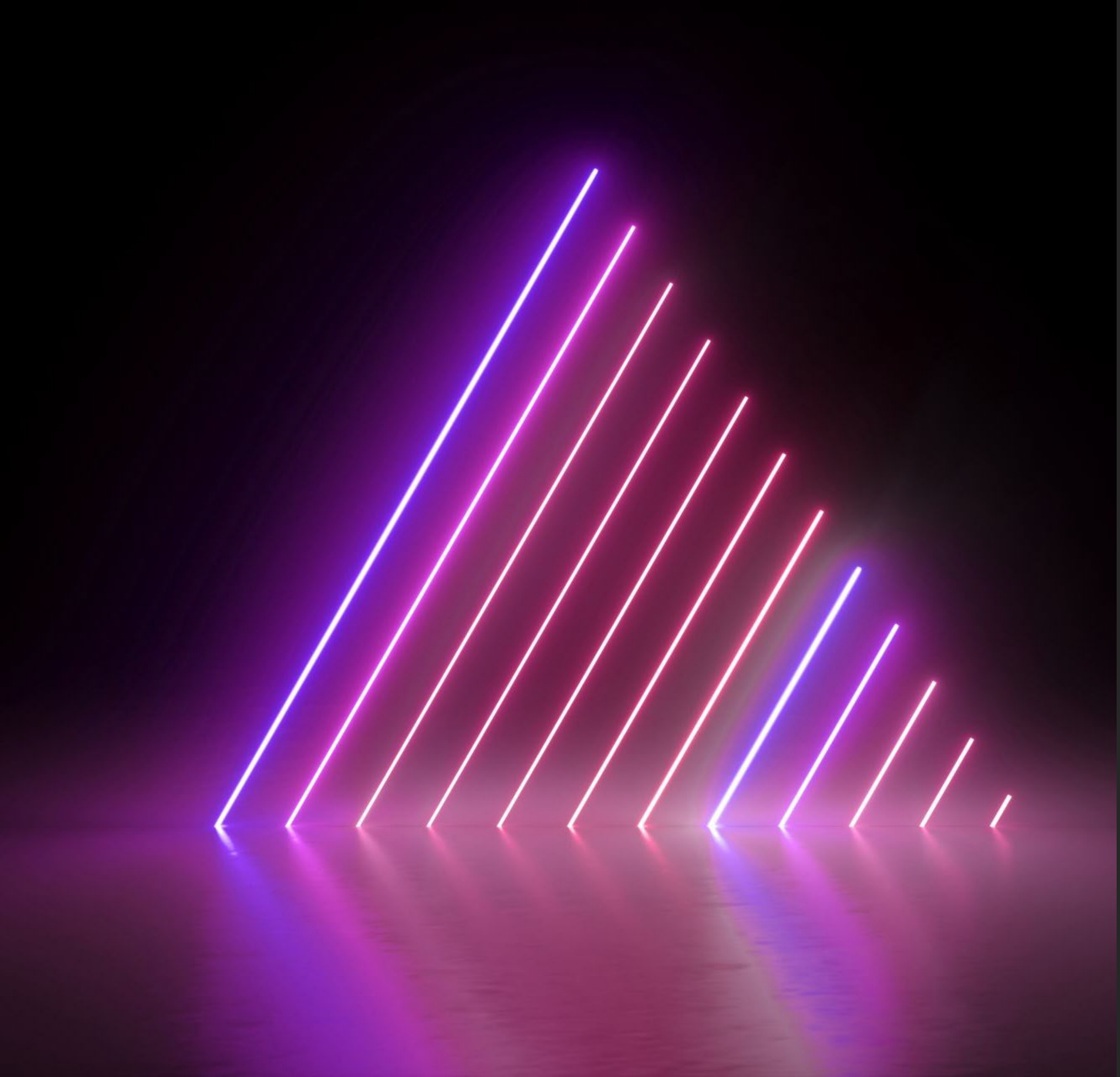
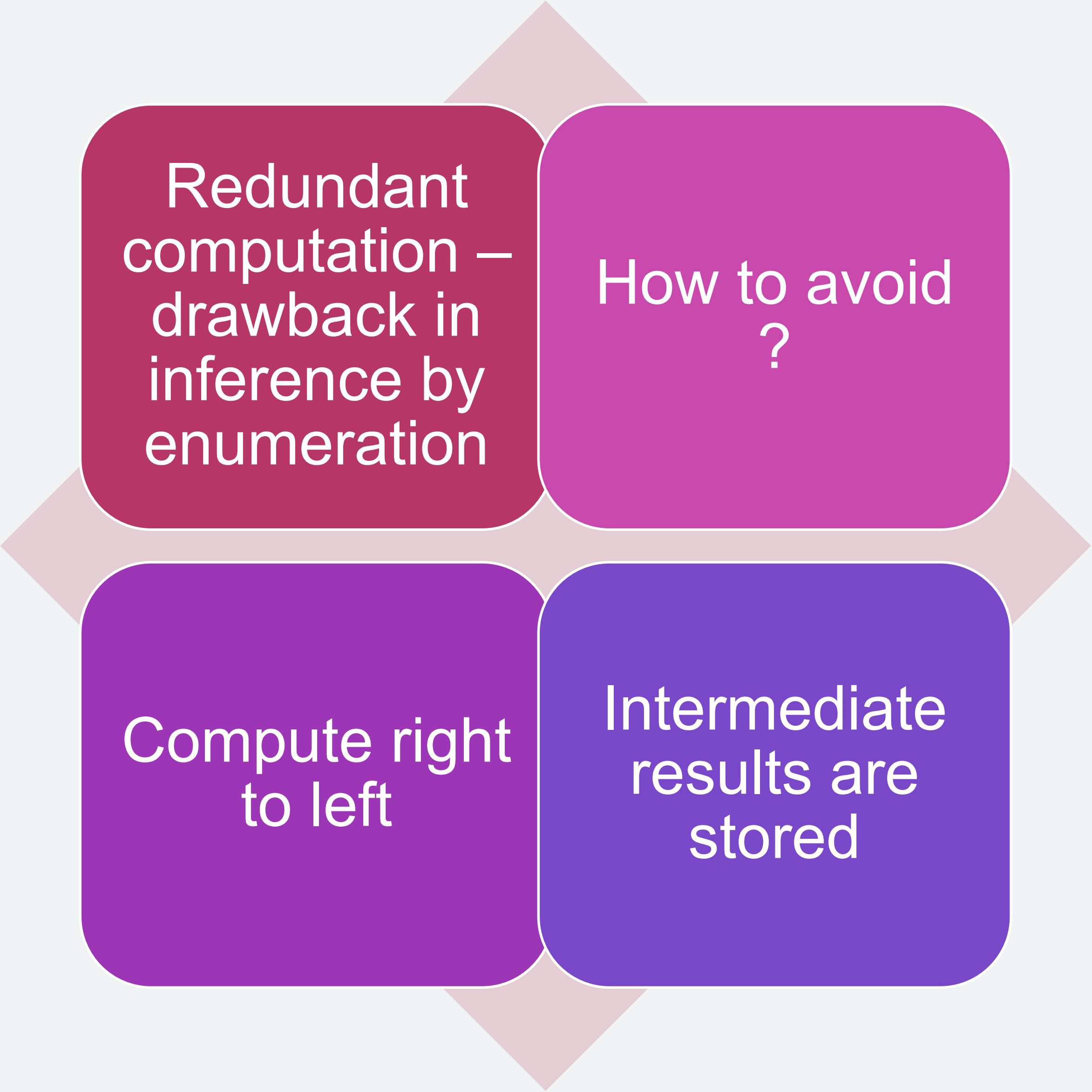


