

CS 561: Artificial Intelligence

Instructor: Sofus A. Macskassy, macskass@usc.edu

TAs: Nadeesha Ranashinghe (nadeeshr@usc.edu)

William Yeoh (wyeoh@usc.edu)

Harris Chiu (chiciu@usc.edu)

Lectures: MW 5:00-6:20pm, OHE 122 / DEN

Office hours: By appointment

Class page: <http://www-rcf.usc.edu/~macskass/CS561-Spring2010/>

This class will use <http://www.uscden.net/> and class webpage

- Up to date information
- Lecture notes
- Relevant dates, links, etc.

Course material:

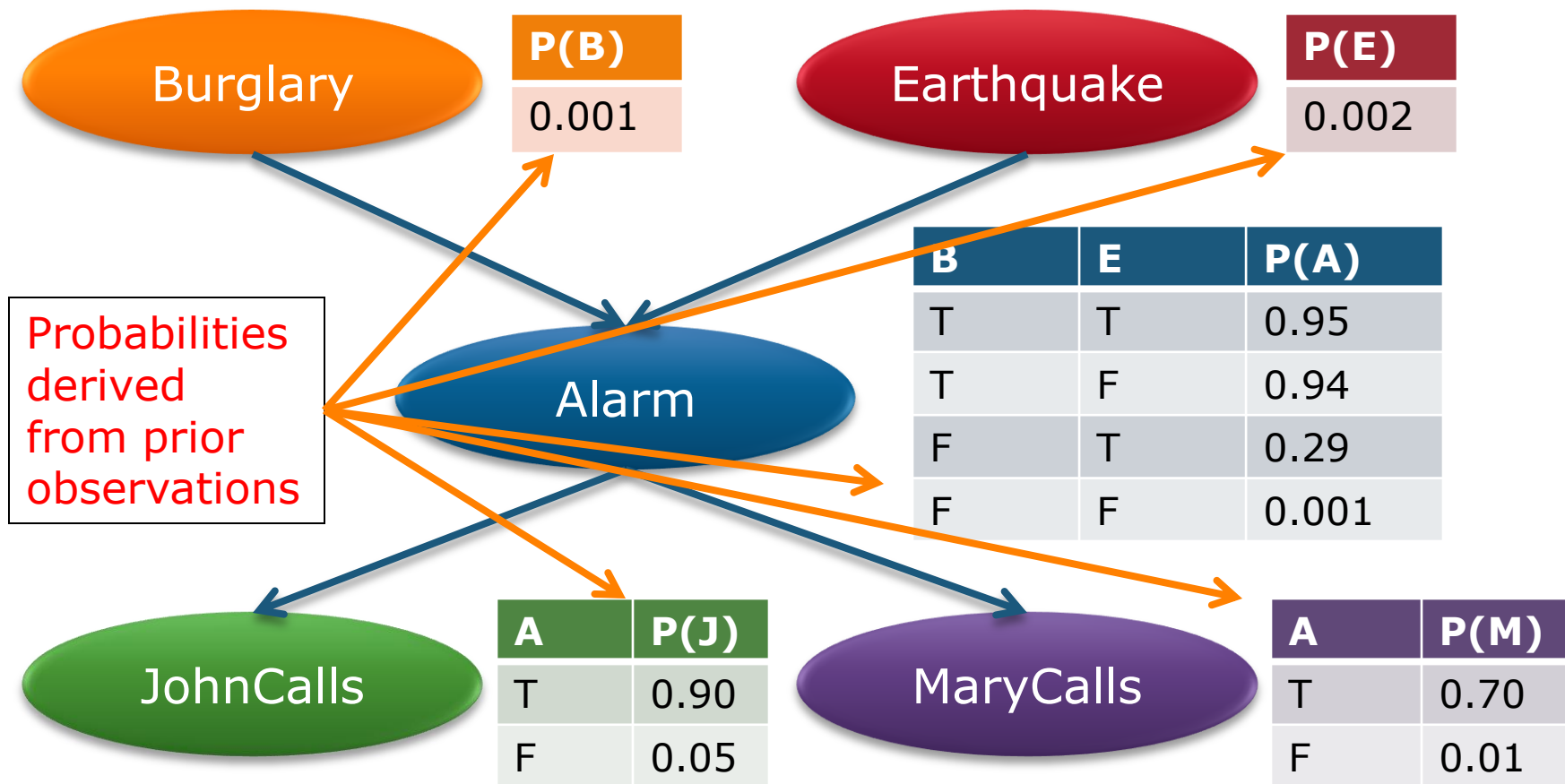
[AIMA] Artificial Intelligence: A Modern Approach,
by Stuart Russell and Peter Norvig. (2nd ed)

Midterm 2

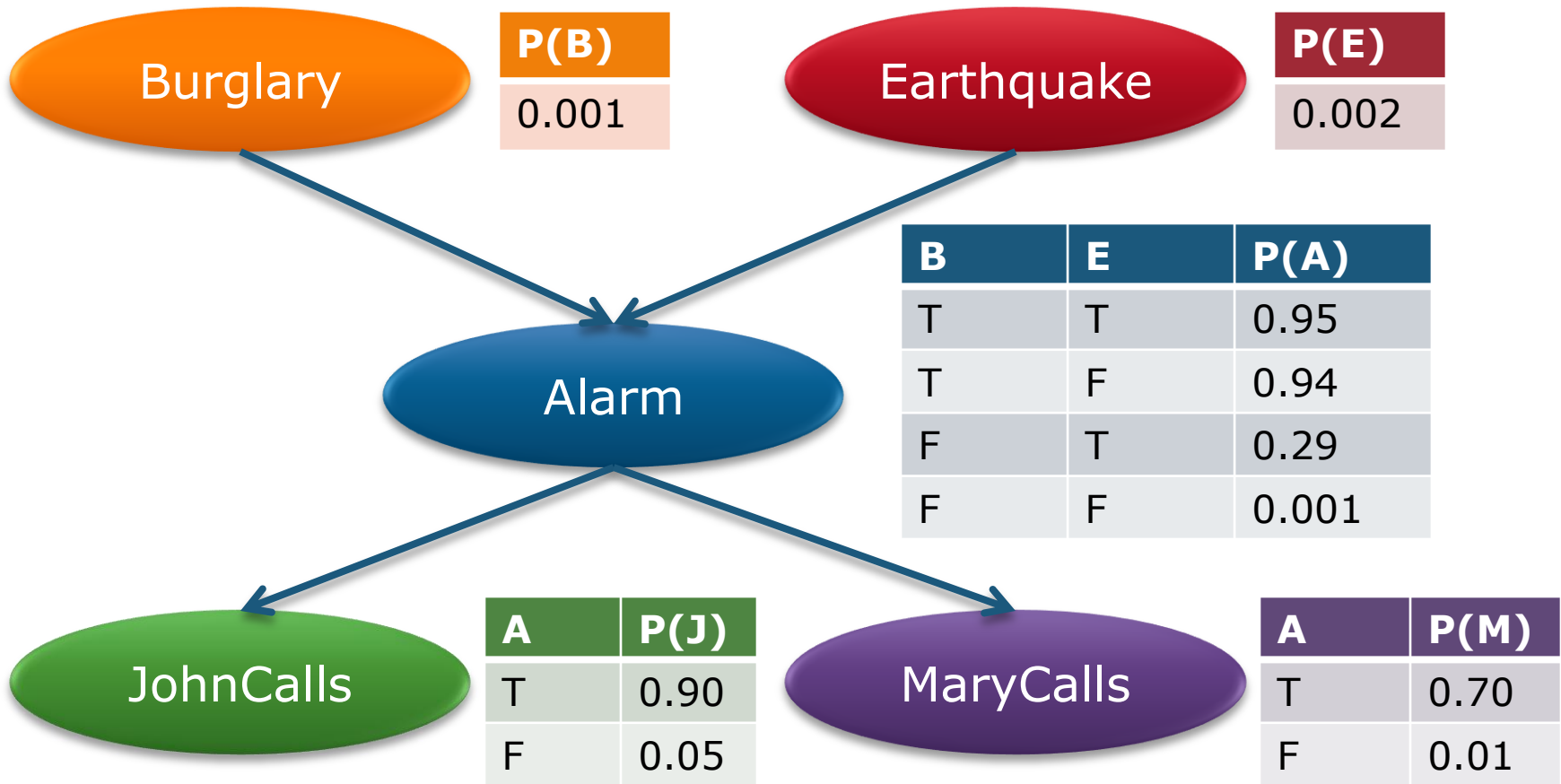
- Midterm 2 is next week (April 12)
- Location: SGM 101
- Time: 5pm-6:20pm
- Covers:
 - Inference in first-order logic
 - Knowledge representation
 - Planning
 - Uncertainty
 - Probabilistic reasoning, inference, reasoning over time
- Is open-book and open-notes

