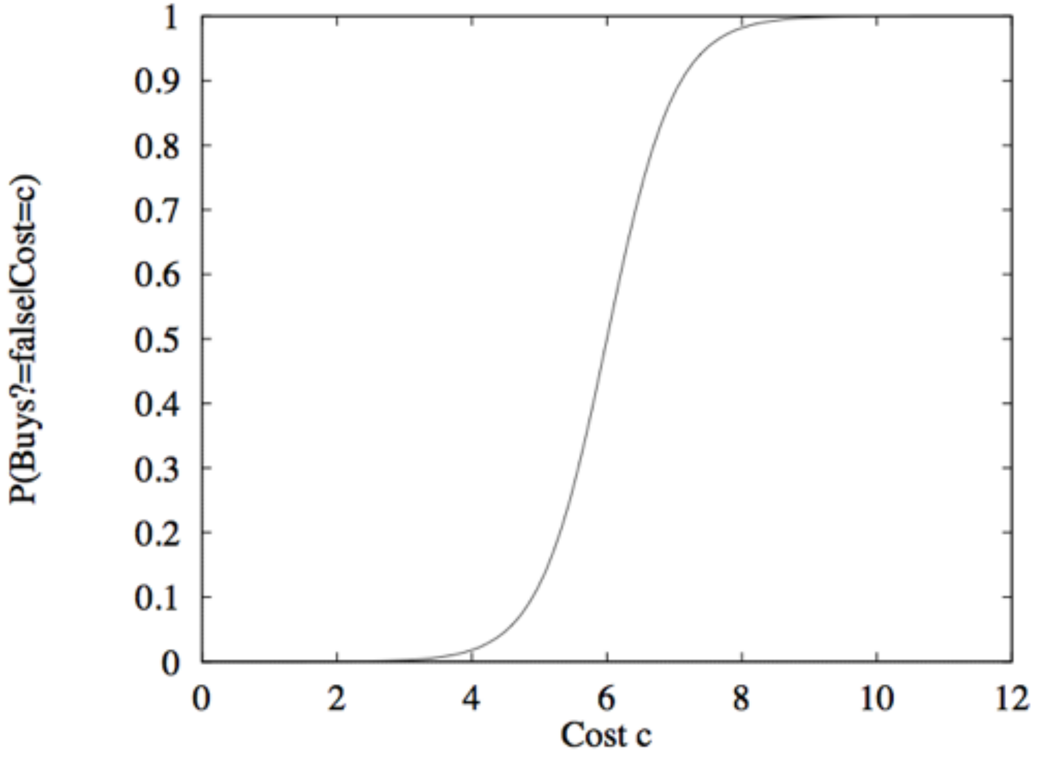


Discrete variable w/ continuous parents

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:



Inference tasks

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

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

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

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?