Exact Inference Bayesian Networks

Variable elimination algorithm



Redundant computation – drawback in inference by enumeration

How to avoid ?

Compute right to left

Intermediate results are stored

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

Steps

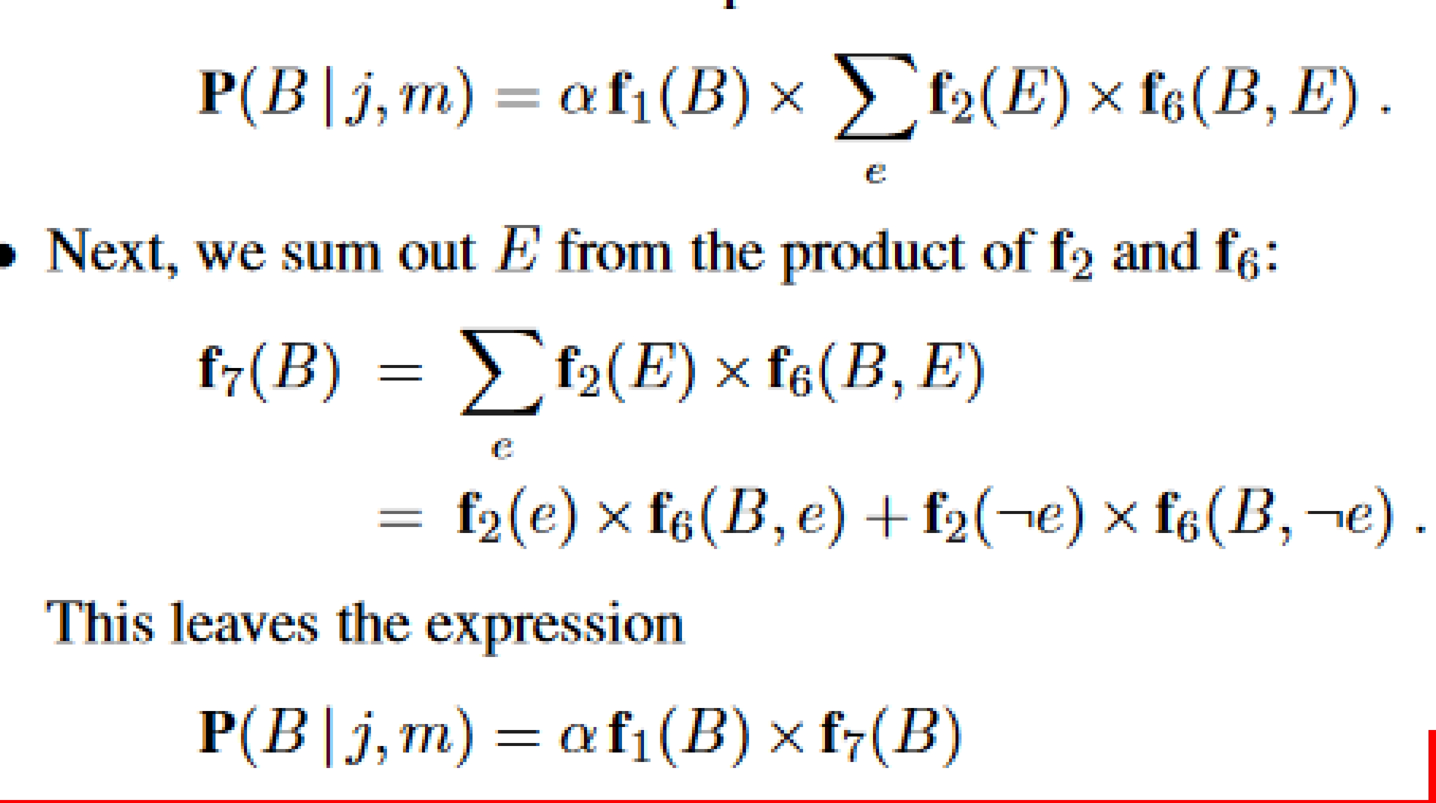
- Replace by functions
- P(b) dependent on b
- P(e) dependent on e
- P(a|B,e) dependent on a, B,e [include evidence]
- Last two terms dependent on a

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

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

Steps

- **In the second part of the equation eliminate A**
- **F6(B,E) = f3(a,b,e) * f4(a) * f5(a) + f3(¬a,b,e) * f4(¬a) * f5(¬a)**
- **Substitute in the equation P(B | j,m)**



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

- Next, we sum out E from the product of $\mathbf{f}_2$ and $\mathbf{f}_6$:

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

This leaves the expression

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

Steps

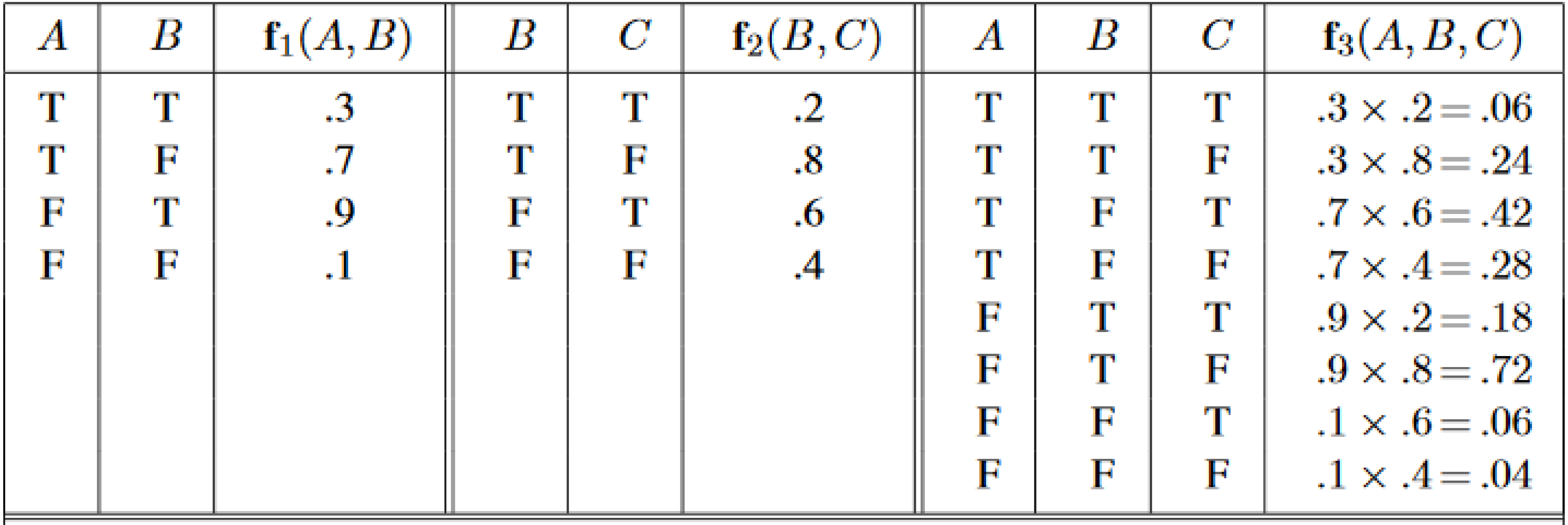
- F7 first normal e then negation of e
- Then substitute in P(B| j,m) finally

Operations on factor

The pointwise product of two factors f1 and f2 yields a new factor f whose variables are the union of the variables in f1 and f2

$f(X1 \,.\,.\,.\, Xj\ , Y1 \,.\,.\,.\, Yk, Z1 \,.\,.\,.\, Zl) = f1(X1 \,.\,.\,.\, Xj\ , Y1 \,.\,.\,.\, Yk)\ f2(Y1 \,.\,.\,.\, Yk, Z, \,.\,.\,.\, Zl).$

Given f1 and f2 calculate f3



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	$.1 \times .4 = .04$

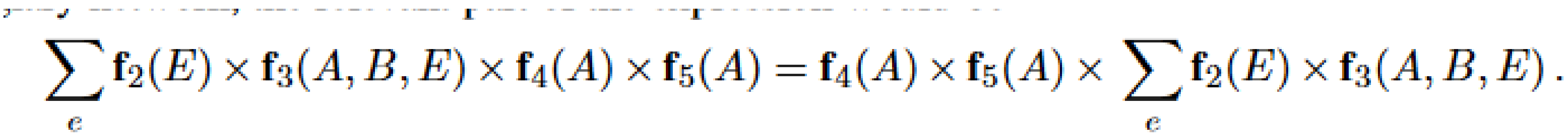
- $f(B, C) = \sum_{a} f3(A, B, C) = f3(a, B, C) + f3(\neg a, B, C)$

So look for a positive and negative (if Summation)

- Represent as matrix values of a true and not true

$$= \begin{pmatrix} .06 & .24 \\ .42 & .28 \end{pmatrix} + \begin{pmatrix} .18 & .72 \\ .06 & .04 \end{pmatrix} = \begin{pmatrix} .24 & .96 \\ .48 & .32 \end{pmatrix}$$

f3(a, B, C) + *f3(¬a, B, C)*

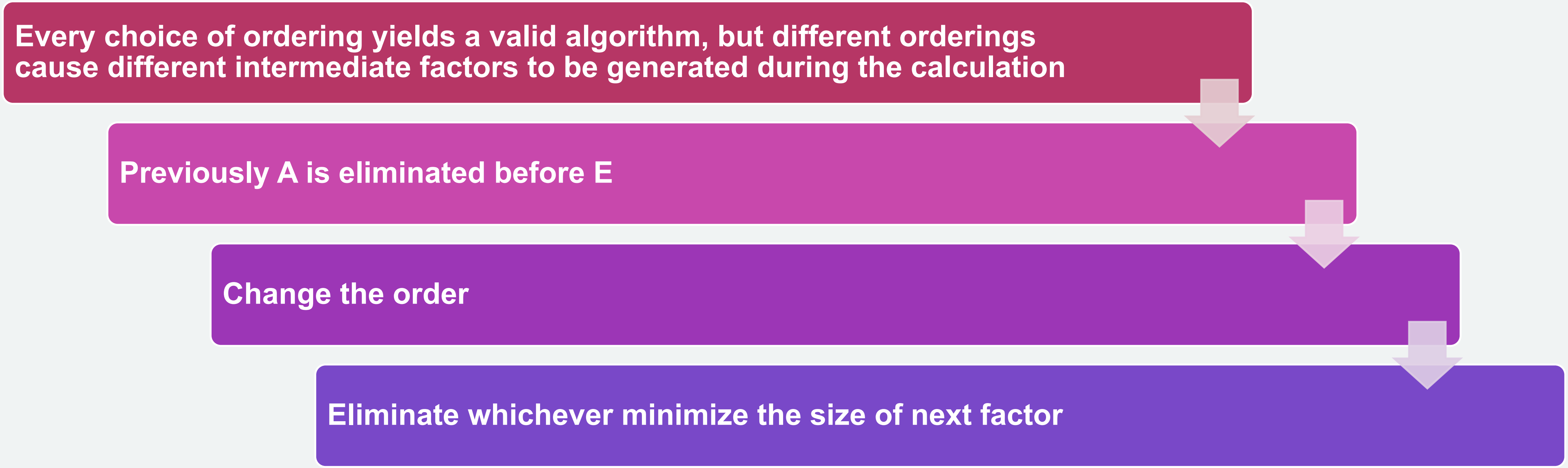


$$\sum_{e} \mathbf{f}_2(E) \times \mathbf{f}_3(A,B,E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A) = \mathbf{f}_4(A) \times \mathbf{f}_5(A) \times \sum_{e} \mathbf{f}_2(E) \times \mathbf{f}_3(A,B,E)\,.$$

Steps

- Any factor that does not fit in the sum take it outside
- Sum of e so move f4 and f5 outside