Probabilistic Reasoning [Ch. 14]

- Bayes Networks – Part 1
 - Syntax
 - Semantics
 - Parameterized distributions
- Bayes Networks – Part2
 - Exact inference by enumeration
 - Exact inference by variable elimination
 - Approximate inference by stochastic simulation
 - Approximate inference by Markov chain Monte Carlo

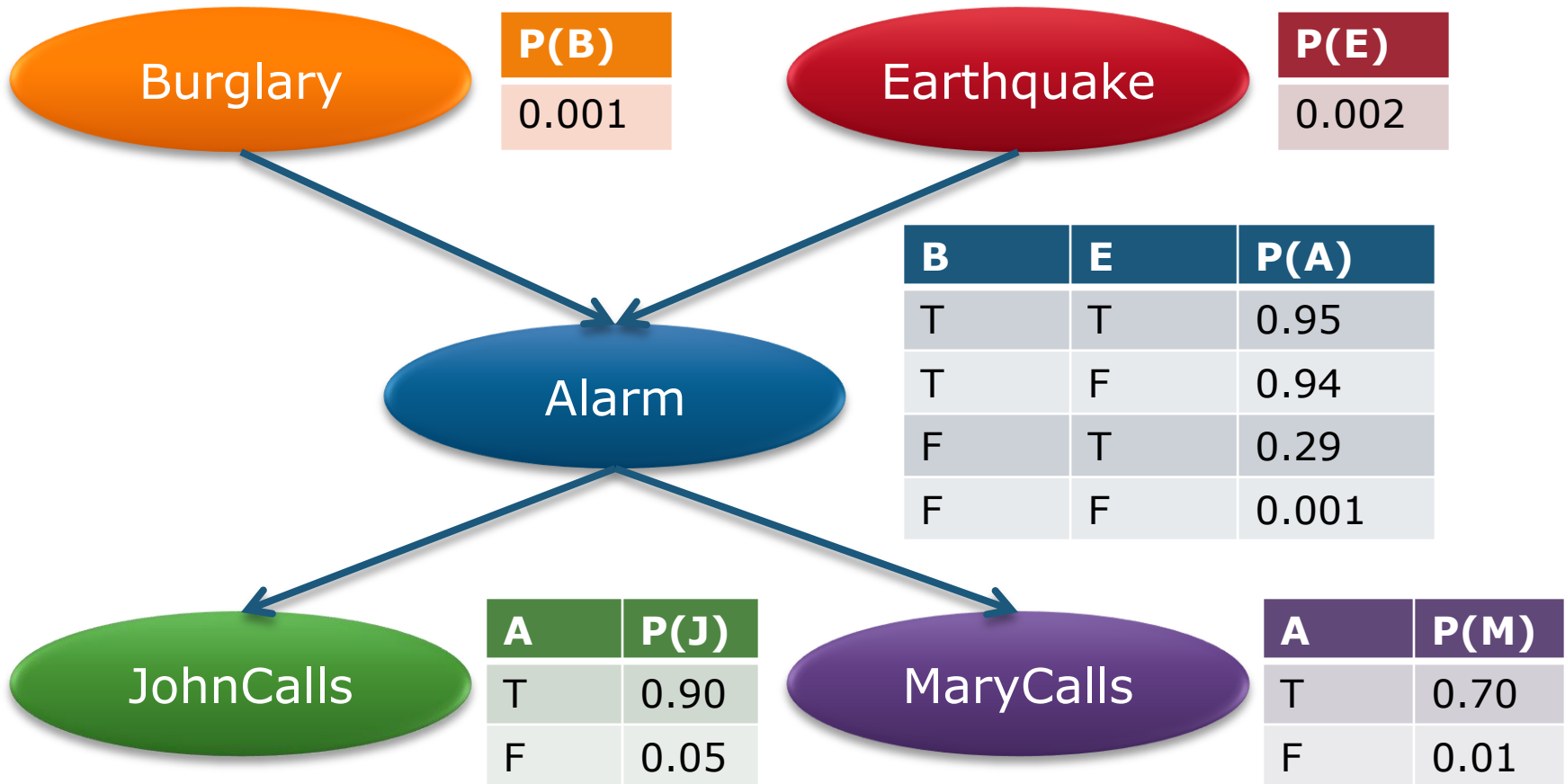


Alarm example revisited



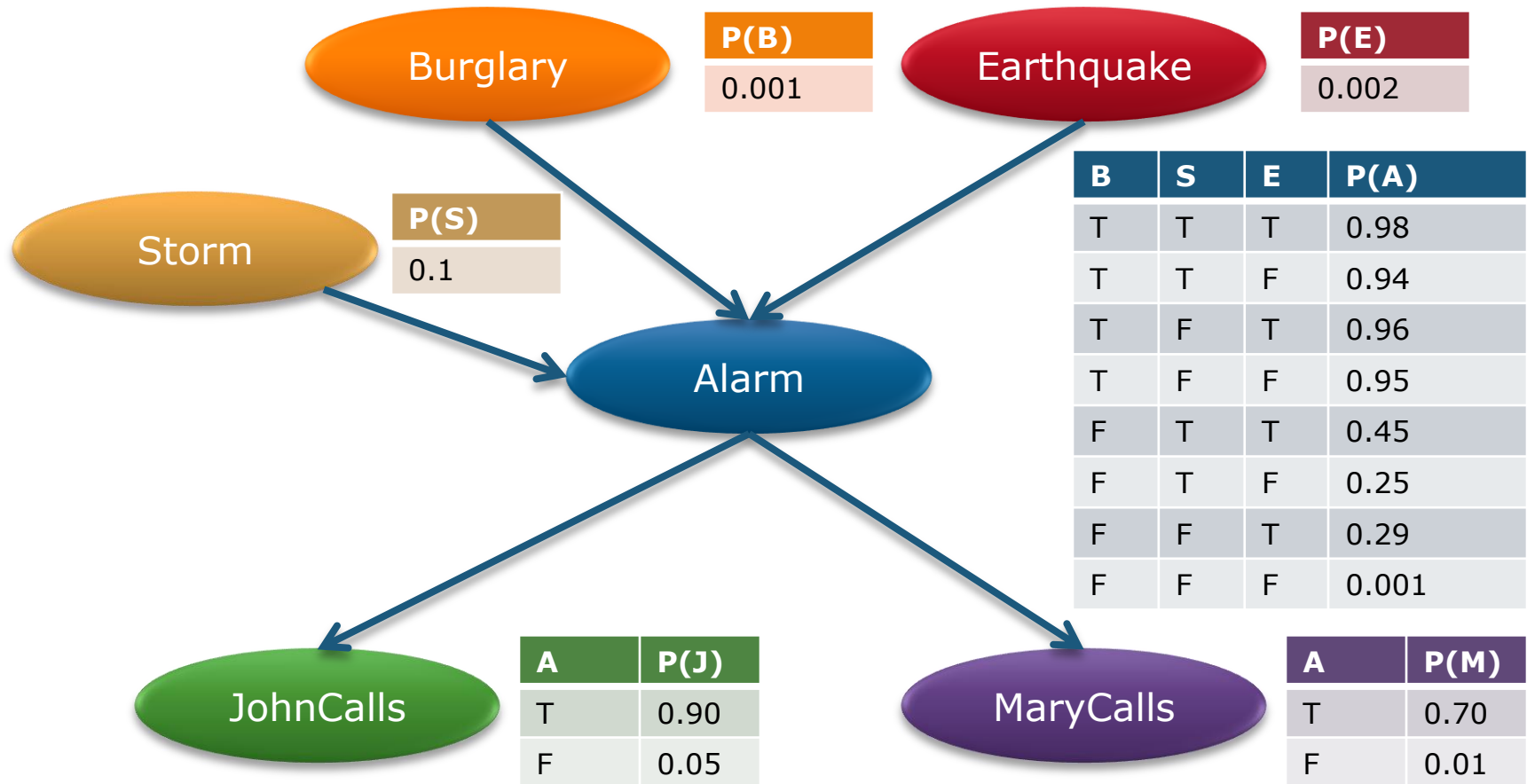
$$P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$$

What is the probability that the alarm has sounded but neither a burglary nor earthquake has occurred and both John and Mary call?