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 Bayesian network with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$
    $\mathbf{Q}(X) \leftarrow$ a distribution over $X$, initially empty
    for each value $x_i$ of $X$ do
        extend $\mathbf{e}$ with value $x_i$ for $X$
        $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL(VARS[$bn$], $\mathbf{e}$)
    return NORMALIZE($\mathbf{Q}(X)$)
```

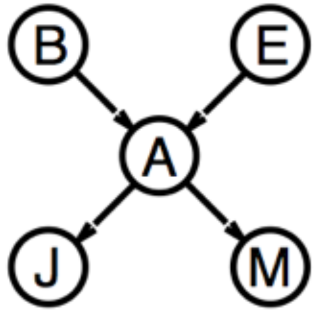
```
function ENUMERATE-ALL($vars$, $\mathbf{e}$) returns a real number
    if EMPTY?($vars$) then return 1.0
    $Y \leftarrow$ FIRST($vars$)
    if $Y$ has value $y$ in $\mathbf{e}$
        then return $P(y \mid Pa(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}$)
        else return $\Sigma_y \; P(y \mid Pa(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_y$)
            where $\mathbf{e}_y$ is $\mathbf{e}$ extended with $Y = y$
```

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:

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



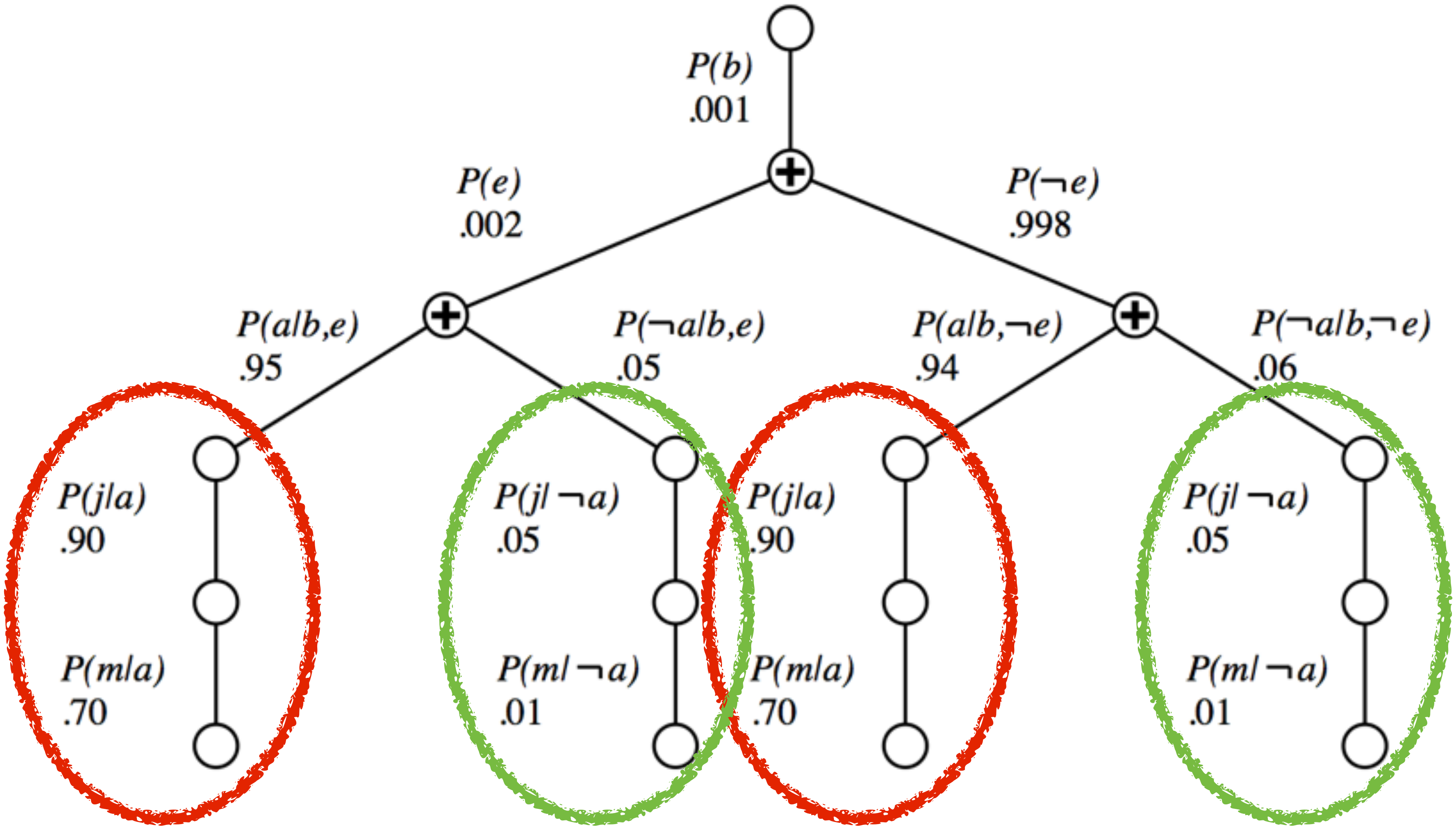
Rewrite full joint entries using product of CPT entries:

$$P(b|j, m) = \alpha \sum_e \sum_a P(b)P(e)P(a|b, e)P(j|a)P(m|a)$$

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

$$\mathbf{P}(B|j, m) = \alpha\langle 0.00059224, 0.0014919\rangle \approx \langle 0.284, 0.716\rangle$$

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

$$\mathbf{P}(B \mid j, m) = \alpha \underbrace{\mathbf{P}(B)}_{\mathbf{f}_1(B)} \sum_e \underbrace{P(e)}_{\mathbf{f}_2(E)} \sum_a \underbrace{\mathbf{P}(a \mid B, e)}_{\mathbf{f}_3(A,B,E)} \underbrace{P(j \mid a)}_{\mathbf{f}_4(A)} \underbrace{P(m \mid a)}_{\mathbf{f}_5(A)}$$

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

$$\mathbf{P}(B \mid j, m) = \alpha\, \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \sum_a \mathbf{f}_3(A, B, E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A)$$

Inference by variable elimination

