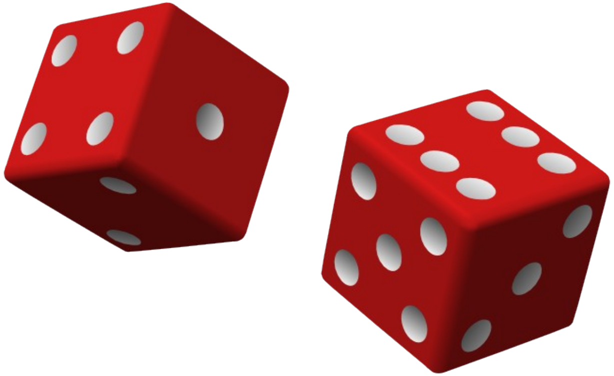
Probabilistic Artificial Intelligence

Lecture 2: Bayesian Networks



CS4375 Probabilistic Artificial Intelligence Techniques
dr. F. Oliehoek

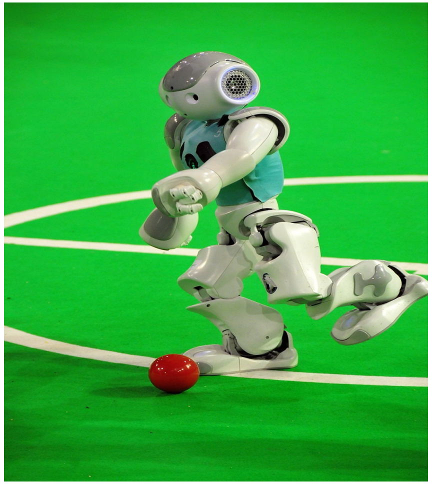
Interactive Intelligence group
Department of Electrical Engineering, Mathematics and Computer Science

TU Delft
Delft University of Technology

Previous Lecture

- What is AI?
  - different perspectives:
    - thinking vs acting,
    - human-like vs. rational

- Agents need to represent beliefs
  - strong arguments: use probability
  - Bayes rule: to update beliefs

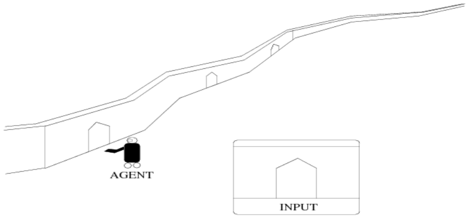


It allows to update a belief

- Bayes rule with State and a particular observation *o*:

  $$\mathbf{P}(State|o) = \mathbf{P}(o|State)\mathbf{P}(State) / P(o)$$

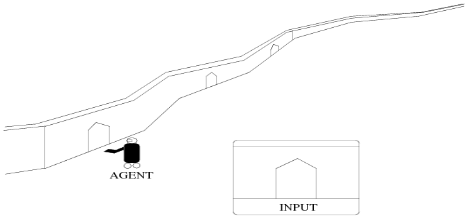
  - $\mathbf{P}(State)$ is our prior belief
  - so we can update our belief, based on observations!

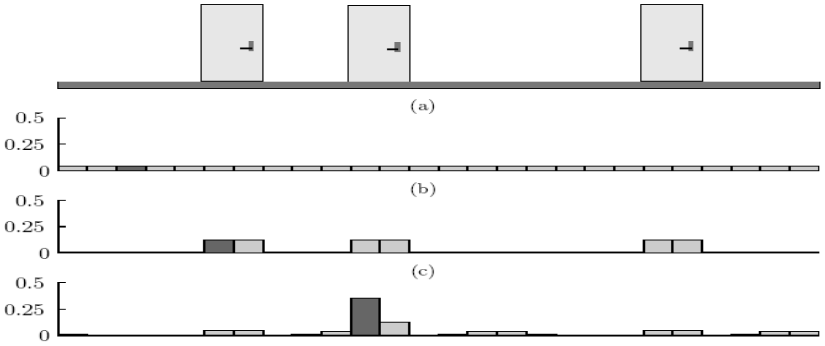


It allows to *maintain* a belief

- Deal with many observations… exploit **conditionally independent** observations
  $\mathbf{P}(o_1,o_2|State) = \mathbf{P}(o_1|State)\mathbf{P}(o_2|State)$

- … then, we can also sequentially update:
  $\mathbf{P}'(State) := \mathbf{P}(o_1|State)\mathbf{P}(State) / P(o_1)$
  $\mathbf{P}''(State) := \mathbf{P}(o_2|State)\mathbf{P}'(State) / P(o_2)$
  - then $\mathbf{P}''(State)=\mathbf{P}(o_1,o_2|State)$
  - (Exercise!)





- Next lecture: incorporate robot movement over time
- Today: how to deal with complex distributions
  - E.g., $\mathbf{P}(State)$ and/or $\mathbf{P}(o|State)$ might be very complex…

This Lecture

- Bayesian Networks
  - What are they?
  - Reasoning with them: Inference
  - Exact Inference
  - Approximate Inference

Compactly representing probability distributions:

Bayesian Networks

Scaling Probabilistic Reasoning

- Probability for beliefs – strong arguments
- Bayes rule: allows updating of beliefs

- The challenge: scaling to many variables – **joint probability tables don't scale**

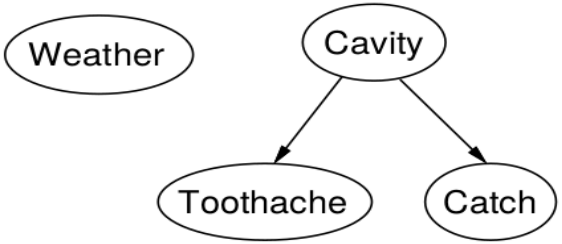
- Important 'hammers':
  - independence – rare
  - conditional independence (CI) – much more common

- Bayesian networks use CI to represent complex problems.

Bayesian Networks (BNs)

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

Topology of network encodes conditional independence assertions:



$Weather$ is independent of the other variables

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

The Burglar Network

Example

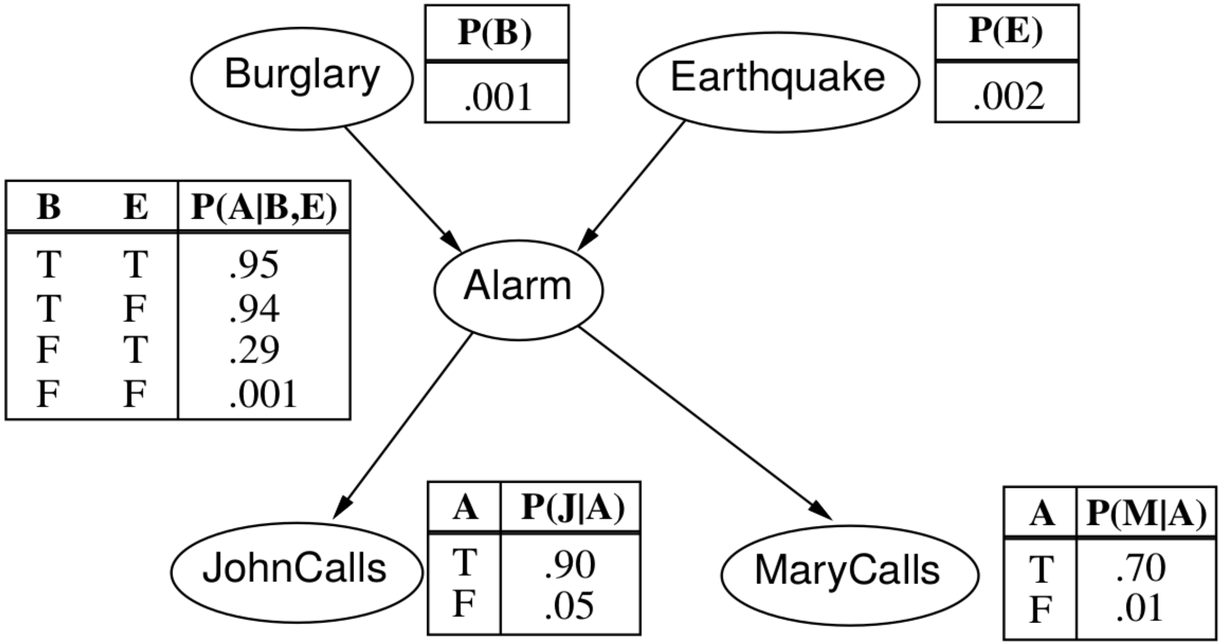
I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

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

The Burglar Network

Example contd.



P(B)
.001

P(E)
.002

B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

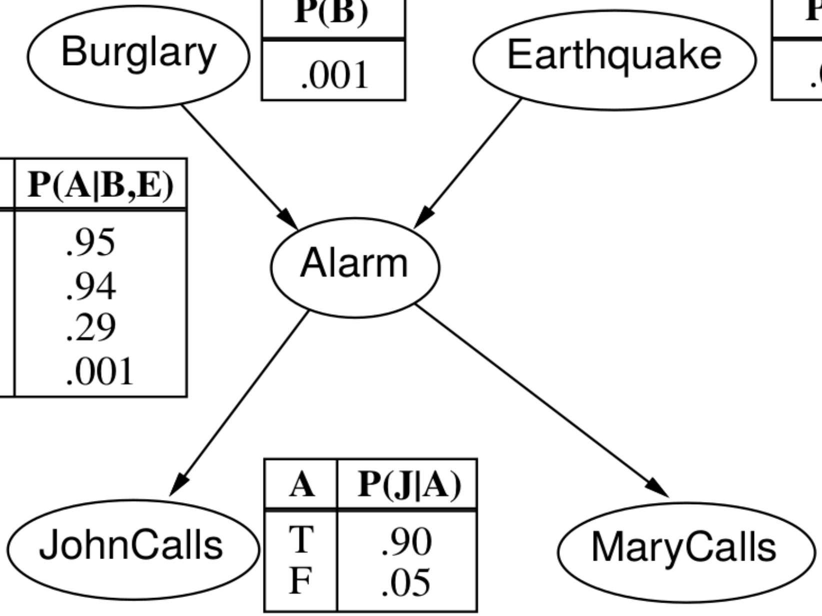
A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01

TU Delft

The Burglar Network

Example contd.



P(B)
.001

P(E)
.002

B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

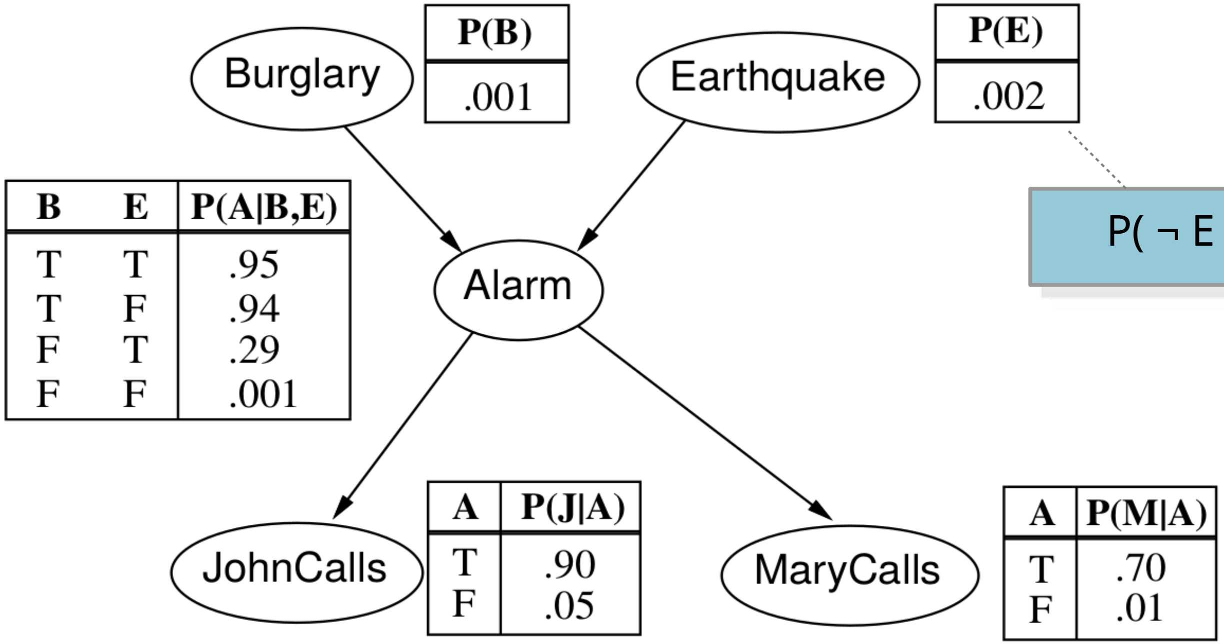
A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01

Note: not all entries are shown
- they sum to 1!