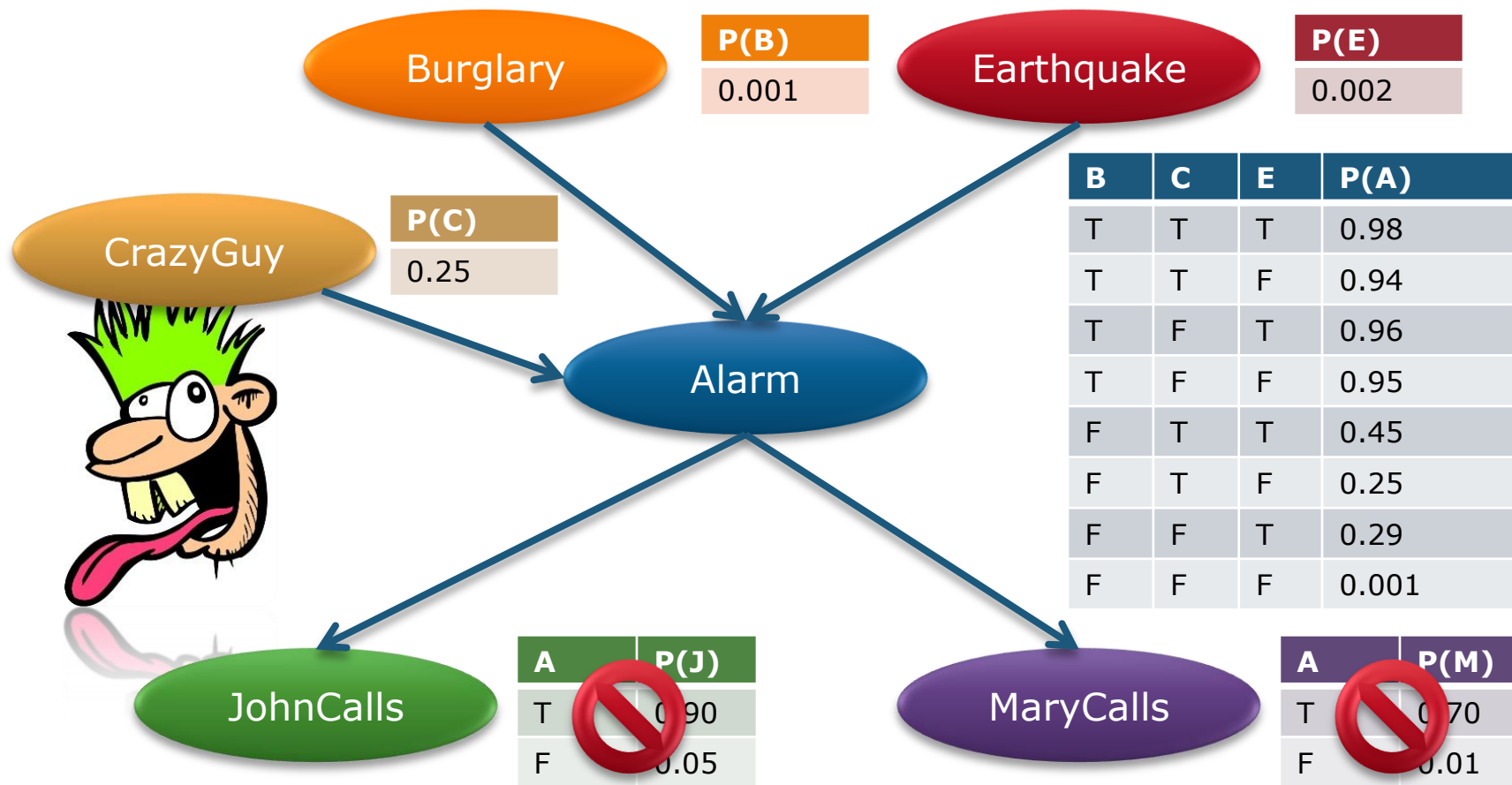
$$\begin{aligned}
 P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) &= P(j|a)P(m|a)P(a|\neg b \wedge \neg e)P(\neg b)P(\neg e) \\
 &= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 \\
 &= 0.00062
 \end{aligned}$$

We find that storms can also set off alarms. We add that into our CPT. Notice that JohnCalls and MaryCalls stay the same since Storms were always there but were just unaccounted for. John and Mary did not change! However, we have better precision at $P(A)$.



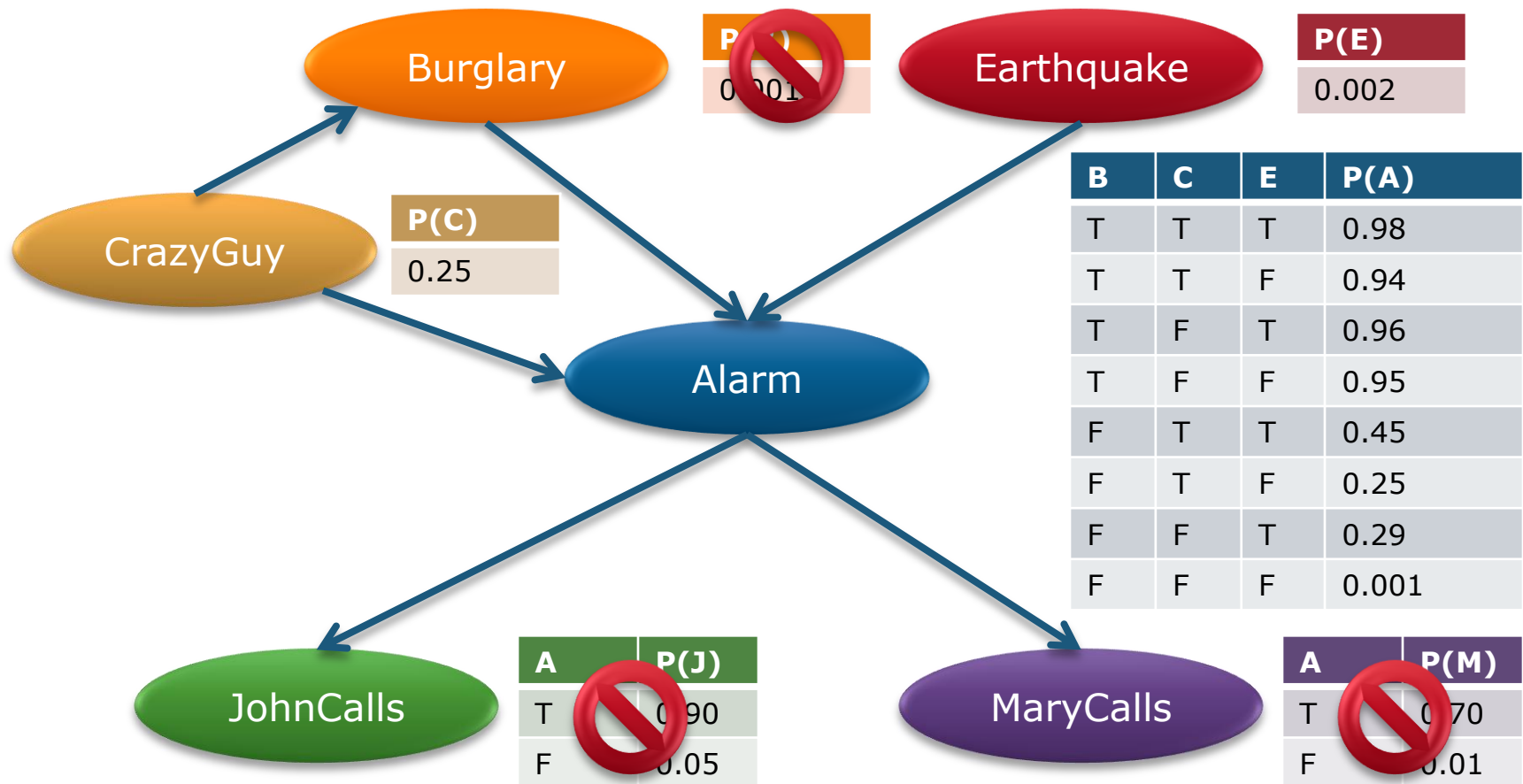
What if we find a new variable?