$$\begin{aligned}\mathbf{f}_6(B,E) &= \sum_a \mathbf{f}_3(A,B,E)\times\mathbf{f}_4(A)\times\mathbf{f}_5(A)\\ &= (\mathbf{f}_3(a,B,E)\times\mathbf{f}_4(a)\times\mathbf{f}_5(a)) + (\mathbf{f}_3(\neg a,B,E)\times\mathbf{f}_4(\neg a)\times\mathbf{f}_5(\neg a))\end{aligned}$$

$$\mathbf{P}(B\mid j,m) = \alpha\,\mathbf{f}_1(B)\times\sum_e \mathbf{f}_2(E)\times\mathbf{f}_6(B,E)$$

$$\begin{aligned}\mathbf{f}_7(B) &= \sum_e \mathbf{f}_2(E)\times\mathbf{f}_6(B,E)\\ &= \mathbf{f}_2(e)\times\mathbf{f}_6(B,e)+\mathbf{f}_2(\neg e)\times\mathbf{f}_6(B,\neg e)\end{aligned}$$

$$\mathbf{P}(B\mid j,m) = \alpha\,\mathbf{f}_1(B)\times\mathbf{f}_7(B)$$

Variable elimination: Basic operations

Pointwise product of factors f_1 and f_2 :

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

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

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

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

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

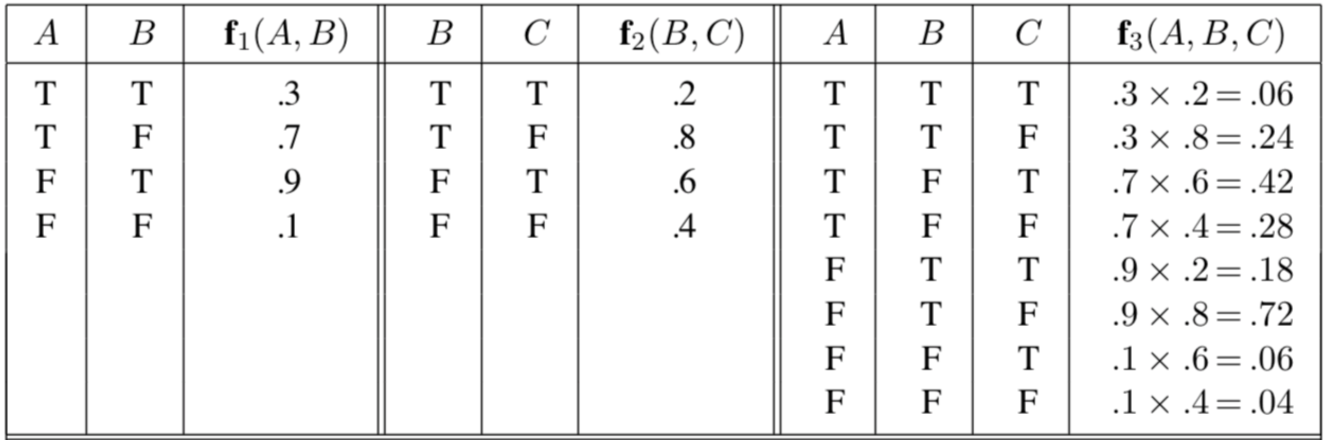
assuming $f_1, \ldots, f_i$ do not depend on X

Variable elimination: Basic operations

Pointwise product of factors f_1 and f_2 :

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

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



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

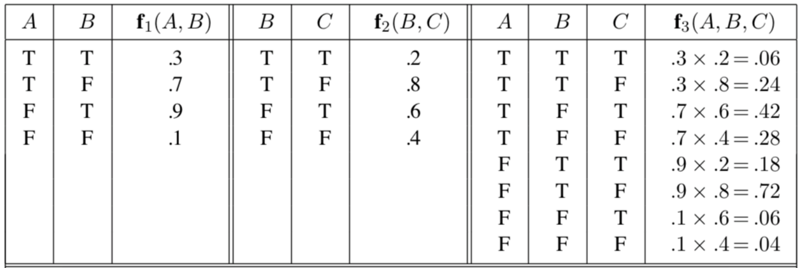
Variable elimination: Basic operations

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

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

assuming $f_1, \ldots, f_i$ do not depend on X *X = variable to be summed out*

$$\begin{aligned}\mathbf{f}(B,C) &= \sum_a \mathbf{f}_3(A,B,C) = \mathbf{f}_3(a,B,C) + \mathbf{f}_3(\neg a,B,C) \\ &= \begin{pmatrix} .06 & .24 \\ .42 & .28 \end{pmatrix} + \begin{pmatrix} .18 & .72 \\ .06 & .04 \end{pmatrix} = \begin{pmatrix} .24 & .96 \\ .48 & .32 \end{pmatrix}.\end{aligned}$$



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

Variable elimination algorithm