Variable ordering and variable relevance



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

Previously A is eliminated before E

Change the order

Eliminate whichever minimize the size of next factor

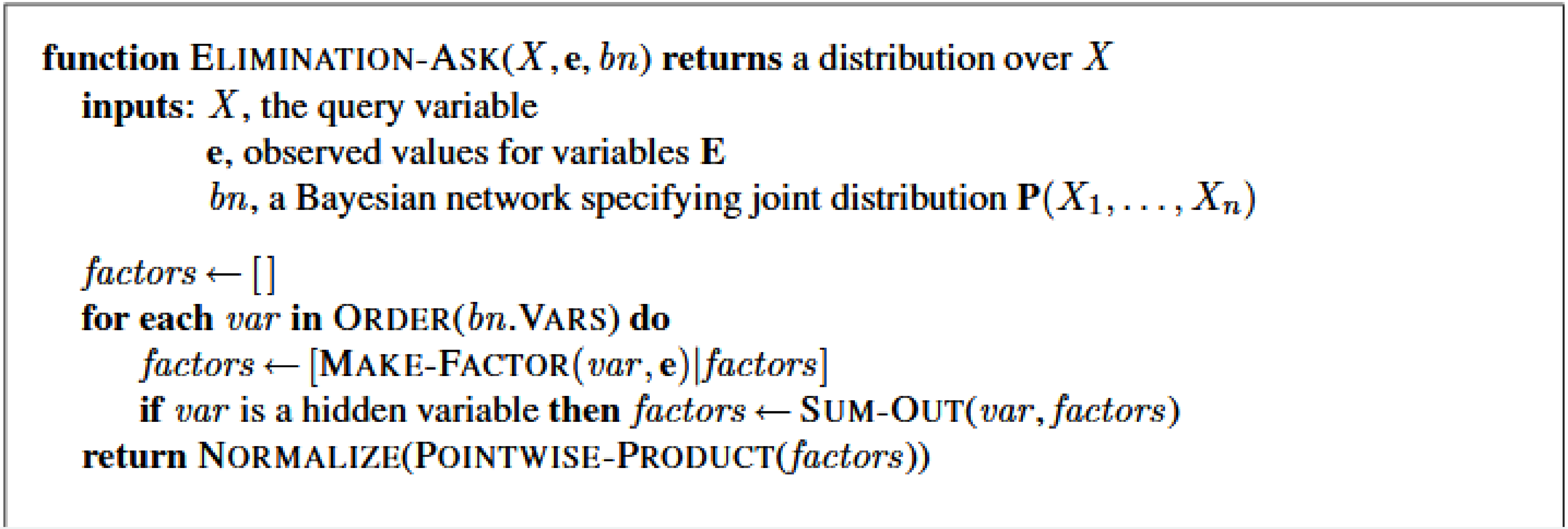
$$\mathbf{P}(B \mid j, m) = \alpha\, \mathbf{f}_1(B) \times \sum \mathbf{f}_4(A) \times \mathbf{f}_5(A) \times \sum \mathbf{f}_2(E) \times \mathbf{f}_3(A, B, E)\ ,$$

- instead of eliminating A eliminate E

Example

- $P(J \mid b) = \alpha\, P(b) \sum_{e} P(e) \sum_{a} P(a \mid b, e)\, P(J \mid a) \sum_{m} P(m \mid a)$
  - *Last term sum will be1*
  - *Remove any leaf that is not query or evidence variable*
  - *Any variable that is not ancestor is irrelavent to query*
  - *So using variable elimination remove all these variables*

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 $\mathbf{P}(X_1, \ldots, X_n)$

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

Complexity of exact inference

There is atmost one undirected path between any two nodes – singly connected networks or poly trees

Time and space complexity in poly tree is linear in the size of the network. (size refers to number of entries in CPT)

For multiply connected network time and space complexity for variable elimination is exponential

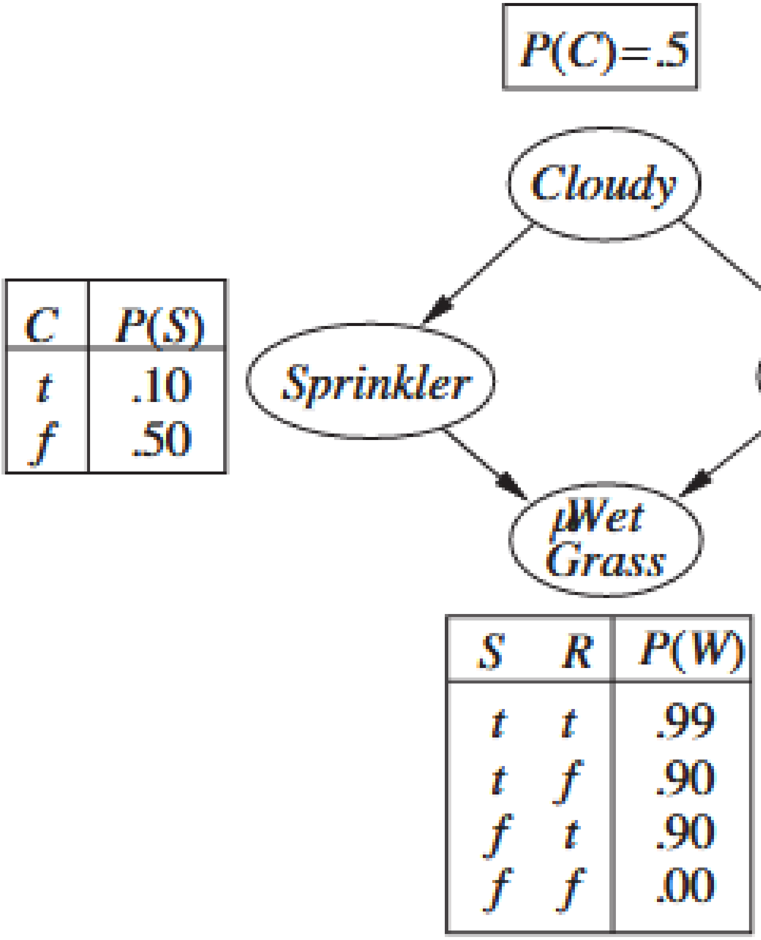
Clustering algorithm

Calculating posterior probability for all variables is less efficient and requires o(n^2)