The Burglar Network

Example contd.



P(B)
.001

P(E)
.002

B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01

$P(\neg E) = 1 - .002 = .998$

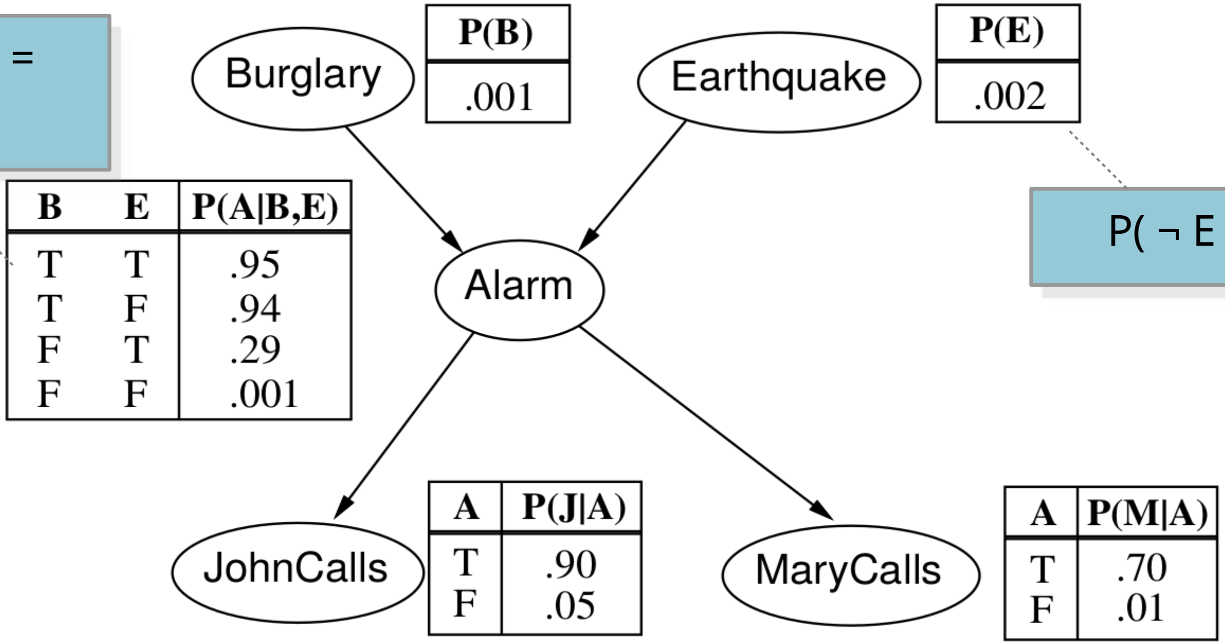
Note: not all entries are shown

- they sum to 1!

The Burglar Network

Example contd.

P(¬ A | B=T, E=T) =
1 - .95= .05



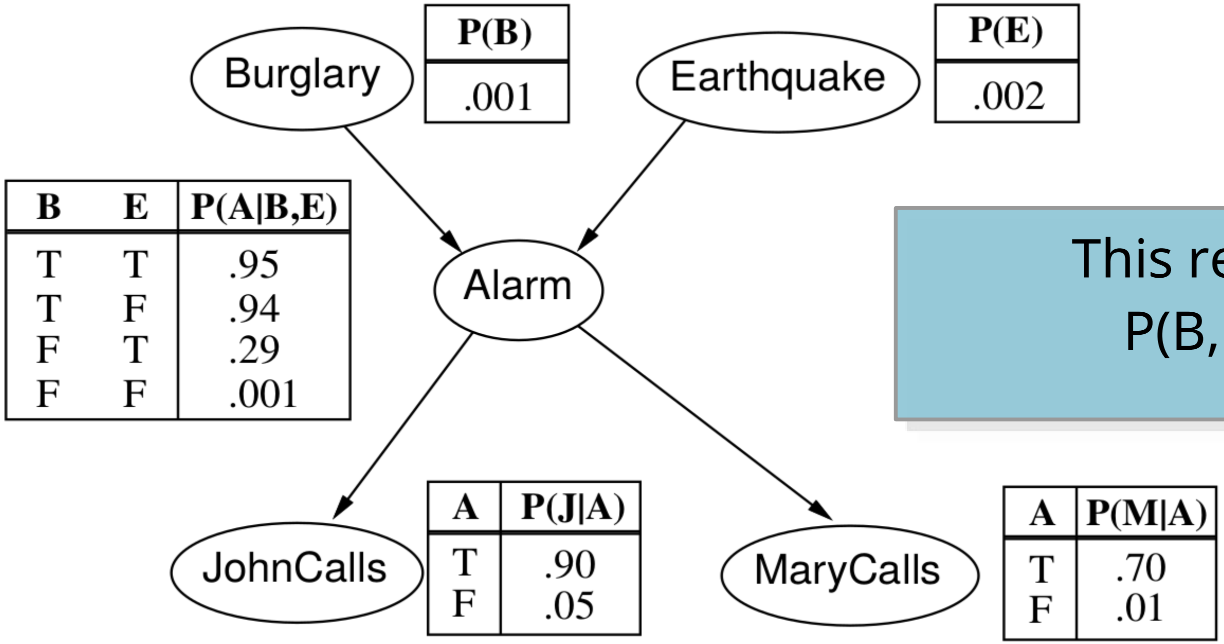
P(¬ E) = 1 - .002= .998

Note: not all entries are shown

- they sum to 1!

The Burglar Network

Example contd.



P(B)
.001

P(E)
.002

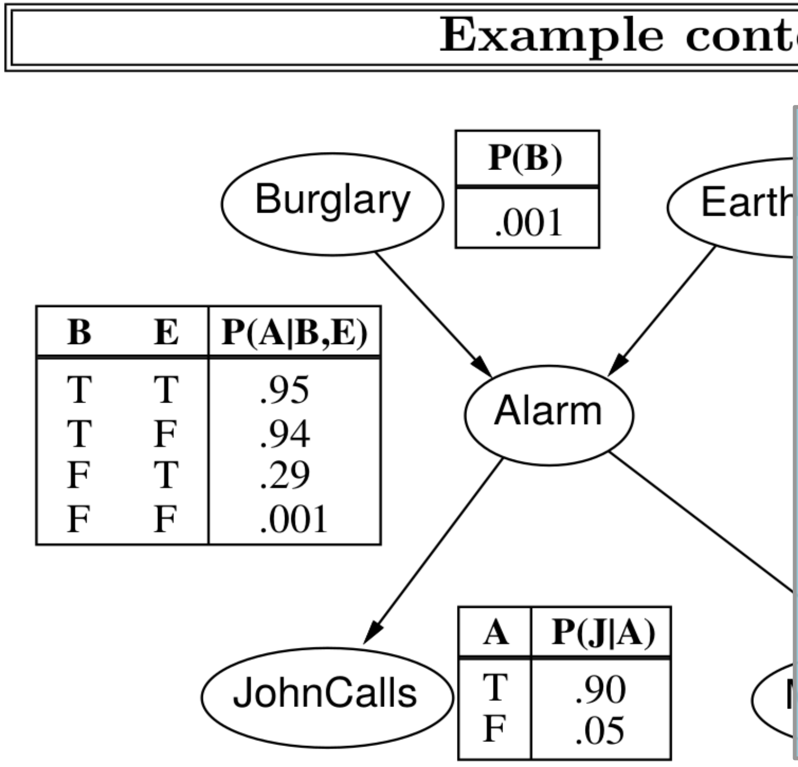
B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01

This represents
P(B,E,A,J,M)

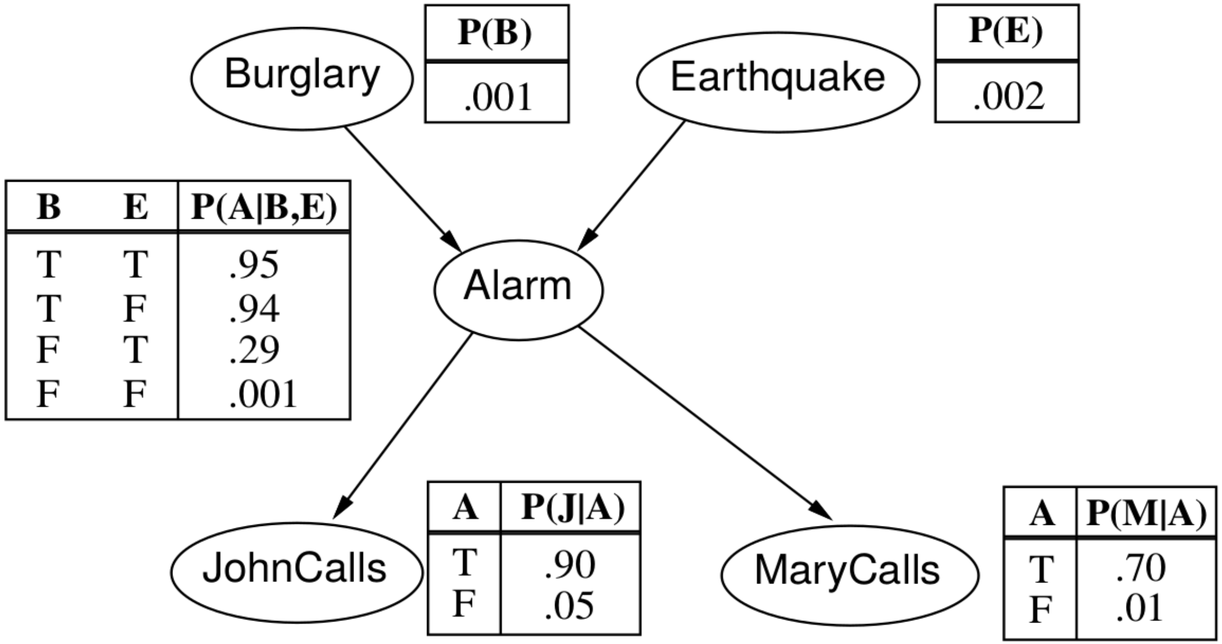
The Burglar Network



- In this lecture, we will assume that the numbers ("the parameters") are given. (e.g., specified by an expert)
- We will use them for **reasoning**, called "**inference**"
- The question of **learning** the parameters will come later.

The Burglar Network

Example contd.



This represents P(B,E,A,J,M)

- but compactly using small **conditional prob. tables (CPTs)**

… how many parameters?

BNs in Practice

- huge number of applications…
  - disaster victim identification
  - Petrophysical decision support (oil, gas drilling)
  - process analysis

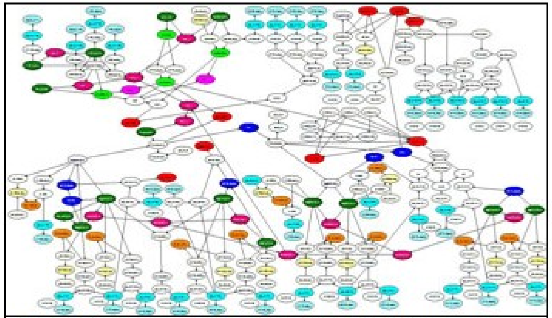
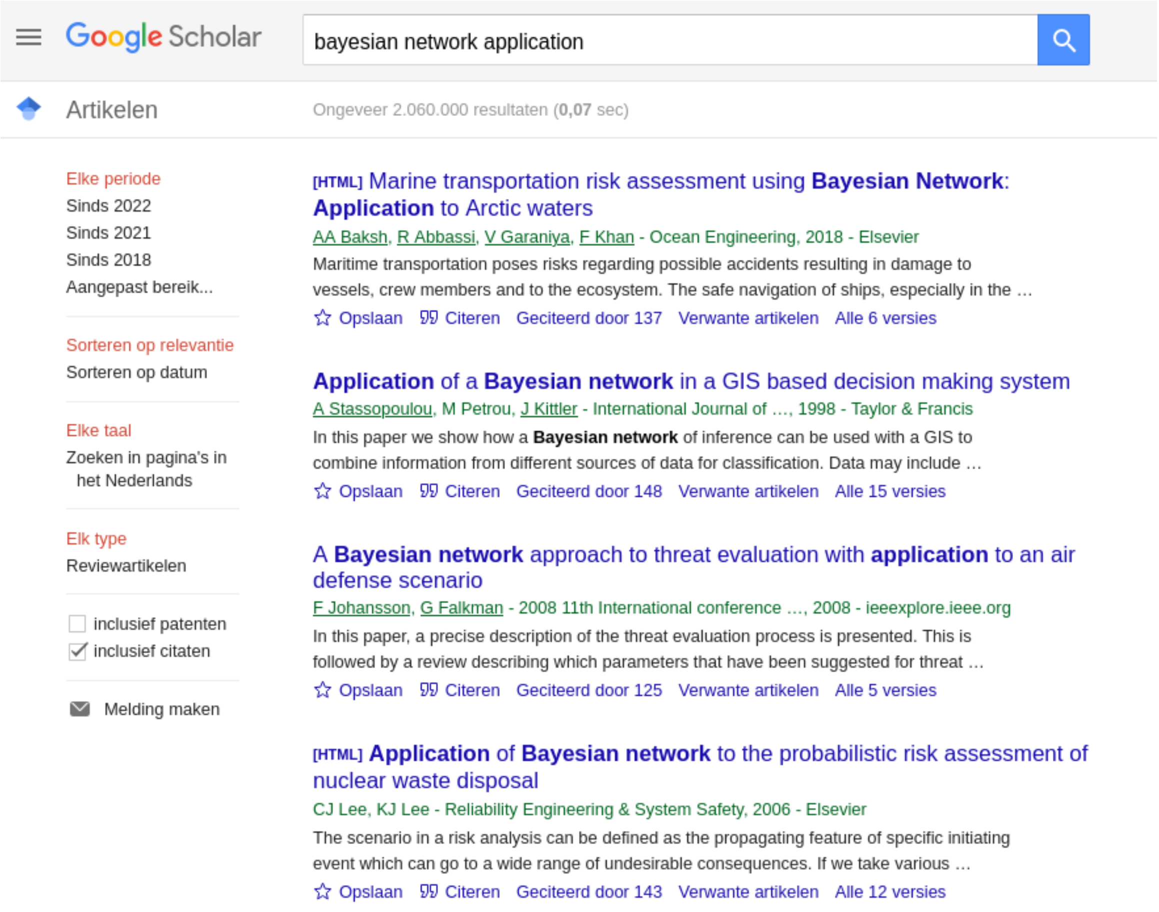