What if we inject a new cause that was not there before. We pay a crazy guy to set off the alarm frequently, JohnCalls and MaryCalls may no longer be valid since we may have changed the behaviors. For instance the alarm goes off so often now that John and Mary are more likely to ignore it.



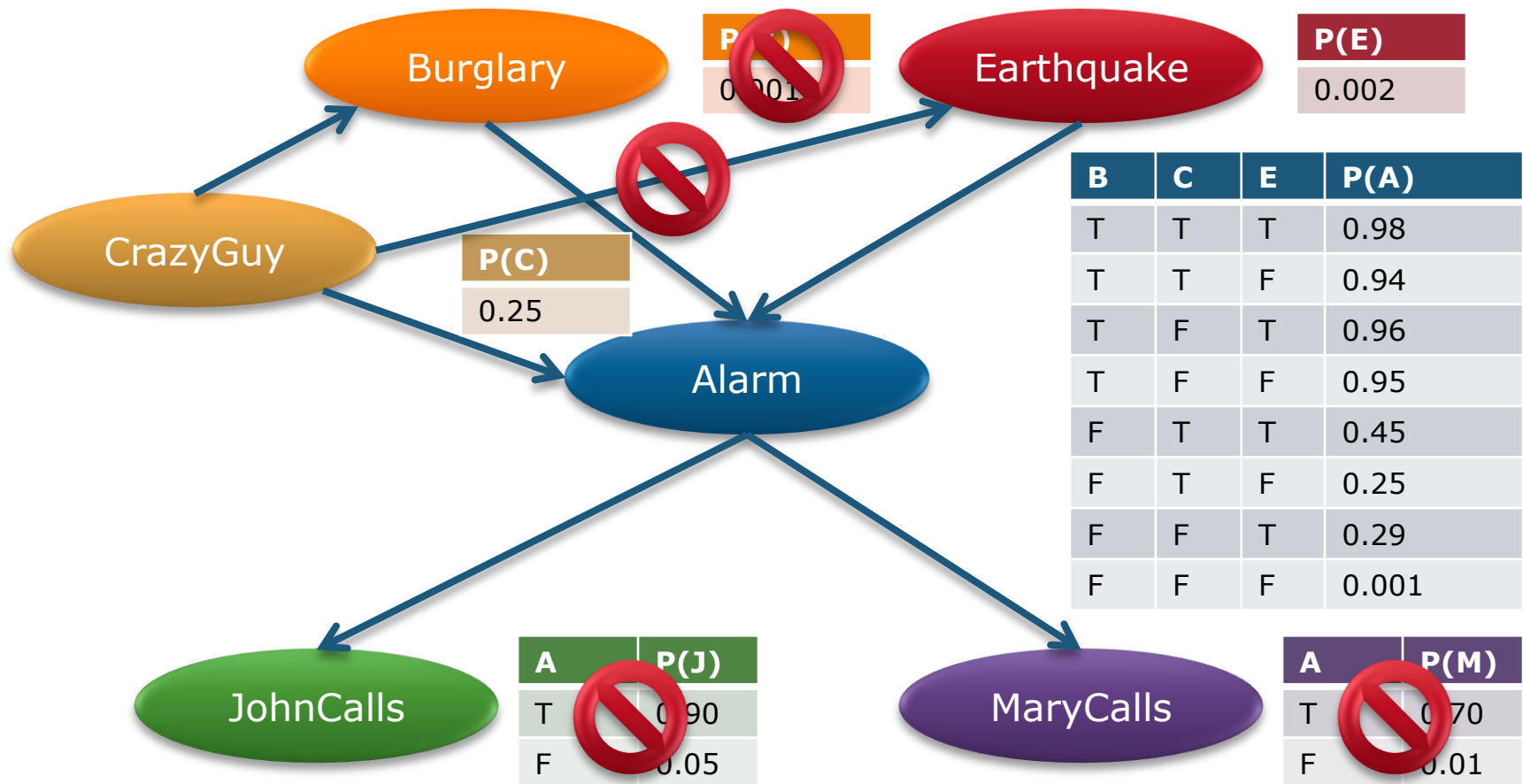
What if we cause a new variable? (or a new variable just occurs)

If the introduced variable is highly erratic, it can invalidate even more of the CPT than we would like.



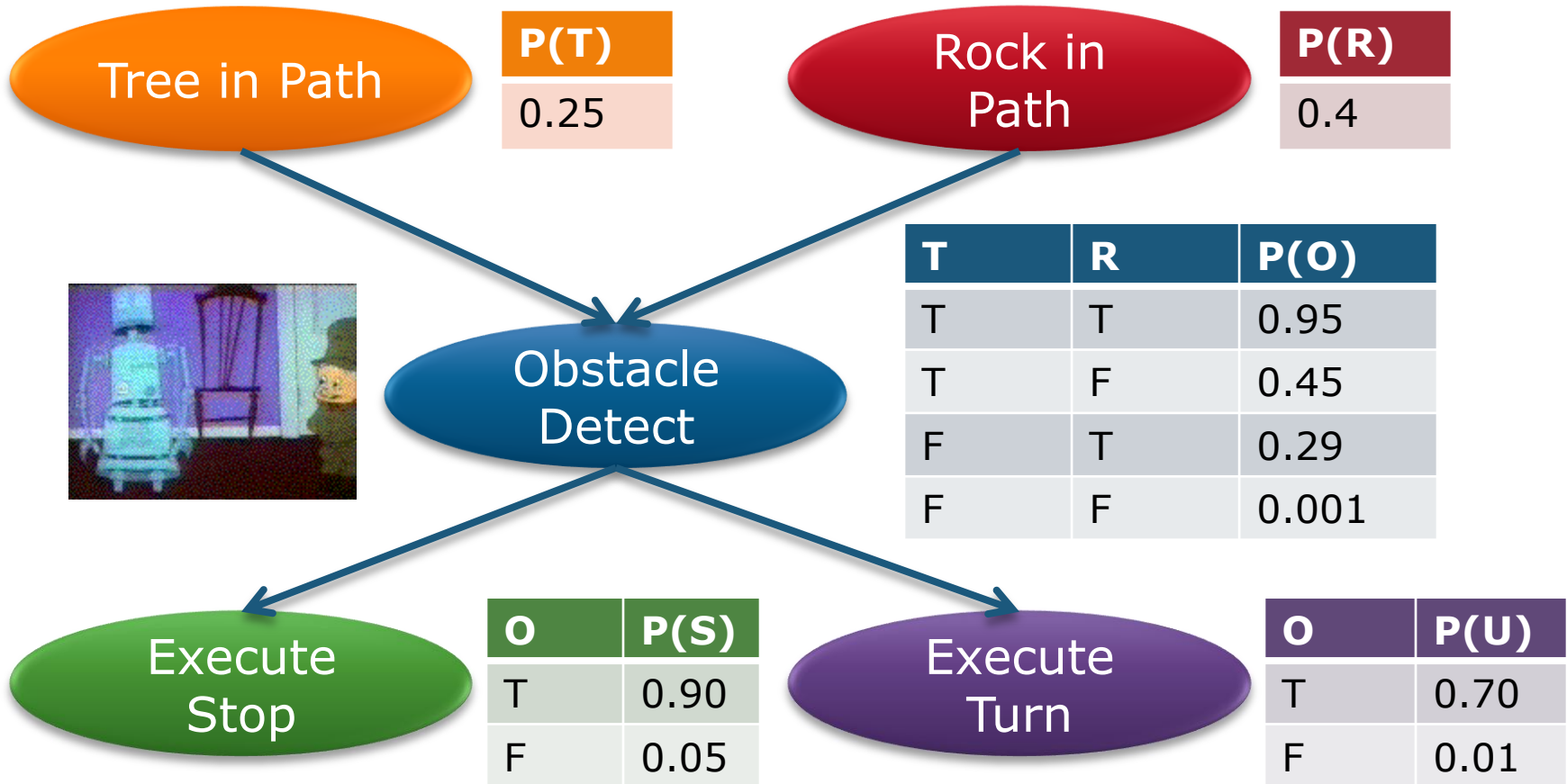
What if we cause a new variable?