the time can be reduced to O(n) using clustering

Clustering – join individual nodes to form clusters

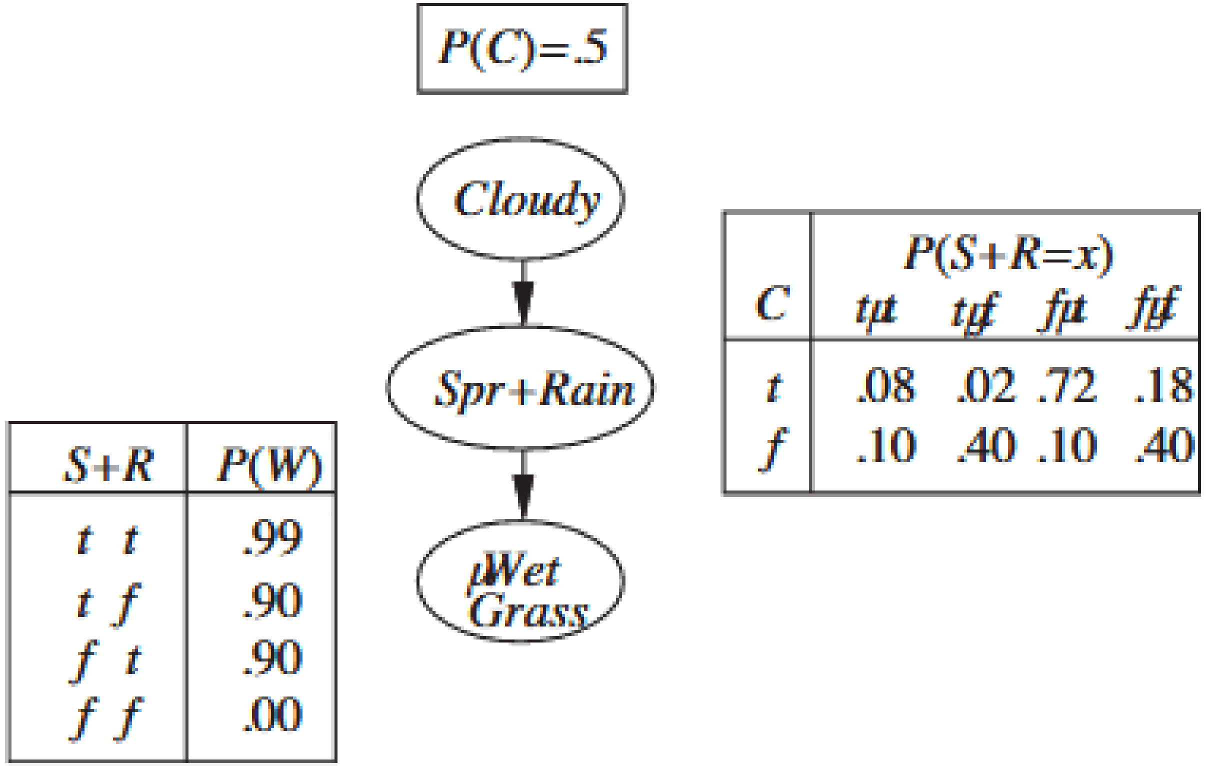
Join to become polytree



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)			
	tµt	tµf	fµt	fµf
t	.08	.02	.72	.18
f	.10	.40	.10	.40

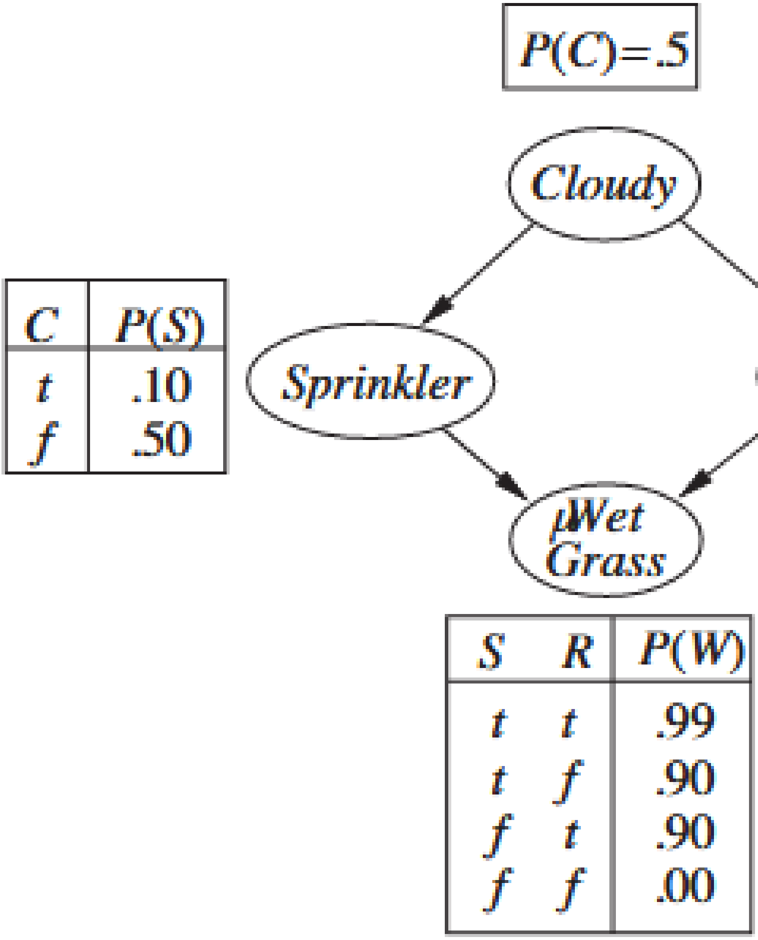
S+R	P(W)
t t	.99
t f	.90
f t	.90
f f	.00