Figure 2: An example of a Bayesian Network for root cause analysis of process operation.



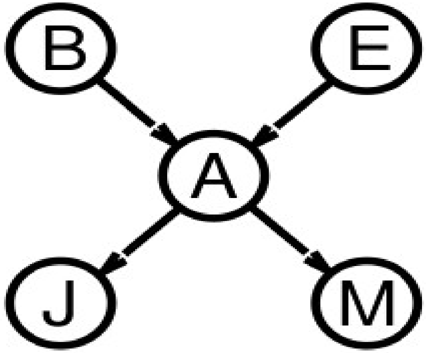
Semantics: global & local

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

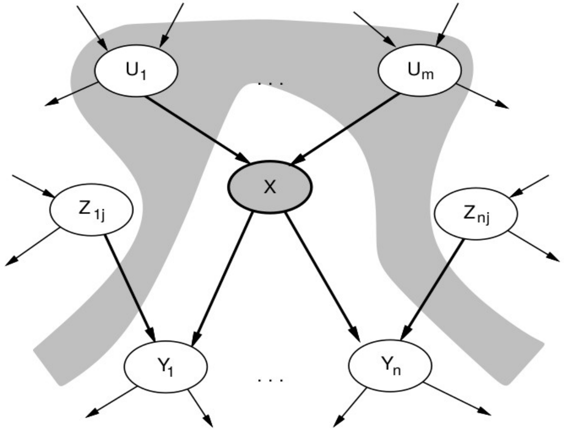
$$P(x_1, \ldots, x_n) = \Pi_{i=1}^{n} P(x_i | parents(X_i))$$

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



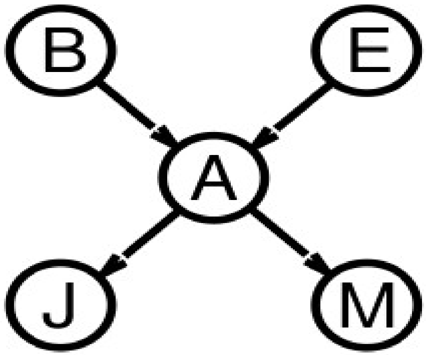
Semantics: global & local

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

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

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

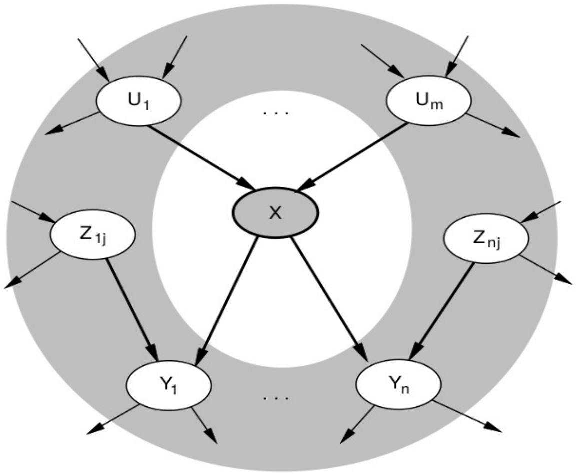
$$\begin{aligned} &= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e) \\ &= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 \\ &\approx 0.00063 \end{aligned}$$



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

Local semantics, a bit stronger:

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



Constructing BNs

- How do we decide how to draw the arrows…?
  - Each PD P(x) might be representable by many BNs…?
- 1 firm rule:
  all conditional independencies implied by the BN need to hold in P(x)
- But many of these may have
  - unnecessary arrows
  - more parameters needed
- Rule of thumb: try to put arrows in causal direction
  - so from cause to effect
  - (figuring out what cause and effect is might be very non-trivial though…!)
- More details: see R&N!

More considerations
(R&N: 13.2.2, 13.2.3 - optional)

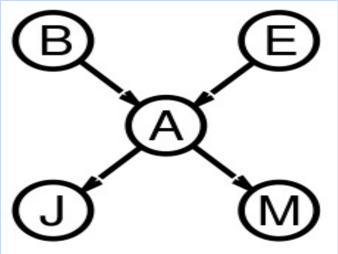
- Compact representations for the conditional probabilities
- Dealing with continuous variables
  - the graphical structure can stay the same!

(Exact) Inference

Inference Tasks

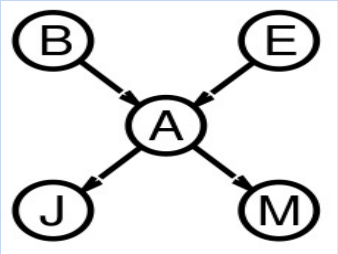
- A typical query:
  - What is $\boldsymbol{P}(X|\boldsymbol{E}=\boldsymbol{e})$?
  - X – query variable
  - **e** – the values of observed evidence vars **E**
  - **Y** – any other variables that are not observed ("**hidden variables**")

P(B|j,m)?



Inference Tasks

P(B|j,m)?

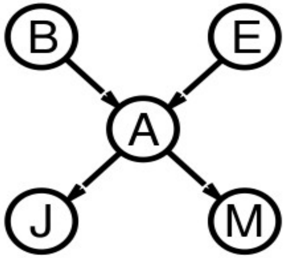


- A typical query:
  - What is $\boldsymbol{P}(X|\boldsymbol{E}=\boldsymbol{e})$?
  - X – query variable
  - **e** – the values of observed evidence vars **E**
  - **Y** – any other variables that are not observed ("**hidden variables**")

- 'conditional probability query'
- others:
  - P(a) – marginal prob. query
  - $\max_x$ P(x) – max. a posteriori (MAP) query

The Burglar Network again

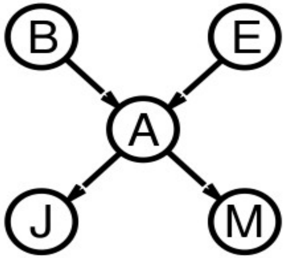
Simple query on the burglary network:
$\mathbf{P}(B|j, m)$



how do we do this?

The Burglar Network again

Simple query on the burglary network:
$\mathbf{P}(B|j,m)$



how do we do this?

Notation:

B – upper case → random variable

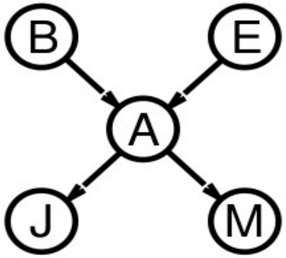
j,m – lowercase → value

P(B|j,m) is a vector:

P(B|j,m) = < P(b|j,m), P(¬b|j,m) >

The Burglar Network again

Simple query on the burglary network:
$\mathbf{P}(B|j,m)$
$= \mathbf{P}(B,j,m)/P(j,m)$



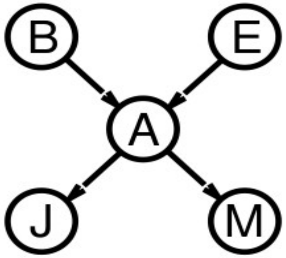
The Burglar Network again

Simple query on the burglary network:

$\mathbf{P}(B|j,m)$

$= \mathbf{P}(B,j,m)/P(j,m)$

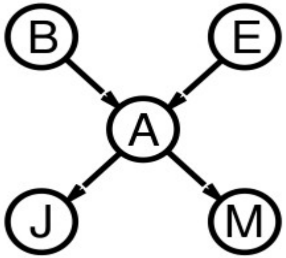
$= \alpha\mathbf{P}(B,j,m)$



The Burglar Network again

Simple query on the burglary network:

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

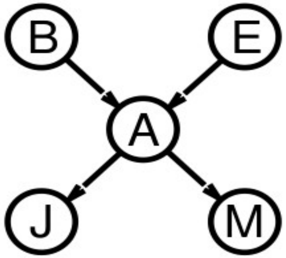


TU Delft

The Burglar Network again

Simple query on the burglary network:

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

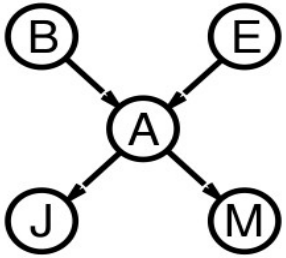


but, we want to avoid constructing **P**(B,E,A,J,M)…

The Burglar Network again

Simple query on the burglary network:

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



but, we want to avoid constructing **P**(B,E,A,J,M)…

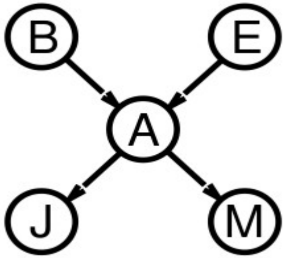
Rewrite full joint entries using product of CPT entries:

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

The Burglar Network again

Simple query on the burglary network:

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



but, we want to avoid constructing **P**(B,E,A,J,M)…

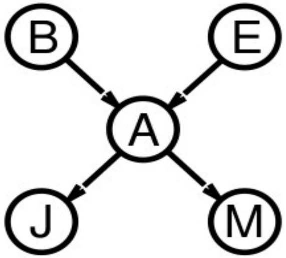
Rewrite full joint entries using product of CPT entries:

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

The Burglar Network again

Simple query on the burglary network:

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



but, we want to avoid constructing **P**(B,E,A,J,M)…

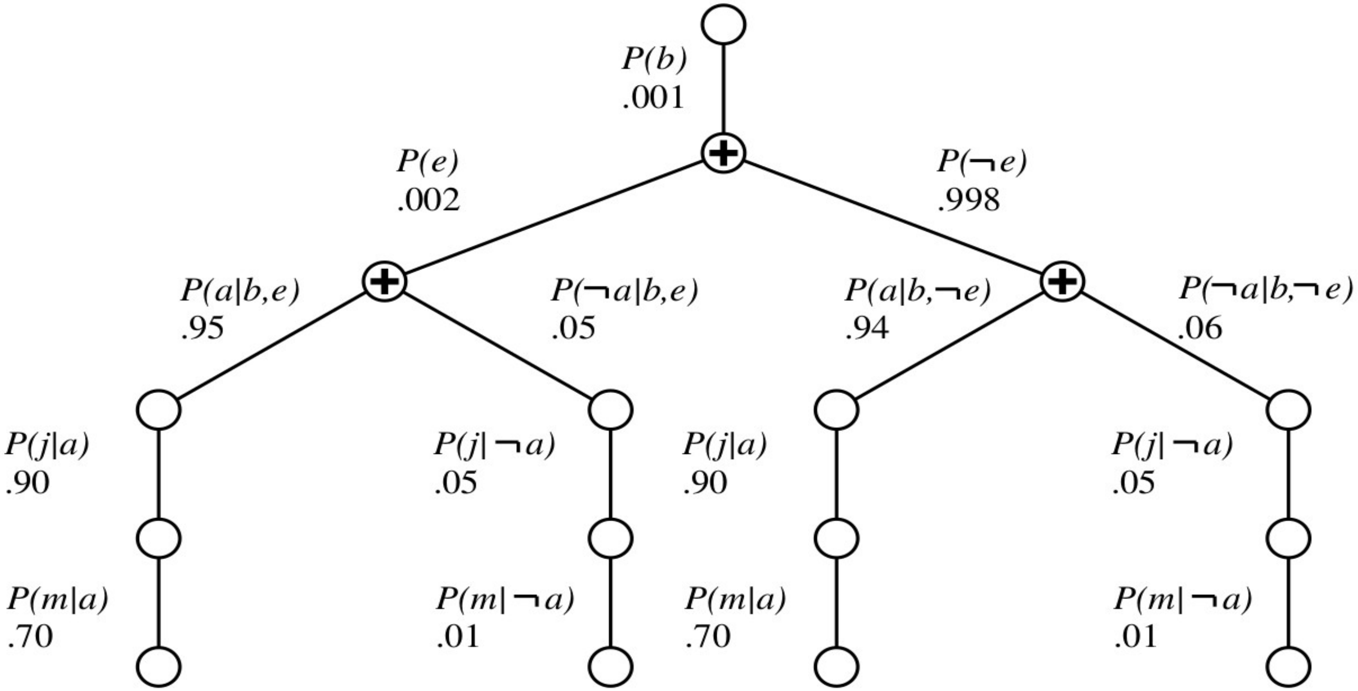
- loop through variable values, to compute summations
- compute *P(b,e,a,j,m)* "on-the-fly"
- called inference by **enumeration** or **search**

Rewrite full joint entries using product of CPT entries:

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

TU Delft

Enumeration Tree

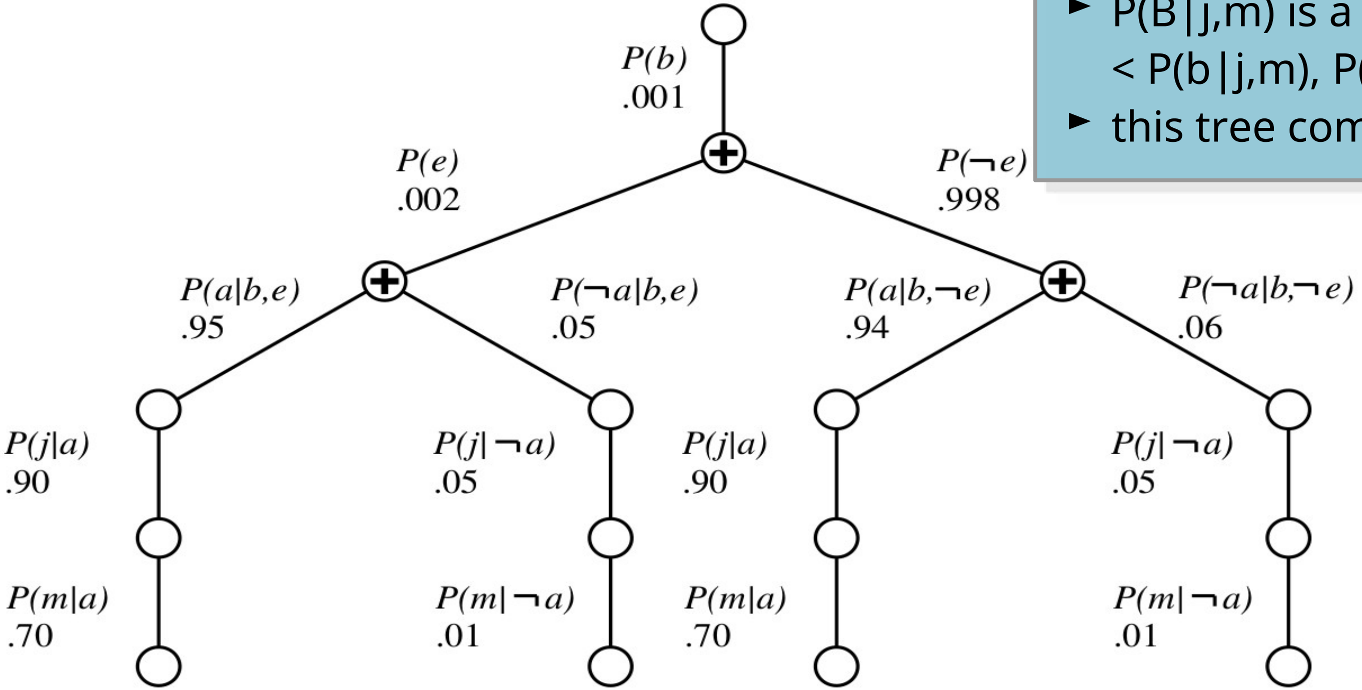


$$\mathbf{P}(B|j,m) = \alpha\mathbf{P}(B)\ \Sigma_e\ P(e)\ \Sigma_a\ \mathbf{P}(a|B,e)P(j|a)P(m|a)$$

Enumeration Tree

Note

- B - random variable
- P(B|j,m) is a vector:
  < P(b|j,m), P(¬b|j,m) >
- this tree computes P(b|j,m)

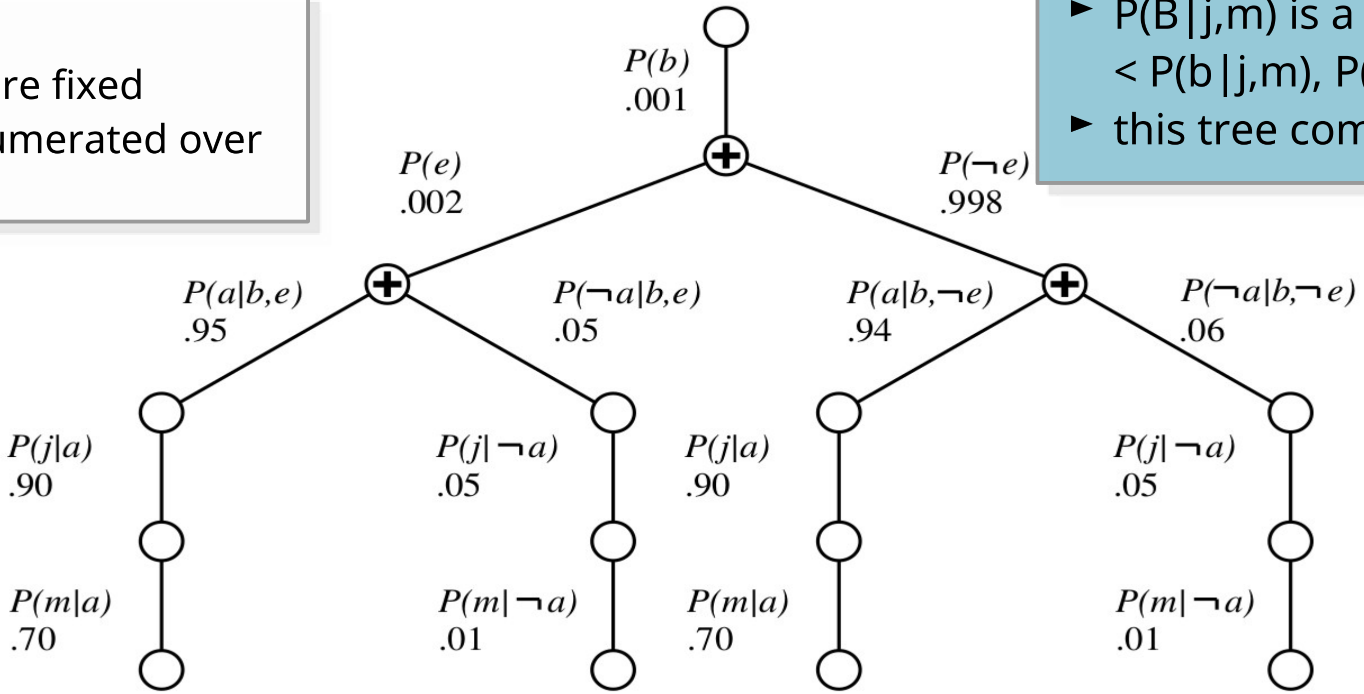


$$\mathbf{P}(B|j,m) = \alpha \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e) P(j|a) P(m|a)$$

Enumeration Tree

So:
- B=b, J=j, M=m are fixed
- A and E are enumerated over

Note
- B - random variable
- P(B|j,m) is a vector:
  < P(b|j,m), P(¬b|j,m) >
- this tree computes P(b|j,m)

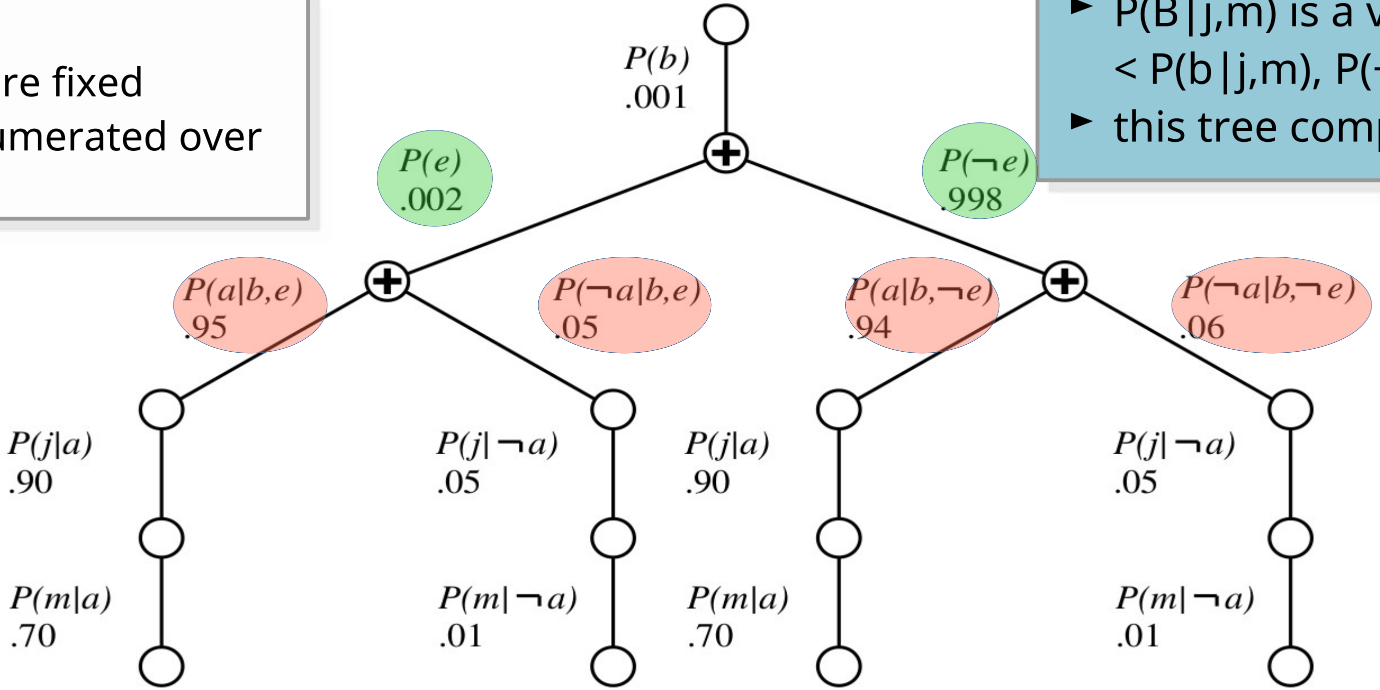


$$\mathbf{P}(B|j,m) = \alpha \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e) P(j|a) P(m|a)$$

Enumeration Tree

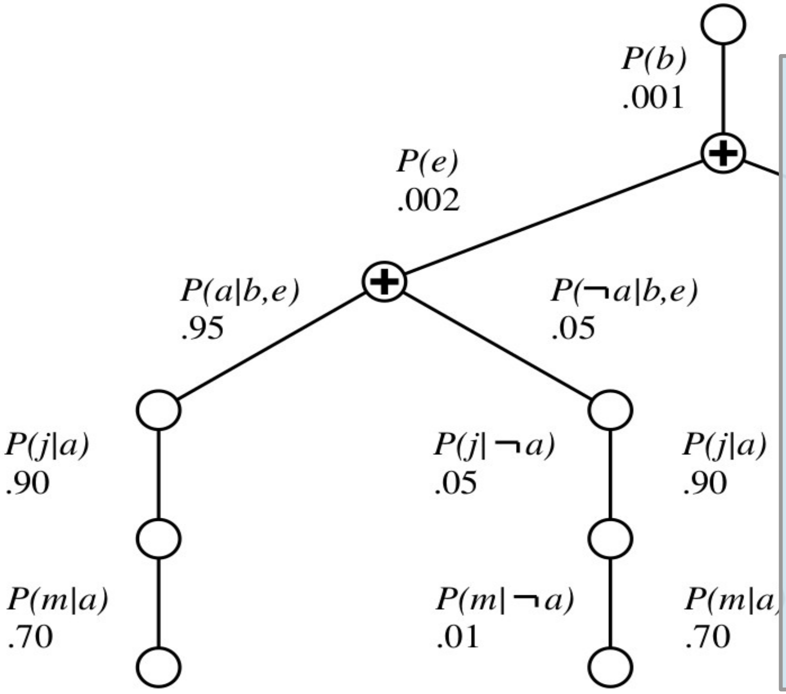
So:
- B=b, J=j, M=m are fixed
- A and E are enumerated over