```
**function** ELIMINATION-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 Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $factors \leftarrow [\,]$
  **for each** $var$ **in** ORDER($bn$.VARS) **do**
      $factors \leftarrow$ [MAKE-FACTOR($var$, $\mathbf{e}$)|$factors$]
      **if** $var$ is a hidden variable **then** $factors \leftarrow$ SUM-OUT($var$, $factors$)
  **return** NORMALIZE(POINTWISE-PRODUCT($factors$))
```

Complexity of exact inference

Singly connected networks (or polytrees):

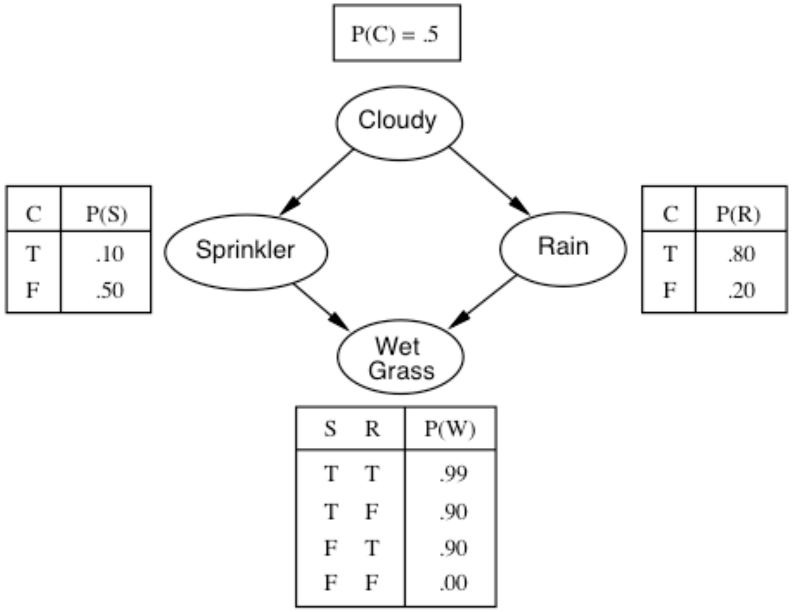
– any two nodes are connected by at most one (undirected) path

– time and space cost of variable elimination are $O(d^k n)$

Multiply connected networks:

– can reduce 3SAT to exact inference $\Rightarrow$ NP-hard

– equivalent to **counting** 3SAT models $\Rightarrow$ #P-complete



C	P(S)
T	.10
F	.50

C	P(R)
T	.80
F	.20

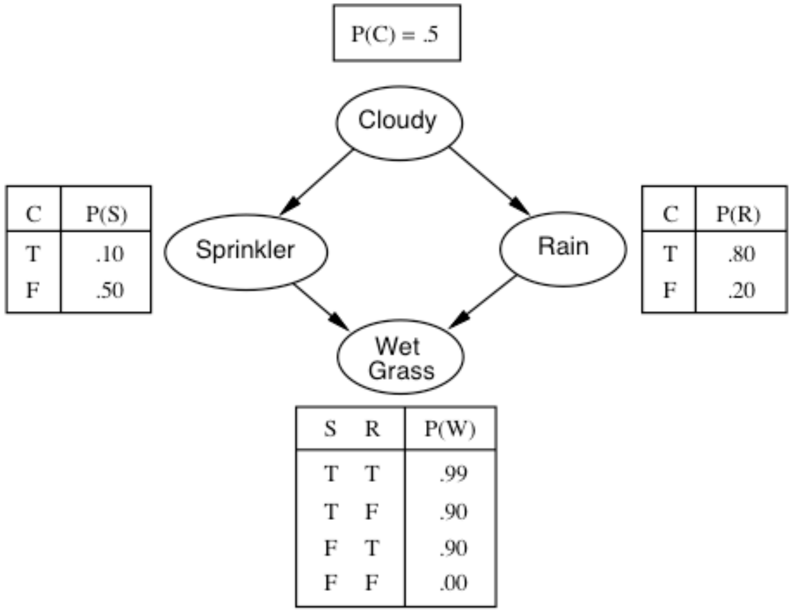
S	R	P(W)
T	T	.99
T	F	.90
F	T	.90
F	F	.00

Clustering algorithms

- Variable elimination algorithm is simple and efficient for answering individual queries
- For computation of posterior probabilities for all the variables in a network it can be less efficient: $O(n^2)$
- Using clustering algorithms (also known as join tree algorithms), this can be reduced to $O(n)$.
- 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.

Clustering algorithms

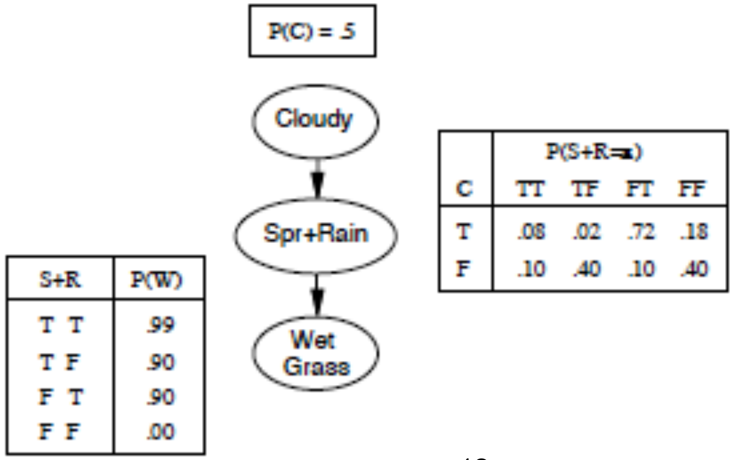