However some changes to the CPT may be absurd so we may never have to worry about them.



What if we cause a new variable?

How would we apply this to Robotics?



The CPT can describe other events and probabilities such as action success given observations.

Inference tasks

- Simple queries: compute posterior marginal $P(X_i|E=e)$
e.g., $P(\text{NoGas}|\text{Gauge=empty}, \text{Lights=on}, \text{Starts=false})$
- Conjunctive queries: $P(X_i, X_j|E=e) = P(X_i|E=e)P(X_j|X_i, E=e)$
- Optimal decisions: decision networks include utility information; probabilistic inference required for $P(\text{outcome}|\text{action}, \text{evidence})$
- 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?

Inference by enumeration

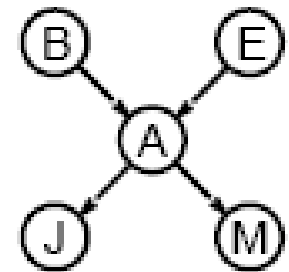
- Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation
- Simple query on the burglary network:

$$P(B | j, m)$$

$$= P(B, j, m) / P(j, m)$$

$$= \alpha P(B, j, m)$$

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



- Rewrite full joint entries using product of CPT entries:

$$P(B | j, m)$$

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

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

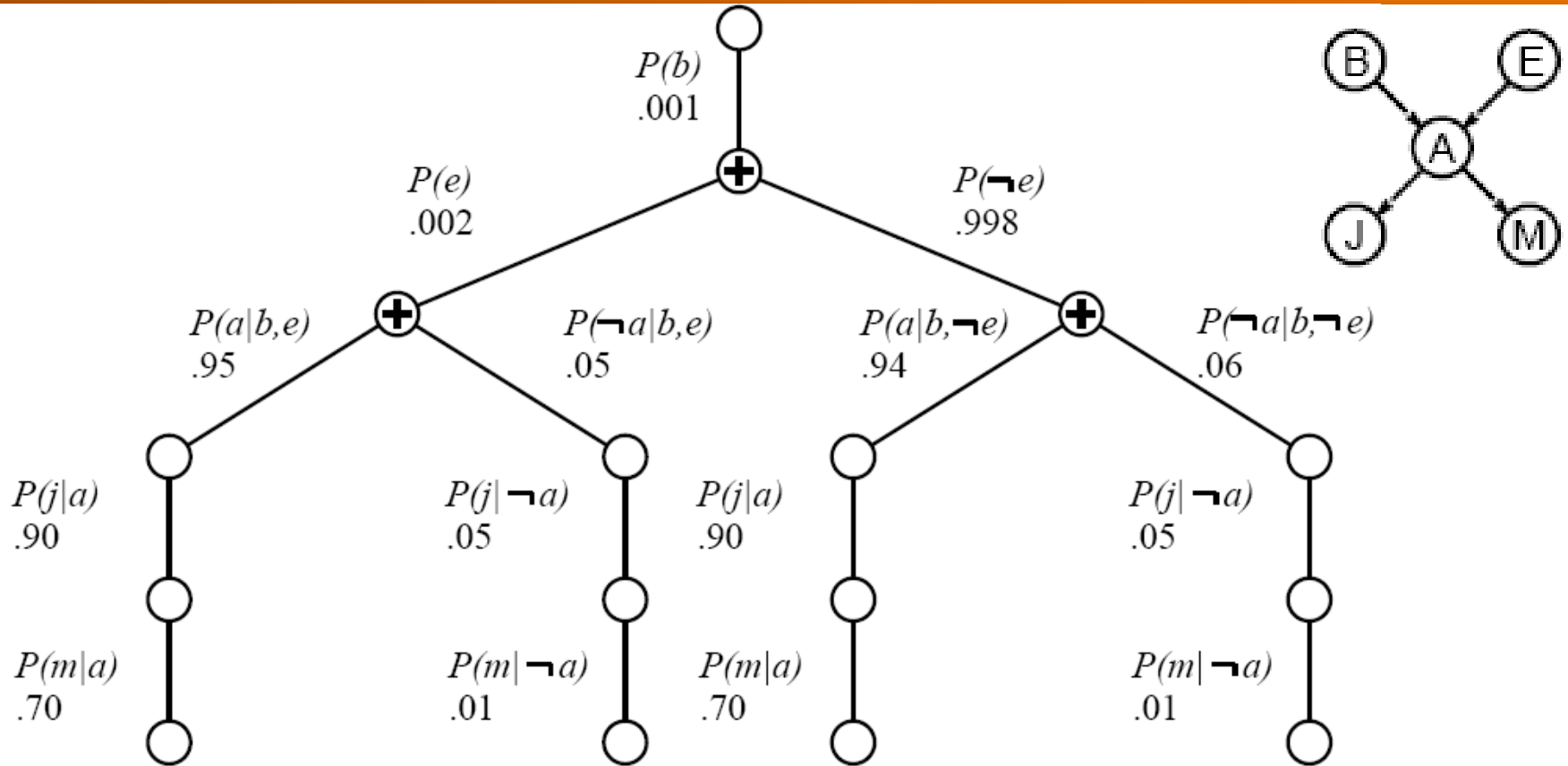
- Recursive depth-first enumeration: $O(n)$ space, $O(d^n)$ time

Enumeration algorithm

```
function ENUMERATION-ASK( $X$ ,  $e$ ,  $bn$ ) returns a distribution over  $X$ 
  inputs:  $X$ , the query variable
          $e$ , observed values for variables  $E$ 
          $bn$ , a Bayesian network with variables  $\{X\} \cup E \cup Y$  /*  $Y$ =hidden vars */
   $Q(X) \leftarrow$  a distribution over  $X$ , initially empty
  for each value  $x_i$  of  $X$  do
    extend  $e$  with value  $x_i$  for  $X$ 
     $Q(x_i) \leftarrow$  ENUMERATE-ALL(Vars[ $bn$ ],  $e$ )
  return NORMALIZE( $Q(X)$ )
```

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

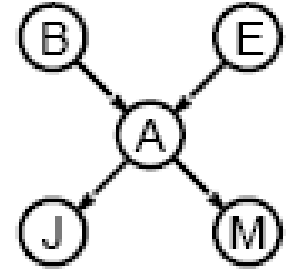
Evaluation tree



- 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



$$\begin{aligned}
 &P(B \mid j, m) \\
 &= \alpha \underbrace{P(B)}_B \sum_e \underbrace{P(e)}_E \sum_a \underbrace{P(a \mid B, e)}_A \underbrace{P(j \mid a)}_J \underbrace{P(m \mid a)}_M \\
 &= \alpha P(B) \sum_e P(e) \sum_a P(a \mid B, e) P(j \mid a) f_M(a) \\
 &= \alpha P(B) \sum_e P(e) \sum_a P(a \mid B, e) f_J(a) f_M(a) \\
 &= \alpha P(B) \sum_e P(e) \sum_a f_A(a, b, e) f_J(a) f_M(a) \\
 &= \alpha P(B) \sum_e P(e) f_{\overline{AJM}}(b, e) \quad (\text{sum out } A) \\
 &= \alpha P(B) f_{\overline{E AJM}}(b) \quad (\text{sum out } E) \\
 &= \alpha f_B(b) \times f_{\overline{E AJM}}(b)
 \end{aligned}$$

$$f_M(a) = \begin{pmatrix} P(m \mid a) \\ P(m \mid \neg a) \end{pmatrix}$$

$$\begin{aligned}
 &f_{\overline{AJM}}(b, e) \\
 &= \sum_a f_A(a, b, e) \times f_J(a) \times f_M(a) \\
 &= f_A(a, b, e) \times f_J(a) \times f_M(a) \\
 &\quad + f_A(\neg a, b, e) \times f_J(\neg a) \times f_M(\neg a)
 \end{aligned}$$