Note
- B - random variable
- P(B|j,m) is a vector: < P(b|j,m), P(¬b|j,m) >
- this tree computes P(b|j,m)



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

Enumeration Tree



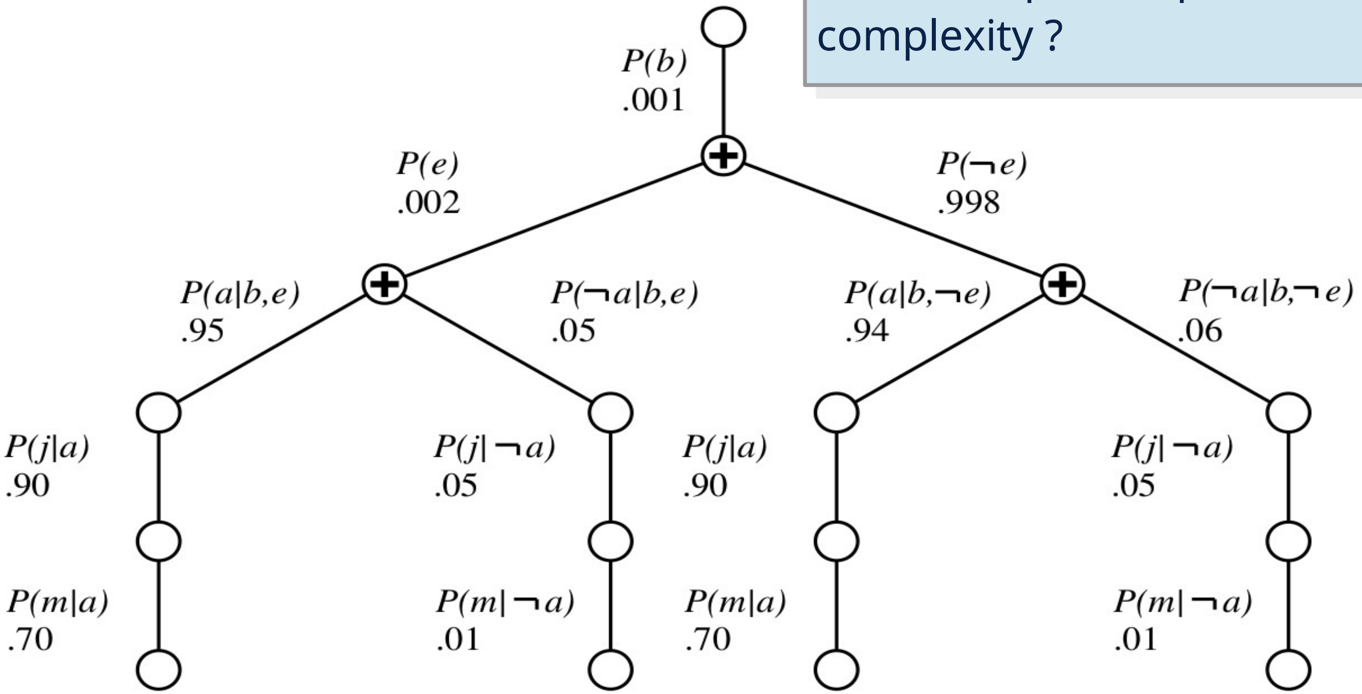
- Given this tree structure…
  can answer these queries by
  depth-first traversal
  - "enumeration-ask" algorithm in R&N.
  - space: O(n)
  - time: $O(2^n)$

- You should know this…
  - "Search" comes from relation to depth-first search (R&N Chap. 3)
  - Big-o notation: O(…)

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

Enumeration Tree

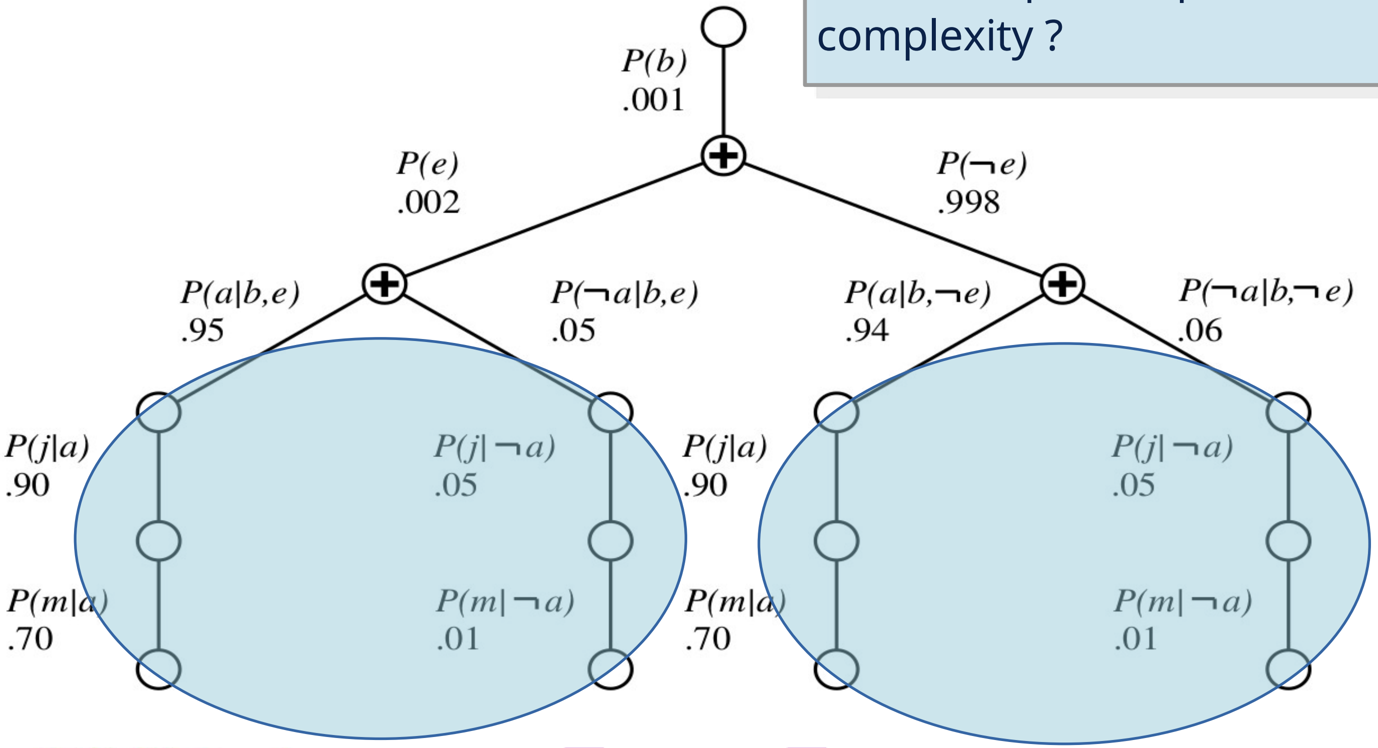
Can we improve upon $O(2^n)$ time complexity ?



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

Enumeration Tree

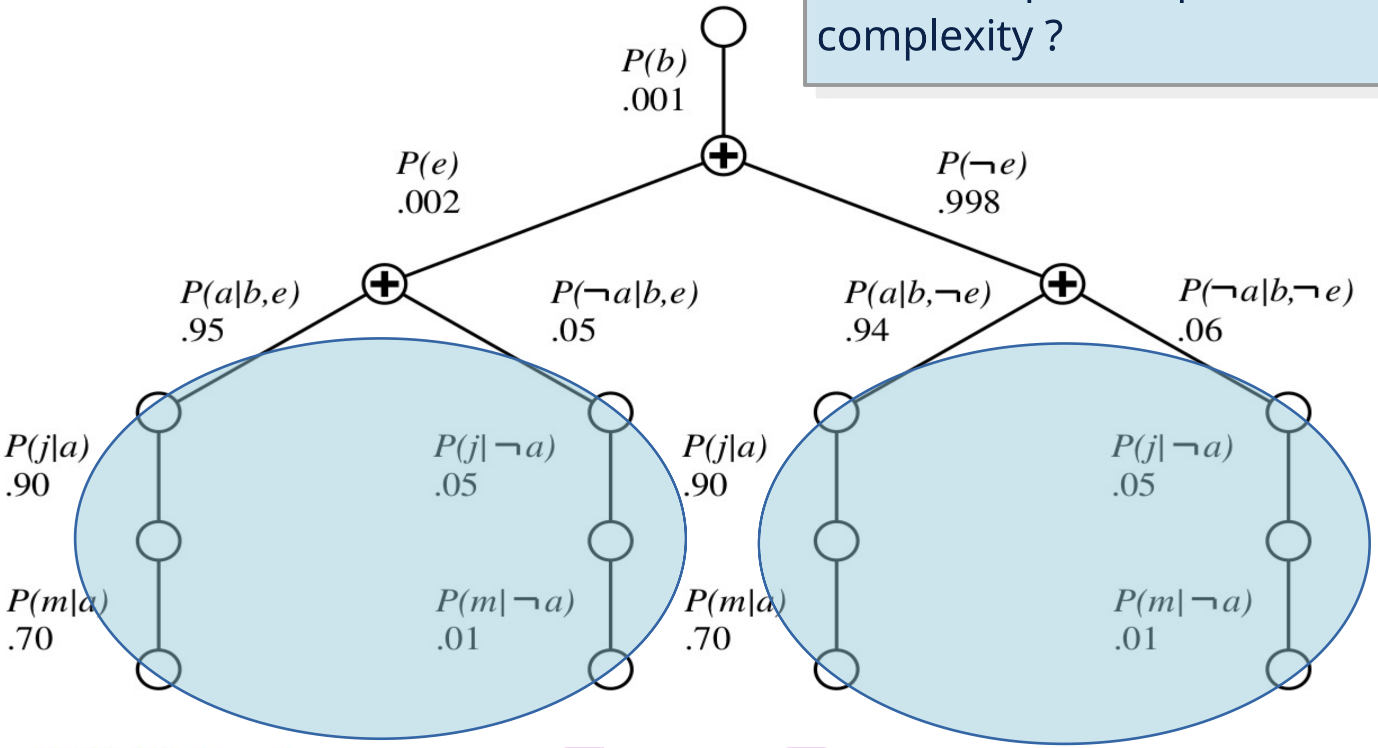
Can we improve upon $O(2^n)$ time complexity ?



$$\mathbf{P}(B|j,m) = \alpha \mathbf{P}(B) \, \Sigma_e \, P(e) \, \Sigma_a \, \mathbf{P}(a|B,e) P(j|a) P(m|a)$$

Enumeration Tree

Can we improve upon $O(2^n)$ time complexity ?



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

Perhaps we can **cache** these replicated computations…?

Variable Elimination (VE)

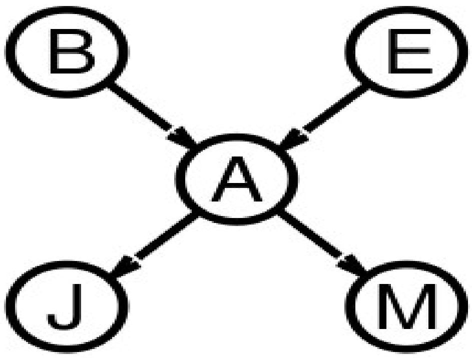
- First push in the summations as far as possible
- Then carry out summations **right-to-left**, caching intermediate results in new **factors**

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

Variable Elimination (VE) – 2

- Implementation in terms of 2 operations:
  - pointwise-product: **f'**(A,B,C)=$\mathbf{f}_1$(A,B) x $\mathbf{f}_2$(B,C)
  - Sum-out: **f''**(A,C) **=** $\sum_b$ **f'**(A,b,C)

- Complexity of VE: **time and space depends on largest factor constructed**.
  - **exponential** in the number of variables that participate in it.
  - trick: don't construct **f'**(A,B,C) explicitly,
    compute entries **f'**(A,b,C)=$\mathbf{f}_1$(A,b) x $\mathbf{f}_2$(b,C) "on-the-fly"



- How big is the largest factor?
  - depends on BN topology and picked 'ordering'
  - cannot bound in general….