- Multiply connected network can be converted into a polytree by combining *Sprinkler* and *Rain* node into cluster node called *Sprinkler+Rain*.
- Two Boolean nodes replaced by a mega-node that takes on four possible values: *TT, TF, FT, FF*. The mega-node has only one parent, the Boolean variable *Cloudy*, so there are two conditioning cases.



C	P(S)
T	.10
F	.50

C	P(R)
T	.80
F	.20

S	R	P(W)
T	T	.99
T	F	.90
F	T	.90
F	F	.00



C	P(S+R=x)			
	TT	TF	FT	FF
T	.08	.02	.72	.18
F	.10	.40	.10	.40

S+R	P(W)
T T	.99
T F	.90
F T	.90
F F	.00

APPROXIMATE INFERENCE IN BAYESIAN NETWORKS

- Randomized sampling algorithms, also called Monte Carlo algorithms
- Provide approximate answers whose accuracy depends on the number of samples generated
- Monte Carlo algorithms are used in many branches of science to estimate quantities that are difficult to calculate exactly.
- Here: sampling applied to the computation of posterior probabilities
- Two families of algorithms: direct sampling and Markov chain sampling

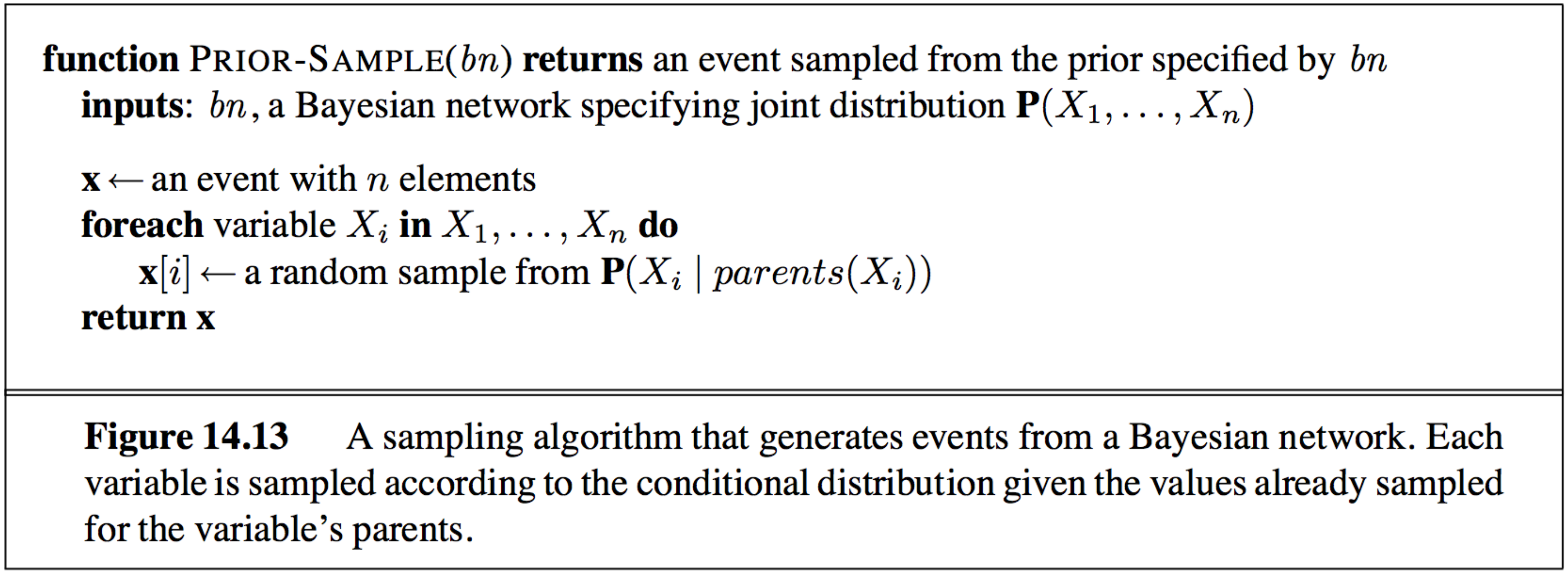
Direct sampling methods

- Generation of samples from a known probability distribution
- Example

$$\mathbf{P}(\text{Coin}) = \langle 0.5, 0.5 \rangle$$

- Sampling from this distribution is exactly like flipping the coin: with probability 0.5 it will return heads , and with probability 0.5 it will return tails.

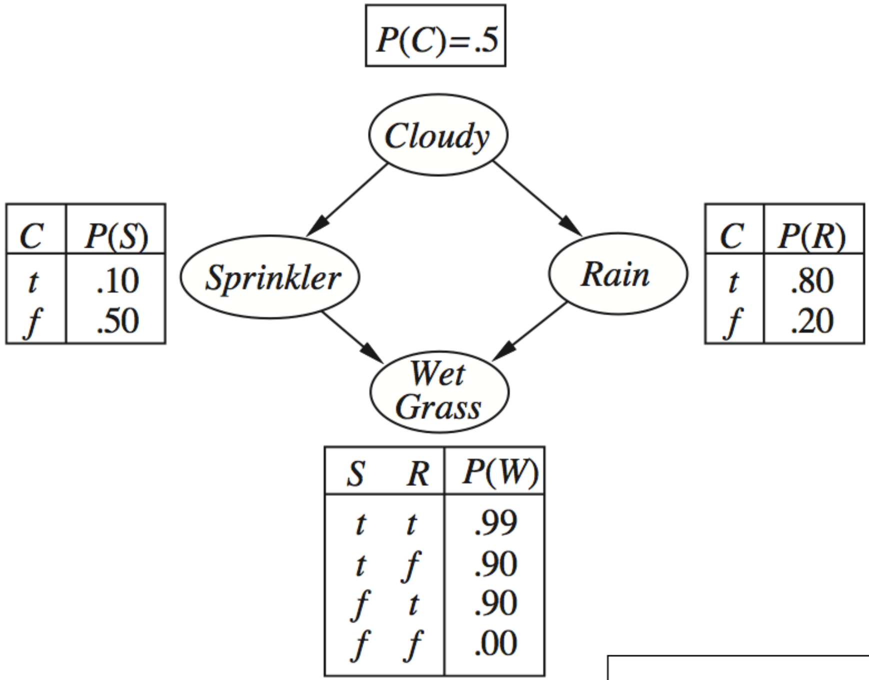
Direct sampling methods