Approximate inference in Bayesian network

- Exact inference not feasible in large, multiply connected network

Direct sampling

- Prior sampling – random sampling
- sample each variable in random order

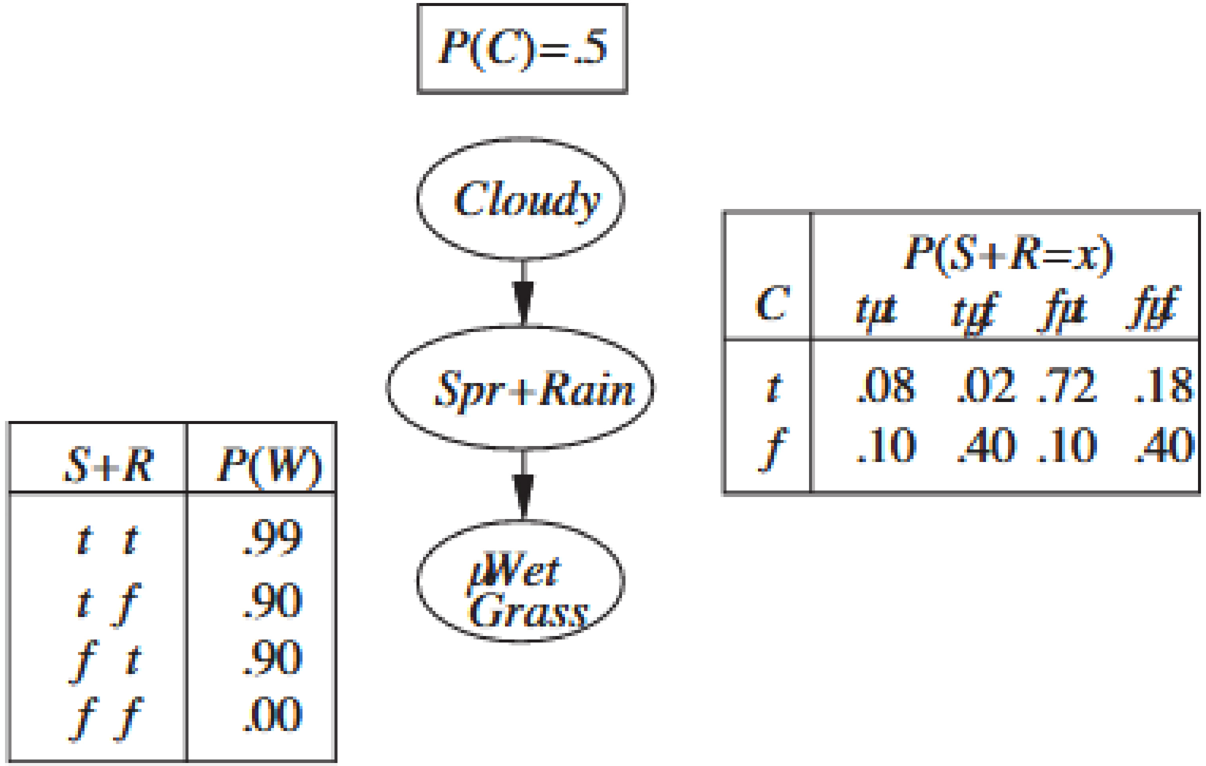


P(C)=.5

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



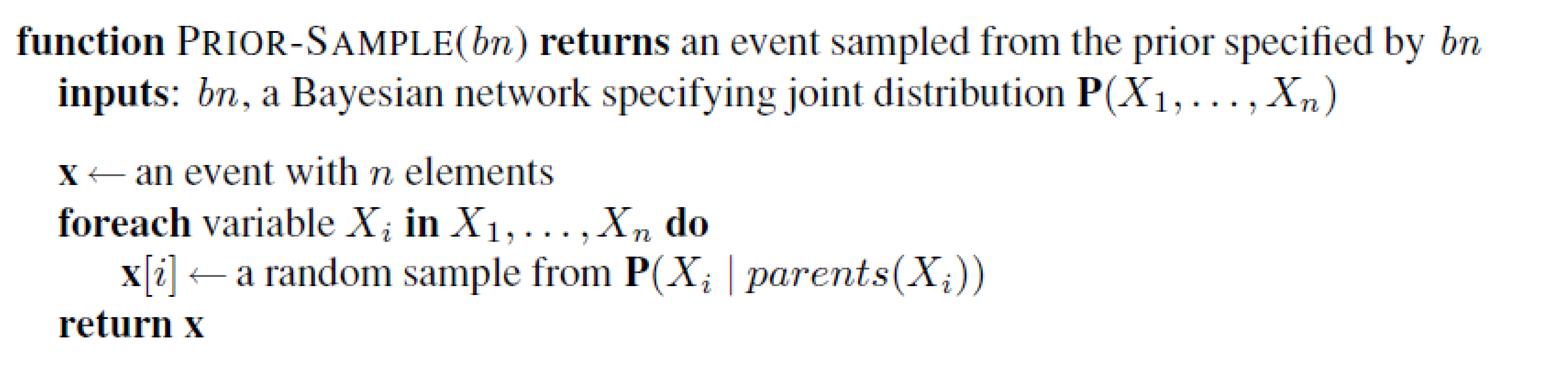
P(C)=.5

C	P(S+R=x)			
	tµt	tµf	fµt	fµf
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

Sample each variable in random order

Sample	Sample from P(Cloudy) = 〈0.5, 0.5〉 , value is true.
Sample	Sample from P(Sprinkler | Cloudy = true) = 〈0.1, 0.9〉 , value is false.
Sample	Sample from P(Rain | Cloudy = true) = 〈0.8, 0.2〉 , value is true.
Sample	Sample from P(WetGrass | Sprinkler = false, Rain = true) = 〈0.9, 0.1〉 , value is true.



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

Probability of specific values

- C S R W
- T F T T
- = 0.5 * 0.9 * 0.8 * 0.9
- = 0.324 (32%)
- SPS specific probability of event
- In any sampling algorithm, the answers are computed by counting the actual samples generated.

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

Probability of specific values

- Suppose there are N total samples, and let NPS (x1, . . . , xn) be the number of times the specific event x1, . . . , xn occurs in the set of samples. We expect this number, as a fraction of the total

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

Rejection sampling in Bayesian networks

- **Rejection sampling** is a general method for producing samples from a hard-to-sample distribution given an easy-to-sample distribution
- It can be used to compute conditional probability

Rejection sampling in Bayesian networks

- First, it generates samples from the prior distribution specified by the network
- Next it rejects all those that do not match the evidence
- Finally estimate conditional probability how often the variable occurs in the remaining samples