- For **polytrees** VE is efficient:
  - runs in time and space linear in 'size' of BN.
  - polytree: between each pairs of nodes at most 1 undirected path

Are there better algorithms?

- Well, we can trade of time for space…
- But in terms of just time, there is little hope
    - Arnie Rosenthal (1977):

> then defined. It is shown that a variable-elimination procedure, nonserial dynamic programming, is optimal in an extremely strong sense among all algorithms in the subclass. The results' strong implications for choosing deterministic, adaptive, and nondeterministic algorithms for the optimization problem, for defining a complexity measure for a pattern of interactions, and for describing general classes of decomposition procedures are discussed. Several possible extensions and unsolved problems are mentioned.

- Exact inference is intractable (see R&N 13.3.3)

Summary so far

- Agents need to represent beliefs
  - strong arguments: use probability
  - Bayes rule: to update beliefs

- Compact representations:
  Bayesian networks exploit conditional independence

- Exact inference:
  - enumeration / search
  - variable elimination
  - intractable in general, polytime on polytrees

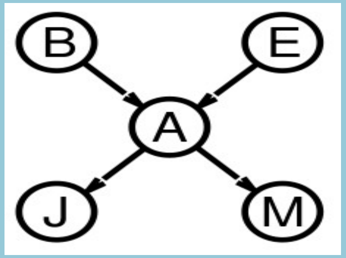
- Next: Approximate inference, preferences & utilities

Approximate Inference

Recap

- Probability: useful for representing beliefs of our agents
- Compactly representing: Bayesian networks
  - exploits conditional independence

P(B|j,m)?



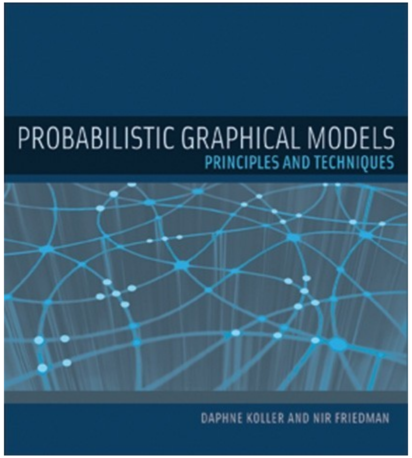
- Inference intractable in general…
  - Need to sum out (enumerate) all hidden variables
  - Variable elimination runs in polytime on polytrees only
- So… consider approximate inference

Approximate inference

- Given the complexity results…
  - …much work on approximate inference!

- Two main strands:
  - based on sampling
  - based on optimization
    (“variational inference”)

- No way that we can cover all this today…
  but see:
  - Daphne Koller & Nir Friedman
  - Daphne’s Coursera courses



Inference by sampling

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

Inference by sampling

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

Inference by sampling

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

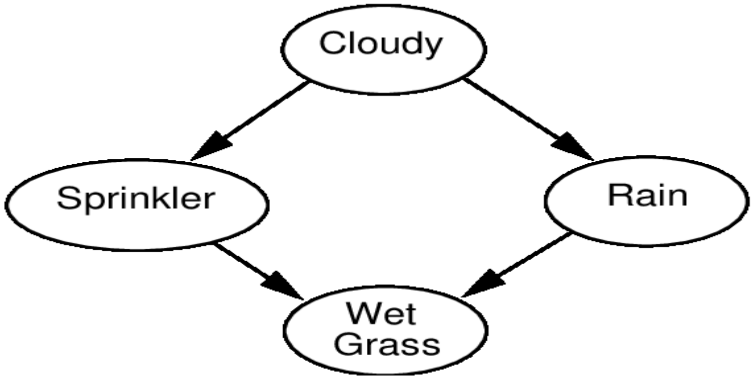
Outline:

– 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

← check it out yourself

Warm up:
Sampling without evidence

- How can we sample from a network without any evidence?
  - (i.e., no observations at all,
    all nodes “hidden”)



Ancestral Sampling

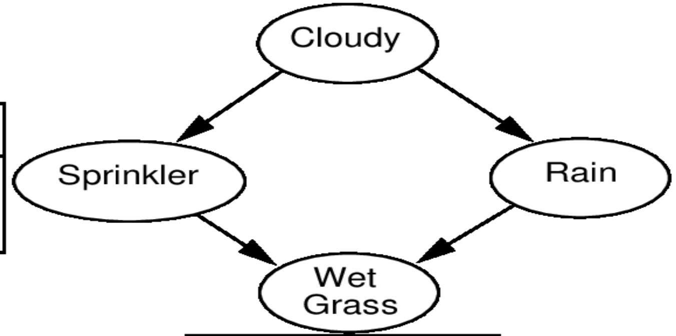
Sampling from an empty network

```
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
```


TU Delft

Ancestral Sampling



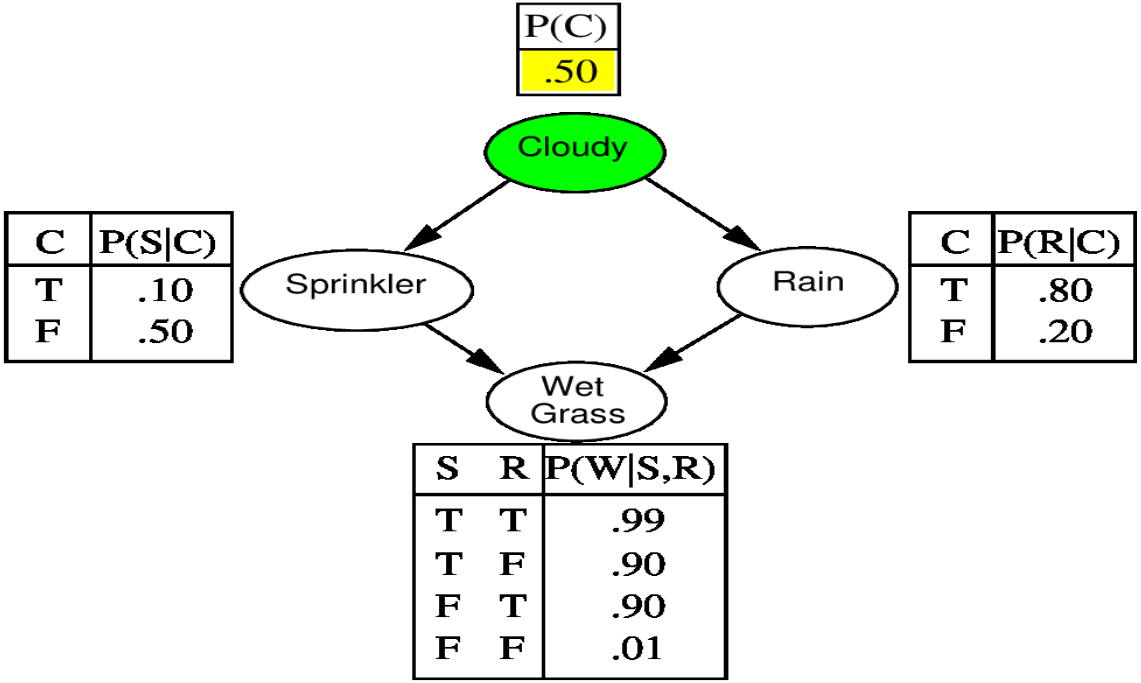
P(C)
.50

C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Ancestral Sampling



P(C)
.50

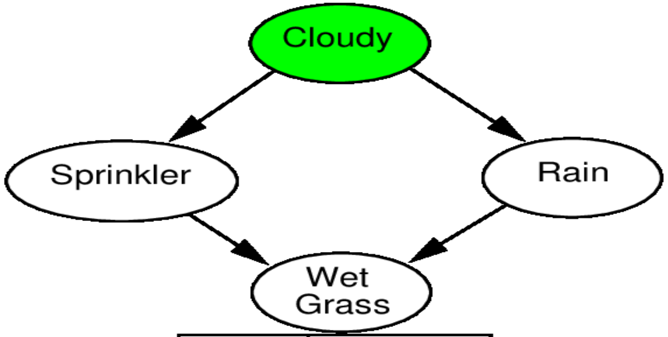
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Ancestral Sampling

P(C)
.50

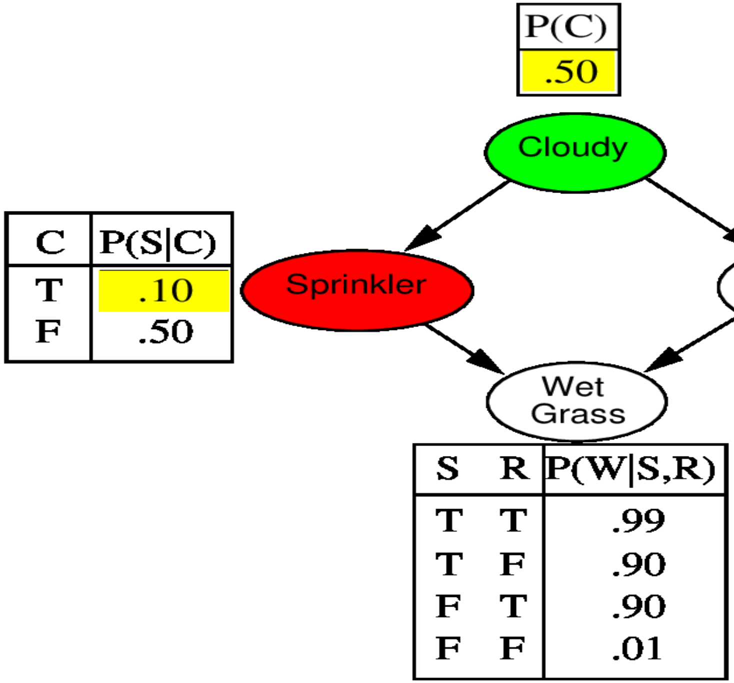


C	P(S|C)
T	.10
F	.50

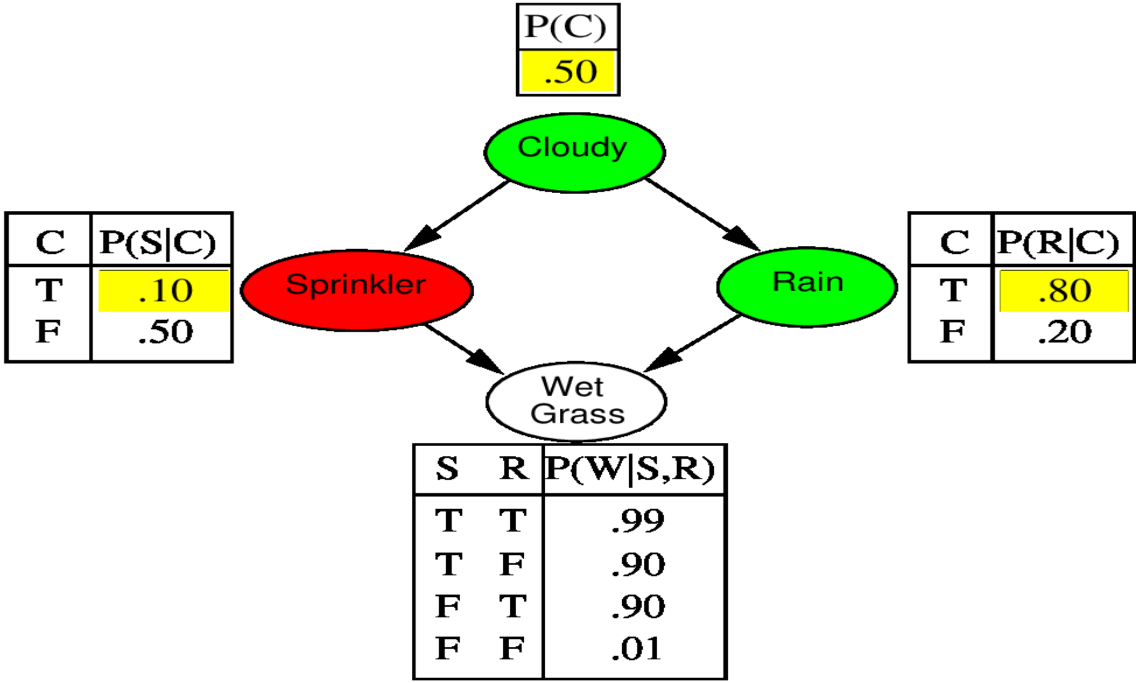
C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