```
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 $\mathbf{x}$
```

Figure 14.13 A sampling algorithm that generates events from a Bayesian network. Each variable is sampled according to the conditional distribution given the values already sampled for the variable's parents.

Direct sampling methods



C	P(S)
t	.10
f	.50

C	P(R)
t	.80
f	.20

S	R	P(W)
t	t	.99
t	f	.90
f	t	.90
f	f	.00

```
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 $\mathbf{x}$
```

Figure 14.13 A sampling algorithm that generates events from a Bayesian network. Each variable is sampled according to the conditional distribution given the values already sampled for the variable's parents.

Direct sampling methods

- PRIOR-SAMPLE generates samples from the prior joint distribution specified by the network

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

- Each sampling step depends only on the parent values

$$S_{PS}(x_1 \dots x_n) = P(x_1 \dots x_n) \ .$$

Computing answers

- Answers are computed by counting the actual samples generated
- Say, *N* total samples and $N_{PS}(x_1, \ldots, x_n)$ number of times the event $x_1, \ldots x_n$ occurs in the samples

$$\lim_{N\to\infty} \frac{N_{PS}(x_1, \ldots, x_n)}{N} = S_{PS}(x_1, \ldots, x_n) = P(x_1, \ldots, x_n) \,.$$

Computing answers

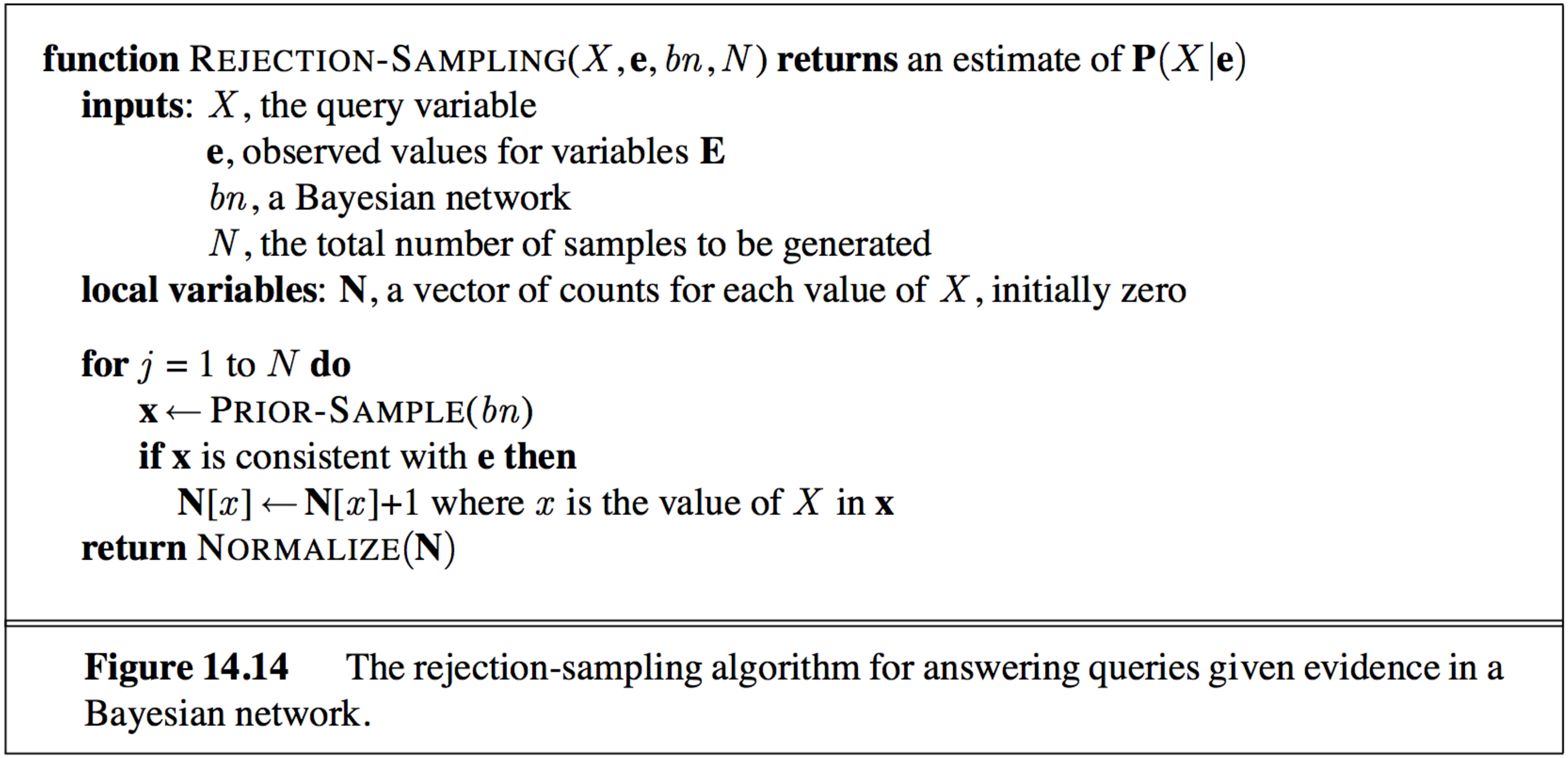
$$\lim_{N\to\infty} \frac{N_{PS}(x_1,\ldots,x_n)}{N} = S_{PS}(x_1,\ldots,x_n) = P(x_1,\ldots,x_n)\,.$$

- For example, consider the event produced earlier: [*true,false,true,true*]. The sampling probability for this event is

$$S_{PS}(\mathit{true}, \mathit{false}, \mathit{true}, \mathit{true}) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324\,.$$

- Hence, in the limit of large *N*, we expect 32.4% of the samples to be of this event.

Rejection sampling



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

  for $j$ = 1 to $N$ do
      $\mathbf{x} \leftarrow$ PRIOR-SAMPLE($bn$)
      if $\mathbf{x}$ is consistent with $\mathbf{e}$ then
         $\mathbf{N}[x] \leftarrow \mathbf{N}[x]+1$ where $x$ is the value of $X$ in $\mathbf{x}$
  return NORMALIZE($\mathbf{N}$)
```