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

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

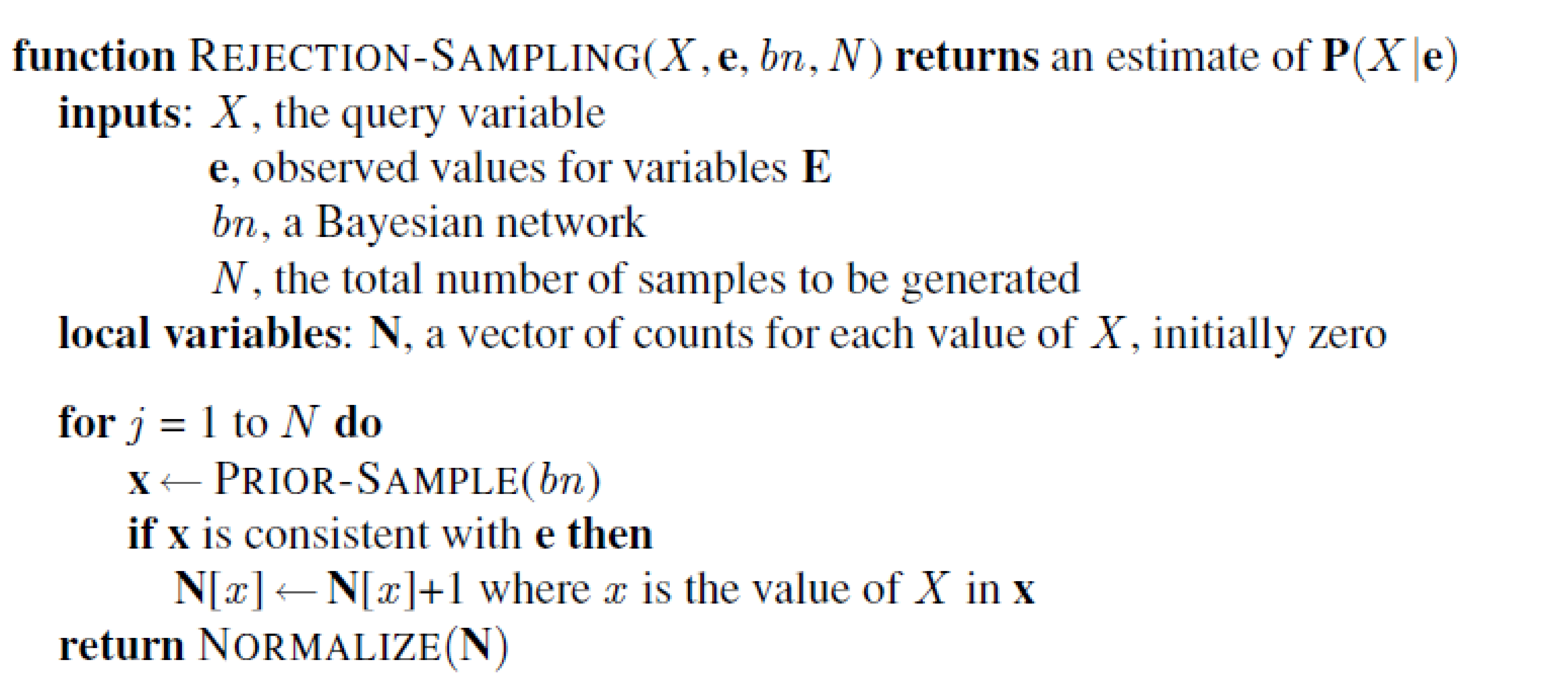
Rejection sampling in Bayesian networks

- we wish to 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.
- Hence

P(Rain | Sprinkler =true) ≈ NORMALIZE(8, 19) = 0.296, 0.704

The true answer is <0.3, 0.7>

As more samples are collected it will converge to true answer



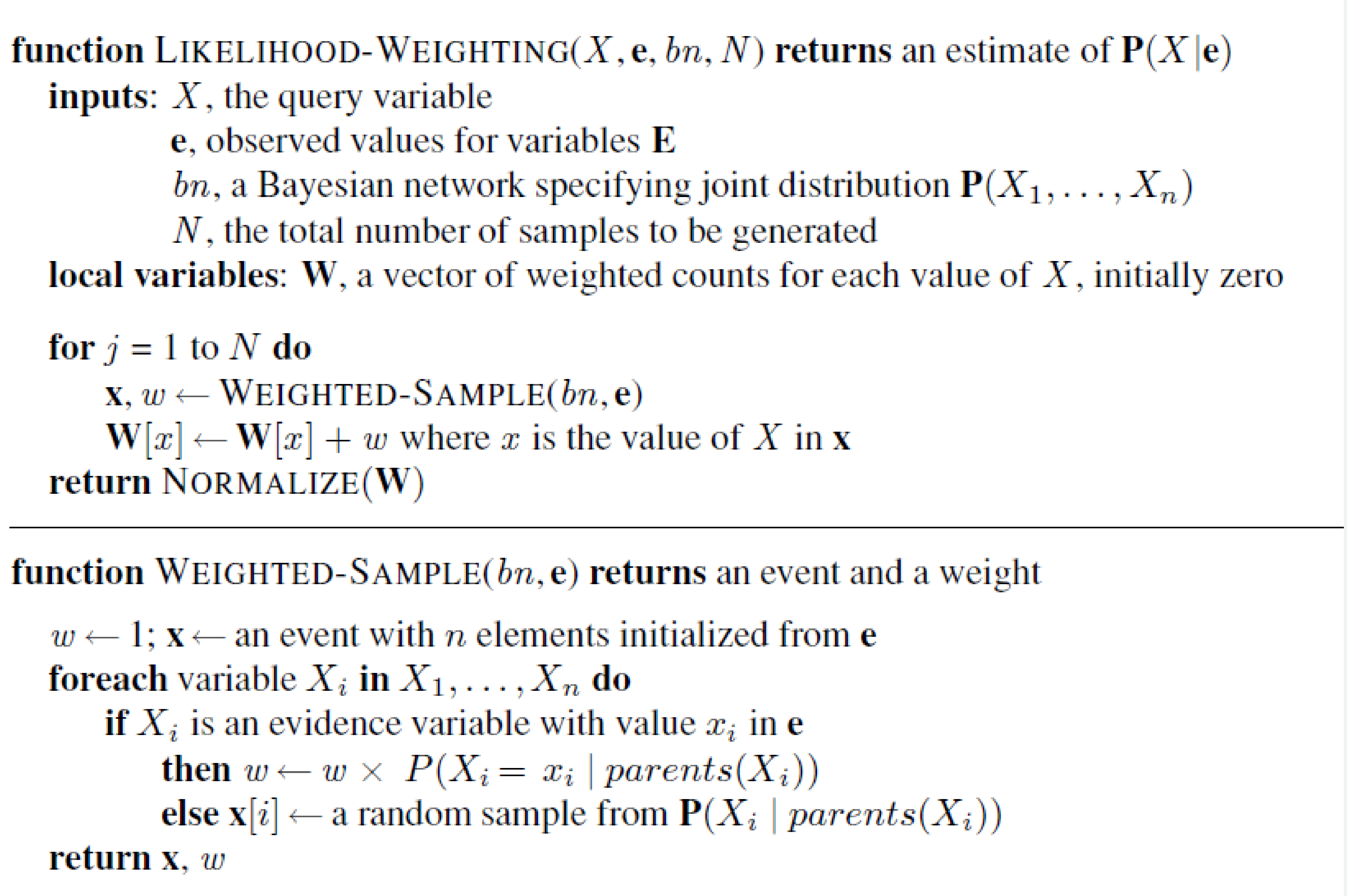
```
function REJECTION-SAMPLING($X$, $\mathbf{e}$, $bn$, $N$) returns an estimate of $\mathbf{P}(X|\mathbf{e})$
  inputs: $X$, the query variable
          $\mathbf{e}$, observed values for variables $\mathbf{E}$
          $bn$, a Bayesian network
          $N$, the total number of samples to be generated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

Drawback of rejection sampling

- It rejects more samples
- Unusable for complex problem – with more evidence variable fraction consistent with evidence drops exponentially

Likelihood weighting

- Likelihood weighting avoids the inefficiency of rejection sampling by generating 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.