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

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

- Assuming f_1, \dots, f_i do not depend on X
- **Pointwise product** of factors f_1 and f_2 :

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

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

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

Pointwise Product: example

A	B	$f_1(A,B)$	B	C	$f_2(B,C)$	A	B	C	$f_3(A,B,C)$
T	T	.3	T	T	.2	T	T	T	.3 x .2
T	F	.7	T	F	.8	T	T	F	.3 x .8
F	T	.9	F	T	.6	T	F	T	.7 x .6
F	F	.1	F	F	.4	T	F	F	.7 x .4
						F	T	T	.9 x .2
						F	T	F	.9 x .8
						F	F	T	.1 x .6
						F	F	F	.1 x .4

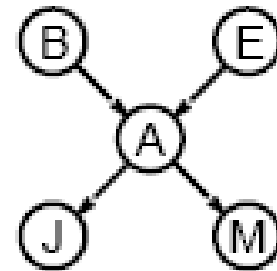
Variable elimination algorithm

```
function ELIMINATION-ASK( $X$ ,  $e$ ,  $bn$ ) returns a distribution over  $X$ 
  inputs:  $X$ , the query variable
          $e$ , evidence specified as an event
          $bn$ , a belief network specifying joint distribution  $P(X_1, \dots, X_n)$ 
  factors  $\leftarrow [ ]$ ;
  vars  $\leftarrow$  REVERSE(VARS[ $bn$ ])
  for each var in vars do
    factors  $\leftarrow$  [MAKE-FACTOR(var;  $e$ )|factors]
  if var is a hidden variable then factors SUM-OUT(var, factors)
  return NORMALIZE(POINTWISE-PRODUCT(factors))
```

Irrelevant variables

- Consider the query $P(\text{JohnCalls} | \text{Burglary} = \text{true})$

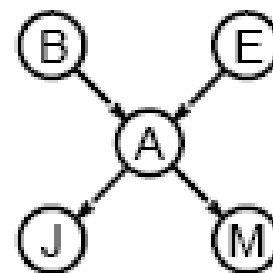
$$P(J | b) = \alpha P(b) \sum_e P(e) \sum_a P(a | b, e) P(J | a) \sum_m P(m | a)$$



- Sum over m is identically 1; M is **irrelevant** to the query
- Thm 1: Y is irrelevant unless $Y \in \text{Ancestors}(\{X\} \cup \mathbf{E})$
- Here, $X = \text{JohnCalls}$, $\mathbf{E} = \{\text{Burglary}\}$, and $\text{Ancestors}(\{X\} \cup \mathbf{E}) = \{\text{Alarm}, \text{Earthquake}\}$ so MaryCalls is irrelevant
- (Compare this to backward chaining from the query in Horn clause KBs)

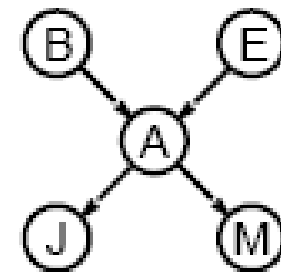
Irrelevant variables contd.

- Defn: moral graph of Bayes net: marry all parents and drop arrows
- Defn: **A** is m-separated from **B** by **C** iff separated by **C** in the moral graph
- Thm 2: **Y** is irrelevant if m-separated from **X** by **E**
- For $P(\text{JohnCalls}|\text{Alarm}=\text{true})$, both **Burglary** and **Earthquake** are irrelevant

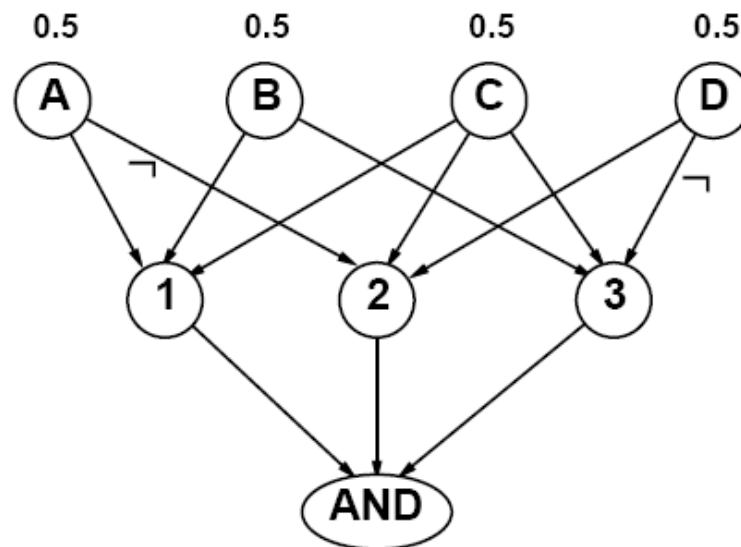


Complexity of exact inference

- **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



- **1. $A \vee B \vee C$**
- **2. $C \vee D \vee \neg A$**
- **3. $B \vee C \vee \neg D$**



Inference by stochastic simulation

- Basic idea:

- 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



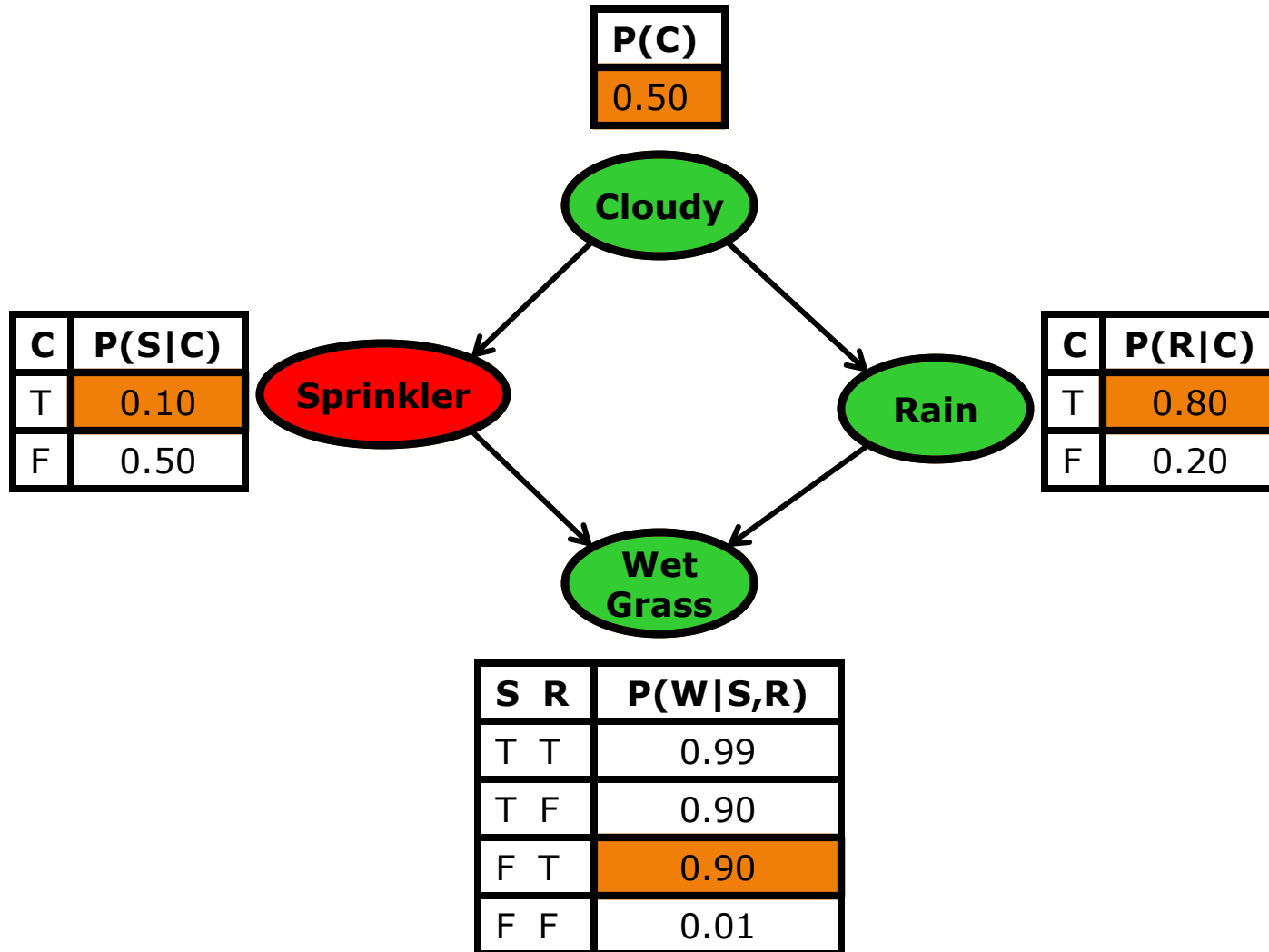
- Outline:

- Sampling from an empty network
- Rejection sampling: reject samples disagreeing with evidence
- Likelihood weighting: use evidence to weight samples
- Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior

Sampling from an empty network

```
function PRIOR-SAMPLE(bn) returns an event sampled from bn
inputs: bn, a belief network specifying
        joint distribution  $P(X_1, \dots, X_n)$ 
 $x \leftarrow$  an event with n elements
for i = 1 to n do
     $x_i \leftarrow$  a random sample from  $P(X_i \mid \text{parents}(X_i))$ 
        given the values of PARENTS( $X_i$ ) in  $x$ 
return  $x$ 
```

Example



Sampling from an empty network contd.

Probability that PriorSample generates a particular event

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

i.e., the true prior probability

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

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

Then we have

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

That is, estimates derived from PriorSample are **consistent**

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

Rejection sampling

- $P(X|e)$ estimated from samples agreeing with 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
     $x \leftarrow$  PRIOR-SAMPLE( $bn$ )
    if  $x$  is consistent with  $e$  then
         $N[x] \leftarrow N[x]+1$  where  $x$  is the value of  $X$  in  $x$ 
return NORMALIZE( $N[X]$ )
```

- E.g., estimate $P(\text{Rain}|\text{Sprinkler}=\text{true})$ using 100 samples
 - 27 samples have $\text{Sprinkler}=\text{true}$
 - Of these, 8 have $\text{Rain}=\text{true}$ and 19 have $\text{Rain}=\text{false}$
- $P(\text{Rain}|\text{Sprinkler}=\text{true}) = \text{Normalize}(8, 19) = \langle 0.296, 0.704 \rangle$
- Similar to basic real-world empirical estimation procedure

Analysis of rejection sampling

$$\begin{aligned} P(X|e) &= \alpha N_{PS}(X, e) && \text{(algorithm defn.)} \\ &= N_{PS}(X; e) = N_{PS}(e) && \text{(normalized by } N_{PS}(e)) \\ &\approx P(X, e) / P(e) && \text{(property of PriorSample)} \\ &= P(X|e) && \text{(defn. of conditional probability)} \end{aligned}$$

Hence rejection sampling returns consistent posterior estimates

Problem: hopelessly expensive if $P(e)$ is small

$P(e)$ drops off exponentially with number of evidence variables!

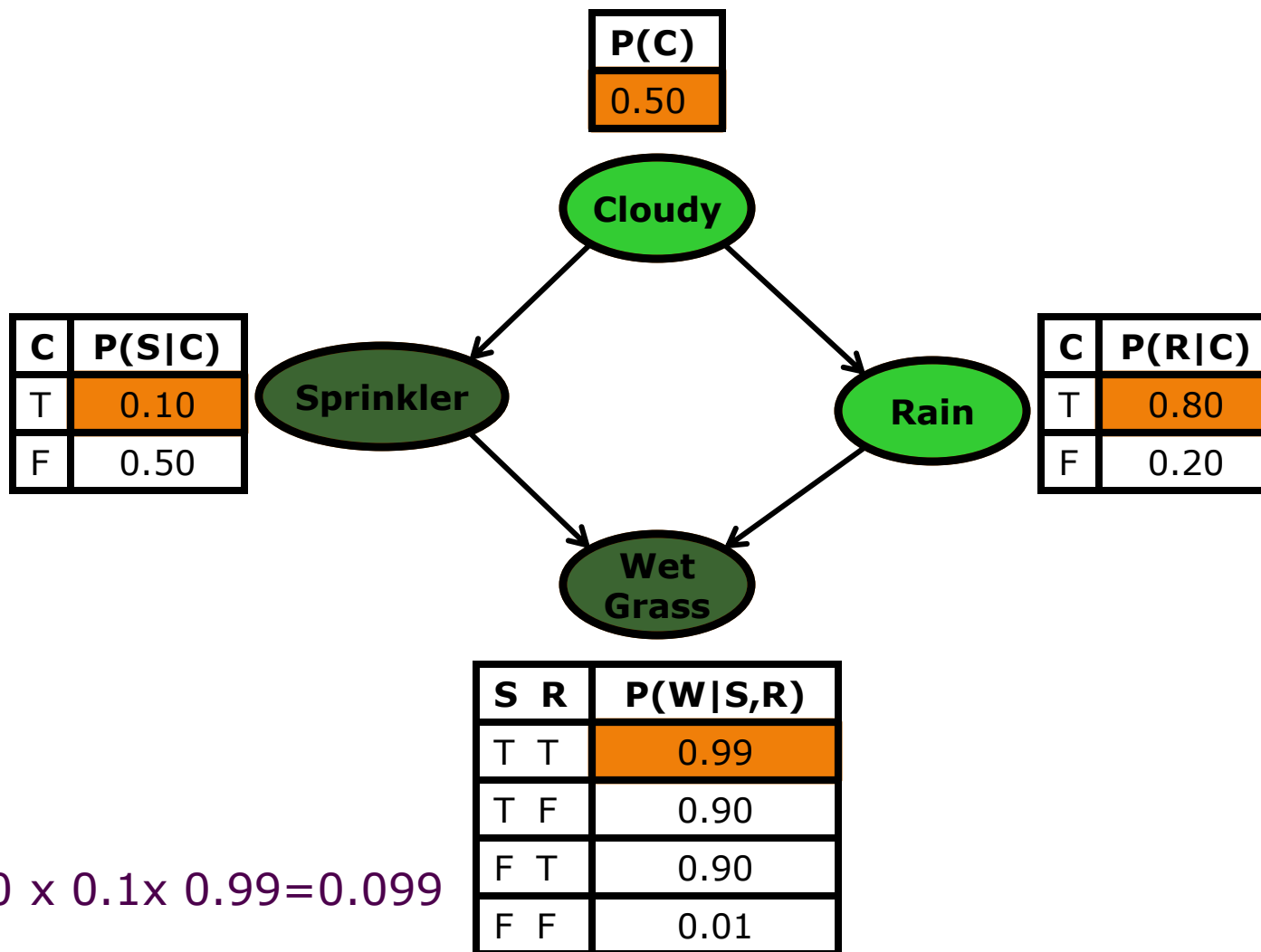
Likelihood weighting

- Idea: fix evidence variables, sample only nonevidence variables, and weight each sample by the likelihood it accords the evidence

```
function LIKELIHOOD-WEIGHTING( $X, e, bn, N$ ) returns an estimate of  $P(X|e)$ 
local variables:  $W$ , a vector of weighted counts over  $X$ , initially zero
for  $j = 1$  to  $N$  do
     $x, w \leftarrow$  WEIGHTED-SAMPLE( $bn$ )
     $W[x] \leftarrow W[x] + w$  where  $x$  is the value of  $X$  in  $x$ 
return NORMALIZE( $W[X]$ )
```