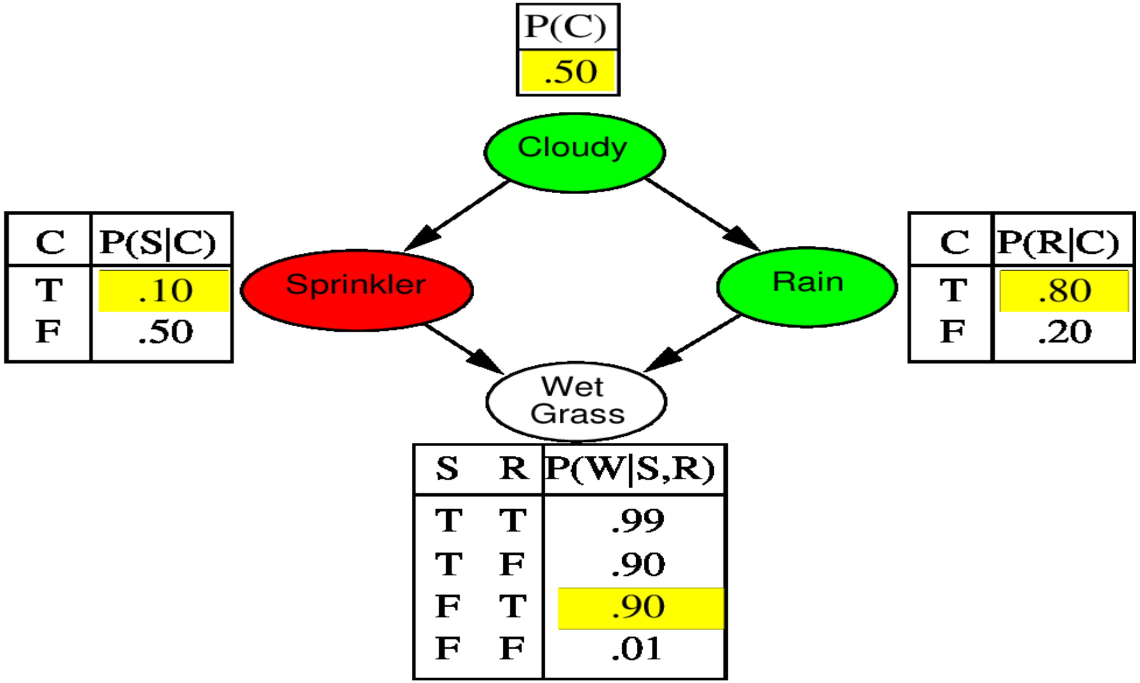
Ancestral Sampling



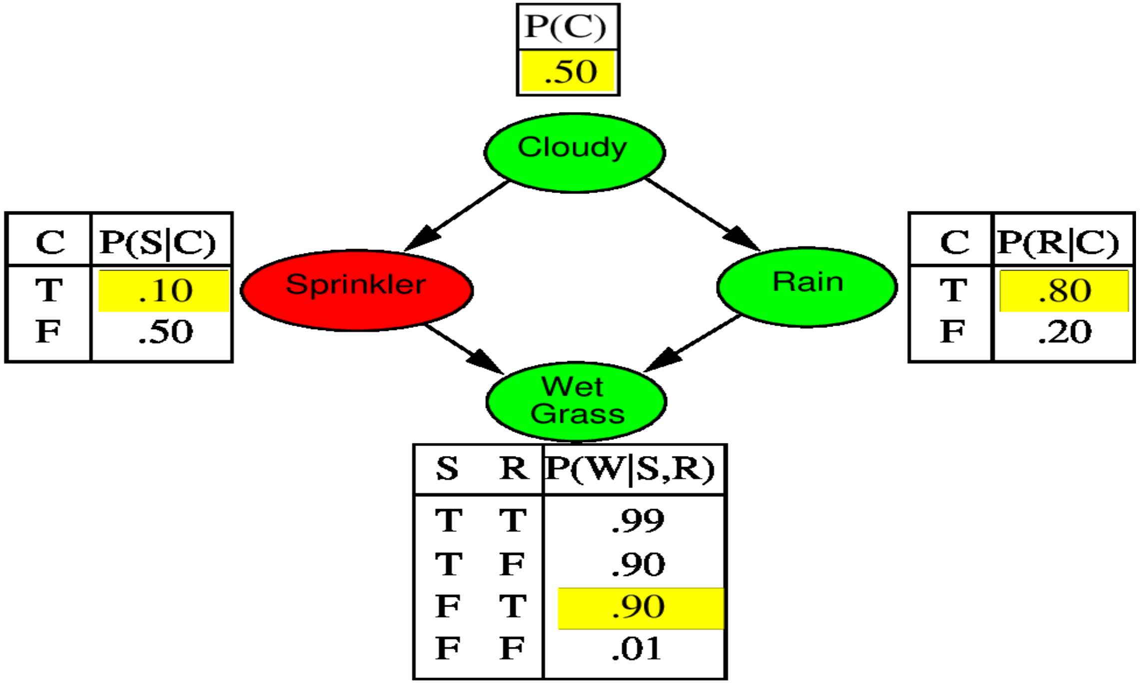
Ancestral Sampling



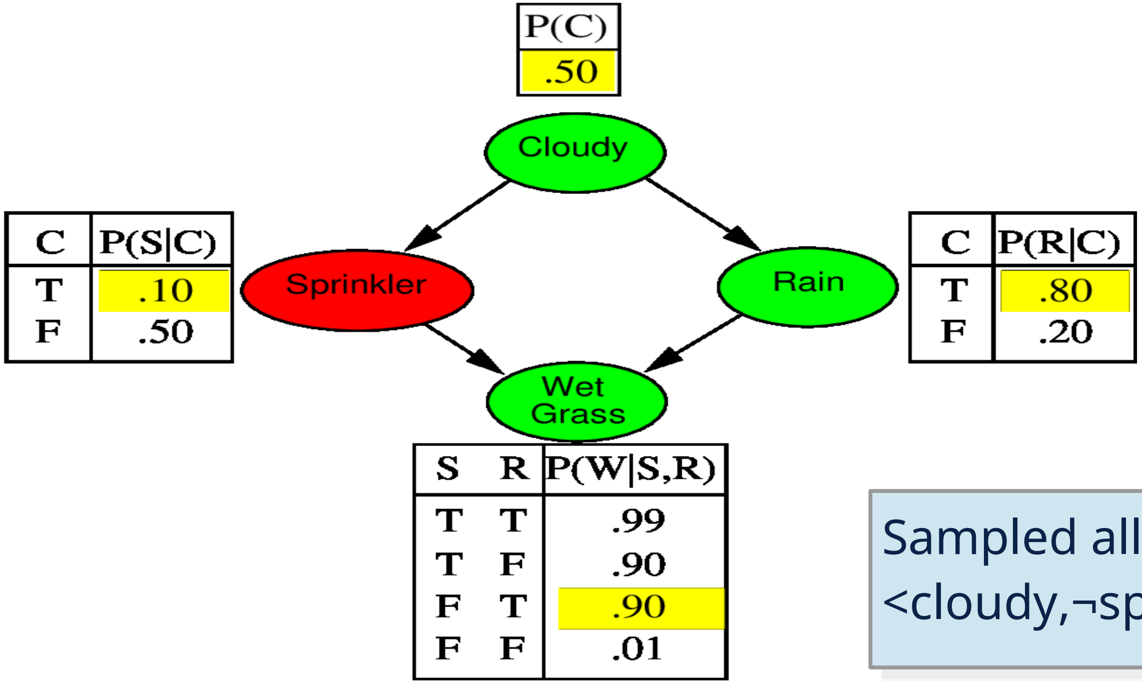
Ancestral Sampling



Ancestral Sampling



Ancestral Sampling



P(C)
.50

C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Sampled all vars:
<cloudy,¬sprinkler,rain,wet>

Using the Samples

- Sampling is nice... but how do we use these samples?
  - to estimate probabilities!

- Draw N samples
- Count how often we saw certain occurrences $N(x_1,\ldots,x_n)$
  - E.g., N(cloudy,¬sprinkler,rain,wet)
- Estimate:

$$\hat{P}(x_1,\ldots,x_n) = N(x_1,\ldots,x_n)/N$$

Using the Samples

- Sampling is nice… but how do we use these samples?
  - to estimate probabilities!

- Draw *N* samples
- Count how often we saw certain occurrences $N(x_1,\ldots,x_n)$
  - E.g., N(cloudy,¬sprinkler,rain,wet)
- Estimate:

$$\hat{P}(x_1,\ldots,x_n) = N(x_1,\ldots,x_n)/N$$

is this estimate any good?

“Consistency”

Probability that PriorSample generates a particular event

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

i.e., the true prior probability

“Consistency”

definition of S_{PS}

Probability that PRIORSAMPLE generates a particular event

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

equality

i.e., the true prior probability

TU Delft

“Consistency”

Probability that PRIORSAMPLE generates a particular event

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

i.e., the true prior probability

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

“Consistency”

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

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

{law of large numbers}

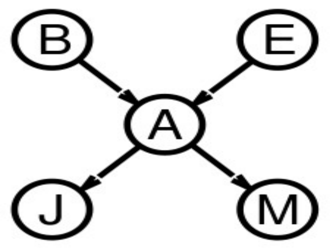
That is, estimates derived from PriorSample are consistent

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

Rejection Sampling

- We usually want to estimate something more complicated… like **P**(X|**e**)

P(B|j,m)?



Rejection Sampling

- We usually want to estimate something more complicated… like **P**(X|**e**)
- Main idea: estimate this 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 ← 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 $Rain = false$.

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

Again: consistency

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

Hence rejection sampling returns consistent posterior estimates

Again: consistency

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

Hence rejection sampling returns consistent posterior estimates

any problems?

TUDelft

Again: consistency

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

Hence rejection sampling returns consistent posterior estimates

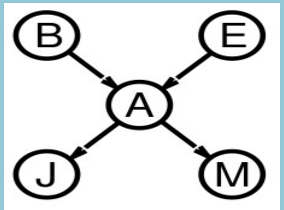
Problem: hopelessly expensive if $P(\mathbf{e})$ is small

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

Likelihood Weighting

- Idea: **only** sample points consistent with evidence **e**

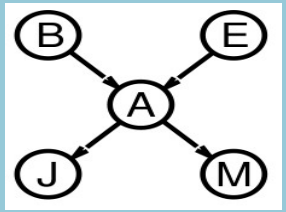
P(B|j,m)?



Likelihood Weighting

- Idea: **only** sample points consistent with evidence **e**
  - sample hidden variables **y** and x
  - form data point **x=**(x,**y**,**e**)
  - compute 'weight' w
  - repeat to construct data set of samples
  - renormalize according to weights
  - answer query based on this renormalized data set

P(B|j,m)?