```
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

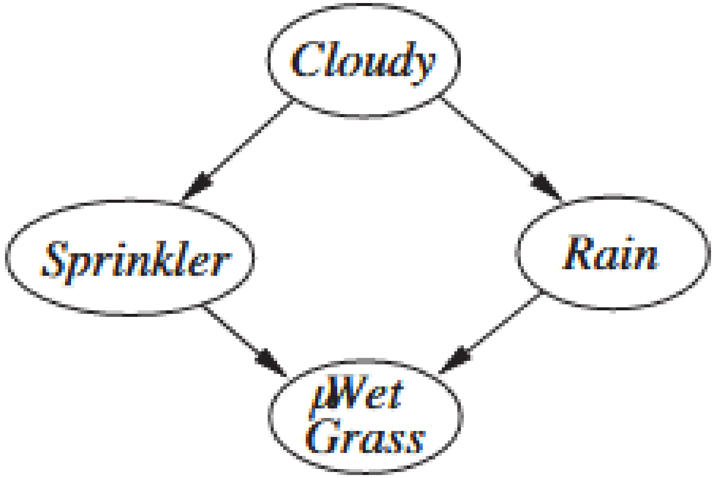
- Likelihood weighting avoids the inefficiency of rejection sampling by generating only events that are consistent with the evidence e.
- It is a particular instance of the general statistical technique of importance sampling, tailored for inference in Bayesian networks.
- Fix the evidence variable
- Sample non evidence variable
- Not all events are equal each event are given weightage by likelihood

Likelihood weighting

query

- P(Rain |Cloudy =true, WetGrass =true) and the ordering Cloudy, Sprinkler, Rain, Wet-Grass. (Any topological ordering will do.)
- The process is → First, the weight w is set to 1.0.
- Then an event is generated:

P(C)=.5

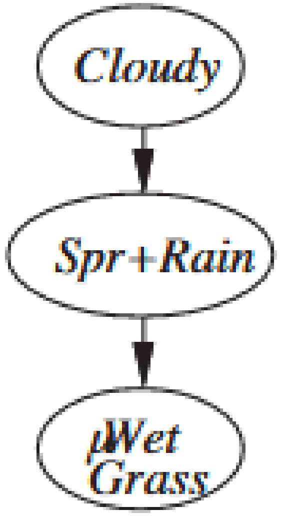


C	P(S)
t	.10
f	.50

C	P(R)
t	.80
f	.20

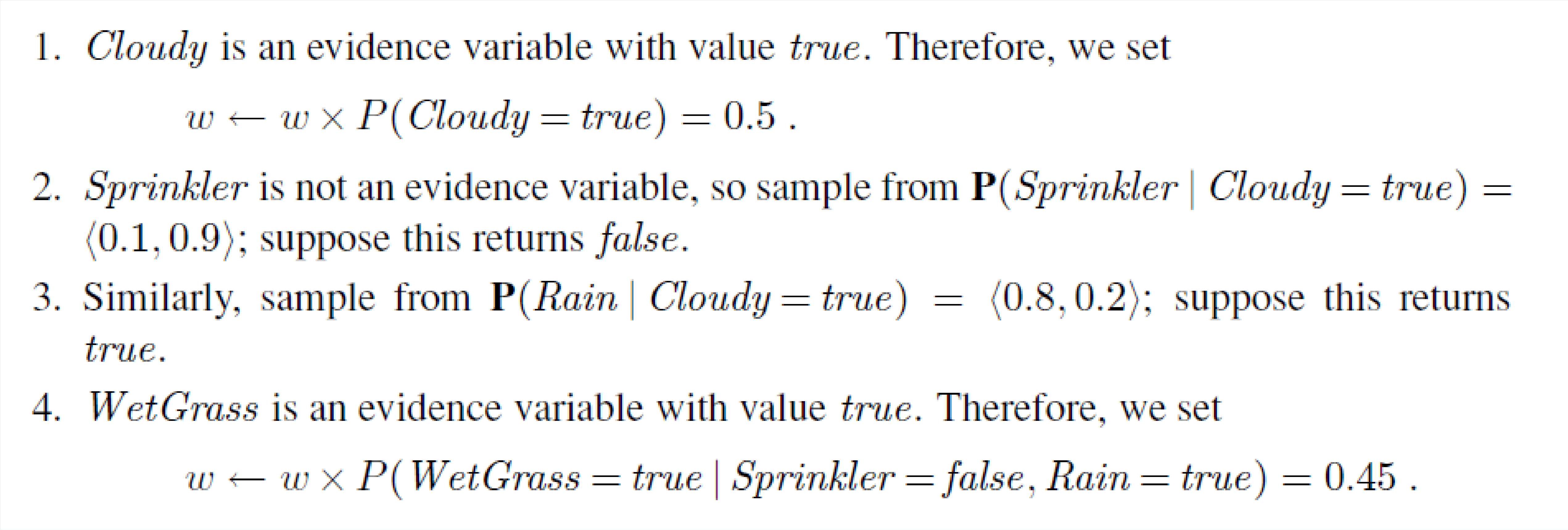
S	R	P(W)
t	t	.99
t	f	.90
f	t	.90
f	f	.00

P(C)=.5



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

C	P(S+R=x)			
	tµt	tµf	fµt	fµf
t	.08	.02	.72	.18
f	.10	.40	.10	.40



1. *Cloudy* is an evidence variable with value *true*. Therefore, we set

$$w \leftarrow w \times P(\mathit{Cloudy} = \mathit{true}) = 0.5\ .$$

2. *Sprinkler* is not an evidence variable, so sample from $\mathbf{P}(\mathit{Sprinkler} \mid \mathit{Cloudy} = \mathit{true}) = \langle 0.1, 0.9 \rangle$; suppose this returns *false*.
3. Similarly, sample from $\mathbf{P}(\mathit{Rain} \mid \mathit{Cloudy} = \mathit{true}) = \langle 0.8, 0.2 \rangle$; suppose this returns *true*.
4. *WetGrass* is an evidence variable with value *true*. Therefore, we set

$$w \leftarrow w \times P(\mathit{WetGrass} = \mathit{true} \mid \mathit{Sprinkler} = \mathit{false}, \mathit{Rain} = \mathit{true}) = 0.45\ .$$

Inference by Markov chain simulation

- Instead of generating each sample from scratch, **Markov chain Monte Carlo** algorithms generate each sample by making a random change to the preceding sample
- From current change make changes and create next state

C	S	R	W
T	F	T	T
F	F	T	T

- Particular form of markov chain is Gibbs sampling

Inference by Markov chain simulation

- In statistics and machine learning, when one wants to infer a random variable with a set of variables, usually a subset is enough, and other variables are useless. Such a subset that contains all the useful information is called a Markov blanket.

Gibbs sampling in Bayesian networks

- **P**(Rain | Sprinkler =true, WetGrass =true)
- Evidence : sprinkler and wet grass are fixed to observed values
- Non evidence : rain & cloudy are initialized randomly
- Initial state [c s r w] → [t t f t]
- Non evidence variable are sampled repeatedly
- Cloudy is sampled, given the current values of its Markov blanket variables
- P(cloudy = False) → [f t f t] (new state)
- Rain sampled → if it yields rain = true then [f t t t] (new state)

Gibbs sampling in Bayesian networks

- If the process visits 20 states where Rain is true and 60 states where Rain is false, then the answer to the query is NORMALIZE(20, 60) = 0.25, 0.75.