```
function WEIGHTED-SAMPLE( $bn, e$ ) returns an event and a weight
 $x \leftarrow$  an event with  $n$  elements;  $w = 1$ 
for  $i = 1$  to  $n$  do
    if  $X_i$  has a value  $x_i$  in  $e$ 
        then  $w \leftarrow w \cdot P(X_i = x_i \mid \text{parents}(X_i))$ 
        else  $x_i \leftarrow$  a random sample from  $P(X_i \mid \text{parents}(X_i))$ 
return  $x, w$ 
```

Likelihood weighting example



Likelihood weighting analysis

- Sampling probability for WeightedSample is

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

- Note: pays attention to evidence in **ancestors** only
 - Somewhere “in between” prior and posterior distribution
- Weight for a given sample \mathbf{z}, \mathbf{e} is

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

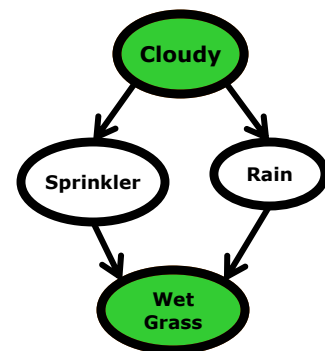
- Weighted sampling probability is

$$S_{WS}(\mathbf{z}, \mathbf{e})w(\mathbf{z}, \mathbf{e})$$

$$= \prod_{i=1}^l P(z_i \mid \text{parents}(Z_i)) \prod_{i=1}^m P(e_i \mid \text{parents}(E_i))$$

$$= P(\mathbf{z}, \mathbf{e}) \quad (\text{by standard global semantics of network})$$

- Hence likelihood weighting returns consistent estimates but performance still degrades with many evidence variables because few samples have nearly all the total weight



Approximate inference using MCMC

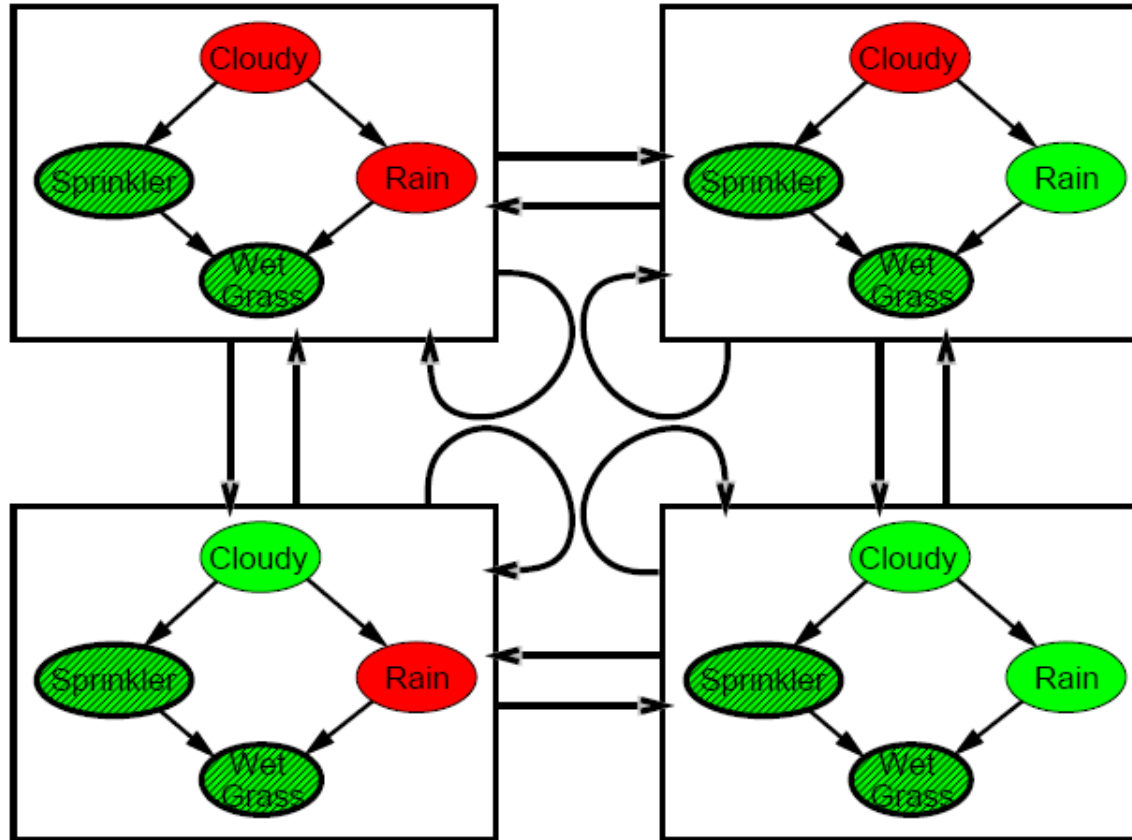
- “State” of network = current assignment to all variables.
- Generate next state by sampling one variable given Markov blanket
- Sample each variable in turn, keeping evidence fixed

```
function MCMC-Ask(X, e, bn, N) returns an estimate of  $P(X|e)$ 
local variables:  $N[X]$ , a vector of counts over  $X$ , initially zero
                  $Z$ , the nonevidence variables in  $bn$ 
                  $x$ , the current state of the network, initially copied from  $e$ 
initialize  $x$  with random values for the variables in  $Y$ 
for  $j = 1$  to  $N$  do
  for each  $Z_i$  in  $Z$  do
    sample the value of  $Z_i$  in  $x$  from  $P(Z_i | mb(Z_i))$  /*  $mb$ =markov blanket */
    given the values of  $MB(Z_i)$  in  $x$ 
     $N[x] \leftarrow N[x] + 1$  where  $x$  is the value of  $X$  in  $x$ 
return NORMALIZE( $N[X]$ )
```

- Can also choose a variable to sample at random each time

The Markov chain

- With *Sprinkler=true*, *Wetgrass=true*, there are four states:



- Wander about for a while, average what you see

MCMC example contd.

- Estimate $P(\text{Rain}|\text{Sprinkler}=\text{true}, \text{WetGrass}=\text{true})$
- Sample **Cloudy** or **Rain** given its Markov blanket, repeat.
- Count number of times **Rain** is true and false in the samples
- E.g., visit 100 states
31 have **Rain=true**, 69 have **Rain=false**
- $\hat{P}(\text{Rain}|\text{Sprinkler}=\text{true};\text{WetGrass}=\text{true})$
= $\text{Normalize}(\langle 31; 69 \rangle) = \langle 0:31; 0:69 \rangle$
- Theorem: chain approaches **stationary distribution**:
long-run fraction of time spent in each state is exactly proportional to its posterior probability

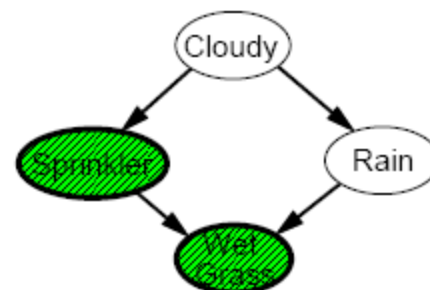
Markov blanket sampling

Markov blanket of **Cloudy** is

Sprinkler and **Rain**

Markov blanket of **Rain** is

Cloudy, **Sprinkler**, and **WetGrass**



Probability given the Markov blanket is calculated as follows:

$$P(x_i' | mb(X_i)) = P(x_i' | parents(X_i)) \prod_{Z_j \in Children(X_i)} P(z_j | parents(Z_j))$$

Easily implemented in message-passing parallel systems

Main computational problems:

- 1) Difficult to tell if convergence has been achieved
- 2) Can be wasteful if Markov blanket is large:

$P(X_i | mb(X_i))$ won't change much (law of large numbers)

Summary

- Exact inference by variable elimination
 - polytime on polytrees, NP-hard on general graphs
 - space = time, very sensitive to topology
- Approximate inference by LW, MCMC:
 - LW does poorly when there is lots of (downstream) evidence
 - LW, MCMC generally insensitive to topology
 - Convergence can be very slow with probabilities close to 1 or 0
 - Can handle arbitrary combinations of discrete and continuous variables