Likelihood Weighting - Algorithm

function LIKELIHOOD-WEIGHTING(X, $\mathbf{e}$, bn, N) **returns** an estimate of $P(X|\mathbf{e})$
  local variables: $\mathbf{W}$, a vector of weighted counts over X, initially zero

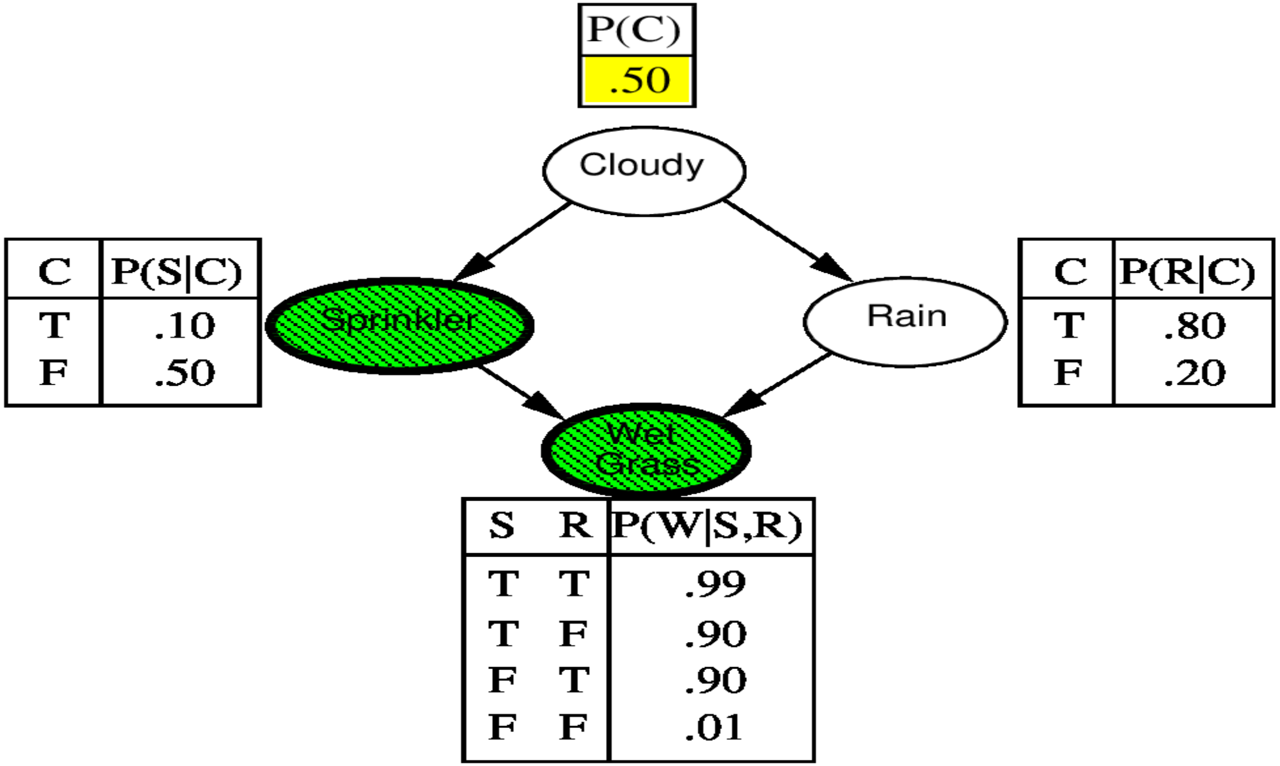
  for $j = 1$ to N **do**
    $\mathbf{x}, w \leftarrow$ WEIGHTED-SAMPLE(bn)
    $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where x is the value of X in $\mathbf{x}$
  return NORMALIZE($\mathbf{W}[X]$)

---

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

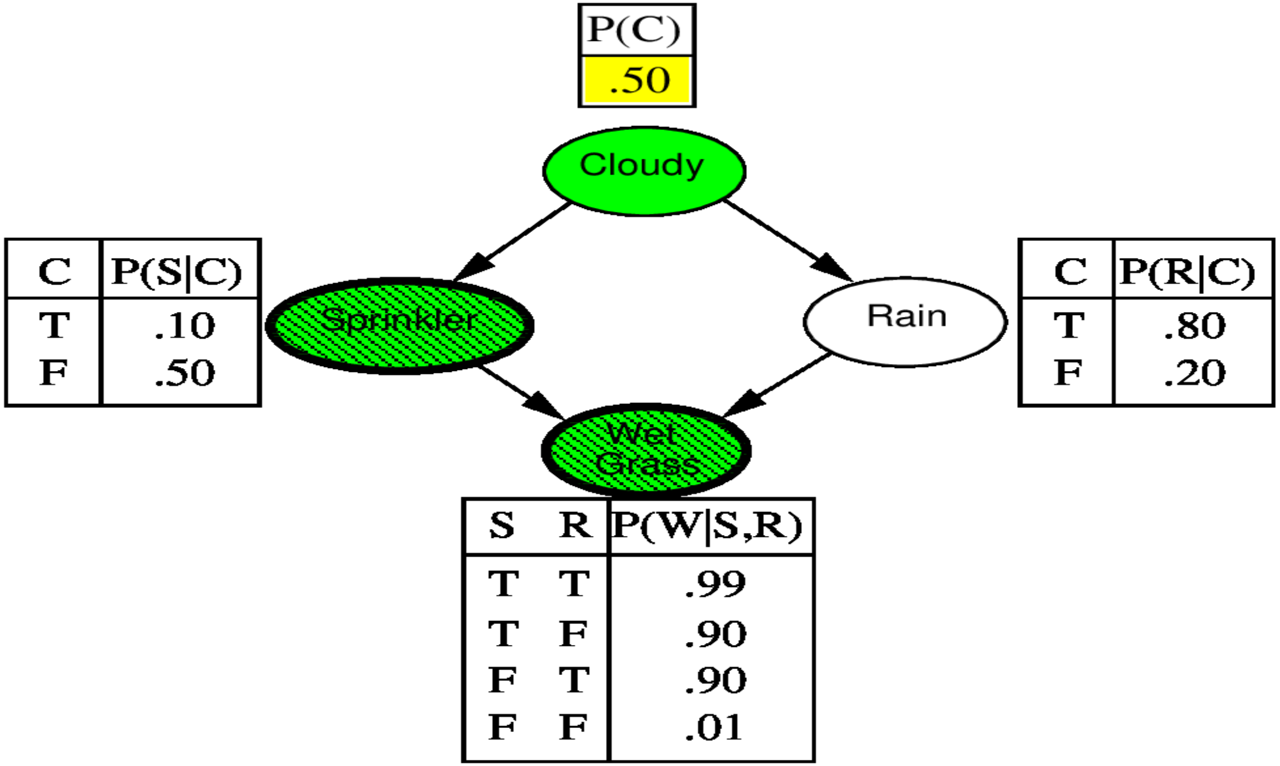
  $\mathbf{x} \leftarrow$ an event with n elements; $w \leftarrow 1$
  for $i = 1$ **to** n **do**
    if X_i has a value x_i in $\mathbf{e}$
      then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
      else $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return $\mathbf{x}$, w

Likelihood Weighting - Example



w=1

Likelihood Weighting - Example



P(C)
.50

C	P(S|C)
T	.10
F	.50

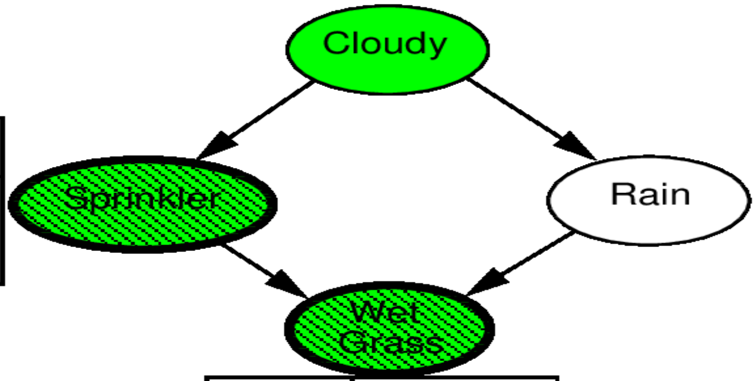
C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

w=1

Likelihood Weighting - Example

P(C)
.50



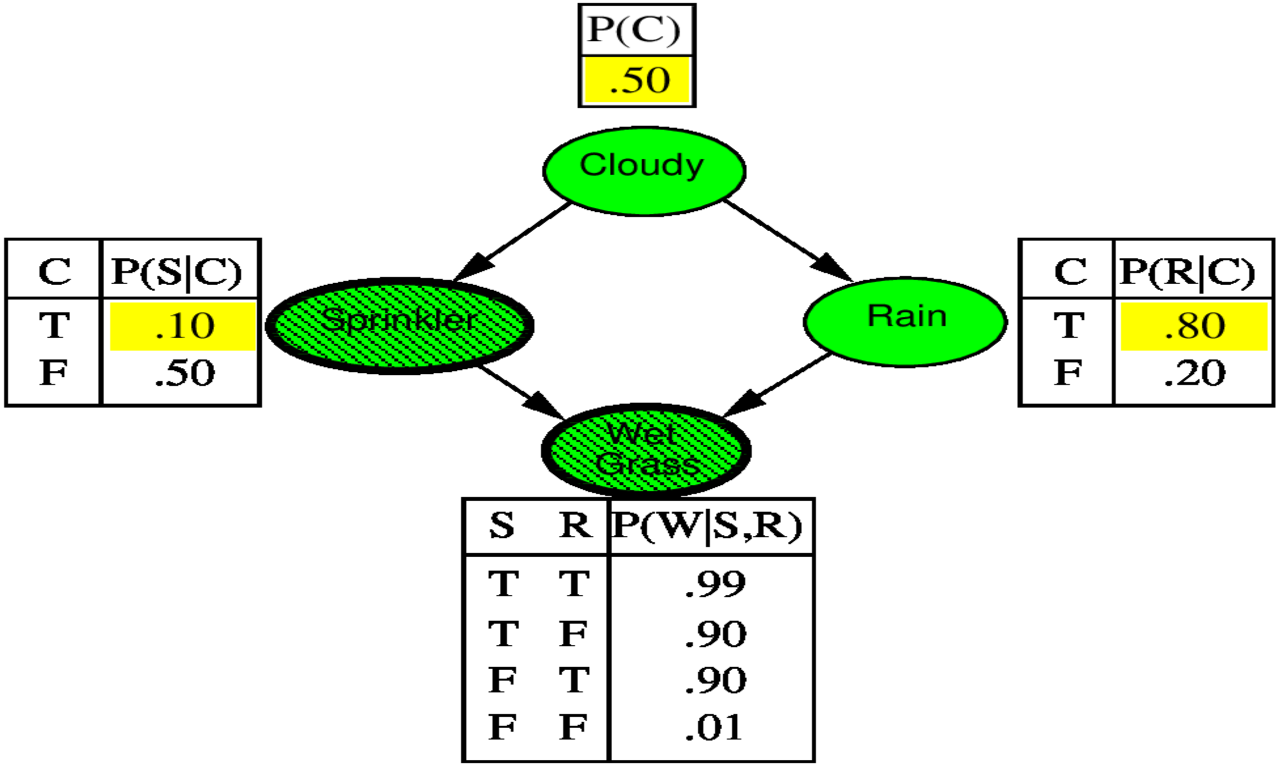
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

w=1*0.1

Likelihood Weighting - Example

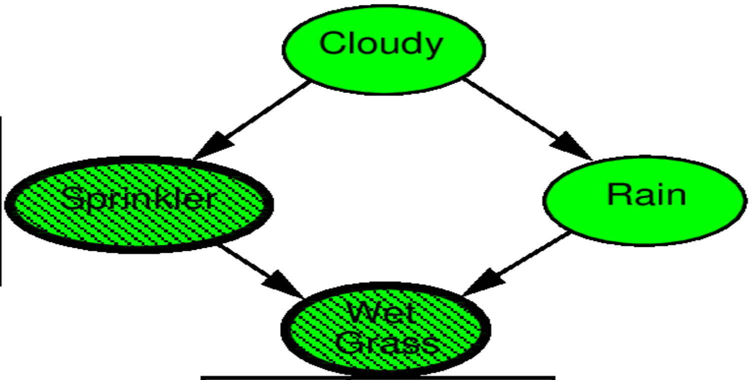


w=1*0.1

TU Delft

Likelihood Weighting - Example

P(C)
.50



C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

w=1*0.1*0.99=0.099

Analysis

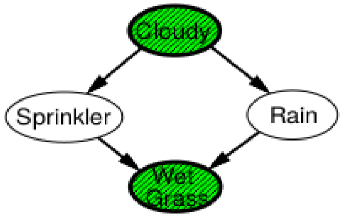
- Can again show this is consistent:

Sampling probability for WEIGHTEDSAMPLE is

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

Note: pays attention to evidence in **ancestors** only

$\Rightarrow$ somewhere "in between" prior and posterior distribution



Weight for a given sample $\mathbf{z},\mathbf{e}$ is

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

Weighted sampling probability is

$$\begin{aligned} S_{WS}(\mathbf{z},\mathbf{e})w(\mathbf{z},\mathbf{e}) \\ &= \Pi_{i=1}^{l} P(z_i|parents(Z_i)) \;\; \Pi_{i=1}^{m} P(e_i|parents(E_i)) \\ &= P(\mathbf{z},\mathbf{e}) \text{ (by standard global semantics of network)} \end{aligned}$$

TU Delft

LW… Problem solved?

- Does likelihood weighting solve all our problems…?

LW… Problem solved?

- Does likelihood weighting solve all our problems…?
- Remaining problems:
  - only takes into account influence of ancestors
    → samples we draw might still be very unlikely given all **e**
    - few samples will have large weight
    - renormalization will put most weight on those
    - many evidence variables → makes things worse…
    - evidence variables late in ordering → makes things worse…
- Many more techniques…
- approximate inference remains an active research topic

Wrap up

Summary Inference

- Agents need to represent beliefs: we consider probability
- Compact representations: Bayesian networks

- Exact inference:
  - Search / Variable Elimination
  - intractable in general, polytime on polytrees

- Approximate inference:
  - "Prior sample" – when no observations
  - Rejection sampling – simple, but not effective
  - likelihood weighting – much better, but problem not solved

- just the simplest sampling methods
  - much more out there…!