Figure 14.14 The rejection-sampling algorithm for answering queries given evidence in a Bayesian network.

Rejection sampling

- Let $\hat{\mathbf{P}}(X|\mathbf{e})$ be the estimated distribution. Then from the definition just given

$$\hat{\mathbf{P}}(X \mid \mathbf{e}) = \alpha\, \mathbf{N}_{PS}(X, \mathbf{e}) = \frac{\mathbf{N}_{PS}(X, \mathbf{e})}{N_{PS}(\mathbf{e})} \, .$$

$$\hat{\mathbf{P}}(X \mid \mathbf{e}) \approx \frac{\mathbf{P}(X, \mathbf{e})}{P(\mathbf{e})} = \mathbf{P}(X \mid \mathbf{e}) \, .$$

- Rejection sampling produces a consistent estimate of the true probability.

Rejection sampling

- Estimate ***P****(Rain | Sprinkler = true)*, using 100 samples. Of the 100 that we generate, suppose that 73 have *Sprinkler = false* and are rejected, while 27 have *Sprinkler = true.*
- Of the 27, 8 have *Rain = true* and 19 have *Rain = false*.
- Thus,

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

Rejection sampling

How often does it rain the day after we have observed aurora borealis? Ignoring all those days with no aurora borealis…

Likelihood weighting

- Likelihood weighting avoids the inefficiency of rejection sampling
- It generates only events that are consistent with the evidence **e**.
- It is a particular instance of the general statistical technique of importance sampling, tailored for inference in Bayesian networks.
- Let’s see how it works…

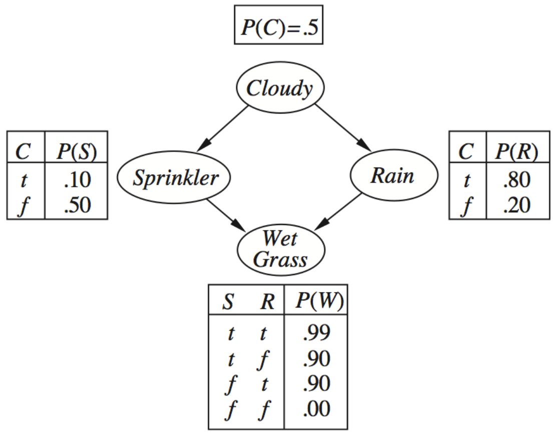
Likelihood weighting

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

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

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

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

Likelihood weighting

For the evidence P(Rain | Cloudy = true , WetGrass = true)



- *Cloudy* is an evidence variable with value *true*. Therefore, we set *w ← w×P(Cloudy=true) = 0.5* .
- Sprinkler is not an evidence variable, so sample from ***P****(Sprinkler | Cloudy = true)* = ⟨0.1,0.9⟩; suppose this returns *false*.
- Similarly, sample from ***P****(Rain|Cloudy=true)* = ⟨0.8,0.2⟩; suppose this returns *true* .
- *WetGrass* is an evidence variable with value *true*. Therefore, we set *w ← w × P(WetGrass=true | Sprinkler=false, Rain=true) = 0.5 x 0.9 = 0.45*

Likelihood weighting

- The weight for a given sample **x** is the product of the likelihoods for each evidence variable given its parents

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

- Multiplying the last two equations we see that the weighted probability of a sample has the particularly convenient form

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

Likelihood weighting

- For any particular value *x* of *X*, the estimated posterior probability can be calculated as follows:

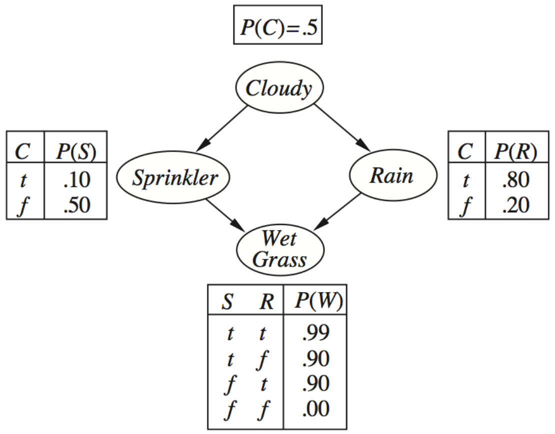
$$
\begin{aligned}
\hat{P}(x \mid \mathbf{e}) &= \alpha \sum_{\mathbf{y}} N_{WS}(x, \mathbf{y}, \mathbf{e}) w(x, \mathbf{y}, \mathbf{e}) && \text{from LIKELIHOOD-WEIGHTING} \\
&\approx \alpha' \sum_{\mathbf{y}} S_{WS}(x, \mathbf{y}, \mathbf{e}) w(x, \mathbf{y}, \mathbf{e}) && \text{for large } N \\
&= \alpha' \sum_{\mathbf{y}} P(x, \mathbf{y}, \mathbf{e}) && \text{by Equation (14.9)} \\
&= \alpha' P(x, \mathbf{e}) = P(x \mid \mathbf{e}) \, .
\end{aligned}
$$

- Hence, likelihood weighting returns consistent estimates.

Inference by Markov chain simulation

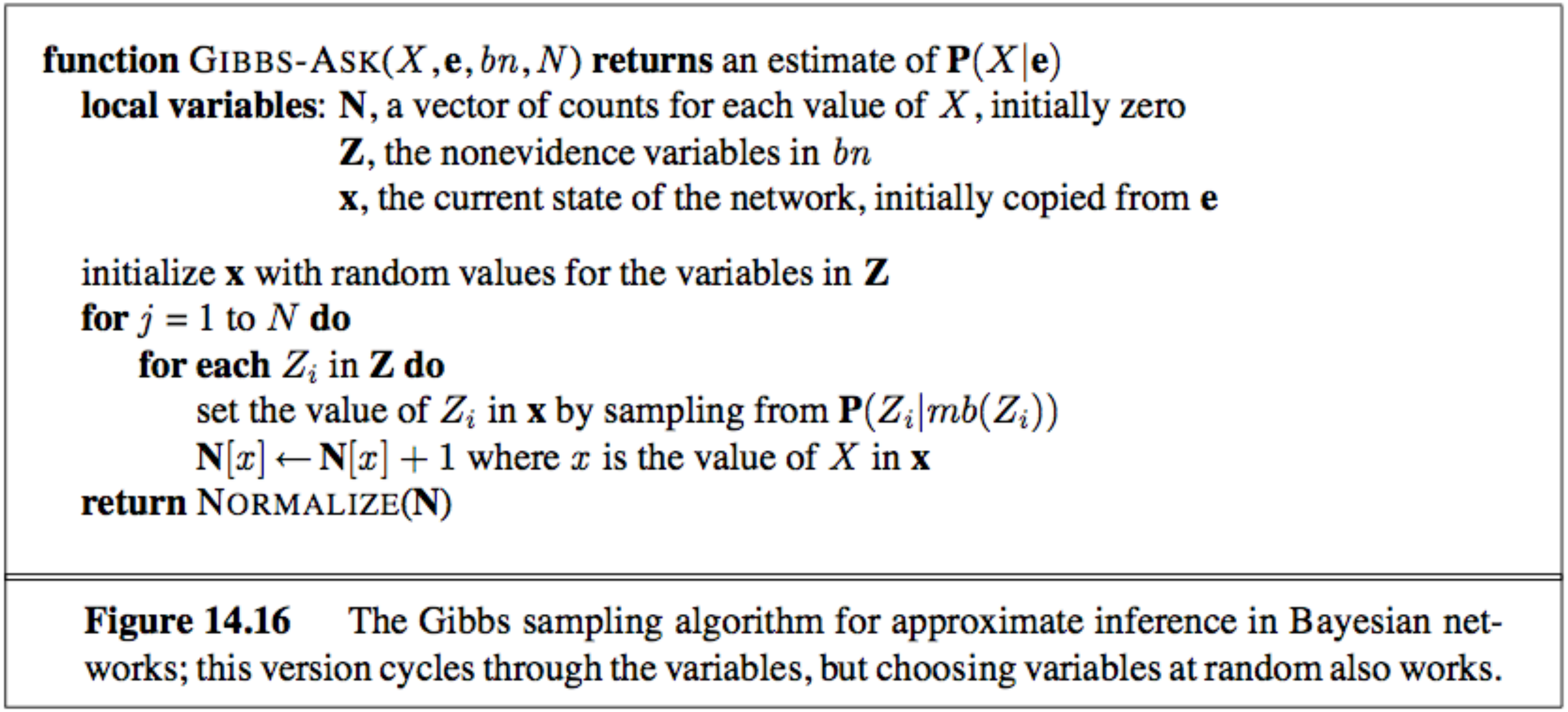
- Markov chain Monte Carlo (MCMC) algorithms work quite differently from rejection sampling and likelihood weighting.
- MCMC state change similar to SA
- Gibbs sampling well suited for BN
- Starts with an arbitrary state and generates a next state by randomly sampling a value for one of the nonevidence variables X_i.
- Sampling for X_i is done conditioned on the current values of the variables in the Markov blanket of X_i.

Inference by Markov chain simulation



- **P**(*Rain* | *Sprinkler=true, WetGrass=true*)
- Initial state is *[true,**true**,false,**true**]*
- Cloudy:
  - **P**(*Cloudy*|*Sprinkler=true,Rain=false*). Suppose the result is Cloudy = false.
  - Then the new current state is [*false,**true**,false,**true***].
- Rain:
  - Given the current values of its Markov blanket variables: **P**(*Rain* | *Cloudy = false , Sprinkler = true , WetGrass = true*). Suppose this yields *Rain = true* .
  - The new current state is [*false,**true**,true,**true***].

Inference by Markov chain simulation



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

  initialize $\mathbf{x}$ with random values for the variables in $\mathbf{Z}$
  for $j$ = 1 to $N$ do
     for each $Z_i$ in $\mathbf{Z}$ do
        set the value of $Z_i$ in $\mathbf{x}$ 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 $\mathbf{x}$
  return NORMALIZE($\mathbf{N}$)
```

Figure 14.16 The Gibbs sampling algorithm for approximate inference in Bayesian networks; this version cycles through the variables, but choosing variables at random also works.

Summary

This chapter has described **Bayesian networks**, a well-developed representation for uncertain knowledge. Bayesian networks play a role roughly analogous to that of propositional logic for definite knowledge.

- A Bayesian network is a directed acyclic graph whose nodes correspond to random variables; each node has a conditional distribution for the node given its parents.
- Bayesian networks provide a concise way to represent **conditional independence** relationships in the domain.
- A Bayesian network specifies a full joint distribution; each joint entry is defined as the product of the corresponding entries in the local conditional distributions. A Bayesian network is often exponentially smaller than the full joint distribution.

Summary (2)

- Inference in Bayesian networks means computing the probability distribution of a set of query variables, given a set of evidence variables. Exact inference algorithms, such as **variable elimination**, evaluate sums of products of conditional probabilities as efficiently as possible.
- In polytrees (singly connected networks), exact inference takes time linear in the size of the network. In the general case, the problem is intractable.
- Stochastic approximation techniques such as **likelihood weighting** and **Markov chain Monte Carlo** can give reasonable estimates of the true posterior probabilities in a network and can cope with much larger networks than can exact algorithms.